

SIMATIC

S7 Software Controller IOT2000EDU




Bedienhandbuch

Wegweiser durch dieses Bedienhandbuch	1
Sicherheitshinweise	2
Beschreibung	3
Vorbereiten des Produkts für die erste Benutzung	4
IOT2000EDU deinstallieren	5
TIA Portal bedienen	6
Shields konfigurieren und bedienen	7
IOT2000EDU bedienen	8
Webserver	9
Funktionsbibliothek	10
OUC Kommunikation	11
Technische Daten	A
Weiterführende Informationen	B
Abkürzungen	C

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

 GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten wird , wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
 WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten kann , wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
 VORSICHT
bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
ACHTUNG
bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.


Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

 WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Inhaltsverzeichnis

1	Wegweiser durch dieses Bedienhandbuch	6
2	Sicherheitshinweise	8
2.1	Security-Hinweise	8
2.2	Einsatz des Produkts	8
3	Beschreibung.....	9
3.1	Funktionsüberblick IOT2000EDU	9
3.2	Erforderliche Komponenten	9
3.3	Betriebsbedingungen	10
4	Vorbereiten des Produkts für die erste Benutzung	11
4.1	Auswahl der Hardware.....	11
4.2	Yocto Linux installieren	12
4.2.1	Voraussetzungen und Einstellungen	12
4.2.2	Installationsschritte	12
4.3	IOT2000EDU installieren	13
4.3.1	Voraussetzungen und Einstellungen	13
4.3.2	IOT2000EDU installieren	14
4.3.3	Installation überprüfen	14
4.3.4	IOT2000EDU bedienen.....	15
4.3.5	IOT2000EDU Support Package im TIA-Portal installieren	17
5	IOT2000EDU deinstallieren.....	19
6	TIA Portal bedienen	20
6.1	Einführung in das TIA-Portal.....	20
6.2	IOT2000EDU im TIA-Portal hinzufügen und konfigurieren.....	20
6.3	IOT2000EDU Programmierung mit TIA-Portal	22
6.4	Laden in Gerät	24
6.5	Online verbinden.....	25
6.6	Diagnoseereignisse	26
7	Shields konfigurieren und bedienen.....	27
7.1	SIMATIC IOT2000, Input/Output Module.....	27
7.2	Arduino Shield konfigurieren.....	32
7.2.1	Address_space zuweisen	34
7.2.2	GPIO zuweisen	35
7.2.3	PWM zuweisen	36
7.2.4	Analog zuweisen.....	37
7.3	Fehlermeldungen	38

8	IOT2000EDU bedienen.....	40
8.1	RUN/STOP/MRES Funktionen via TIA-Portal	40
8.2	RUN/STOP/MRES Funktionen via Kommandozeile.....	41
8.3	RUN/STOP/MRES via Webserver	43
9	Webserver	44
9.1	Einleitung	44
9.2	Voraussetzungen	44
9.3	Webserver mit Linux Kommandos bedienen	44
9.4	Auf Webserver zugreifen	45
9.5	Webserver Seiten.....	45
9.5.1	Einleitung	45
9.5.2	Startseite	46
9.5.3	Identifikation	48
9.5.4	Diagnosepuffer	49
10	Funktionsbibliothek	50
10.1	Überblick über die CPU Funktionsbibliothek	50
10.2	Benutzerdefinierte SO-Datei erstellen	51
10.2.1	Funktion „Execute“ programmieren	52
10.2.2	Weitere Funktionen programmieren	54
10.2.3	Data Access Helper Klassen	55
10.2.4	Datentypen.....	59
10.3	Funktionsbibliothek-Programme mit Eclipse erstellen	60
10.3.1	Alternative Entwicklungsumgebungen	61
10.3.2	SDK Installation in Windows	62
10.3.3	SO-Datei mit dem Beispielprojekt kompilieren	65
10.3.4	Cross Compilen mit der IOT2000 SDK Installation.....	67
10.3.5	SO-Datei auf IOT20X0 laden	69
10.4	TIA Portal Projekt für die Applikation vorbereiten	70
10.4.1	Referenz der IOT2000EDU Funktionsbibliothek Anweisungen	74
10.4.1.1	CREA_COM (SFB65001)	74
10.4.1.2	EXEC_COM (SFB65002).....	75
10.5	Funktionsbibliothek-Programme mit Matlab Simulink erstellen	77
10.5.1	Voraussetzungen	77
10.5.2	Simulink Modell erstellen	78
10.5.3	Beschreibung der Simulink Parameter	80
10.5.4	Simulink Modell in Eclipse integrieren	81
10.5.5	TIA Portal für das Simulink-Modell vorbereiten	84
10.5.6	Datentyp-Umwandlung	86

11	OUC Kommunikation	87
11.1	Verbindungsaufbau	87
11.2	TCON und TDISCON im TIA Portal konfigurieren	88
11.3	Datenaustausch	89
11.4	TSEND, TRCV und TUSEND, TURCV im TIA Portal konfigurieren	90
A	Technische Daten	92
A.1	Technische Daten	92
B	Weiterführende Informationen	95
C	Abkürzungen.....	96

Wegweiser durch dieses Bedienhandbuch

1

Zweck der Dokumentation

Die Informationen des vorliegenden Bedienhandbuchs ermöglichen Ihnen, die Software "SIMATIC S7 Software Controller IOT2000EDU" zu installieren und zu benutzen.

Weiterhin lernen Sie das Arbeiten mit TIA-Portal-Projekten kennen, um die IOT2000EDU zu programmieren, zu starten und zu stoppen.

Leserkreis

Diese Dokumentation wendet sich an Studenten in Bildungseinrichtungen. Sie dient zu Schulungszwecken und vermittelt den Umgang mit TIA Produkten.

Erforderliche Grundkenntnisse

Zum Verständnis der Dokumentation sind folgende Kenntnisse erforderlich:

- Allgemeine Kenntnisse auf dem Gebiet der Automatisierungstechnik
- Kenntnisse des Industrie-Automatisierungssystems SIMATIC
- Kenntnisse der Programmierung mit C/C++
- Kenntnisse im Umgang mit MATLAB/ Simulink und Simulink Coder
- Kenntnisse im Umgang mit Eclipse
- Kenntnisse im Umgang mit Microsoft und Linux Betriebssystemen
- Kenntnisse im Umgang mit Computertechnik

Gültigkeitsbereich der Dokumentation

Die vorliegende Dokumentation ist gültig für folgende Produkte:

- IOT2000EDU: **6ES7671-0LE00-0YB0**
- IOT2020: **6ES7647-0AA00-0YA2**
- IOT2040: **6ES7647-0AA00-1YA2**
- SIMATIC IOT2000, Input/Output Module: **6ES7647-0KA01-0AA2**

Hinweise

Beachten Sie auch die folgendermaßen gekennzeichneten Hinweise:

Hinweis

Ein Hinweis enthält wichtige Informationen zum in der Dokumentation beschriebenen Produkt, zur Handhabung des Produkts oder zu dem Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll.

Definitionen und Namenskonventionen

In dieser Dokumentation werden folgende Oberbegriffe verwendet:

- **Arduino Shield:** ARDUINO UNO (Rev3)
- **SIEMENS Shield:** SIMATIC IOT2000, Input/Output Module
- **IOT2000EDU:** SIMATIC S7 Software Controller IOT2000EDU
- **IOT20x0:** SIMATIC IOT2020 und SIMATIC IOT2040
- **STEP 7:** Zur Bezeichnung der Projektier- und Programmiersoftware verwenden wir in der vorliegenden Dokumentation "STEP 7" als Synonym für die Version "STEP 7 V15 (TIA Portal)".
- **SO:** Shared Object

2.1 Security-Hinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.

Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.

Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.

Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter (<https://www.siemens.com/industrialsecurity>).

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter (<https://www.siemens.com/industrialsecurity>).

2.2 Einsatz des Produkts

Hinweis

Verwenden Sie dieses Produkt nicht im Produktiveinsatz oder zur Steuerung möglicherweise gefährlicher Prozesse. Dieses Produkt richtet sich nur an Studenten und dient daher ausschließlich zu Bildungszwecken.

Beschreibung

3.1 Funktionsüberblick IOT2000EDU

Die IOT2000EDU realisiert eine SIMATIC S7-kompatiblen Software Controller auf dem IOT2020 oder IOT2040 und ist für Ausbildungszwecke konzipiert. Sie ist speziell für die Erlernung Programmierung von Steuerungen mit dem SIMATIC TIA Portal vorgesehen.

Weiterhin ermöglicht Ihnen der TIA-Portal-Programmeditor das Programmieren der IOT2000EDU. Dabei stehen Ihnen Organisations-, Funktions- und Datenbausteine, sowie mit Funktionen zur Verfügung. Mithilfe dieser Bausteine und den im TIA-Portal vorhandenen Libraries können Sie Ihre Programme schreiben.

Das TIA-Portal bzw. das Kommandozeilen Programm "CPU_Control" ermöglicht es Ihnen, die IOT2000EDU zu steuern. Weiterhin können Sie Ihr IOT20x0 Gerät konfigurieren, programmieren sowie die Funktionen RUN, STOP und MRES ausführen.

Hinweis

Einsatz des Produkts

Verwenden Sie dieses Produkt nicht im Produktiveinsatz oder zur Steuerung möglicherweise gefährlicher Prozesse. Dieses Produkt richtet sich nur an Studenten und dient daher ausschließlich zu Bildungszwecken.

3.2 Erforderliche Komponenten

Überprüfen Sie das Produkt auf Vollständigkeit. Folgende Teile gehören zum Lieferumfang der IOT2000EDU:

- IOT2000EDU IPK-Datei zur Installation der Runtime auf dem SIMATIC IOT2020 oder IOT2040
- IOT2000EDU HSP-Datei für die Bereitstellung der Runtime in SIMATIC STEP7 V15
- IOT2000EDU Handbuch als PDF-Datei (deutsch und englisch)
- IOT2000EDU Lizenzaufkleber zum Anbringen auf dem SIMATIC IOT2020 oder IOT2040

3.3 Betriebsbedingungen

Einsatzvoraussetzungen

Die IOT2000EDU ist nur zu Bildungszwecken vorgesehen. Das bedeutet, dass sie nicht in Produktionsanlagen oder sicherheitskritischen Bereichen eingesetzt werden darf.

Hardwarevoraussetzungen

- SIMATIC IOT2020 oder IOT2040
- Stromversorgung 24 V DC
- Empfohlen: Arduino Shield oder Siemens Shield

Hinweis

Benutzen der IOT2000EDU ohne Shield

Wenn Sie kein Shield konfigurieren, läuft die IOT2000EDU ohne Zugriff auf die I/O-Schnittstellen via der Arduino Pins.

Außerdem erscheint folgende Meldung in "var/log/messages": "Can not open IO configuration file, IO Module not initialized."

- microSD-Karte
- LAN-Kabel
- PC mit Host OS

Softwarevoraussetzungen

- SIMATIC IOT2000EDU
- TIA-Portal V15 oder höher
- Hardware Support Package "HSP_V15_0245_001_S7_IOT2000EDU_1.1.isp15"
- Yocto Linux OS
- OPKG Package Manager Version 0.3.3 und höher
- MRAA 1.6.1
- Libc6 und libstdc++6
- SSH oder installiertes Serial Terminal
- 200 MB freier Speicher auf dem root-Dateisystem
- Root Privilegien

Vorbereiten des Produkts für die erste Benutzung

4.1 Auswahl der Hardware

Es gibt zwei SIMATIC IOT20x0 Versionen:

- SIMATIC IOT2020
- SIMATIC IOT2040

SIMATIC IOT2020

- Intel Quark® x1000
- 512 MB RAM
- 1 Ethernet-Schnittstelle
- 1 USB Host Type A
- 1 USB Client microUSB

SIMATIC IOT2040

- Intel Quark® x1020
- 1 GB RAM
- 2 Ethernet-Schnittstellen
- 1 USB Host Type A
- 1 USB Client microUSB
- 2 RS232/485 Schnittstellen
- Batterie gepufferte RTC

4.2 Yocto Linux installieren

4.2.1 Voraussetzungen und Einstellungen

Die SIMATIC IOT2000EDU verwendet für den Betrieb ein Yocto Linux Betriebssystem, welches auf einer Micro-SD Karte installiert sein muss. Voraussetzung dafür ist, dass Sie über eine Micro-SD Karte verfügen, die eine Speichergröße von 8 GB bis 32 GB besitzt.

Diese Dokumentation geht davon aus, dass Sie bereits eine Inbetriebnahme der IOT durchgeführt haben. Ergänzende Informationen zur Inbetriebnahme finden Sie hier (<https://support.industry.siemens.com/tf/ww/en/posts/setting-up-the-simatic-iot2000/155642?page=0&pageSize=10>).

4.2.2 Installationsschritte

SD-Karten Beispielimage herunterladen oder selbst erstellen

Um den vollen Funktionsumfang der IOT2000EDU nutzen zu können, benötigen Sie eine SD-Karte mit einem installierten Yocto Linux Betriebssystem. Dieses wird für Sie im Siemens Industry Online Support Portal (SIOS-Portal) zur Verfügung gestellt. Sie können sich es hier (<https://support.industry.siemens.com/cs/document/109741799/simatic-iot2000-sd-card-beispielimage?dti=0&lc=de-WW>) herunterladen.

Sie können sich aber auch selbst ein Image erstellen. Eine Anleitung dazu finden Sie im Internet hier (<https://github.com/siemens/meta-iot2000>).

Wenn Sie dieser Anleitung folgen und nicht das Beispielimage aus dem SIOS-Portal verwenden wollen, müssen Sie vor dem Image Building Vorgang folgende Software installieren:

- OPKG Package Manager Version 0.3.3 und höher
- MRAA 1.6.1
- Libc6 und libstdc++6

OPKG Package Manager installieren

Den OPKG Package Manager installieren Sie mithilfe des Yocto Build System, indem Sie das image feature "package-management" in die Variable "EXTRA_IMAGE_FEATURES" einfügen und `PACKAGE_CLASSES ?= "package_ipk"` setzen.

Detaillierte Informationen zu Yocto-Variablen finden Sie im Internet hier (<http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#ref-features-image>) und hier (http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#var-PACKAGE_CLASSES).

MRAA 1.6.1 installieren

Die IOT2000EDU benötigt Intels MRAA Library (1.6.1) für Arduino basierte IO-Aufgaben. Weitere Informationen finden Sie hier (<https://github.com/intel-iot-devkit/mraa#installing-on-intel-32bit-yocto-based-opkg-image>).

Libc6 und libstdc++6 installieren

Weiterhin müssen Sie folgende zusätzliche Variablen setzen, bevor Sie den Image Building Vorgang beginnen:

```
IMAGE_INSTALL_append = " libstdc++"
```

Um weitere Packages wie ssh-server-openssh zu nutzen, ergänzen Sie die Datei "bbblayer.conf" um den Eintrag `meta-oe meta layer`.

Weiterhin benötigt die IOT2000EDU libjson-c 0.12, was in dem ipk-Package enthalten ist.

4.3 IOT2000EDU installieren

4.3.1 Voraussetzungen und Einstellungen

Damit die Installation der IOT2000EDU reibungslos ablaufen kann, sollten Sie folgende Voraussetzungen erfüllen sowie ihr Linux Betriebssystem entsprechend der folgenden Empfehlungen vorbereiten.

- IOT2020 bzw. IOT2040
- Installiertes Linux OS auf einer microSD-Karte
 - Linux mit TCP/IPv4 Support
 - libc6 und libstdc++6
 - mraa 1.6.1
 - OPKG Package Manager Version 0.3.3 und höher
 - IKP Package Ökosystem
 - 200 MB freier Speicher auf dem root-Dateisystem
 - Root Privilegien
- Ein Host OS mit:
 - SSH oder installiertem Serial Terminal
 - Tastatur
 - Monitor
 - Serial FTDI Kabel oder LAN Kabel

4.3.2 IOT2000EDU installieren

Voraussetzung

- Root Rechte

Vorgehensweise

1. Überprüfen Sie die opkg Installation: `opkg -v`
2. Installieren Sie IOT2000EDU_<version-info>_i586.ipk mittels des OPKG Package Managers mit dem Command: `opkg install IOT2000EDU_<version-info>_i586.ipk`

Beispiel für die Version 1.1.0: `opkg install IOT2000EDU_1.1.0_i586.ipk`

Ergebnis

Sie finden die ausführbaren Dateien "IOT2000EDU" und "CPU_Control" unter "/usr/local/bin".

Die voreingestellte IO-Konfigurationsdatei "io.conf" finden Sie in den Verzeichnissen "/usr/local/etc".

Die "Libjson-c"-Dateien finden Sie in den Verzeichnissen "/usr/lib" und "/usr/include/json-c".

4.3.3 Installation überprüfen

1. Überprüfen Sie die installierten Packages mit folgendem OPKG List Command:
 - `opkg info IOT2000EDU`
Dieser Command stellt den Packagenamen dar, der von der OPKG-Datenbank abgerufen wird.
2. Die ausführbaren Dateien und deren Versionen lassen sich mit den Parametern `-v` oder `--version` überprüfen:
 - `/usr/local/bin/IOT2000EDU -v`
 - `/usr/local/bin/CPU_Control -v`

Der OPKG Manager zeigt eine Fehlermeldung an und beendet die Installation, wenn Abhängigkeiten (Dependencies) wie zum Beispiel "mraa", fehlen.

Wenn Sie die Installation des Packages erneut starten, zeigt der OPKG eine Fehlermeldung an und beendet die Installation.

Während der Installation des IOT2000EDU Installationspakets "IOT2000EDU_1.1.0_i586.ipk" wird auch der Webserver mitinstalliert. Im Kapitel "Webserver (Seite 44)" erfahren Sie mehr dazu.

4.3.4 IOT2000EDU bedienen

Starten der IOT2000EDU

Starten Sie die Applikation "IOT2000EDU" im Installationspfad mit einer User-Rolle.

Sie können die IOT2000EDU Applikation mit dem "&" Suffix starten, damit diese im Hintergrund ausgeführt wird. Dieser Vorgang befreit die Shell für die nächsten Benutzereingaben.

Beispiel:

```
/usr/local/bin/IOT2000EDU&
```

Mit dem Präfix `./`, der das aktuelle Verzeichnis kennzeichnet, können Sie die Applikation vom aktuellen Pfad ausführen.

Um die Applikation von einem beliebigen Pfad auszuführen, müssen Sie den vollständigen Ablagepfad anstelle des aktuellen Pfadpräfixs angeben.

Weiterhin können Sie auch den `init.d` Autostart Service verwenden, welcher bereits durch das IOT2000 Yocto Image vorinstalliert ist. Wenn Sie den `init.d` Service für die IOT2000 einstellen, startet die IOT2000EDU automatisch beim Hochfahren des Linux OS.

Logs

Logs werden in die Datei `"/var/log/messages"` erstellt.

I/O Konfiguration

Während der Installation des IOT2000EDU_<version>_i586.ipk Pakets, wird die vorgefertigte Beispieldatei `io.config.sample` in das Verzeichnis `/usr/local/etc` installiert. Diese Beispieldatei beinhaltet die Pin-Konfigurationseinstellungen "SIMATIC IOT IOT2000, Input/Output Module".

Wenn die IOT2000EDU das erste Mal gestartet wird und der Ordner "IOT2000EDU" im "\$HOME"-Verzeichnis nicht existiert, wird der Ordner angelegt und die Beispieldatei `"$HOME/IOT2000EDU/io.conf.sample"` angelegt.

Um die Pins der IOT20x0 zu verwenden, passen Sie entweder die Datei `"io.conf.sample"` an und ändern Sie den Namen in `"io.conf"`, oder erstellen Sie manuell eine `io.conf`-Datei.

Wenn die IOT2000EDU beim Starten die Datei `"$HOME/IOT2000EDU/io.conf"` nicht vorfindet, startet die IOT2000EDU weiter und die Arduino Pins werden nicht initialisiert.

Alternativ können Sie die manuell erstellte `io.conf`-Datei auch mit den Parametern `-c` oder `--conf` an die IOT2000EDU weitergeben.

Beispiel:

- `/usr/local/bin/IOT2000EDU -c <Dateiname mit Pfad der manuell erstellten io.conf-Datei>`
- `/usr/local/bin/IOT2000EDU --conf <Dateiname mit Pfad der manuell erstellten io.conf-Datei>`

Wenn Sie den Zustand der CPU auf "STOP" oder "SHUTDOWN" einstellen, z.B. mit "CPU_Control" oder dem TIA-Portal, dann bleiben die PWMs stehen und die GPIO Outputs werden auf "0" gesetzt. Am Anfang des "run"-Zustands werden alle Pins auf die von Ihnen in der Datei `io.conf` eingetragenen Standardeinstellungen zurückgesetzt.

waf-Dateien

Die IOT2000EDU überprüft beim Start das Verzeichnis `"$HOME/IOT2000EDU"` nach instanzspezifischen `waf`-Dateien. Die IOT2000EDU speichert das vom TIA-Portal geladene Steuerungsprogramm und die Hardware-Konfiguration in der `waf`-Datei. Die `waf`-Datei wird beim Start geladen und ausgeführt, wenn die PLC im Zustand "RUN" ist. Wenn die IOT2000EDU diese `waf`-Dateien nicht finden kann, generiert sie projektspezifische `waf`-Dateien in das Verzeichnis `"$HOME/IOT2000EDU"`.

Die IOT2000EDU verwendet dabei voreingestellt das Verzeichnis `"$HOME/IOT2000EDU"`. Sie können jedoch mithilfe der Parameter `-p <Projektverzeichnis>` oder `--projectdir <Projektverzeichnis>` den Pfad des Projektverzeichnisses anpassen.

Beispiel:

- `/usr/local/bin/IOT2000EDU -p <Pfad zu anderem Projektverzeichnis>`
- `/usr/local/bin/IOT2000EDU --projectdir <Pfad zu anderem Projektverzeichnis>`

Versionsinformationen anzeigen

Um die Versionsinformationen anzuzeigen, verwenden Sie die Parameter `-v` oder `--version`.

Beispiel:

- `/usr/local/bin/IOT2000EDU -v`
- `/usr/local/bin/IOT2000EDU --version`

Ändern des Instanznamens der Runtime

Um den Instanznamen der IOT2000EDU zu ändern, verwenden Sie die Parameter `-i` oder `--instancename`.

Beispiel:

- `/usr/local/bin/IOT2000EDU -i <neuer Instanzname>`
- `/usr/local/bin/IOT2000EDU --instancename <neuer Instanzname>`

Hilfenachrichten anzeigen

Um die Hilfenachrichten der IOT2000EDU anzuzeigen, verwenden Sie die Parameter `-h` oder `--help`.

Beispiel:

- `/usr/local/bin/IOT2000EDU -h`
- `/usr/local/bin/IOT2000EDU --help`

4.3.5 IOT2000EDU Support Package im TIA-Portal installieren

In diesem Kapitel erfahren Sie, wie Sie das Hardware Support Package (HSP) der IOT2000EDU im TIA-Portal installieren. Dieses ist notwendig, damit Sie mit der IOT2000EDU im TIA-Portal arbeiten können.

Voraussetzungen

- Sie haben das TIA-Portal V15 (oder höher) installiert.
- Ihnen liegt das HSP "HSP_V15_0245_001_S7_IOT2000EDU_1.1.isp15" vor.

Vorgehensweise

1. Starten Sie das TIA-Portal als Administrator.
2. Klicken Sie im TIA Portal in der Menüleiste "Extras" den Eintrag "Support Packages". Der Dialog "Detailinformation" wird geöffnet. In einer Tabelle werden alle Support Packages aus dem Verzeichnis aufgelistet, das Sie als Speicherort für Support Packages in den Einstellungen ausgewählt haben.
3. Klicken Sie auf "Aus dem Dateisystem hinzufügen"
4. Wählen Sie die Datei "HSP_V15_0245_001_S7_IOT2000EDU_1.1.isp15" aus.
5. Wählen Sie das Support Package aus und klicken Sie auf "Installieren".
6. Beenden Sie das TIA Portal und starten Sie es erneut.

Ergebnis

Nach der erfolgreichen Installation steht Ihnen nun die IOT2000EDU im TIA-Portal Gerätecatalog zur Verfügung, sowie alle anderen Funktionen wie Konfiguration, Laden oder Online verbinden.

IOT2000EDU deinstallieren

So gehen Sie vor, wenn Sie die IOT2000EDU wieder deinstallieren möchten:

1. Starten Sie die Deinstallation mit dem folgenden OPKG Command:

```
opkg remove IOT2000EDU
```

Durch diesen Command werden folgende Dateien entfernt:

- "io.conf.sample" unter "/usr/local/etc/"
- "IOT2000EDU" und "CPU_Control" unter "/usr/local/bin/"
- libjson-c Dateien in unter "/usr/lib/" und "/usr/include/json-c/"

Die benutzerspezifischen Dateien im Verzeichnis "IOT2000EDU" werden bei der Deinstallation nicht entfernt.

Wenn Sie versuchen, das Paket mehrfach zu deinstallieren, erscheint eine Meldung, dass kein Paket zum entfernen vorhanden ist.

TIA Portal bedienen

6.1 Einführung in das TIA-Portal

Das Totally Integrated Automation Portal (TIA-Portal) integriert verschiedene SIMATIC Produkte in eine Software-Anwendung, mit der Sie Ihre Produktivität und Effizienz erhöhen können. Innerhalb des TIA-Portals arbeiten die TIA-Produkte zusammen und unterstützen Sie in allen Bereichen bei der Erstellung einer Automatisierungslösung.

Die wichtigsten Projektierungsschritte sind:

- Anlegen des Projekts
- Konfigurieren der Hardware
- Vernetzen der Geräte
- Programmieren der Steuerung
- Projektieren der Visualisierung
- Laden der Projektierungsdaten
- Verwenden der Online- und Diagnosefunktionen

Im folgenden Kapitel erfahren Sie, wie Sie die IOT2000EDU im TIA-Portal anlegen, konfigurieren und programmieren können.

6.2 IOT2000EDU im TIA-Portal hinzufügen und konfigurieren

In diesem Kapitel erfahren Sie, wie Sie die IOT2000EDU im TIA-Portal hinzufügen.

Um ein Gerät im TIA-Portal hinzuzufügen, gibt es mehrere Wege. Dieses Kapitel beschreibt die Vorgehensweise über den Dialog "Neues Gerät hinzufügen".

Voraussetzung

- Sie haben das TIA-Portal gestartet.
- Sie befinden sich in der Portalansicht.
- Sie haben ein neues Projekt angelegt.

Neues Gerät hinzufügen

1. Klicken Sie auf die Option "Ein Gerät konfigurieren".
2. Klicken Sie auf die Option "Neues Gerät hinzufügen".
Es öffnet sich der Dialog "Neues Gerät hinzufügen".
3. Klicken Sie auf "PC-Systeme".
4. Navigieren Sie zu dem Eintrag unter "PC Systeme > SIMATIC IOT2000EDU" > "6ES7 671-0LE00-0YB0" und wählen Sie ihn aus.
Auf der rechten Seite des Dialogs wird Ihnen ein Vorschaubild der IOT sowie deren Artikelnummer, Version und Beschreibung angezeigt.
5. Klicken Sie auf die Schaltfläche "Hinzufügen".

Ergebnis

Die Projektansicht öffnet sich und die IOT2000EDU wurde Ihrem Projekt hinzugefügt.

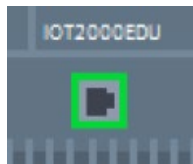
IOT2000EDU konfigurieren

Nachdem Sie die IOT2000EDU als neues Gerät angelegt haben, können Sie die IOT2000EDU und deren Schnittstelle einsehen und konfigurieren, sowie den Namen ändern.

- Klicken Sie im Arbeitsbereich auf das Gerät "IOTPLC_1".
Im Inspektorfenster öffnet sich die Registerkarte "Eigenschaften" der IOT2000EDU.



- Klicken Sie im Arbeitsbereich auf die grün umrandete Schnittstelle.
Im Inspektorfenster öffnet sich die Registerkarte "Eigenschaften" der IOT-Schnittstelle.



In der Registerkarte "Eigenschaften" können Sie die IP- und Subnetzmaske-Adresse eingeben.

Hinweis

Die in den Schnittstellen-Eigenschaften eingestellte IP-Adresse muss die gleiche Adresse sein, wie die der IOT20x0.

6.3 IOT2000EDU Programmierung mit TIA-Portal

In diesem Kapitel erfahren Sie, welche Mittel Ihnen bei der Programmierung der IOT2000EDU im TIA-Portals zur Verfügung stehen.

Dabei lernen Sie, wie Sie Organisations-, Funktions- und Datenbausteine und Funktionen einfügen können.

Organisationsbausteine

Folgende Organisationsbausteine werden von der IOT2000EDU unterstützt und können im TIA-Portal konfiguriert werden:

OB	Beschreibung
OB 1	Freier Zyklus
OB 10	Uhrzeitalarme
OB 20	Verzögerungsalarme
OB 30 - OB 38	Weckalarme ¹⁾
OB 100, OB 102	Anlauf
OB 80, OB 84, OB 85	Asynchrone Fehleralarme
OB 121	Synchrone Fehleralarme

1) Wenn ein Weckalarm-OB die Zykluszeit überschreitet, wird der Zeitfehler OB (OB 80) gestartet.

Dialog "Neuen Baustein hinzufügen"

1. Doppelklicken Sie in der Projektnavigation auf den Eintrag "Neuen Baustein hinzufügen", welcher sich im Ordner "Programmbausteine" befindet.
Es öffnet sich der Dialog "Neuen Baustein hinzufügen".

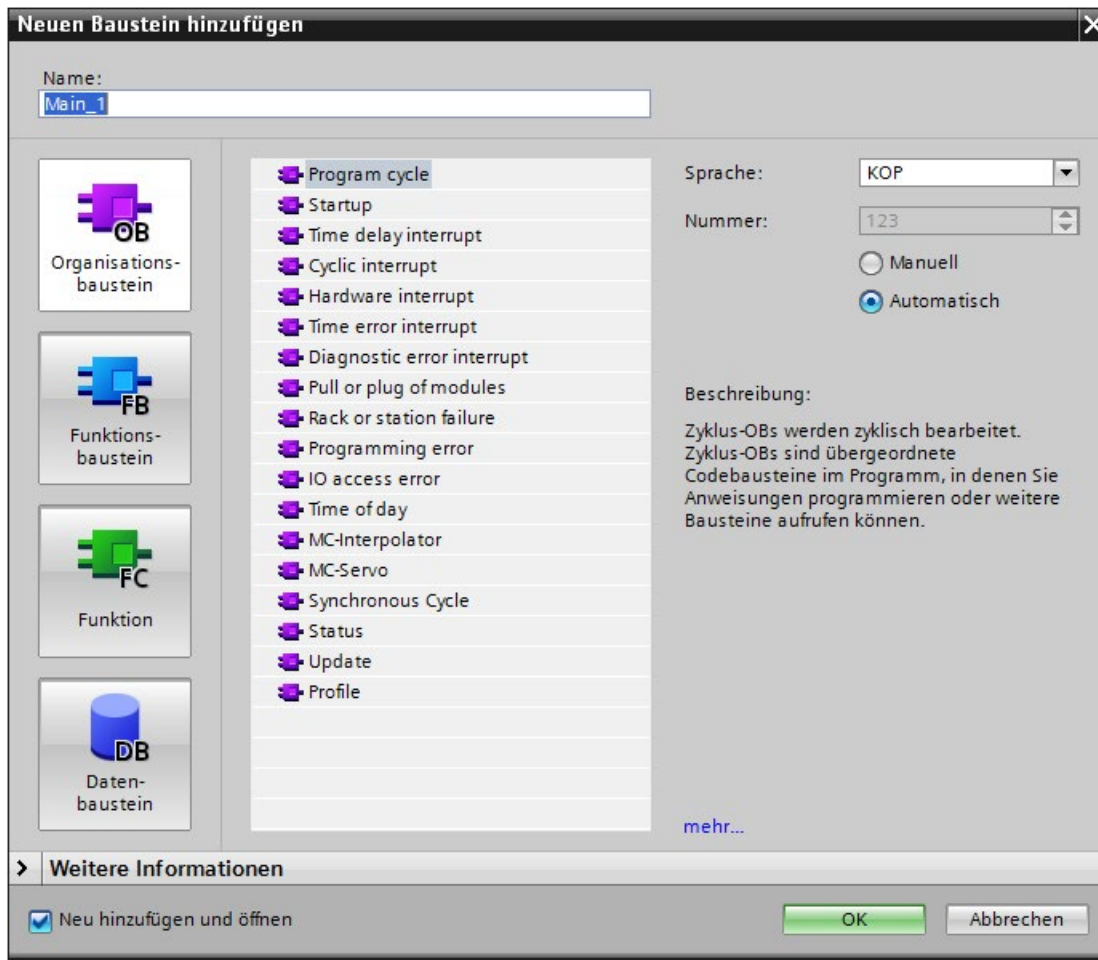


Bild 6-1 Bausteine

2. Wählen Sie im Dialog "Neuen Baustein hinzufügen" einen Organisations-, Funktions- oder Datenbaustein oder einer Funktion aus.
3. Im Feld "Sprache" können Sie sich für eine Programmiersprache entscheiden.
4. Bestätigen Sie die Auswahl mit "OK".

Palette "Anweisungen"

In der Palette "Anweisungen" stehen Ihnen alle verfügbaren Bibliotheken zur Verfügung, welche Sie bei Ihrer Programmierung unterstützt. Nicht verfügbare Bibliotheken werden ausgegraut dargestellt und können nicht verwendet werden.

1. Doppelklicken Sie auf ein Bibliothekselement.
Es öffnet sich der Dialog "Aufrufoptionen".
2. In diesem Dialog können Sie den Name und die Nummer des Elements ändern.
3. Bestätigen Sie mit "OK".


6.4 Laden in Gerät

In diesem Kapitel erfahren Sie, wie Sie Ihre getroffenen Einstellungen auf das Gerät laden.

Voraussetzungen

- Sie haben die IOT20x0 per Ethernet mit Ihrem PC verbunden.
- Sie haben die IOT2000EDU gestartet.
- Sie haben das Gerät im TIA-Portal angelegt.
- Sie haben in den Eigenschaften der Schnittstelle eine IP-Adresse vergeben.

Vorgehensweise

1. Klicken Sie in der Funktionsleiste auf die Schaltfläche "Laden in Gerät" .
Wenn Sie ein Projekt zum ersten Mal auf ein Gerät laden, wird automatisch der Dialog "Erweitertes Laden" geöffnet.
2. Weisen Sie In diesem Dialog, abhängig von der verwendeten Runtime des Geräts, Protokoll und Schnittstelle oder den Zielpfad für das Projekt zu.
3. Wählen Sie die PG/PC-Schnittstelle aus, an der Ihr PC mit der IOT2000EDU verbunden ist.
4. Wählen Sie im Dropdown-Menü eine Suche aus.

Zielgerät auswählen:			
Gerät	Gerätetyp	Schnittstellentyp	Ad
—	—	PN/IE	Zu

Alle kompatiblen Teilnehmer anzeigen
▼

Geräte mit gleichen Adressen anzeigen
Alle kompatiblen Teilnehmer anzeigen
Erreichbare Teilnehmer anzeigen

5. Klicken Sie auf "Suche starten".

6. Sobald die Suche die angeschlossene IOT2000EDU gefunden hat, klicken Sie auf "Laden".
Das Projekt wird übersetzt. Warnungen und Fehler beim Übersetzen werden im Dialog "Vorschau laden" angezeigt.
7. Klicken Sie nach dem erfolgreichen Übersetzen auf Laden.

Ergebnis

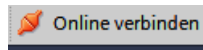
Das Projekt wurde erfolgreich auf das Gerät geladen.

6.5 Online verbinden

Dieses Kapitel erklärt Ihnen, wie Sie das Gerät online verbinden können und bringt Ihnen die Funktionen RUN, STOP und MRES der SIMATIC IOT2000EDU via TIA-Portal näher.

Online verbinden

Nachdem Sie die Funktion "Laden in Gerät" erfolgreich ausgeführt haben, klicken Sie auf die Schaltfläche "Online verbinden" in der Funktionsleiste, um eine Verbindung mit dem Gerät herzustellen.



"RUN", "STOP" und "MRES"

Sobald Sie mit dem Gerät online verbunden sind, stehen Ihnen die Funktionen RUN, STOP und MRES zur Verfügung. Mithilfe dieser Funktionen können Sie die CPU starten, stoppen und Umlöschen. Wie Sie diese Funktionen nutzen erfahren Sie ausführlich im Kapitel "RUN/STOP/MRES Funktionen via TIA-Portal (Seite 40)".

Hinweis

Führen Sie die RUN-Funktion nur aus, wenn sich die IOT2000EDU im "STOP"-Modus befindet. Umgekehrt führen Sie die STOP-Funktion nur aus, wenn sich die IOT2000EDU im "RUN"-Modus befindet. Andernfalls erscheint eine Fehlermeldung.

6.6 Diagnoseereignisse

Nachdem Sie sich mit dem Gerät online verbunden haben, können sie die Funktion "Online & Diagnose" nutzen.

Vorgehensweise

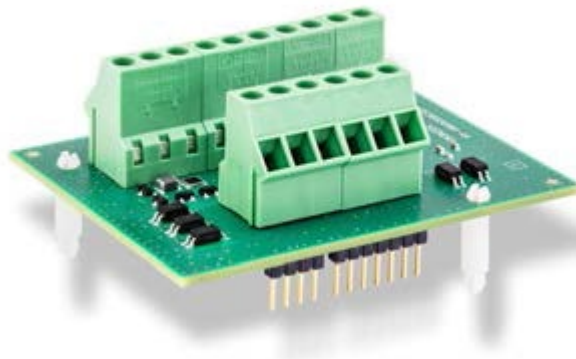
1. Doppelklicken Sie im Projektnavigator auf "Online & Diagnose".
Es öffnet sich die Ansicht "Online & Diagnose" im Arbeitsbereich.
2. Öffnen Sie das Menü "Diagnosepuffer" unter "Diagnose" > "Diagnosepuffer".

Im Diagnosepuffer werden Ihnen die Ereignisse in der Reihenfolge ihres Auftretens angezeigt.

Shields konfigurieren und bedienen

7.1 SIMATIC IOT2000, Input/Output Module

In diesem Kapitel erfahren Sie anhand des folgenden Beispiels, welche Einstellungen Sie für die Pin-Konfiguration eines Siemens Arduino Shields treffen müssen.



Der Siemens Shield besitzt 5 digitale Eingänge, 2 digitale Ausgänge und 2 analoge Eingänge. Die analogen Eingänge lassen sich als Strom und Spannung auswählen. Detaillierte Informationen zum Siemens Shield "Input/Output Module" finden Sie hier (<https://support.industry.siemens.com/cs/document/109745681/iot2000-io-input-output-module?dti=0&lc=de-DE>).

Adresse zuweisen

Weisen Sie zunächst die I/O-Adressen zu. Die I/O-Adressen dürfen sich dabei nicht im gleichen Prozessabbild (Eingang/Ausgang) Speicherbereich überschneiden. Die Input- und Output-Adressen müssen ohne Überschneidung angeordnet sein.

Beispiel:

Digital_in_start kann ein Wert zwischen 0 und 509 sein. Wenn Sie den Wert "100" wählen, werden die Werte "101" und "102" zugewiesen. Ein anderer Input-Adressbereich, z. B. der Wert für "analog_in_start" muss zwischen 0 und 500 sein und nicht 100, 101 und 102.

Programmcode

```
"address_space": [{
  "digital_in_start":100,
  "digital_out_start":100,
  "pwm_out_start":103,
  "analog_in_start":103,
}],
```

GPIO zuweisen

Die 5 digitalen Eingänge des Siemens Shields haben die Pinnummern 4, 9, 10, 11 und 12. Wenn Sie für "digital_in_start" den Wert "100" eingeben, bedeutet dies:

I-Adresse	102					101								100							
Bit	Reserviert	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Arduino-Pin	Reserviert	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Arduino Pin	Prozessabbild (I-Bereich)	I/O Beschreibung auf dem Shield
4	I100.4	DI4
9	I101.1	DI3
10	I101.2	DI2
11	I101.3	DI1
12	I101.4	DI0

Die 2 digitalen Ausgänge des Siemens Shields haben die Pinnummern 7 und 8. Wenn Sie für "digital_out_start" den Wert "100" eingeben, bedeutet dies:

Q-Adresse	102					101								100							
Bit	Reserviert	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Arduino-Pin	Reserviert	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Arduino Pin	Prozessabbild (I-Bereich)	I/O Beschreibung auf dem Shield
7	Q100.7	DQ1
8	Q101.0	DQ0

PWM Ausgänge zuweisen

Das Siemens Arduino Shield besitzt keine PWM Ausgänge.

Analog zuweisen

Die 2 analogen Eingänge können Sie entweder als "0 bis 10 V" oder "0 bis 20 mA" eingeben. Die Pinauflösung dieser Eingänge beträgt 9 Bit bei diesem Shield. Wenn Sie für "analog_in_start" den Wert 103 eingeben, bedeutet dies:

I-Adresse	113	111	109	107	105	103
Arduino-Pin	A5	A4	A3	A2	A1	A0

Arduino Pin	Prozessabbild (I-Bereich)	I/O Beschreibung auf dem Shield
A0	IW103	U0
A1	IW105	I0
A2	IW107	U1
A3	IW109	I1

Inhalt der Beispieldatei io.conf.sample des Siemens IOT2000 Arduino Shield

Programmcode	Kommentar
<pre>{ " comments": [{ "name": "compatibility": "supports": "gpio": "analog": "address": }], "address_space": [{ "digital_in_start":100, "digital_out_start":100, "pwm_out_start":103, "analog in start":103, }], "gpio": [{ "pin": 4, "is_output": 0, "initial_value": 0 }, { "pin": 9, "is_output": 0, "initial_value": 0 }, { "pin": 10,</pre>	<pre>"SIMATIC IOT2000, Input/Output Module", "Arduino Uno R3", "IOT2020 & IOT2040 Devices", "5 Digital Input Pin: DI4,DI3,DI2,DI1,DI0, "2 Digital Output Pin: DQ1, DQ0", "2 Analog Input Pin: U0,I0,U1,I1 (0 .. 10 V or 0 ... 20 mA can be selected), Resolution:9-bit", "Look at the surface of IOT2000 io module and see port names. I/O Description Arduino Pin PLC Adress DI4 4 I100.4 DI3 9 I101.1 DI2 10 I101.2 DI1 11 I101.3 DI0 12 I101.4 DQ1 7 Q100.7 DQ0 8 Q101.0 U0 A0 IW 103 I0 A1 IW 105 U1 A2 IW 107 I1 A3 IW 109",</pre>

Programmcode	Kommentar
<pre> "is_output": 0, "initial_value": 0 }, { "pin": 11, "is_output": 0, "initial_value": 0 }, { "pin": 12, "is_output": 0, "initial_value": 0 }, { "pin": 7, "is_output": 1, "initial_value": 0 }, { "pin": 8, "is_output": 1, "initial_value": 0 }], "analog": [{ "pin": 0, "pin_resolution":9 }, { "pin": 1, "pin_resolution":9 }, { "pin": 2, "pin_resolution":9 }, { "pin": 3, "pin_resolution":9 }] } </pre>	

Beispiel:

Demo_V04 ▶ IOT2000 Sending [IOT2000EDU] ▶ Program blocks ▶ Main [OB1]

Block interface

1	L	"U0 - A0"	%IW103
2	T	"DB_Send".Pot_U0	%DB1.DBW2
3			
4	L	"I0 - A1"	%IW105
5	T	"DB_Send".Pot_I0	%DB1.DBW4
6			
7	L	"U1 - A2"	%IW107
8	T	"DB_Send".Pot_U1	%DB1.DBW6
9			
10	L	"I1 - A3"	%IW109
11	T	"DB_Send".Pot_I1	%DB1.DBW8

▼ Network 2: Digital Inputs

▶ The code only reads the digital inputs (buttons)...

1	//L	"Tag_3"	
2	//T	"Tag_4"	
3	//L	"Tag_5"	
4	//T	"Tag_6"	
5			
6	A	"Button 1"	%I100.4
7	=	"DB_Send".Button1	%DB1.DBX0.0
8			
9	A	"Button 2"	%I101.1
10	=	"DB_Send".Button2	%DB1.DBX0.1
11			
12	A	"Button 3"	%I101.2
13	=	"DB_Send".Button3	%DB1.DBX0.2
14			
15	A	"Button 4"	%I101.3
16	=	"DB_Send".Button4	%DB1.DBX0.3
17			
18	A	"Button 5"	%I101.4
19	=	"DB_Send".Button5	%DB1.DBX0.4

7.2 Arduino Shield konfigurieren

Es ist möglich, statt dem Siemens Shield ein Arduino Shield zu verwenden.

Das folgende Kapitel erläutert die notwendigen Schritte, wie Sie ein ARDUINO UNO (Rev3) konfigurieren müssen und erklärt Ihnen, wie Sie die Konfigurationsdatei `io.conf` anpassen müssen.

Schnittstellen des IOT20x0 Arduino Shields

Die folgende Abbildung zeigt das Motherboard der SIMATIC IOT20x0, die Schnittstellen X10, X11, X12 und X13 sowie die Anordnung der Pins. Die orangenen Rechtecke markieren den ersten Pin an der jeweiligen Schnittstelle.

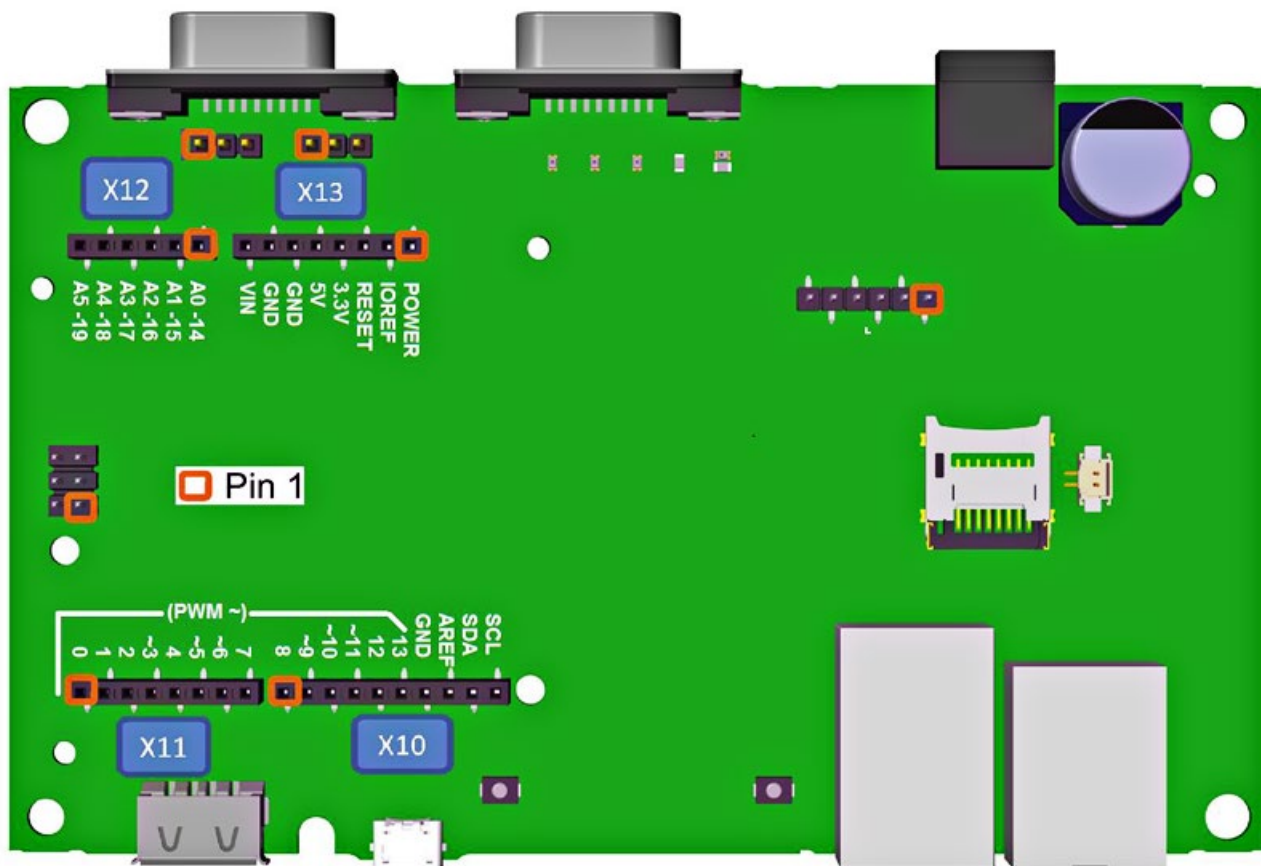


Bild 7-1 IOT20x0 Motherboard mit Schnittstellen

Die folgende Tabelle zeigt die Pinnummern der Schnittstellen in der Datei io.conf in Abhängigkeit des Operationsmodus.

Schnittstelle	Pin	io.conf Pinnummern		
		Digital	Analog	PWM
X11	1	0		
	2	1		
	3	2		
	4	3		3
	5	4		
	6	5		5
	7	6		6
	8	7		
X10	1	8		
	2	9		9
	3	10		10
	4	11		11
	5	12		
	6	13		
	7			
	8			
	9	18	4	
	10	19	5	
X12	1	14	0	
	2	15	1	
	3	16	2	
	4	17	3	
	5	18	4	
	6	19	5	

Die Schnittstelle X13 wird nicht in der Software konfiguriert.

Konfigurationsdatei io.conf

Befüllen Sie die folgenden Abschnitte in der Konfigurationsdatei:

1. "address_space": Festlegung der Startadresse von PWM, GPIO und AIO.
2. "gpio": Bestimmung eines Arduino Pins als Digitaler Input/Output.
3. "pwm": Bestimmung eines Arduino Pins als PWM mit seiner Periode.
4. "analog": Bestimmung eines Arduino Pins als AIO mit seiner Auflösung.

In den folgenden Abschnitten erfahren Sie, wie Sie die einzelnen Abschnitte in der Konfigurationsdatei befüllen müssen.

7.2.1 Address_space zuweisen

Definieren Sie PIN Adressen im folgenden Format:

Programmcode

```
"address_space": [{
    "digital_in_start" : gpioIN_address,
    "digital_out_start" : gpioOut_address,
    "pwm_out_start" : pwm_address,
    "analog_in_start" : aio_address,
}],
```

Diese Adressen korrespondieren zu dem voreingestellten Prozessabbild (OB1 PI) Adressen im PLC Programm. Nur der Adressbereich 0 bis 511 kann in Form eines Bytes verwendet werden.

Innerhalb dieses Adressenabschnitts befinden sich:

- "digital_in_start" (verwendet 3 Bytes) und "analog_in_start" (verwendet 12 Bytes) Adressen im Prozessabbild (Eingänge) Speicherbereich
- "digital_out_start" (verwendet 3 Bytes) und "pwm_out_start" (verwendet 6 Bytes) Adressen im Prozessabbild (Ausgänge) Speicherbereich

Die Input- und Outputadressen müssen in ihrem Adressbereich ohne Überschneidungen angeordnet sein.

Die Minimum und Maximum Werte lassen sich als Startadressen bei der IO Konfiguration verwenden:

Tabelle 7- 1 Prozessabbild Adressen

Prozessabbild	adress_space	min	max
Input	digital_in_start	0	509
Output	digital_out_start	0	509
Output	pwm_out_start	0	506
Input	analog_in_start	0	500

Verwenden Sie folgende Speicherbereiche im TIA-Portal zusammen mit den Adressen:

- I, IB, IW, ID : Prozessabbild-Speicherbereiche für das Lesen des Inputs
- Q, QB, QW, QD : Prozessabbild-Speicherbereiche für das Schreiben von Outputs
- PIB, PIW, PID : Direkter Zugriff für das Lesen des Peripherie-Inputs
- PQB, PQW, PQD: Direkter Zugriff für das Schreiben des Periepherie-Outputs

7.2.2 GPIO zuweisen

Pro Input/Output Pin ist jeweils nur 1 Bit zugeordnet.

Das IOT20x0 besitzt 20 Pins (0 bis 19). Insgesamt sind je 3 Bytes für den Process Input Image (I, PI) und den Process Output Image (Q, PQ) reserviert. Pin Adressen werden dem "digital_in_start" oder dem "digital_out_start" sequenziell vom "address_space"-Bereich in der Konfigurationsdatei io.conf zugewiesen.

Wählen Sie "digital_in_start" als gpioIN_adresse aus, d. h.:

I-Adresse	gpioIN_adress + 2					gpioIN_adress + 1								gpioIN_adress							
Bit	Reserviert	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Arduino-Pin		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Wählen Sie "digital_out_start" als gpioOut_adress aus, d. h.:

Q-Adresse	gpioOut_adress + 2					gpioOut_adress + 1								gpioOut_adress							
Bit	Reserviert	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Arduino-Pin		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Folgender Abschnitt zeigt Ihnen beispielhaft, wie Sie zwei Pins als GPIO in die Datei io.conf eintragen:

Programmcode	Kommentar
"gpio": [{	
"pin": 4,	// Arduino-Pin (0-19)
"is_output": 1,	// Output
"initial value": 0	// Logik '0' oder '1'
}, {	
"pin": 9,	// Arduino-Pin
"is_output": 0,	// Input
"initial value": 0	// Logik '0' oder '1'
}]	

Detaillierte Informationen zur Zuordnung der GPIOs finden Sie hier

(<https://support.industry.siemens.com/tf/WW/en/posts/how-is-the-assignment-of-the-gpios/155609?page=0&pageSize=10>).

7.2.3 PWM zuweisen

Jedem PWM Output ist jeweils 1 BYTE zugeordnet.

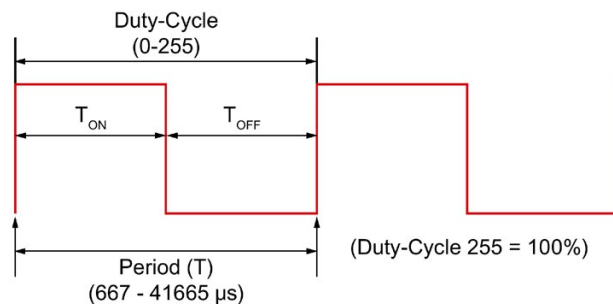
Das IOT20x0 besitzt 6 PWM Pins: 3, 5, 6, 9, 10 und 11. Insgesamt sind 6 Bytes für Process Output Image memory area (Q, PQ) reserviert, um die PWMs zu steuern. Die Adress-Bytes werden der "pwm_address" sequenziell zugewiesen.

Wählen Sie "pwm_out_start" als pwm_address aus, d. h.:

Q-Adresse	pwm_address +5	pwm_address +4	pwm_address +3	pwm_address +2	pwm_address +1	pwm_address
Arduino-Pin	11	10	9	6	5	3

Im PWM existieren zwei zu steuernde Parameter, "Period" und "Duty-Cycle".

"Period" ist das Zeitintervall, das den Zeitunterschied zwischen zwei aufeinander folgenden Wellen oder Signalen darstellt. Die "Period" tragen Sie in die Datei io.conf ein. Sein Wert liegt zwischen 667 µs und 41665 µs. Dies sind die von der Plattform unterstützten Periodengrenzen in Mikrosekunden (µs). Wenn Sie einen Wert außerhalb dieser Grenzen eintragen, wird der Wert auf den nächsten Maximum- oder Minimum-Wert geändert.



"Duty-cycle" ist ein Parameter, welcher den Wert der Zeit definiert, in der das Signal "an" bleibt. Dieser Wert ist angegeben in Prozent. Der Wert kann in STEP7 Professional eingegeben und geändert werden.

Wenn Sie ein Rechtecksignal generieren wollen, muss der Wert auf 50 % geändert werden, d. h. Sie müssen den Wert auf 127 ändern. In diesem Fall beträgt die Auflösung des duty-cycle 8 Bits (max: 255).

$$\text{Duty cycle} = T_{\text{on}} / (T_{\text{on}} + T_{\text{off}}) = x / 255$$

X ist der Wert, den Sie in STEP7 Professional eintragen.

L 127

T QB pwm_address

// Pin3 wird aktiviert, um ein Signal bei 50 % zu generieren

Folgender Abschnitt zeigt Ihnen beispielhaft, wie Sie zwei Pins als PWM in die Datei io.conf eintragen:

Programmcode	Kommentar
"pwm": [{	
"pin": 3,	// Arduino-Pin
"period us": 10000	// Period (Mikrosekunde)
}, {	
"pin": 5,	
"period us": 10000	
}],	

Hinweis

Beachten Sie das die Werte für "period_us" bei allen Pins gleich sein müssen.

7.2.4

Analog zuweisen

Jedem Analog Input sind 2 BYTE zugeordnet, aufgrund der 12-Bit-Auflösung.

Das IOT20x0 besitzt 6 AIO Pins: A0, A1, A2, A3, A4, A5. Insgesamt sind 12 Bytes für Process Input Image Memory Area (I, PI) reserviert, um die AIOs zu steuern.

Wählen Sie "analog_in_start" als aio_adress aus, d. h.:

I-Adresse	aio_adress +10	aio_adress +8	aio_adress +6	aio_adress +4	aio_adress +2	aio_adress
Adruino-Pin	A5	A4	A3	A2	A1	A0

Neben der PIN-Nummer gibt es einen weiteren Parameter, den Sie in die Datei io.conf eintragen: Die Auflösung (Resolution).

Geben Sie für die Auflösung einen Wert zwischen 1 und 12 Bit ein. Wenn Sie einen Wert außerhalb dieser Begrenzung eingeben, wird der Wert der Auflösung standardmäßig auf 10 Bit geändert.

Folgender Abschnitt zeigt Ihnen beispielhaft, wie Sie zwei Pins als AIO in die Datei io.conf eintragen:

Programmcode	Kommentar
"analog": [{	
"pin": 0,	// Arduino-Pin (A0)
"pin resolution":12	// Auflösung
}, {	
"pin": 1,	// A1
"pin resolution":10	
}],	

7.3 Fehlermeldungen

Die Datei io.conf wird in der IOT2000EDU während der Initialisierung der IOT2000EDU geparsed. Wenn einer der Parameter nicht ordnungsgemäß konfiguriert wurde und/oder ein Schreibfehler enthalten ist, kann die IOT2000EDU nicht starten und beendet mit einer Fehlermeldung.

Die folgenden Punkte werden in der Software geprüft. Wenn die Datei io.conf nicht valide ist, wird die entsprechende Meldung angezeigt:

1. Adressbereiche müssen kompatibel mit der Tabelle "Prozessabbild Adressen (Seite 34)" sein.
Meldung:
"Digital Input Start Address should be greater than 0 and less than 509"
"Digital Output Start Address should be greater than 0 and less than 509"
"PWM Start Address should be greater than 0 and less than 506"
"Analog Input Start Address should be greater than 0 and less than 500"
2. Adressen dürfen sich nicht überlappen. Zum Beispiel: Wenn "digital_in_start=100" ist, darf analog_in_start nicht "101" sein, es muss mindestens "103" sein.
Meldung:
"Invalid address space configuration."
"Please be sure that Start Address values don't overlap"
"GPIO allocates 3 Bytes, PWM allocates 6 Bytes and Analog Input allocates 12 Bytes"
3. GPIO Pinnummer muss zwischen 0 und 19 liegen.
Meldung:
"GPIO Pin number should be 0 – 19"
4. GPIO Is_output muss "0" oder "1" sein.
Meldung:
"is_output should be 1 or 0"
5. GPIO initial_value muss "0" oder "1" sein.
Meldung:
"initial_value should be 1 or 0"
6. PWM Pinnummern müssen 3, 5, 6, 9, 10, 11 sein.
Meldung:
"Only 3,5,6,9,10,11 pins can be used as PWM"
7. PWM Periode muss zwischen 667 und 41665 µs liegen. Sonst wird der Wert auf den nächsten Minimal- bzw. Maximalwert geändert. Es folgen einige Informationsmeldungen.
Meldung:
"PWM Period %d is more than maximum supported value. It will be set as %d"
"PWM Period %d is less than minimum supported value. It will be set as %d"
8. AIO Pins müssen zwischen 0, 1, 2, 3, 4 und 5 liegen.
Meldung:
"Analog Pin numbers should be between 0-5"
9. AIO Auflösung muss zwischen 1 und 12 liegen. Andernfalls wird die AIO Auflösung standardmäßig auf den Wert "10" geändert.
Meldung:
"Analog Resolution should be between 1-12; it will be set as 10"

10. Einige Pins sind multifunktional. Gehen Sie daher vorsichtig bei der Konfiguration der Pins vor. Zum Beispiel: Pin 3 lässt sich sowohl als PWM als auch als GPIO einstellen.

Meldung:

"PWM Pin Number: %d has already been configured. Please check multifunctional GPIO/PWM Pins. Invalid PWM Pin Configuration"

Alle Fehlermeldungen werden in die Datei "/var/log/messages" gespeichert. Sie können sich die Meldungen auf verschiedenen Wegen anzeigen lassen:

```
cat /var/log/messages
```

```
tail-f /var/log/messages (Zeigt das Ende der Datei an.)
```

Siehe auch

Address_space zuweisen (Seite 34)

IOT2000EDU bedienen

8.1 RUN/STOP/MRES Funktionen via TIA-Portal

Dieses Kapitel bringt Ihnen die Funktionalitäten RUN, STOP und MRES der SIMATIC IOT2000EDU via TIA-Portal näher.

Sobald Sie mit dem Gerät online verbunden sind, stehen Ihnen folgende Funktionen zur Verfügung:

- RUN, STOP und MRES: Über das CPU-Bedienpanel im Arbeitsbereich "Online & Diagnose".
- RUN und STOP: Über die Schaltflächen "CPU starten" und "CPU stoppen" in der Funktionsleiste.

"RUN" und "STOP"

1. Klicken Sie in der Funktionsleiste auf die Schaltfläche "CPU starten", wenn Sie die IOT2000EDU in den Betriebszustand RUN setzen wollen, bzw. "CPU stoppen", wenn Sie die IOT2000EDU in den Betriebszustand STOP setzen wollen.



2. Beantworten Sie die Sicherheitsabfrage mit "OK".

Oder:

1. Aktivieren Sie die Task Card "Online Tools" der IOT2000EDU.
2. Klicken Sie in der Palette "CPU-Bedienpanel" auf die Schaltfläche "RUN", wenn Sie die IOT2000EDU in den Betriebszustand RUN setzen wollen, bzw. "STOP", wenn Sie die IOT2000EDU in den Betriebszustand STOP setzen wollen.
3. Beantworten Sie die Sicherheitsabfrage mit "OK".

Hinweis

Führen Sie die RUN-Funktion nur aus, wenn sich die IOT2000EDU im "STOP"-Modus befindet. Umgekehrt führen Sie die STOP-Funktion nur aus, wenn sich das Gerät im "RUN"-Modus befindet. Sonst erscheint eine Fehlermeldung.

MRES

Die Funktion MRES ermöglicht Ihnen das Urlöschen der CPU. Dabei wird der Speicher des Geräts gelöscht und das Gerät in den so genannten "Anfangszustand" versetzt.

Das bedeutet:

- Alle Benutzerdaten werden gelöscht
- Abhängig vom Zielsystem können Benutzerprogramme und Hardware-Konfigurationen gelöscht werden
- Alle Verbindungen zum Modul werden getrennt

Um das Urlöschen durchzuführen, gehen Sie wie folgt vor:

1. Aktivieren Sie die Task Card "Online Tools" der IOT2000EDU.
2. Klicken Sie in der Palette "CPU-Bedienpanel" auf die Schaltfläche "MRES".
3. Beantworten Sie die Sicherheitsabfrage mit "OK".

8.2 RUN/STOP/MRES Funktionen via Kommandozeile

Im folgenden Kapitel erfahren Sie, wie Sie mithilfe des Kommandozeilenprogramms "CPU_Control" den Betriebszustand der IOT2000EDU ändern. Das Programm wurde zusammen mit der Applikation "IOT2000EDU" im Verzeichnis "/usr/local/bin" installiert.

Zurzeit kann "CPU_Control" 5 Kommandos ausführen: RUN, STOP, MRES, Shutdown und Status

Betriebszustand RUN

Mit folgendem Kommando versetzen Sie die IOT2000EDU in den Betriebszustand RUN:

```
CPU_Control run
```

Betriebszustand STOP

Mit folgendem Kommando versetzen Sie die IOT2000EDU in den Betriebszustand STOP:

```
CPU_Control stop
```

MRES

Beachten Sie, dass Sie vor der Ausführung dieses Kommandos die IOT2000EDU in den Betriebszustand STOP versetzen.

Mit folgendem Kommando lösen Sie das Urlöschen des IOT2000EDU Controllers aus:

```
CPU_Control mres
```

Shutdown

Mit folgendem Kommando beenden Sie den IOT2000EDU Controller:

```
CPU_Control shutdown
```

Status

Mit folgendem Kommando überprüfen Sie den momentanen Betriebszustand des IOT2000EDU Controllers:

```
CPU_Control status
```

So können Sie auch überprüfen, ob ein vorhergehendes Kommando erfolgreich war oder fehlgeschlagen ist.

Hilfe

Wenn Sie das Kommando CPU_Control ohne einen der 5 Parameter ausführen, wird Ihnen der folgende Hilfebildschirm angezeigt:

```
Usage :
```

```
Switch to RUN state : CPU_Control run
```

```
Switch to STOP state : CPU_Control stop
```

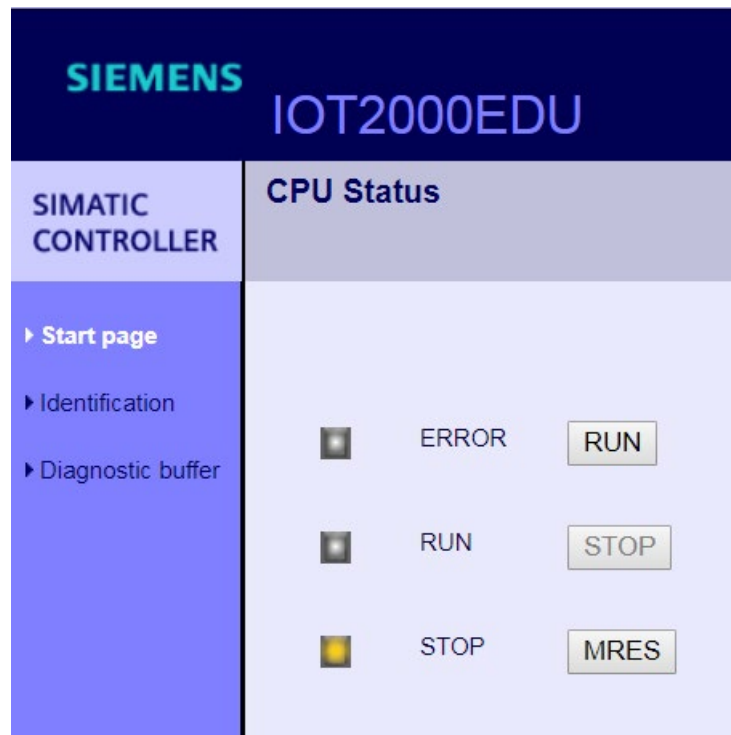
```
Trigger MASTER RESET : CPU_Control mres
```

```
Shutdown the Controller : CPU_Control shutdown
```

```
Get Status : CPU_Control status
```

```
Show version info : CPU_Control -v
```

8.3 RUN/STOP/MRES via Webserver



Auf dieser Seite des Webserver können Sie den Betriebszustand der IOT2000EDU mithilfe der Schaltflächen ändern, sowie den Speicher der IOT2000EDU urlöschen. Weitere Informationen dazu finden Sie im Kapitel "Startseite (Seite 46)".

Webserver

9.1 Einleitung

Das folgende Kapitel beschreibt die IOT2000EDU Webserver Aktivierung, Inhalte von Webseiten und die Interaktion mit der IOT2000EDU.

Der IOT2000EDU Webserver wird automatisch bei der Installation des IOT2000EDU Installationspakets "IOT2000EDU_1.1.0_i586.ipk" mitgeliefert.

9.2 Voraussetzungen

Folgende Voraussetzungen sind für eine erfolgreiche Installation der IOT2000EDU V1.1 erforderlich:

- "init system" muss im Betriebssystem vorhanden sein
Ob "init system" vorhanden ist, sehen Sie daran, ob "/etc/init.d" im Betriebssystem vorhanden ist.
- "node v6.9.2" Paket muss im Betriebssystem vorhanden sein

9.3 Webserver mit Linux Kommandos bedienen

Die IOT2000EDU Installation fügt die Webserver Applikation automatisch dem Autostart Service hinzu und aktiviert den Prozess. Bei Systemüberlastungen oder möglichen Abstürzen können Sie den IOT2000EDU Webserver mit dem folgenden Kommando manuell starten: `/etc/init.d/IOT2000Web start`

Während des Boot-Vorgangs der IOT20x0 wird der IOT2000EDU Webserver automatisch aktiviert. Ihnen stehen weiterhin folgende Kommandos zur Verfügung:

- Webserver stoppen: `/etc/init.d/IOT2000Web stop`
- Webserver Status anzeigen: `/etc/init.d/IOT2000Web status`

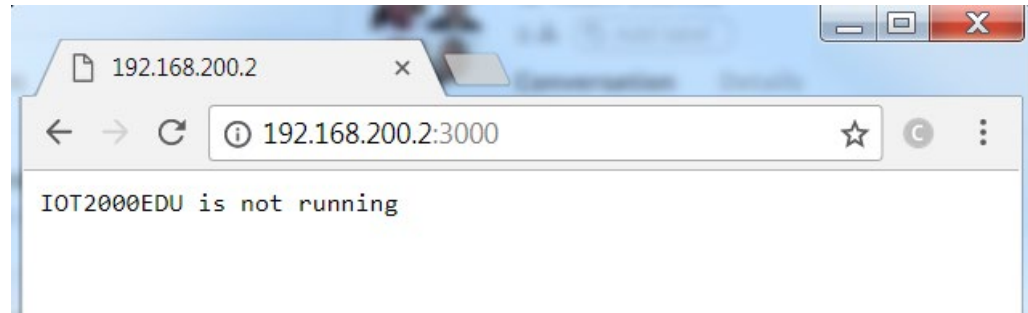
9.4 Auf Webserver zugreifen

Um auf den Webserver zuzugreifen, nutzen Sie die IOT20x0 IP Adresse über den Port "3000" als `ipaddress:3000`

Beispiel:

192.168.200.1:3000

Wenn die IOT2000EDU nicht auf einem IOT20x0 Gerät läuft, verweist der Server auf die folgende Fehlerseite:



9.5 Webserver Seiten

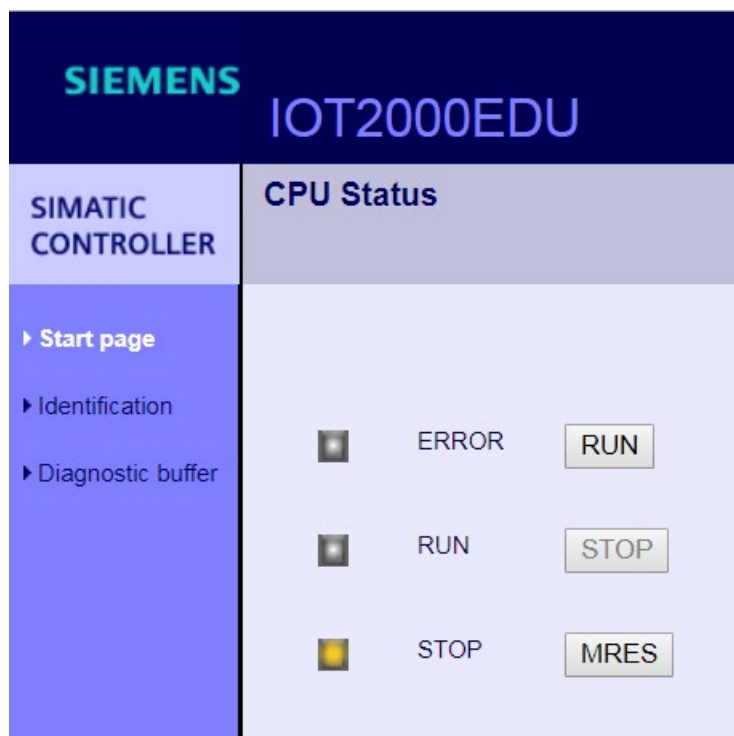
9.5.1 Einleitung

Die folgenden 3 Webseiten sind der Hauptbestandteil des IOT2000EDU Webserver:

- Startseite
- Identifikation
- Diagnosepuffer




Webseiten werden nicht automatisch aktualisiert. Daher müssen Sie die Seiten manuell laden, wenn Sie die Webseite aktualisieren wollen.

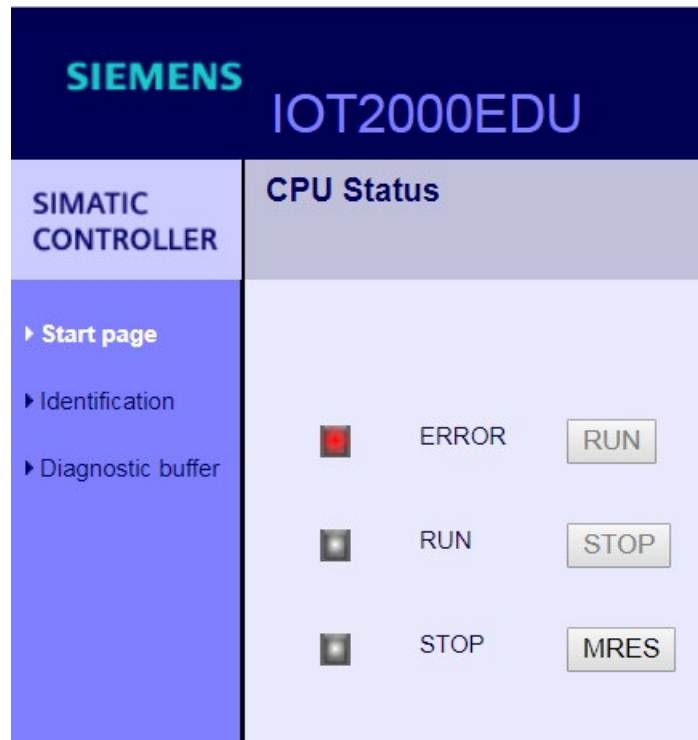
9.5.2 Startseite



Die Startseite ist die voreingestellte Webseite des IOT2000EDU Webservers. Auf dieser Seite sehen Sie den Status der IOT2000EDU anhand der farbigen LEDs. Jede LED hat bei Aktivität ihre eigene Farbe. Jedoch kann immer nur eine LED gleichzeitig leuchten. Sie können den Betriebszustand der IOT2000EDU mithilfe der Schaltflächen "RUN" und "STOP" ändern, sowie den Speicher der IOT2000EDU urlöschen mithilfe der "MRES" Schaltfläche.

Die LEDs können folgende Zustände der IOT2000EDU anzeigen:

"RUN" 	Die Schaltfläche "RUN" ist ausgegraut und die "RUN" LED leuchtet grün.
"STOP" 	Die Schaltfläche "STOP" ist ausgegraut und die "STOP" LED leuchtet gelb.
"ERROR" 	Die Schaltflächen "RUN" und "STOP" sind ausgegraut und die "ERROR" LED leuchtet rot.



9.5.3 Identifikation

SIEMENS IOT2000EDU	
SIMATIC CONTROLLER	Identification
<ul style="list-style-type: none"> ▶ Start page ▶ Identification ▶ Diagnostic buffer 	<p>Identification:</p> <p>Device Name: IOTPLC_1</p> <p>Module Name: IOTPLC_1</p> <p>Plant designation: plant</p> <p>Location identifier: location</p> <p>Order number:</p> <p>MLFB: 6ES7 671-0LE00-0YB0</p> <p>Firmware: V1.1.0</p>

Diese Seite enthält Identifikations-, Wartungs- und grundlegende Projektinformationen, welche vom TIA Portal auf die IOT2000EDU geladen wurden.

9.5.4 Diagnosepuffer

<div> <div>SIEMENS</div> <div>IOT2000EDU</div> </div>				
SIMATIC CONTROLLER	Diagnostic buffer			
	Number	Time	Date	Event
► Start page	1	00:01:38:52	18/03/18	Mode transition from STARTUP to RUN
	2	00:01:38:50	18/03/18	Request for manual warm restart
► Identification	3	00:01:38:49	18/03/18	Mode transition from STOP to STARTUP
	4	00:01:38:49	18/03/18	New startup information in STOP mode
► Diagnostic buffer	5	00:01:36:06	18/03/18	New startup information in STOP mode
	6	00:01:35:88	18/03/18	New startup information in STOP mode
	7	00:01:35:65	18/03/18	New startup information in STOP mode
	8	00:01:35:65	18/03/18	STOP caused by PG stop operation or by SFB 20 "STOP"
	9	00:01:08:17	18/03/18	Mode transition from STARTUP to RUN
	10	00:01:08:14	18/03/18	Request for manual warm restart
	11	00:01:08:12	18/03/18	Mode transition from STOP to STARTUP
	12	00:01:08:12	18/03/18	New startup information in STOP mode
	13	00:01:08:09	18/03/18	Memory reset executed
	14	00:01:08:09	18/03/18	Memory reset started automatically (power on not backed up)
Mode transition from STARTUP to RUN Startup information: - Time for time stamp at the last non backed up power on - Single processor operation Current/last startup type: - Warm restart triggered via MPI; last power on not backed up Permissibility of certain startup types: - Manual warm restart permitted - Automatic warm restart permitted Last valid operation or setting of automatic startup type at power on: - Warm restart triggered via MPI; last power on not backed up Previous operating mode: STARTUP (warm restart) Requested operating mode: RUN Incoming Event				

Diese Seite enthält alle Diagnose Logs, welche durch ein Ereignis in der IOT2000EDU erstellt wurden.

Im oberen Teil der Seite finden Sie die Ereignisse sowie dazugehörige allgemeine Informationen, wie z. B. Ereignis ID, Zeit und Datum.

Wenn Sie einen Eintrag aus der Liste auswählen, werden im unteren Teil der Seite die Detailinformationen angezeigt.

Die CPU Funktionsbibliothek ist eine Binärdatei, die ausführbare Funktionen enthält. Die CPU Funktionsbibliothek kann von der IOT2000EDU aufgerufen werden.

Dadurch haben Sie die Möglichkeit, Ihre eigene Software in einer höheren Programmiersprache (C/C++) zu implementieren und mit der IOT2000EDU zusammenarbeiten zu lassen.

10.1 Überblick über die CPU Funktionsbibliothek

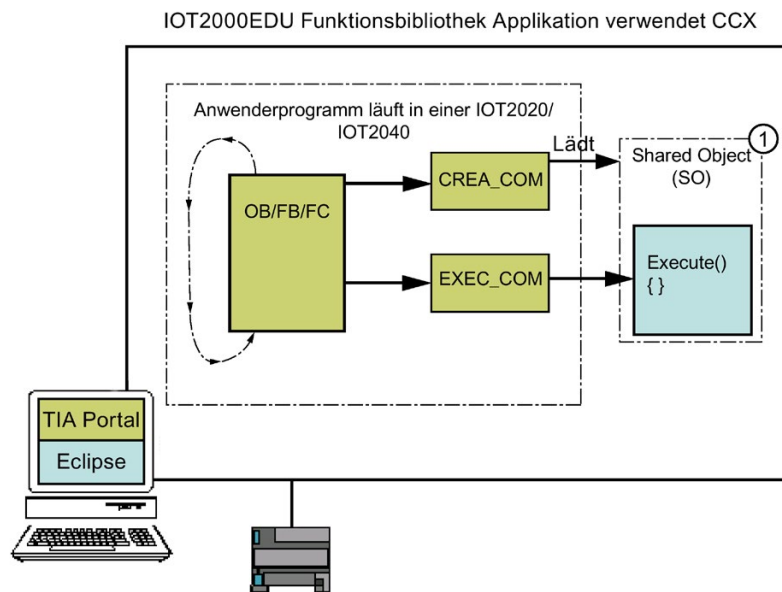
Mit einer Entwicklungsumgebung wie Eclipse können Sie eine CPU Funktionsbibliothek als "Shared Object (SO)" erstellen. Das SO enthält Ihre in der Programmiersprache C/C++ geschriebene Software. Sie programmieren Ihr Anwenderprogramm im TIA Portal entsprechend so, dass das TIA Portal die Anweisungen "CREA_COM" (SFB65001) und "EXEC_COM" (SFB65002) aufruft. Dadurch wird das SO dynamisch geladen und Ihr Code direkt vom IOT2000EDU Anwenderprogramm ausgeführt.

Um sich mit Ihrer Software zu verbinden, rufen Sie die Anweisungen aus dem IOT2000EDU Programm auf. Mithilfe der Anweisung "CREA_COM" lädt die IOT2000EDU ein SO. Die Anweisung "EXEC_COM" führt das geladene SO aus. Das TIA Portal enthält diese beiden Anweisungen.

Indem Sie die CPU Funktionsbibliotheken zur Integration Ihrer Software in ein Anwenderprogramm verwenden, können Sie das Leistungsvermögen eines IOT2000EDU Anwenderprogramms erweitern.

Der Einsatz einer CPU Funktionsbibliothek bietet Ihnen folgende Vorteile:

- Aufnehmen spezieller in C oder C++ geschriebener Kontroll-Logik
- Verwenden von komplexen oder proprietären Algorithmen oder Regelfunktionen, z. B. PID oder Gasfluss. Diese Algorithmen können mithilfe üblicher Entwicklungsumgebungen (IDE) wie z. B. Eclipse in C/ C++ geschrieben und gewartet werden, oder aus modellbasierten Entwicklungsumgebungen wie Matlab/ Simulink integriert werden.
- Verbinden mit anderen Applikationen oder Geräten, wie zum Beispiel Motion Control oder Arduino Shields von Drittherstellern. Viele Hersteller verteilen ihre Arduino Treiber in C/C++.
- Zugriff auf Funktionen des IOT20x0 Boards, auf die nicht mit normalen Programmiersprachen zugegriffen werden kann.



① Verwenden Sie CCX um Ihren C/C++ Code in ein SO einzubetten.

Die CPU Funktionsbibliothek ermöglicht es Ihnen, Ihre eigenen Thread Funktionen zu erstellen. Dadurch können Sie Ihren eigenen Code außerhalb des Main Thread der Ausführung ausführen.

Programmieren mehrerer Funktionsbibliotheken

Die IOT2000EDU kann mehr als ein SO laden. Jedes SO kann mehrere Aufgaben ausführen. Wenn die Anweisung "CREA_COM" ein SO lädt, gibt dieses einen eindeutigen Object Handle (OBJHandle) aus. Ein Aufruf der Anweisung "EXEC_COM" nutzt den OBJHandle eines spezifischen Objekts, um die Software für das gleiche SO auszuführen.

10.2 Benutzerdefinierte SO-Datei erstellen

Die Funktionen einer Funktionsbibliothek werden über die Datei „EDU_Function.cpp“ umgesetzt. Die Datei „EDU_Function.cpp“ finden Sie im mitgelieferten Beispielprojekt. Die wichtigste Funktion innerhalb der Datei ist die „Execute“ Funktion, in der Sie die eigentlichen Funktionen Ihrer Funktionsbibliothek einbetten.

10.2.1 Funktion „Execute“ programmieren

Die Funktion „Execute“ ist die Funktion, die aus dem Anwenderprogramm mit Hilfe der Anweisung „EXEC_COM“ gestartet wird. Beim Aufruf werden folgende Parameter übergeben:

- Zeiger auf den Puffer der Eingangsdaten
- Zeiger auf den Puffer der Ausgangsdaten
- Länge der Puffer
- Command-Variable

```
extern "C" SEA_EXT CALL EDU RESULT Execute (unsigned long command, long nInBytes, byte bInData [],
long nOutBytes, long * pnUsedOutBytes, byte bOutData [])
{
    CWinLCReadData Input(nInBytes, bInData);
    CWinLCReadWriteData Output(nOutBytes, bOutData);
    EDU RESULT retVal = EDU SUCCESS;

    switch(command)
    {
        case 0:
            // add your functions here
            retVal = Standard Deviation(Input, Output);
            break;
        case 1:
            // add your functions here
            retVal = Function 1(Input, Output);
            break;
        default:
            retVal = EDU COMMAND NOT IMPLEMENTED;
    }
    *pnUsedOutBytes = Output.EDU LastByteChanged() + 1;

    return retVal;
} //
Execute
```

Das Interface der Funktion „Execute“ darf nicht geändert werden. Die Parameter sind wie folgt definiert:

- **Unsigned long command:**
Der Parameter „command“ ermöglicht es, verschiedene Funktionen innerhalb einer Funktionsbibliothek zu implementieren. Diese Funktionen werden über eine „Switch-Case“ Anweisung identifiziert und aufgerufen. Damit lassen sich nahezu beliebig viele Funktionen innerhalb einer Funktionsbibliothek implementieren.
- **Long nInBytes:**
Dieser Parameter enthält die Länge des Inputdatenpuffer in Byte.
- **byte bInData []:**
Der Parameter ist ein Zeiger auf den Inputdatenpuffer. Die einzelnen Variablen liegen hintereinander, wie sie vom Anwenderprogramm der IOT2000EDU abgelegt wurden. Sie einzelnen Eingangswerte können mit Hilfe der „Data Access Helper“ Klassen ausgelesen werden. Details dazu siehe Kapitel "Data Access Helper Klassen (Seite 55)".
- **long nOutBytes:**
Dieser Parameter enthält die Länge des Outdatenpuffer in Byte.

- `long * pnUsedOutBytes`:
Dieser Parameter enthält die Länge der tatsächlich genutzten Länge des Outdatenpuffer.
- `byte bOutData []`:
Dieser Parameter ist ein Zeiger auf den Ausgangspuffer. Die einzelnen Variablen liegen hintereinander, wie sie vom Anwenderprogramm der IOT2000EDU verwendet werden sollen. Einzelne Eingangswerte können mit Hilfe der „Data Access Helper“ Klassen ausgelesen bzw. geschrieben werden. Details dazu siehe Kapitel "Data Access Helper Klassen (Seite 55)".
- **Return Value:**
`EDU_Result` gibt den Statuswert bzw. Fehlermeldungen der aufgerufenen Funktion zurück. Die Standard-Rückgabewerte (z. B. `EDU_SUCCESS`) sind in der Datei „EDUso.h“ definiert.

10.2.2 Weitere Funktionen programmieren

Neben der „Execute“ Funktion gibt es weitere Funktionen, die in bestimmten Situationen automatisch aufgerufen werden. Diese Funktionen müssen im Code enthalten sein, können aber ansonsten leer gelassen werden.

Die Schnittstellenfunktionen der CPU Funktionsbibliothek sind initial leer. Das erlaubt Ihnen entsprechend den Anforderungen Ihrer Applikation zu programmieren.

Die IOT2000EDU Runtime ruft diese Schnittstellenfunktionen automatisch auf, wenn:

- das IOT2000EDU Benutzerprogramm das SO lädt.
- das IOT2000EDU Benutzerprogramm das SO gleichzeitig ausführt.
- die IOT2000EDU von STOP zu STARTUP wechselt, von HALT zu RUN, oder wenn das SO das erste Mal geladen wird.
- die IOT2000EDU in den STOP oder HALT Modus wechselt.

Die Schnittstellenfunktionen:

ODKCreate:

Die Funktion "ODKCreate" initialisiert Daten oder erstellt Objekte, nachdem die Anweisung "CREA_COM" das SO geladen hat. Wenn das Shared Object initial geladen wird und Sie Auswertungen durchführen wollen, laden Sie es in "ODKCreate". Andernfalls lassen Sie "ODKCreate" leer.

Activate:

Die Funktion "Activate" wird aufgerufen, bevor die IOT2000EDU von STOP oder HALT Modus nach STARTUP oder RUN Modus wechselt.

"Activate" wird immer aufgerufen, nachdem das SO geladen wird und bevor der erste Aufruf der "Execute" Funktion beginnt. Wenn Sie Auswertungen durchführen wollen, bevor der erste Aufruf der "Execute" Funktion, können Sie es in die "Activate" Funktion laden. Andernfalls lassen Sie "Activate" leer.

DeActivate:

Die Funktion "DeActivate" wird aufgerufen, nachdem die IOT2000EDU von STARTUP oder RUN Modus in den STOP oder HALT Modus wechselt. Wenn Sie Auswertungen durchführen wollen, nach dem Wechsel von STOP oder HALT, können Sie es in die "DeActivate" Funktion laden. Andernfalls lassen Sie die Funktion leer.

ODKRelease:

Die Funktion "ODKRelease" ist verantwortlich für das Veröffentlichen/Entladen des Shared Objects. Wenn Sie Auswertungen durchführen wollen, wenn das Object veröffentlicht wird, laden Sie es in ODKRelease. Andernfalls lassen Sie die Funktion leer.

Die IOT2000EDU veröffentlicht das SO bei einem Memory Reset (MRES) oder bei einem Controller Shutdown. Weil sowohl der MRES als auch der Controller Shutdown den Controller in den STOP Modus versetzen, ruft die IOT2000EDU immer "DeActivate" auf, bevor sie "ODKRelease" aufruft.

Scan cycle impact:

Ihre Software in der "Execute" Funktion und die Sub-Funktionen sind Teil des Main Program Scan Zyklus, wenn sie von der Anweisung "EXEC_COM" aufgerufen werden. Das IOT2000EDU Programm führt die Anweisung aus, welche Ihre Software als eine einfache Instruction aufruft. Dieser Instruction Aufruf ist nicht unterbrechbar. Während der Ausführung Ihrer Software sind der Watchdog Timer, die OBs und die Prozessalarme nicht verfügbar. Sie werden bedient, nachdem der Aufruf der Anweisung Ihrer Software abgeschlossen ist.

Hinweis

Anwenderspezifische Software, die die Zykluszeiten deutlich verlängert, verzögert die Auswertung kritischer Aktivitäten im Controller.

Um Verzögerungen im OB Scanzzyklus zu vermeiden, schreiben Sie keine Auswertungen in die "Execute" Funktion oder Sub-Funktionen, welche den Scanzzyklus über eine akzeptable Zykluszeit hinaus verlängern. Das schließt auch Aufrufe einer nichtdeterministischen Funktion wie "Printf" ein.

Hinweis

Die CPU Funktionsbibliothek garantiert, dass die IOT2000EDU diese Funktionen von einem einzelnen Ausführungsstrang aufruft.

10.2.3 Data Access Helper Klassen

Die Data Access Helper Klassen erlauben den Zugriff auf die von der IOT2000EDU übergebenen Variablen in den Eingangs- und Ausgangsparametern.

Die beiden Data Access Helper Klassen sind:

- CWinLCReadData: Lesen der Variablen aus den Eingangsparametern
- CWinLCReadWriteData: Lesen und Schreiben der Variablen in den Ausgangsparametern

Die Steuerung IOT2000EDU übergibt die Eingangsparameter im S7-Format und erwartet die Rückgabe der Ausgangsparameter ebenfalls im S7-Format. Dabei werden alle Eingangs- und Ausgangsvariablen ab den übergebenen Zeigern hintereinander abgelegt. Dabei ist folgendes zu beachten:

Um die Daten im C/C++ Code verwenden zu können, müssen sie mit Hilfe der Zugriffsfunktionen in den Data Access Helper Klassen konvertiert werden. Genau so müssen Daten vor Rückgabe an die IOT2000EDU in das S7-Format gewandelt werden.

Die Methoden EDU_ReadS7<data type> und EDU_WriteS7<data type> führen diese Typkonvertierungen automatisch durch und erlauben so einen einfachen und sicheren Zugriff auf die Eingangs- und Ausgangsvariablen.

Im Beispielprojekt wird die Verwendung der Helper Klassen an einem einfachen Beispiel gezeigt:

```
EDU RESULT Function 1 ( CWinLCReadData& rInput ,CWinLCReadWriteData& rOutput )
{
    short   MyInteger;
    float   MyReal;
    bool    MyBool;
    // extract input variables from input data buffer
    rInput.EDU ReadS7INT(0,MyInteger);
    rInput.EDU ReadS7REAL(2,MyReal);
    rInput.EDU ReadS7BOOL(6,0,MyBool);
    // Add your Code here
    /* printf("\n----- InputData ----- \nMyInteger: %d \n MyReal: %.5f
    \n MyBool: %s \n ",MyInteger,MyReal,MyBool ? "TRUE" : "FALSE");
    MyInteger +=5;
    MyReal += (float)3.0000;
    MyBool = false;
    /* printf("\n----- OutputData ----- \nMyInteger: %d \n MyReal: %.5f
    \n MyBool: %s \n ",MyInteger,MyReal,MyBool ? "TRUE" : "FALSE");

    // put output variables into output data buffer
    rOutput.EDU Writes7INT(0,MyInteger);
    rOutput.EDU Writes7REAL(2,MyReal);
    rOutput.EDU Writes7BOOL(6,0,MyBool);

    return EDU SUCCESS;
}
```

Im Beispiel-Projekt wird als Eingangsparameter eine UDT mit den folgenden Elementen verwendet:

```
UDT:
Integer      MyInteger
Real         MyReal
BOOL         MyBool
END UDT
```

Damit sind die einzelnen Variablen bzw. Mitglieder der UDT im Eingangsparameter wie folgt abgelegt:

Offset	0		2				6
	MyInteger		MyReal				MyBool

Unterstützte S7-Datentypen

Die Helper Klassen unterstützen die folgenden S7-Datentypen:

Tabelle 10- 1 CWinLCReadData

EDU_GetBuffer	Gibt einen Pointer zum Datenbereich zurück.
EDU_SetBuffer	Initialisiert den "Input Data"-Bereich und Dateigröße.
EDU_GetBufferSize	Gibt die Größe des Datenbereich zurück (in Bytes).
EDU_ReadS7BYTE	Liest ein Byte (1 Byte) aus dem Datenbereich.
EDU_ReadS7WORD	Liest ein Word (2 Byte) aus dem Datenbereich.
EDU_ReadS7DWORD	Liest ein Double Word (4 Byte) aus dem Datenbereich.
EDU_ReadS7S5TIME	Liest ein 16-Bit-Zeitwert (2 Byte) aus.

EDU_ReadS7DATE	Liest ein Datumswert (2 Byte) aus dem Datenbereich.
EDU_ReadS7TIME_OF_DAY	Liest die Uhrzeit (4 Byte) aus dem Datenbereich.
EDU_ReadS7INT	Liest ein Integer (2 Byte) aus dem Datenbereich.
EDU_ReadS7DINT	Liest ein Double Integer (4 Byte) aus dem Datenbereich.
EDU_ReadS7REAL	Liest eine Gleitpunktzahl (4 Byte) aus dem Datenbereich.
EDU_ReadS7TIME	Liest einen Zeitwert (4 Byte) aus dem Datenbereich.
EDU_ReadS7CHAR	Liest einen Character (1 Byte) aus dem Datenbereich.
EDU_ReadS7BOOL	Liest einen boolschen Wert (1 Bit) aus dem Datenbereich.
EDU_ReadS7STRING_LEN	Liest die String-Längeninformation eines S7 Strings im Datenbereich.
EDU_ReadS7STRING	Liest einen S7 String aus dem Datenbereich und gibt ihn als C++ Character String zurück.
EDU_ReadS7DATE_AND_TIME	Liest einen generischen Datums- und Zeitbereich.

Tabelle 10- 2 CWinLCReadWriteData

EDU_GetBuffer	Gibt einen Pointer zum Datenbereich zurück.
EDU_SetBuffer	Initialisiert den "Input Data"-Bereich und Dateigröße.
EDU_GetBufferSize	Gibt die Größe des Datenbereich zurück (in Bytes).
EDU_WriteS7BYTE	Schreibt ein Byte (1 Byte) in den Datenbereich.
EDU_WriteS7WORD	Schreibt ein Word (2 Byte) in den Datenbereich.
EDU_WriteS7DWORD	Schreibt ein Double Word (4 Byte) in den Datenbereich.
EDU_WriteS7INT	Schreibt ein 16-Bit-Zeitwert (2 Byte).
EDU_WriteS7DINT	Schreibt ein Datumswert (2 Byte) in den Datenbereich.
EDU_WriteS7S5TIME	Schreibt die Uhrzeit (4 Byte) in den Datenbereich.
EDU_WriteS7TIME	Schreibt ein Integer (2 Byte) in den Datenbereich.
EDU_WriteS7DATE	Schreibt ein Double Integer (4 Byte) in den Datenbereich.
EDU_WriteS7TIME_OF_DAY	Schreibt eine Gleitpunktzahl (4 Byte) in den Datenbereich.
EDU_WriteS7CHAR	Schreibt einen Zeitwert (4 Byte) in den Datenbereich.
EDU_WriteS7REAL	Schreibt einen Character (1 Byte) in den Datenbereich.
EDU_WriteS7BOOL	Schreibt einen boolschen Wert (1 Bit) in den Datenbereich.
EDU_WriteS7STRING	Schreibt einen S7 String in den Datenbereich und gibt ihn als C++ Character String zurück.
EDU_WriteS7DATE_AND_TIME	Schreibt einen generischen Datums- und Zeitbereich.

Im Kapitel "Datentypen (Seite 59)" erhalten Sie Informationen zu den relevanten Datentypen.

Weitere Informationen zu den unterstützten Datentypen der IOT2000EDU finde Sie in der Online Hilfe des TIA Portals. In der IOT2000EDU kann der gleiche Umfang an Datentypen wie bei einer S7-300 Steuerung verwendet werden.

Hinweis

Die Methoden der Data Access Helper Klassen dürfen nur innerhalb der Funktion „Execute“ angewendet werden. Dadurch helfen Ihnen die Helper Klassen Programmierfehler zu vermeiden, z. B. Out-Of-Range Fehler oder das Schreiben zu ungültigen Zeigern.

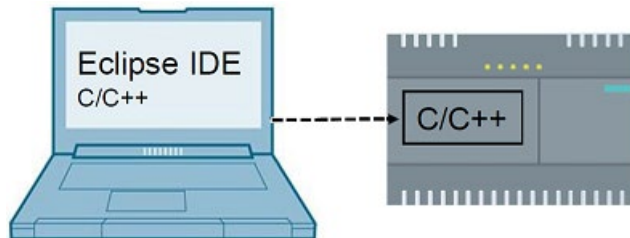
10.2.4 Datentypen

In der folgenden Tabelle sind alle relevanten Datentypen beispielhaft aufgelistet:

C/C++ (Simulink Coder)	C/C++ (IOT2000EDU Funktionsbibliothek)	TIA Portal	Bytes	Helper Funktionen EDU_ReadS7<data type> EDU_WriteS7<data type>
boolean_T	bool	BOOL	1 bit	EDU_ReadS7BOOL EDU_WriteS7BOOL
int8_T, uint8_T, char_T, uchar_T, byte_T	char	CHAR	1	EDU_ReadS7CHAR EDU_WriteS7CHAR
int16_T, uint16_T	short	INT	2	EDU_ReadS7INT EDU_WriteS7INT
int32_T, uint32_T, int_T, uint_T	long	DINT	4	EDU_ReadS7DINT EDU_WriteS7DINT
ulong_T	long	DINT	4	EDU_ReadS7DINT EDU_WriteS7DINT
real_T	float	REAL (4 Byte) Hinweis: Datenverlust	4	EDU_ReadS7REAL EDU_WriteS7REAL
real32_T	float	REAL	4	EDU_ReadS7REAL EDU_WriteS7REAL
"Typ"_T["Länge"]	"Typ" ["Länge"]	Array [lo .. hi] of type	abhängig von Größe und Typ	
Struct (Simulink: BUS-Element)	struct	Struct	abhängig von Größe und Typ	

10.3 Funktionsbibliothek-Programme mit Eclipse erstellen

Dieses Kapitel beschreibt die Funktionalität und Implementation eines CPU Funktionsbibliothek Beispielprogramms, das in Eclipse IDE erstellt wird. Weiterhin lernen Sie, wie Sie das Programm kompilieren und auf die IOT20x0 Geräte schreiben. Zusätzlich erfahren Sie, wie Sie ein einfach TIA Portal Projekt anlegen und wie Sie ein Shared Object (SO) im PLC Anwenderprogramm verwenden können. Das nachfolgende Bild bietet einen Überblick über die CPU Funktionsbibliothek Applikationen:



Beispiel CPU Funktionsbibliothek Projekt in Eclipse

Um mit der Erstellung einer Funktionsbibliothek (SO) anzufangen, können Sie das Eclipse-Beispielprojekt verwenden. Dieses Beispielprojekt enthält alle Schnittstellenfunktionen mit einer einfachen kundenspezifischen Applikation. Diese Applikation berechnet die "Standard Deviation" einer Nummerngruppe.

Zusätzlich zum IOT2000EDU Installationspaket wird das Beispielprojekt als separate Datei ausgeliefert.

Sie können außerdem neben Eclipse auch ein eigenes Projekt in einer anderen IDE erstellen. Beachten Sie jedoch die Systemvoraussetzungen.

Sie können mit Windows oder einem Linux Betriebssystem arbeiten. Wenn Sie mit Windows arbeiten, benötigen Sie einen Cross-Compiler.

Wenn die IOT2000EDU auf den IOT20x0 Geräten installiert wurde, sind bereits folgende Pakete vorinstalliert:

- libc6
- libstdc++6

Die IOT2000EDU benötigt die libc6 und libstdc++6 Bibliotheken. Daher muss sowohl der native Compiler, als auch der Cross-Compiler, diese Bibliotheken unterstützen, um die SO-Datei zu generieren. Ein paar alternative Entwicklungsumgebungen werden Ihnen für die Generierung des SO empfohlen.

10.3.1 Alternative Entwicklungsumgebungen

Windows: Eclipse via ODK 1500S installieren

Wenn ODK 1500S bereits im System installiert ist, können Sie auf die Eclipse Version (Kepler) unter folgendem Pfad zugreifen:

"C:\Program Files (x86)\Siemens\Automation\ODK1500S\V2.5\eclipse\Eclipse.exe"

Importieren Sie das Beispielprogramm mit dem Build "Release" oder erstellen Sie ein eigenes Projekt mit Hilfe von Kepler Eclipse. Um ein SO für die IOT2000EDU zu generieren, installieren Sie das IOT2000 SDK für das Cross-Compilen für Windows Betriebssysteme. Folgen Sie der unten stehenden IOT2000 SDK Anleitung für die Cross-Compiler Einstellungen.

Windows: Eclipse herunterladen und installieren

Sie benötigen "Eclipse IDE for C/C++ Developers", um ein neues C/C++ Projekt anzulegen bzw. um das Beispielpjekt "EDU_StandardDeviation", zu öffnen. Sie können es unter folgendem Link downloaden: <https://www.eclipse.org/downloads/eclipse-packages/> (<https://www.eclipse.org/downloads/eclipse-packages/>)

Um ein SO für die IOT2000EDU zu generieren, installieren Sie das IOT2000 SDK für das Cross-Compilen für Windows Betriebssysteme. Folgen Sie der IOT2000 SDK Anleitung (Seite 62) für die Cross-Compiler Einstellungen.

Linux: Eclipse installieren

Wenn Sie ein Linux Betriebssystem verwenden, laden Sie eine Eclipse Version herunter. Unter Linux steht Ihnen ein nativer Compiler zur Verfügung, der kompatibel mit den Bibliotheken libc6 and libstdc++6 ist.

Zusätzlich ist es möglich, einen Cross-Compiler für Yocto zu erstellen.

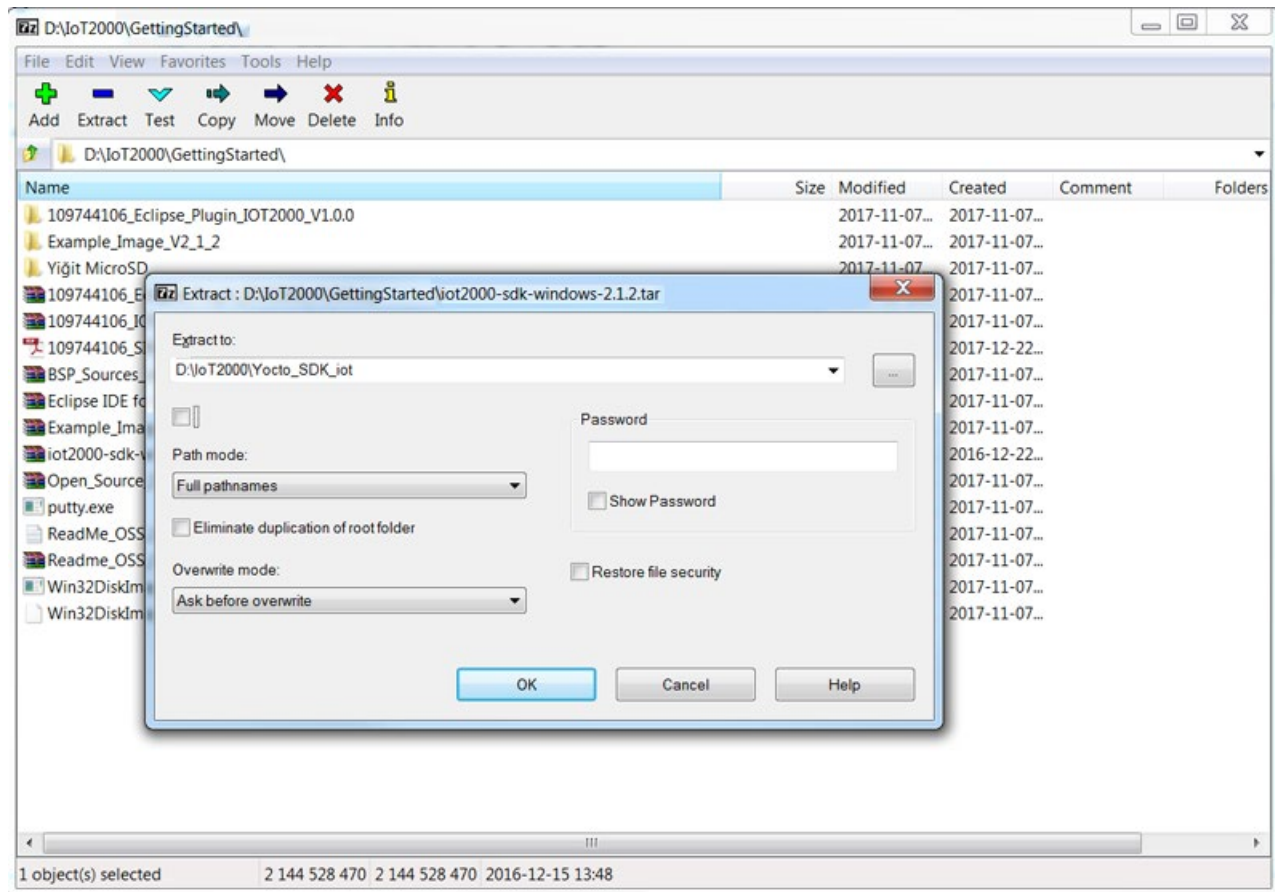
10.3.2 SDK Installation in Windows

Die IOT2000 SDK Version muss zu der Version des Beispielimages passen und funktioniert ausschließlich mit einem Windows Betriebssystem. Unter folgendem Link können Sie sich das Plugin herunterladen: SIMATIC IOT2000 Eclipse Plugin

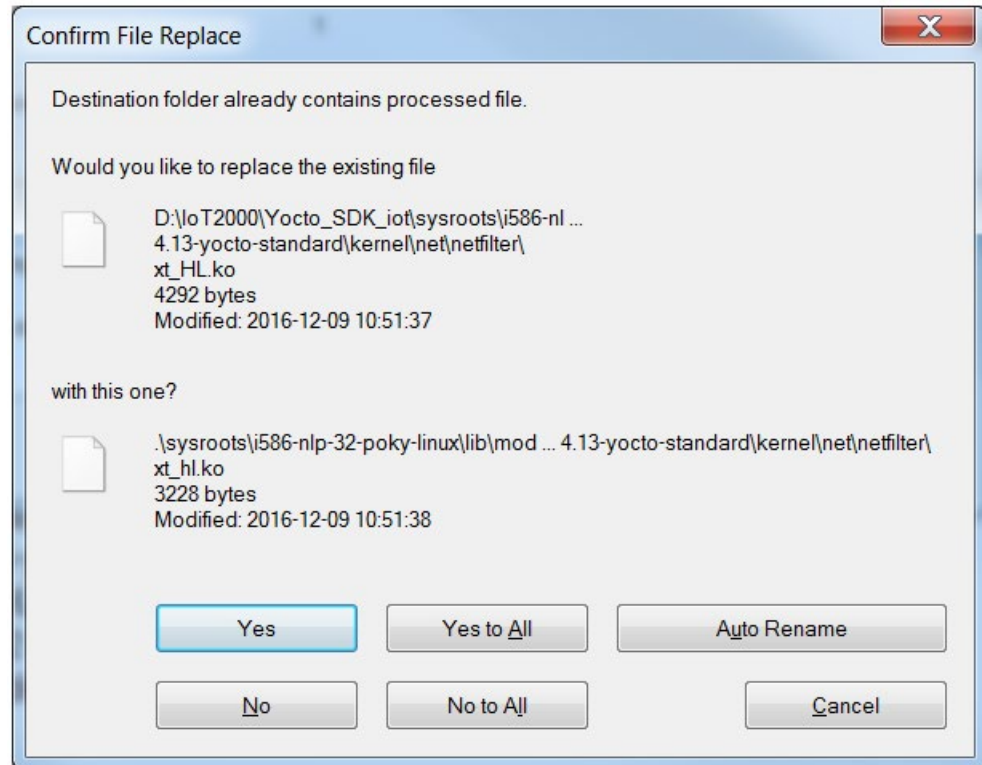
(<https://support.industry.siemens.com/cs/document/109744106/simatic-iot2000-eclipse-plugin?dti=0&lc=de-WW>)

Um das SDK zu installieren, gehen Sie wie folgt vor:

1. Starten Sie 7-Zip als Administrator.
2. Entzippen Sie die heruntergeladene Datei "IOT2000_sdk_windows_2.1.2.zip"
Beachten Sie, dass seit der Erstellung dieser Anleitung bereits eine neuere Version des SDKs erhältlich sein kann.
Die Datei "iot2000_sdk_windows.tar" wird entpackt.
3. Entzippen Sie die Datei "iot2000_sdk_windows.tar".



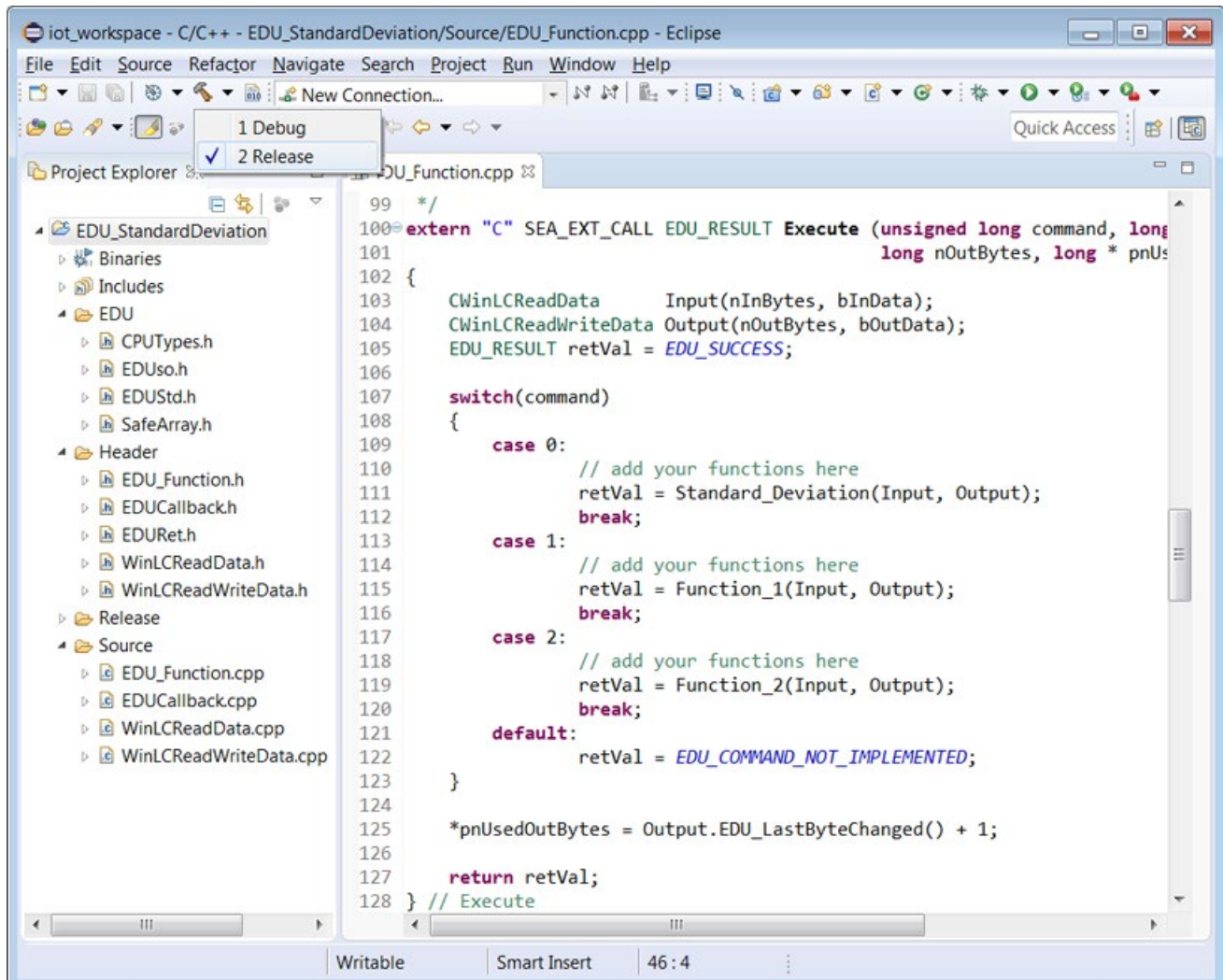
Ein Dialog öffnet sich, der Sie danach fragt, ob Sie die IOT2000 SDK spezifischen Dateien überschreiben wollen.



4. Klicken Sie auf die Schaltfläche "Ja, alle" um den Dialog zu bestätigen.

10.3.3 SO-Datei mit dem Beispielprojekt kompilieren

Die Projektdateien und Orderstrukturen werden im folgenden Bild dargestellt:



Sie können Ihre Software in die Dateien "EDU_Function.cpp" und "EDU_Function.h" schreiben. Diese Dateien enthalten alle Schnittstellenfunktionen der CPU Funktionsbibliothek.

Die "Execute" Funktion wird jedes mal aktiviert, wenn der SFB65002 (EXEC_COM) aufgerufen wird. Die Anweisung EXEC_COM im TIA Portal enthält den Parameter "Command". Dieser korrespondiert mit der Bedingung "switch case" in der SO-Datei.

Fügen Sie Ihre eigenen "Case x" Status mithilfe der Customfunktionen hinzu. Funktionsprototypen müssen in der EDU_Function.h Datei angelegt werden. Die Beispiel Funktion "Standard_Deviation (Input, Output)" wird unter den "Case 0" Status platziert. Das bedeutet, wenn EXEC_COM mit dem Command "0" aufgerufen wird, betrifft dies "Case 0" und die Funktion "Standard_Deviation" wird ausgeführt. Wenn Sie Ihr eigenes Projekt auf irgendeinem Betriebssystem oder IDE erstellen, müssen Sie folgende Parameter hinzufügen:

```
_M_IX86
```

Definiertes Symbol oder Präprozessoren

```
RELEASE
```

```
Position Independent Code (-fPIC)
```

```
-c -m32 -march=i586
```

Compiler Flags

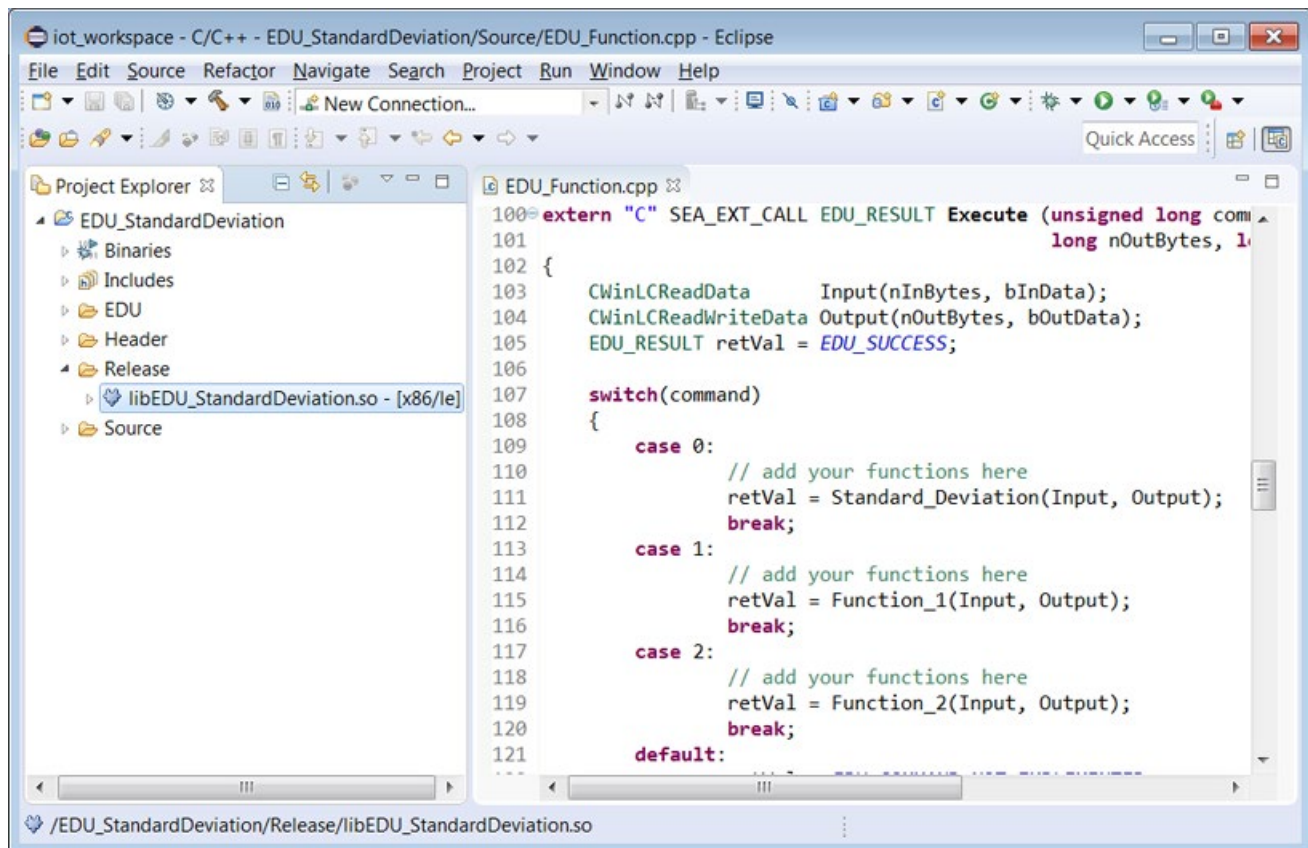
```
"${workspace}/${ProjName}/EDU"
```

Einschließlich Verzeichnisse

```
"${workspace}/${ProjName}/Header"
```

Kompilieren Sie das Projekt, nachdem Sie die notwendigen Compile- und Link-Parameter eingestellt haben.

Die SO-Datei "libEDU_StandardDeviation.so" wird generiert.



10.3.4 Cross Compilen mit der IOT2000 SDK Installation

Um ein neues Eclipse Projekt zu erstellen, müssen Sie ein Cross-Compiler Verzeichnis angeben und Compiler Flags einstellen. Außerdem müssen Sie die folgenden Ordner des Beispielprojekts in das Verzeichnis des neuen Projekts kopieren:

- Header
- EDU
- Source

Geben Sie die folgenden Compiler Optionen in den Eigenschaften des neuen Projekts ein:

Tabelle 10- 3 C/C++ Build

Eigenschaft			Compiler Option	Kommentar
Builder type:			Internal builder	
Environment:	POKY_HOME:		D:\IoT2000\Yocto_SDK_iot\sysroots\i586-nlp-32-poky-linux	
	PATH:		D:\IoT2000\Yocto_SDK_iot\sysroots\i686-pokysdk-mingw32\usr\bin\i586-poky-linux	Fügen Sie diesen Pfad an das Ende der PATH Variable ein.
Current toolchain:			Cross GCC	
Settings:	Cross GCC Compiler:	Command:	i586-poky-linux-gcc	Geben Sie hier die gleichen Einstellungen ein, wie bei Cross G++ Compiler.
	Cross G++ Compiler:	Command:	i586-poky-linux-g++	
		Defined symbols (-D):	RELEASE, _M_IX86	
		Includes:	"\${ workspace}/\${ProjName}/Header" "\${ workspace}/\${ProjName}/EDU" "\${POKY_HOME}\usr\include" "\${POKY_HOME}\usr\include\c++\5.3.0" "\${POKY_HOME}\usr\include\c++\5.3.0\i586-poky-linux" "\${POKY_HOME}\usr\include\mraa" "\${POKY_HOME}\usr\include\upm"	
		Miscellaneous:	-O0 -g3 -Wall -c -fmessage-length=0 -m32 -march=i586 -c -ffunction-sections -fdata-sections	
			Position Independent Code (-fPIC)	Aktivieren Sie diese Option.
	Cross G++ linker	Command:	i586-poky-linux-g++	
		Library search path (-L):	"\${POKY_HOME}"	
		Miscellaneous:	-m32 -fno-use-linker-plugin --sysroot=D:\IoT2000\Yocto_SDK_iot\sysroots\i586-nlp-32-poky-linux	
	Cross GCC Assembler	Command:	i586-poky-linux-as	

Diese Optionen wurden bereits für das Beispielprojekt angelegt.

Hinweis

Beachten Sie, dass Sie Ihren konkreten Projektnamen und SDK Pfad verwenden, z. B. bei "\${ workspace}/\${ProjName}", POKY_HOME, PATH und --systemroot.

10.3.5 SO-Datei auf IOT20X0 laden

Je nachdem mit welcher Entwicklungsumgebung Sie die SO-Datei kompiliert haben, können Sie die SO-Datei auf einen von 3 Wegen auf das IOT20X0 Gerät laden.

Hinweis

In der folgenden Beschreibung wurde Beispiel SO-Datei "libEDU_StandardDeviation.so".

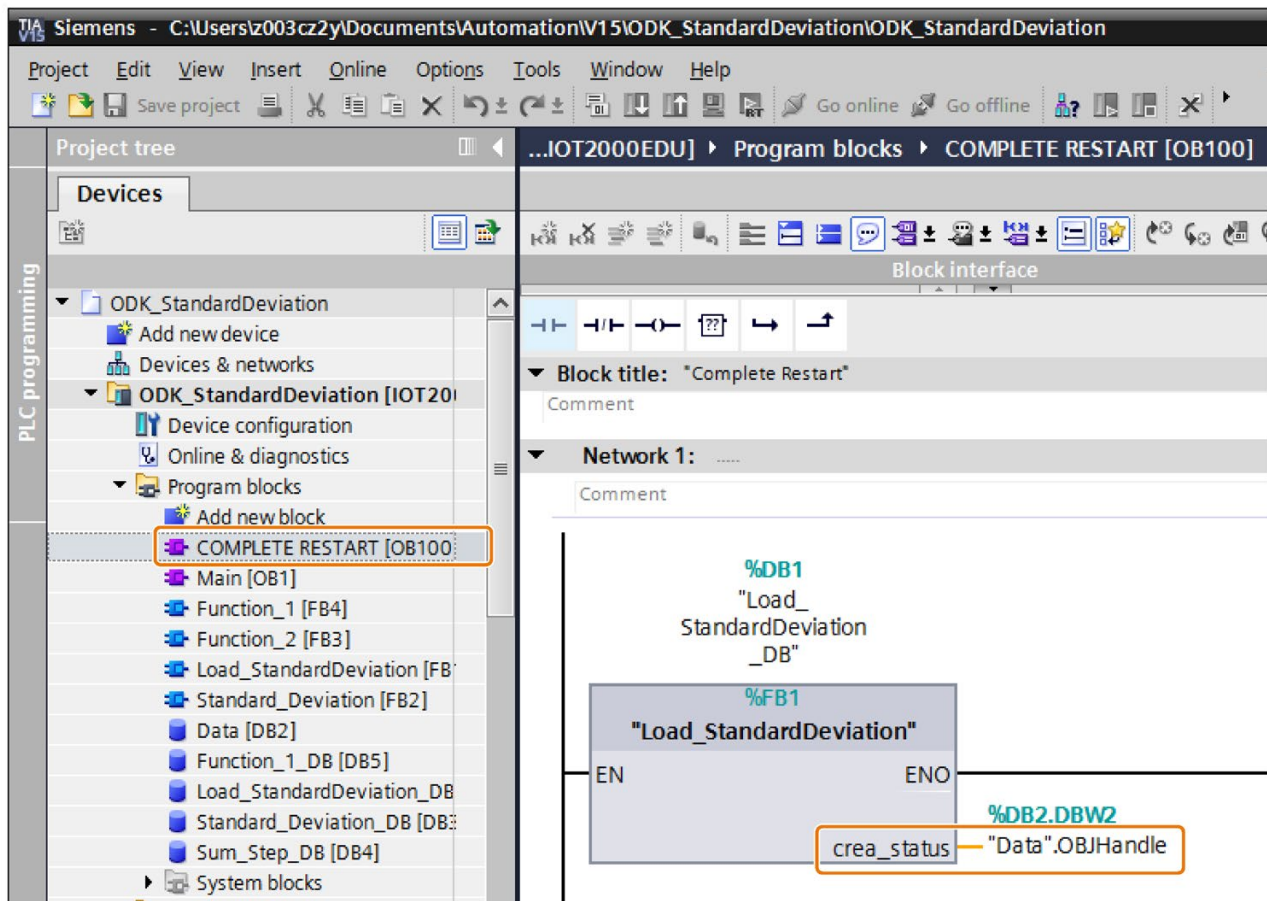
- Sie nutzen ein Windows Betriebssystem und haben den Cross-Compiler benutzt:
 - Übertragen Sie die Datei "libEDU_StandardDeviation.so" mithilfe des Programms "pscp.exe". Dieses können Sie sich kostenlos im Internet herunterladen.
 - Öffnen Sie das Windows Kommandozeilen Programm und starten Sie die pscp.exe mit dem Command "-scp".
 Zum Beispiel: `pscp.exe -scp ..\EDU_StandardDeviation (Eclipse)\Release\libEDU_StandardDeviation.so root@192.169.200.1:/home/root/`
 - In diesem Beispiel wird `root@192.168.200.1:/home/root/` für die IOT2000 Geräte-IP und für das Verzeichnis verwendet. Sie können das SO in ein beliebiges Verzeichnis verschieben.
- Sie haben Ihre Software auf einem Linux Betriebssystem kompiliert:
 - Verwenden Sie das Command "scp" in einem Terminal.
 - Zum Beispiel: `sudo scp /home/Alex/EDU_StandardDeviation (Eclipse)\Release\libEDU_StandardDeviation.so root@192.168.200.1:/home/root/`
- Sie können das SO auch auf einen USB-Stick kopieren:
 - Wenn Sie den USB-Stick an das IOT20x0 Gerät anschließen, wird ein Geräte-Name unter `/dev` folder angezeigt. Danach kann der USB-Stick in jedes temporäre Verzeichnis gemountet werden. Sie können das SO jetzt kopieren.
 Zum Beispiel:

```
Alex@ubuntu:~$ mount /dev/sdx /media (sdx: may be sdb, sdc ...)
Alex@ubuntu:~$ cp /media/ libEDU_StandardDeviation.so /home/Alex/
```

10.4 TIA Portal Projekt für die Applikation vorbereiten

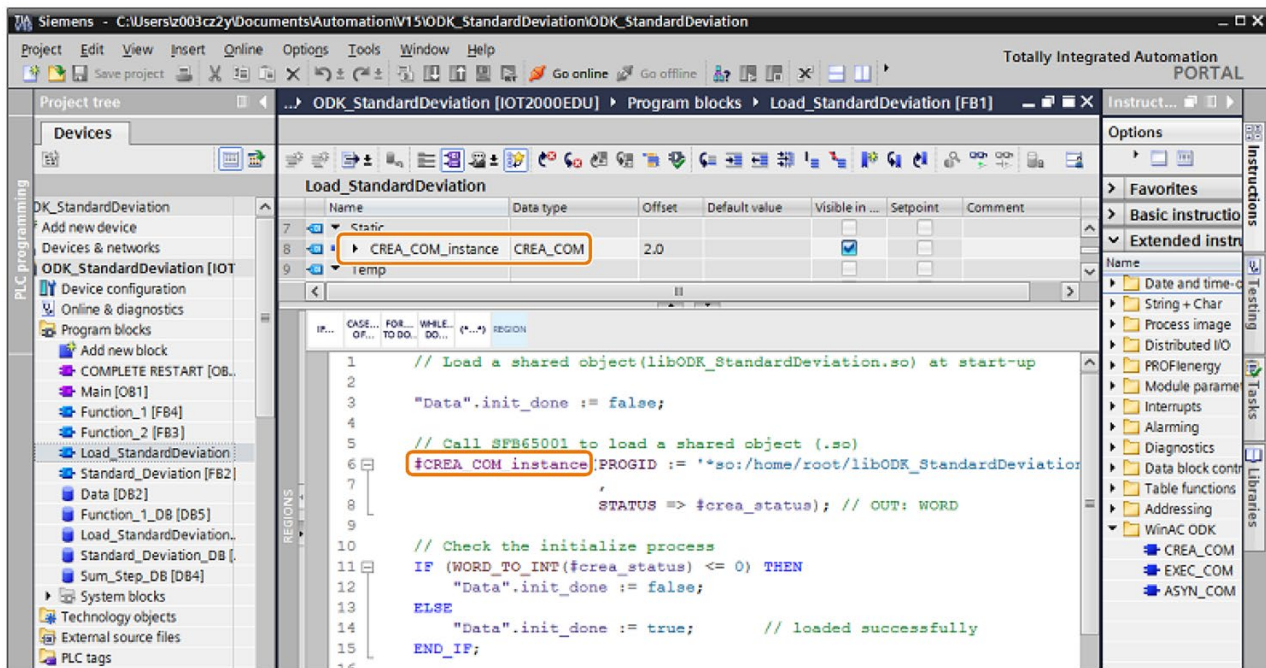
Um ein SO zu laden und auszuführen, müssen Sie die Anweisungen mit den richtigen Parametern im TIA Portal Projekt aufrufen:

1. Stellen Sie sicher, dass Ihre SO-Datei auf dem IOT2000 Gerät geladen wurde. Damit die IOT2000EDU ein SO laden kann, wird die Anweisung CREA_COM mit 2 Parametern aufgerufen:
 - PROGID: Zeigt auf den Objektnamen mit dem vollständigen Verzeichnispfad. Dieser Parameter hat den Typ "string".
 *so:/home/root/libEDU_StandardDeviation.so'
 Der *.so Präfix muss vor dem Objektnamen geschrieben werden, welcher auf die *.so Datei zeigt.
 - STATUS: Gibt einen Handle (OBJHandle) oder einen Fehlercode zurück. Wenn der Rückgabewert zwischen 0x0001 und 0x7FFF liegt, ist es ein Object Handle. Liegt er zwischen 0x8001 und 0x810C, ist es ein Fehlercode.
2. Allgemein werden Anweisungen in einen Funktionsblock angelegt, hier dem "Load_StandardDeviation[FB1]". Um das SO zu laden, ruft anschließend ein OB die Anweisungen auf, z. B. OB100.

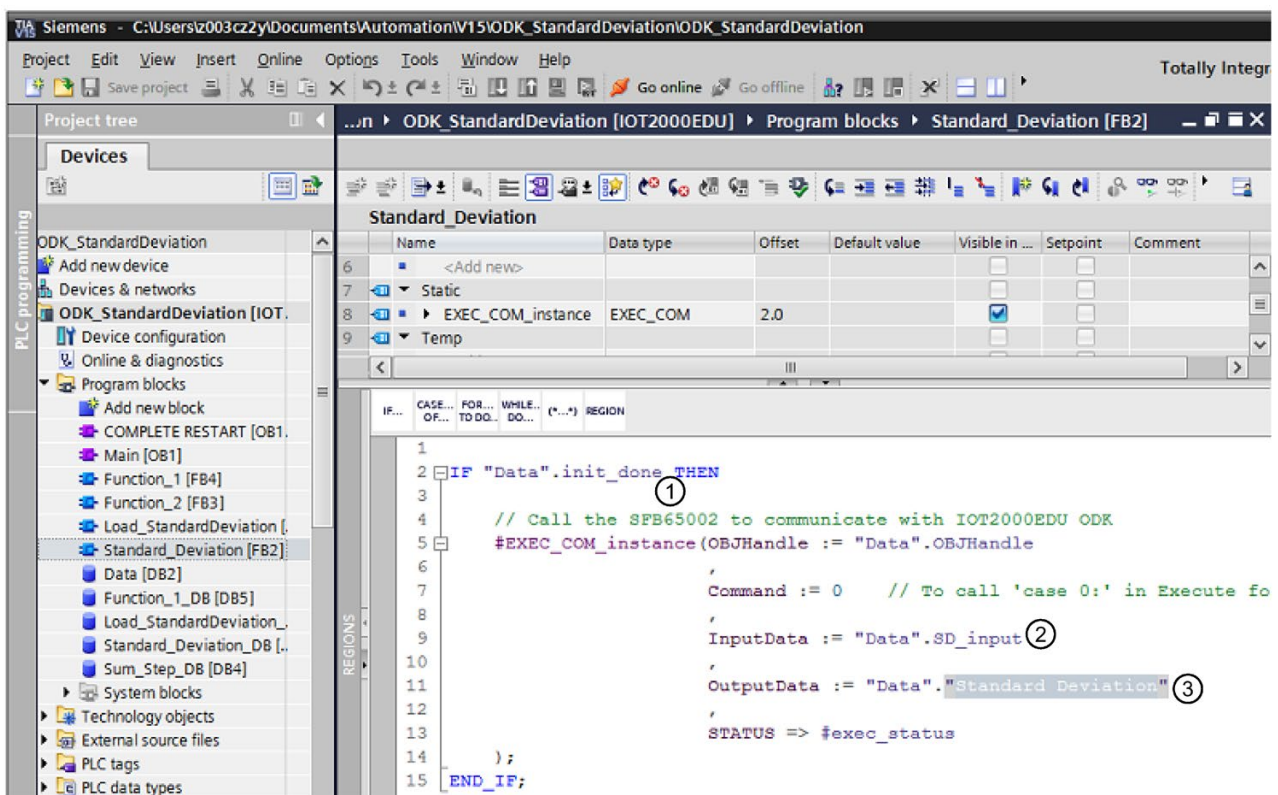


3. Ziehen Sie die Anweisungen direkt aus der Palette "Erweiterte Anweisungen" unter der Task Card "Anweisungen" in den Programmiereditor.

4. Bei Bedarf können Sie auch den statischen Namen der Anweisungen in der Baustein-Schnittstelle ändern, z. B. "CREA_COM_instance" und "EXEC_COM_instance".



5. Die Anweisung "EXEC_COM" ruft die "Execute" Funktion des SOs auf. Diese Funktion wird durch den OBJHandle Parameter spezifiziert.
Der OBJHandle ist der Rückgabewert der Anweisung "CREA_COM".
Wenn Sie 2 oder mehr geteilte Bibliotheken laden wollen, wird der OBJHandle verwendet, um zwischen den geteilten Bibliotheken unterscheiden zu können.
Die Anweisung "EXEC_COM" wird in einen neuen Funktionsblock eingefügt, hier Standard_Deviation[FB2], und wird Main[OB1] benannt.
6. Wichtig ist außerdem der Parameter "Command". Der Parameter "Command" spezifiziert einen speziellen switch-case Status in der "Execute" Funktion des SOs.
In diesem Beispiel wird die "Standard_Deviation(Input,Output)" Funktion unterhalb von "case 0" platziert. Daher muss "Command" den Wert "0" haben, damit die Funktion ausgeführt werden kann.



① "Data".init_done

Wenn das SO erfolgreich geladen wurde, ist dieser Wert "TRUE".

② "Data".SD_input

Ein Satz Nummern, der im "Data[DB2]" platziert wird. Sie können die Größe erweitern oder die Nummern modifizieren, wenn dies eine andere Standard Deviation Berechnung benötigt.

③ "Data"."Standard Deviation"

Ist der Output der Berechnung.

Verarbeiten der In- und Outputdaten der SO-Datei

Jede Funktion im switch-case hat zwei Inputwerte, so genannte Objekte:

- Inputdaten (r_{Input})
- Outputdaten (r_{Output})

Um Variablen aus diesen Objekten herauszubekommen, verwenden Sie die von den Helper Klassen unterstützten Datentypen, z. B. "EDU_ReadS7INT".

Beispiel:

Zu erst müssen alle Strukturen, z. B. "UDT", im Datenbaustein "Data" im TIA Portal definiert werden.

	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoi
	▼ Static						
	crea_status	Word	0.0	16#0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	OBJHandle	Word	2.0	16#0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	init_done	Bool	4.0	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	exec_StandDev	Bool	4.1	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	exec_Function_1	Bool	4.2	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	exec_SumStep	Bool	4.3	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	exec_status	Word	6.0	16#0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	SD_input	Struct	8.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
0	▼ UDT	"User_data_type_1"	18.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
1	MyInteger	Int	18.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	MyReal	Real	20.0	0.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	MyBool	Bool	24.0	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	Standard Deviation	Real	26.0	0.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	Sum_Inputs	Struct	30.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	Sum_Output	Int	34.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Diese Struktur kann sowohl Input- als auch Outputwert für die Anweisung "EXEC_COM" im TIA Portal sein.

```
#EXEC_COM_instance(OBJHandle := "Data".OBJHandle
/
Command := 1 // TO call 'case 1:' in Execute
/
InputData := "Data".UDT
/
OutputData := "Data".UDT
/
STATUS => #exec_status
```

"InputData := \"Data\".UDT" im TIA Portal korrespondiert mit r_{Input} in der SO-Datei. Weiterhin müssen alle struct-Variablen einzeln aus r_{Input} entnommen werden.

Zum Beispiel: die struct-Variable "MyBool" in UDT. Diese Variable wurde mithilfe von `rInput.EDU_ReadS7Bool(6,0,Mybool);` aus der SO-Datei entnommen.

10.4.1 Referenz der IOT2000EDU Funktionsbibliothek Anweisungen

Die IOT2000EDU Funktionsbibliothek stellt 2 Anweisungen im TIA Portal zur Verfügung:

- CREA_COM (SFB65001)
- EXEC_COM (SFB65002)

10.4.1.1 CREA_COM (SFB65001)

Die Anweisung "CREA_COM" lädt eine Instanz des SOs, spezifiziert durch den PROGID Parameter. Das IOT2000EDU Programm vergibt den InstanceID Parameter.

Die folgende Tabelle zeigt die Schnittstellen der Anweisung "CREA_COM":

Adresse	Declaration	Name	Datentyp	Kommentar
0.0	In	PROGID	STRING[254]	ID des zu ladenden SOs
256	Out	Status	WORD	SFB Rückgabecode: Fehlercode oder OBJHandle Code

Die PROGID ist ein String, der den Dateinamen und Ablagepfad des SOs enthält.

Zum Beispiel:

```
*so:/home/Benutzername/Desktop/Final_Project/CalculateStandartDeviation.so
```

Der SO-Name ist "CalculateStandartDeviation.so".

Die Anweisung "CREA_COM" evaluiert die Eingabebedingungen und führt folgende Aktionen durch:

1. Wenn das SO noch nicht geladen wurde, ruft "CREA_COM" die Funktion "ODKCreate" auf, um das SO zu erstellen. Um eine ClassID zu erstellen, wird der InstanceID Parameter verwendet. "CREA_COM" erstellt nur eine Instanz dieses Objekts. "CREA_COM" fügt die Objektinstanz der internen Liste der erstellten IOT2000EDU Objekten hinzu.
2. Wenn das SO bereits erstellt wurde, behält "CREA_COM" den IOT2000EDU Funktionsbibliothek-handle für zuvor erstellte Objekte bei (ein Index um den Object Pointer zu lokalisieren).
3. Wenn dies der erste Aufruf des "CREA_COM" nach dem Verlassen des STOP Modus ist, oder wenn das SO gerade erst erstellt wurde, ruft "CREA_COM" die Active Funktion auf.
4. "CREA_COM" stellt den Status Parameter zum IOT2000EDU Funktionsbibliothek-handle ein (oder Fehlercode) und stellt das BR Bit ein.

Rückgabecodes

Rückgabecode	Meldung	Bedeutung
0x0001 - 0x7FFF	OBJ_HANDLE	Zurückgegebener "object handle".
0x807F	ERROR_INTERNAL	Ein interner Fehler ist aufgetreten.
0x8001	E_EXCEPTION	Eine Ausnahme ist aufgetreten.
0x8102	E_CLSID_FAILED	Der Aufruf von CLSIDFromProgID ist fehlgeschlagen.
0x8103	E_COINITIALIZE_FAILED	Der Aufruf von CoInitializeEd ist fehlgeschlagen.
0x8104	E_CREATE_INSTANCE_FAILED	Der Aufruf von CoCreateInstance ist fehlgeschlagen.
0x8105	E_LOAD_LIBRARY_FAILED	Die Bibliothek wurde nicht geladen.
0x8106	E_NT_RESPONSE_TIMEOUT	Eine Antwort-Zeitüberschreitung in Windows ist aufgetreten.
0x8107	E_INVALID_OB_STATE	Der Controller ist in einem ungültigen Betriebszustand um ein OB zu planen.
0x8108	E_INVALID_OB_SCHEDULE	Zeitplaninformationen für den OB ist ungültig.
0x8109	E_INVALID_INSTANCEID	Instance ID für den Aufruf der Anweisung "CREA_COM" ist ungültig.
0x810A	E_START_ODKPROXY_FAILED	Der Controller konnte nicht die proxy DLL laden.
0x810B	E_CREATE_SHAREMEM_FAILED	Der Controller konnte keine Shared Memory Area erstellen oder initialisieren.
0x810C	E_OPTION_NOT_AVAILABLE	Der Versuch auf eine nicht verfügbare Option zuzugreifen ist aufgetreten.

10.4.1.2 EXEC_COM (SFB65002)

Die Anweisung "EXEC_COM" ruft die "Execute" Funktion des SO auf, spezifiziert durch den OBJHandle Parameter.

Die folgende Tabelle zeigt die Schnittstellen der Anweisung "EXEC_COM":

Adresse	Declaration	Name	Datentyp	Kommentar
0.0	In	OBJHandle	WORD	Zurückgegebener Handle von "CREA_COM"
2.0	In	Command	DWORD	Index der auszuführenden Funktion oder des auszuführenden Kommandos
6.0	In	InputData	ANY	Pointer zur Funktion Input Area
16.0	In	OutputData	ANY	Pointer zur Funktion Output Area
26.0	Out	Status	WORD	Anweisungsfehlercode oder Rückgabecode von der "Execute" Funktion

Die Anweisung "EXEC_COM":

1. Verifiziert, dass "CREA_COM" aufgerufen wurde und dass der Object Handle valide ist.
2. Verarbeitet die ANY Pointers und gibt die Fehlercodes für invalide ANY Pointer Parameter zurück.
3. Ruft die "Execute" Funktion des SOs auf.
4. Weist die Input und Output Pointer Areas den IOT2000EDU Data Access Helper Klassen zu.
5. Stellt den Status Parameter zum "Execute" Rückgabecode ein, außer vorher ist ein Fehler aufgetreten, und geht zurück zum IOT2000EDU Programm.

Hinweis

Beachten Sie, dass sich die Laufzeit der aufgerufenen Funktion zur Zykluszeit hinzu addiert.

Rückgabecodes

Rückgabecode	Meldung	Bedeutung
0x0000	NO_ERRORS	Keine Fehler
0x807F	ERRORS_INTERNAL	Ein interner Fehler ist aufgetreten.
0x8001	E_EXCEPTION	Eine Ausnahme ist aufgetreten.
0x8002	E_NO_VALID_INPUT	Input: Der ANY Pointer ist ungültig.
0x8003	E_INPUT_RANGE_INVALID	Input: Die ANY Pointer range ist ungültig.
0x8005	E_NO_VALID_INPUT	Output: Der ANY Pointer ist ungültig.
0x8005	E_OUTPUT_RANGE_INVALID	Output: Die ANY Pointer range ist ungültig.
0x8006	E_OUTPUT_OVERFLOW	Es wurden mehr Bytes durch das Shared Object in den Output Puffer geschrieben als zugewiesen wurden.
0x8007	E_NOT_INITIALIZED	Funktionsbibliothek-System wurde nicht initialisiert: Bisher kein Aufruf der Anweisung "CREA_COM".
0x8008	E_HANDLE_OUT_OF_RANGE	Der zugeteilte handle Wert passt zu keinem gültigen Shared Object.
0x8009	E_INPUT_OVERFLOW	Es wurden mehr Bytes durch das Shared Object in den Input Puffer geschrieben als zugewiesen wurden.

10.5 Funktionsbibliothek-Programme mit Matlab Simulink erstellen

Mathworks MATLAB ist eine Software zur vorrangigen Lösung mathematischer Problemstellungen und deren Visualisierung.

Simulink ist ein Addon für MATLAB zur grafischen Modellierung von Systemen und deren Simulation.

Mit dem Simulink Coder Addon können Sie aus einem Simulink Modell direkt C/C++ Code kompilieren. Die IOT2000EDU Funktionsbibliothek ermöglicht es Ihnen, C/C++ Code in IOT20x0 Geräten ablaufen zu lassen.

Hinweis

Die nachfolgenden Screenshots der Codebeispiele können unter Umständen vom ausgelieferten Produkt abweichen.

10.5.1 Voraussetzungen

Sie benötigen folgende Software:

- TIA Portal V15
- Matlab 2017b in folgender Konfiguration:
 - Matlab 9.3
 - Matlab Coder 3.4
 - Embedded Coder 6.13
 - Simulink 9.0
 - Simulink Coder 8.13
- Eclipse Kepler
- IOT2000 SDK



10.5.2 Simulink Modell erstellen

Voraussetzung

MATLAB R2017b ist geöffnet.

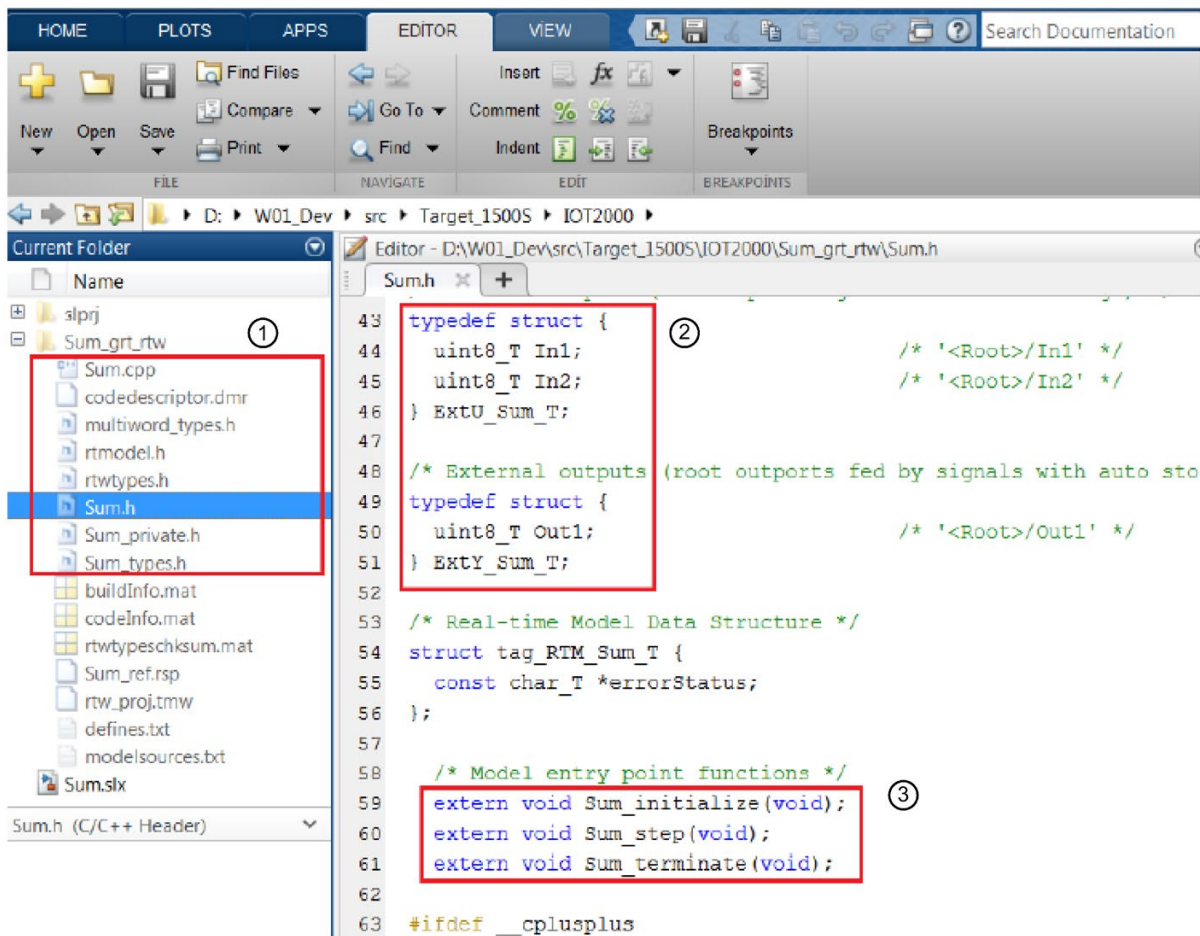
Vorgehen

Das Vorgehen wird im Folgenden am Beispiel einer einfachen mathematischen Operation beschrieben.

1. Um ein neues Simulink-Modell zu erstellen, klicken Sie in der Menüleiste auf Simulink-Symbol .
2. Fügen Sie aus dem Simulink Library Browser  folgende Blöcke per Drag&Drop ein:
 - Input: unter "Simulink > Source > In1"
 - Input: unter "Simulink > Source > In2"
 - Output: unter "Simulink > Source > Out1"
 - Adding Block: unter "Simulink > Math Operation > Add"
3. Um die jeweiligen Block Parameter für einen Input zu öffnen, doppelklicken Sie auf den entsprechenden Input-Block im Simulink-Modell.
4. Ändern Sie bei beiden Input-Blöcken im Fenster "Block Parameter" unter "Signal Attributes" den Parameter "Data type" in "uint8".
5. Passen Sie die Simulink Parameter (Seite 80) an.
6. Speichern Sie das Modell ab.
Verwenden Sie für dieses Beispiel den Namen "Sum".
7. Um den Build-Prozess in Simulink zu starten, wählen Sie in der Menüleiste den Befehl "Code > C/C++ Code > Build Model".


Ergebnis

Das System erstellt den Ordner "Sum_grt_rtw" und legt darin die generierten *.cpp und *.h Dateien ab. Die Dateinamen erhalten den Modellnamen als Prefix, in diesem Fall also "Sum".



- ① Generierte Dateien
- ② Input- und Outputparameter
- ③ Interface Funktionen des Sum-Modells

10.5.3 Beschreibung der Simulink Parameter

Um die Simulink Parameter für den Build-Prozess anzupassen, öffnen Sie die "Configuration Parameters" über das Symbol  in der Menüleiste.

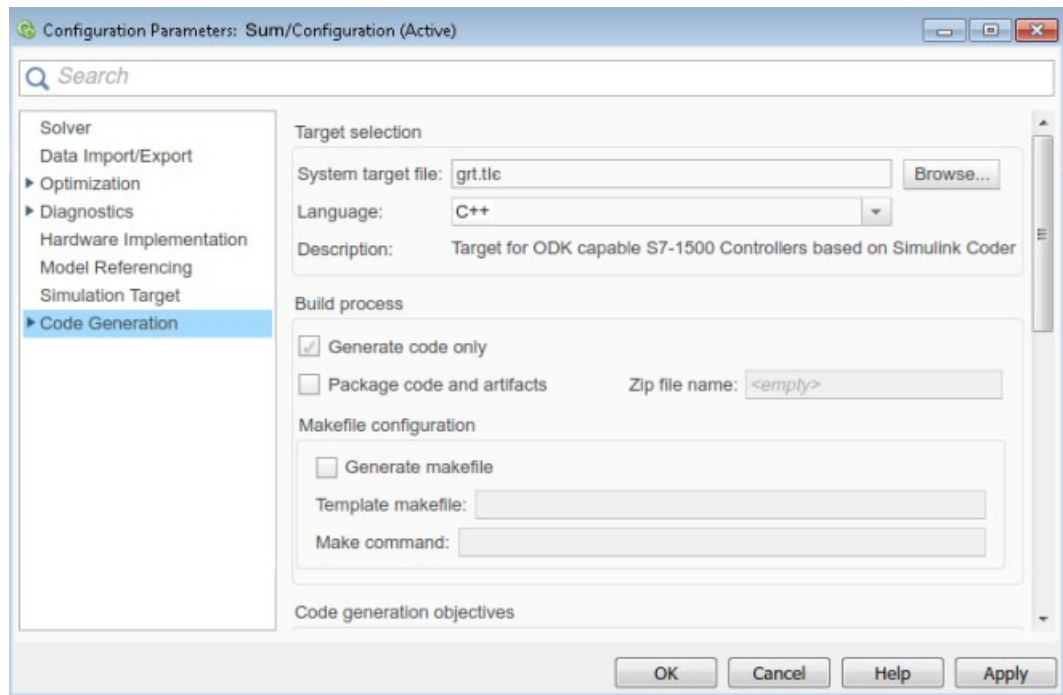


Bild 10-1 Configuration Parameters

Relevante Parameter

Solver

In Simulink wird ein dynamisches System als mathematische Berechnung modelliert. Um die Ausführung des Systems zu simulieren, wird diese Berechnung in bestimmten Zeitintervallen durchgeführt. Die Größe dieses Zeitintervalls wird als "Step-Size" bezeichnet. Das Verfahren zur Berechnung der Zustände eines Modells wird hier als Auflösen (solving) des Modells bezeichnet.

- Solver options
 - Bestimmen Sie die Solver-Auswahl
 - Type: Fixed Step
 - Solver: auto
- Fixed-step size: auto

Code Generation

- Target Selection:
 - System target file: grt.tlc (Create Visual C/C++ Solution File for Simulink Coder)
 - Language: C++
- Build process:
 - Generate code only: Option "Generate code only" aktiviert
 - Generate makefile: Option "Generate makefile" deaktiviert
- Interface:
 - Code interface packaging: Nonreusable function
 - MAT-file logging: Option "MAT-file logging" deaktiviert
 - ASAP2 interface: Option "ASAP2 interface" deaktiviert
 - External mode: Option "External mode" deaktiviert

10.5.4 Simulink Modell in Eclipse integrieren

Um den vom Simulink Coder generierten C/C++ Code auf IOT2000EDU Geräten ablaufen zu lassen, muss der Code von der IOT2000EDU Funktionsbibliothek konvertiert werden. Dafür steht Ihnen Eclipse zur Verfügung.

Vorgehen

1. Starten Sie Eclipse.
2. Legen Sie im "Project Explorer" in Ihrem Eclipse-Projekt "EDU_StandardDeviation" den Ordner "Matlab" an.
3. Kopieren Sie die generierten *.cpp und *.h Dateien in den Ordner "Matlab".
4. Öffnen Sie im "Project Explorer" unter "Source" die EDU_Function.cpp.
5. Um auf die Interface-Funktionen zugreifen zu können, fügen Sie die Header-Datei "Sum.h" in die EDU_Function.cpp ein.

Lokalisieren Sie die 3 Funktionen:

- `Sum_initialize()` unter `ODKCreate`
 ODKCreate wird aufgerufen, wenn ein Shared Object (*.so) geladen wird.
- `Sum_terminate()` unter `ODKRelease`
 ODKRelease wird aufgerufen, wenn eine Shutdown-Operation von der IOT2000EDU ausgeführt wird.
- `Sum_step()` in jeder von der IOT2000EDU Funktionsbibliothek mittels der Anweisung "EXEC_COM" aufgerufenen Case-Anweisung unter dem Execute Callback.

6. Im Beispielprojekt befindet sich die "Function_2 (Input, Output)" unter dem "case 2:" Statement. Diese Funktion besitzt Input und Output Werte, `Sum_step` fehlen jedoch diese Werte. Daher müssen Sie die Inputwerte vor `Sum_step` und die Outputwerte danach zuweisen und aufeinander anpassen:

```

113         break;
114     case 1:
115         // add your functions here
116         retVal = Function_1(Input, Output);
117         break;
118     case 2:
119         // add your functions here
120         retVal = Function_2(Input, Output);
121         break;
122     default:
123         retVal = EDU_COMMAND_NOT_IMPLEMENTED;
124     }
125
126     *pnUsedOutBytes = Output.EDU_LastByteChanged() + 1;
127
128     return retVal;
129 } // Execute
130
131 /**
132  * @ingroup User_defined_functions
133  * @details User-defined function that is called from the Execute method on a call
134  * to SFB65002 within IOT2000EDU.
135  */
136 EDU_RESULT Function_2 (CWinLCReadData& rInput, CWinLCReadWriteData& rOutput )
137 {
138
139     //////////// READ INPUTS ////////////
140     short input_In1;
141     rInput.EDU_ReadS7INT(0, input_In1 );
142     Sum_U.In1 = input_In1;
143
144     short input_In2;
145     rInput.EDU_ReadS7INT(2, input_In2);
146     Sum_U.In2 = input_In2;
147
148     //////////// CALL THE STEP FUNCTIONS ////////////
149     Sum_step();
150
151     //////////// WRITE OUTPUTS ////////////
152     rOutput.EDU_WriteS7INT(0, (short)Sum_Y.Out1);
153
154     return EDU_SUCCESS;
155 }

```

```

typedef struct {
    uint8_T In1;
    uint8_T In2;
} ExtU_Sum_T;

typedef struct {
    uint8_T Out1;
} ExtY_Sum_T;

```

7. Klicken Sie mit der rechten Maustaste auf das Eclipse-Projekt "EDU_StandardDeviation" und wählen Sie im Kontextmenü den Befehl "Properties" aus.

Der Dialog "Properties for EDU_StandardDeviation" öffnet sich.

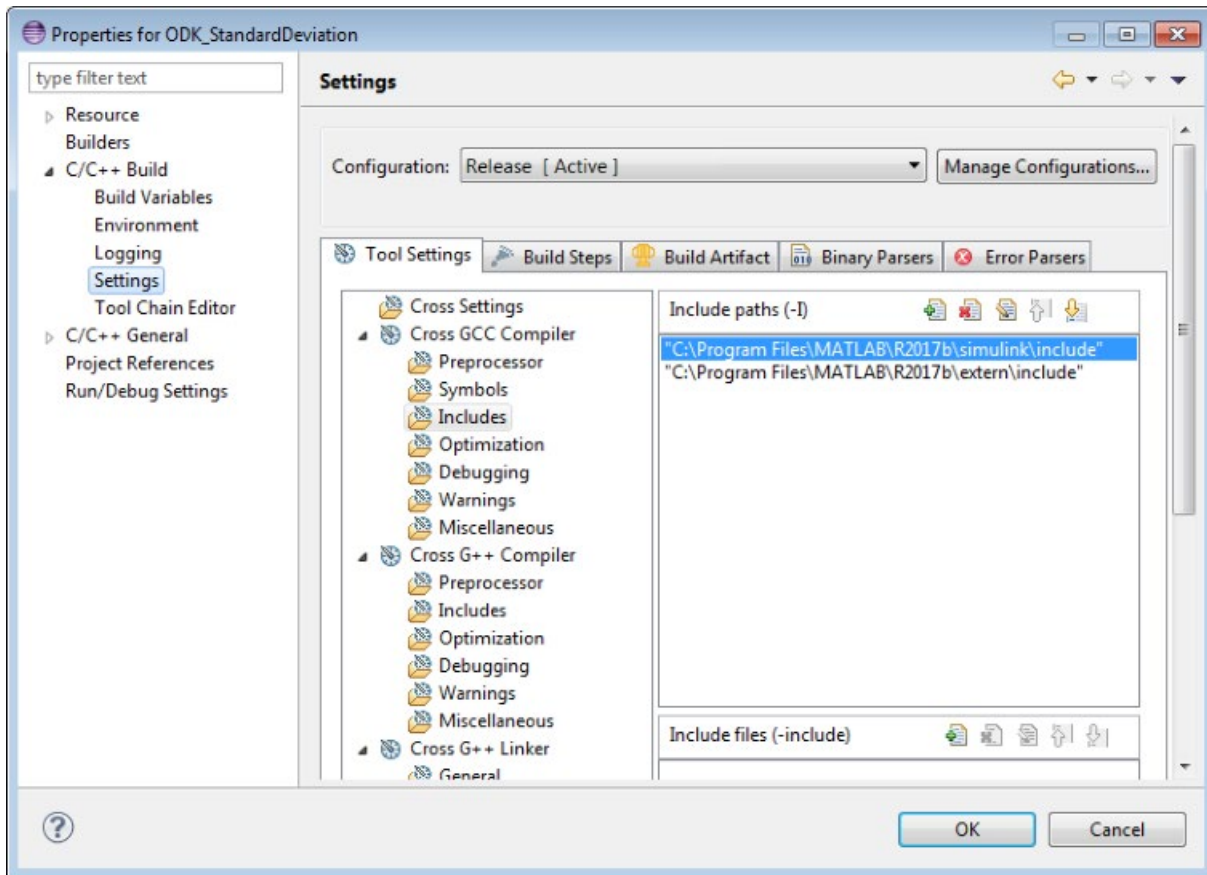


Bild 10-2 Dialog "Properties for EDU_StandardDeviation"

8. Fügen Sie unter "C/C++ Build > Settings" alle Pfade ein, unter deren Verzeichnissen Matlab-/Simulink-spezifische Dateien liegen und bestätigen Sie mit "OK".

Z. B.:

- tmwtypes.h: C:\Program Files\MATLAB\R2017b\extern\include
- simstruc_types.h: C:\Program Files\MATLAB\R2017b\simulink\include

9. Um das Projekt zu kompilieren, klicken Sie auf das Symbol "🔧".

Ergebnis

Die SO-Datei "libEDU_StandardDeviation.so" wird im "Project Explorer" unter "Release" erstellt.

10.5.5 TIA Portal für das Simulink-Modell vorbereiten

Um die Funktion "Sum_Step" von der IOT2000EDU aufzurufen, müssen Sie Ihr TIA-Programm anpassen.

Voraussetzung

Sie haben das Programm "EDU_StandardDeviation" im TIA Portal geöffnet.

Vorgehen

Anpassung am Beispiel des Programms "EDU_StandardDeviation".

1. Öffnen Sie den Datenbaustein "Data".

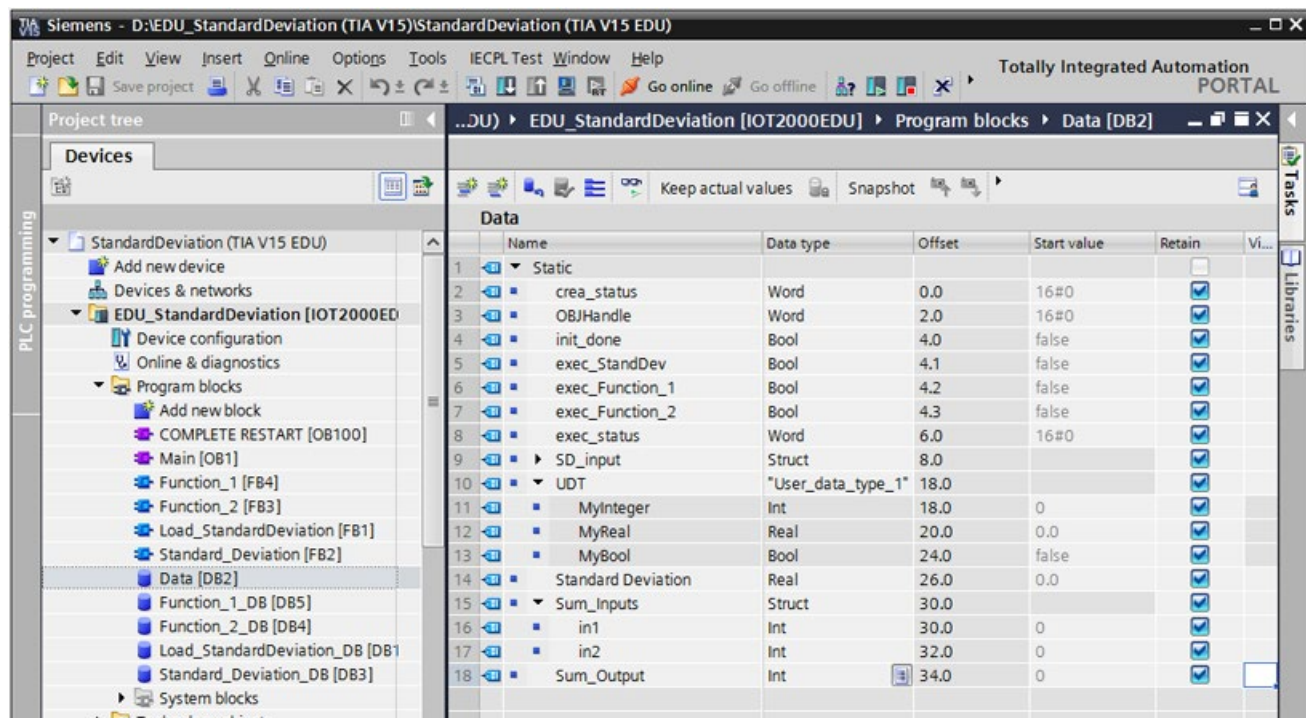


Bild 10-3 Ein- und Ausgabeparameter vergeben

2. Fügen Sie den Eingabeparameter (Sum_Inputs) und den Ausgabeparameter (Sum_Output) hinzu, die an die Funktion "Sum_Step" übergeben werden.

Vergeben Sie für "Sum_Inputs" den Datentyp "Struct" mit 2 Variablen (In1, In2) mit dem Datentyp "Int".

Vergeben Sie für "Sum_Ouput" den Datentyp "Int".

3. Weisen über den Funktionsblock "Function_2" die Ein- und Ausgabeparameter der Anweisung "EXEC_COM" zu.

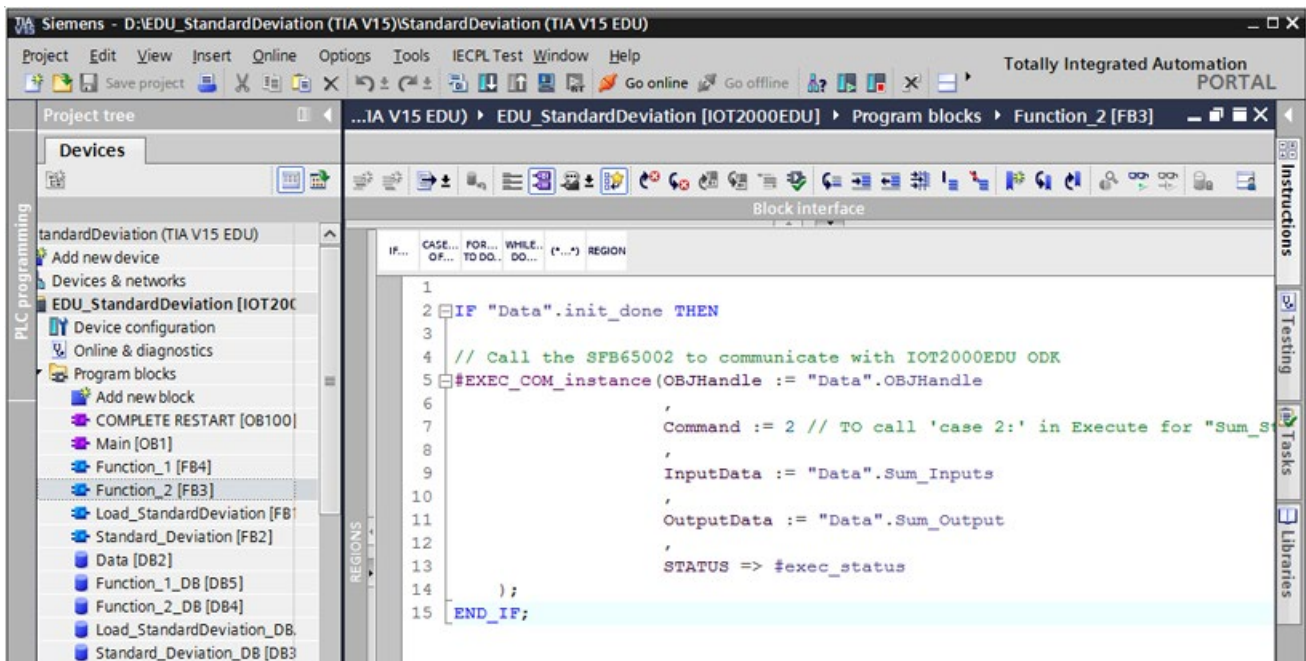


Bild 10-4 Ein- und Ausgabeparameter "Sum_Step" zuweisen

4. Laden Sie das TIA-Programm in das IOT20x0 Gerät.

Hinweis

Voraussetzungen zum erfolgreichen Laden

- Sie haben die SO-Datei "libEDU_StandardDeviation.so" in das IOT20x0 Gerät kopiert.
- Sie haben den Pfad der SO-Datei im TIA Portal im Funktionsblock "Load_StandardDeviation" folgendermaßen definiert:
 "so:/home/root/libEDU_StandardDeviation.so"
- Die IOT2000EDU läuft.

5. Testen und Beobachten Sie über die Beobachtungstabelle den Funktionsblock "Function_2", indem Sie in die Zeile "exec_Function_2" in der Spalte "Steuerwert" den Wert "TRUE" setzen.

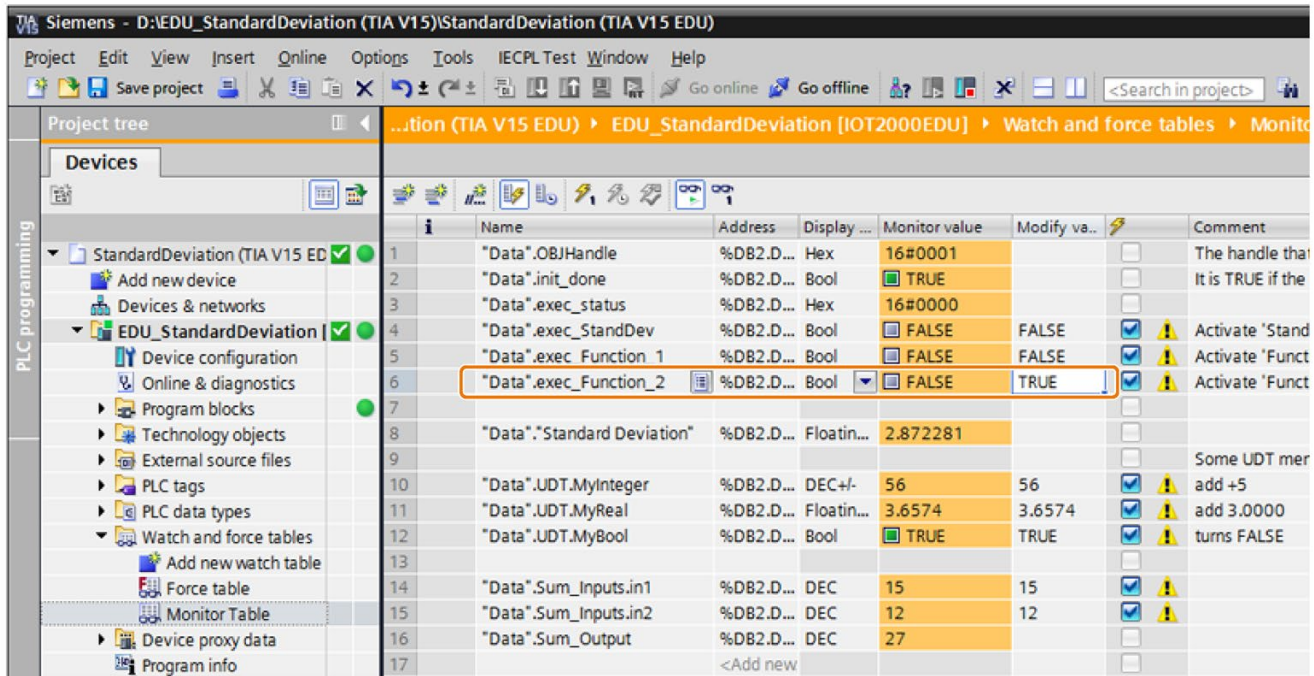


Bild 10-5 Sum_Step testen

6. Um den Beobachtungswert des Ausgangsparameters zu verändern, passen Sie den Steuerwert der Eingabeparameter an und klicken Sie auf das Symbol "⚡".

10.5.6 Datentyp-Umwandlung

Führen Sie eine Datentyp-Umwandlung zwischen der IOT2000EDU Funktionsbibliothek, Simulink Coder und TIA Portal durch. Achten Sie darauf, dass die umgewandelten Datentypen zueinanderpassen.

Die IOT2000EDU Funktionsbibliothek unterstützt maximal 32-bit (4-bytes). Wenn der Datentyp im Programm der IOT2000EDU Funktionsbibliothek größer als der im TIA Portal genutzte Datentyp ist, kann es zu Datenverlust kommen.

Informationen zu den Datentypen finden Sie im Kapitel "Datentypen (Seite 59)".

OUC Kommunikation

11.1 Verbindungsaufbau

Die IOT2000EDU unterstützt die "Open User Communication (OUC)" mithilfe des TCP/IP Stacks. Mit der OUC-Funktionalität lässt sich die IOT2000EDU als Server oder Client einsetzen, je nach den Anforderungen Ihres Programms.

Die IOT2000EDU unterstützt "OUC TCP"- und UDP-Verbindungstypen.

Hinweis

"ISO on TCP" ist nicht unterstützt.

Die IOT2000EDU unterstützt die folgenden OUC-Anweisungen für den Verbindungsaufbau:

- "TCON" (SFC 133)
- "TDISCON" (SFC 134)

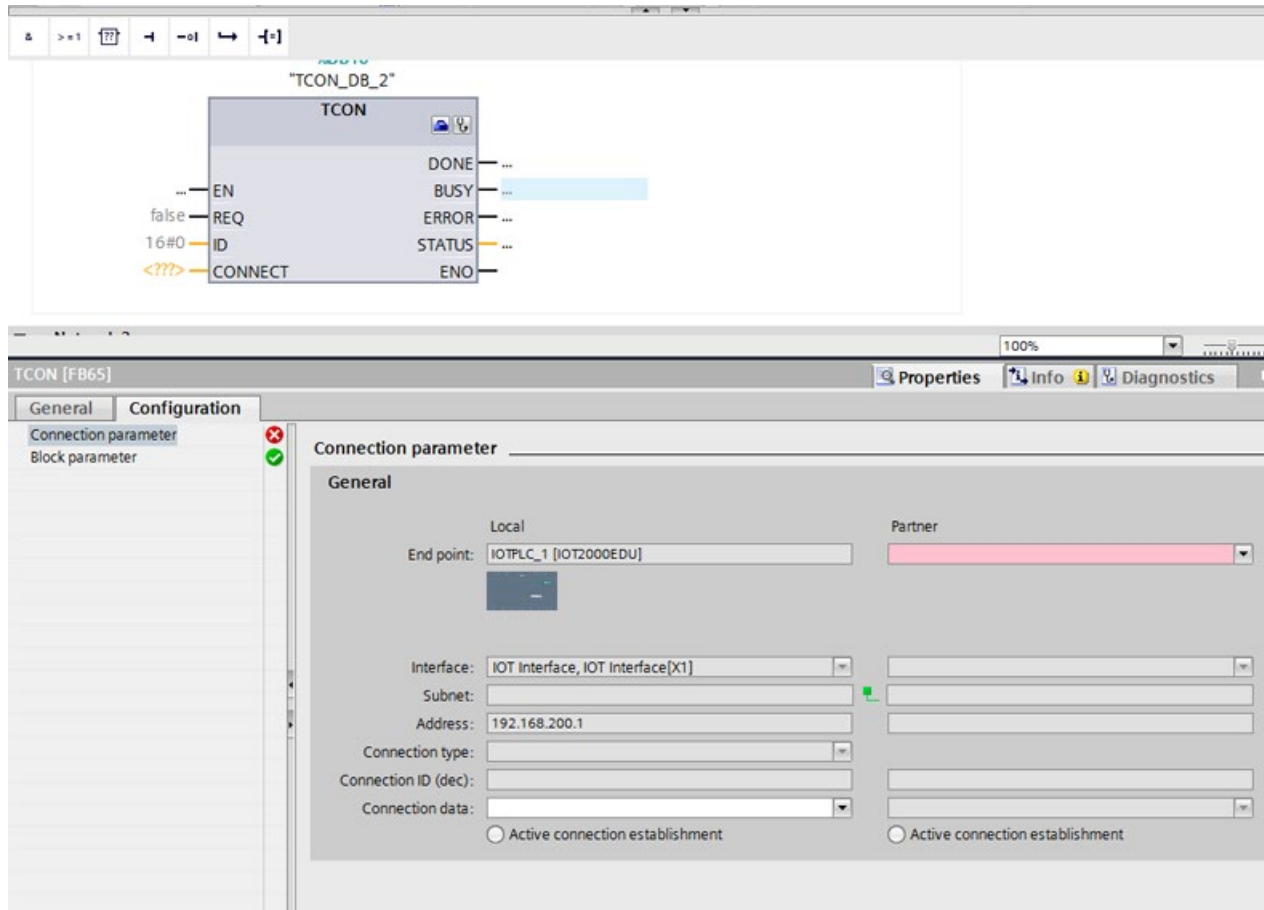
Communication		
Name	Description	Version
▶ S7 communication		V1.3
▼ Open user communication		V5.1
TCON	Establish communication connection	V4.0
TDISCON	Terminate communication connection	V2.1

Diese Anweisungen finden Sie im TIA Portal in der Task Card "Anweisungen" unter "Kommunikation > Open User Communication".

Sowohl TCP-, als auch UDP-basierte OUC-Verbindungen benötigen die Anweisungen "TCON" und "TDISCON".

11.2 TCON und TDISCON im TIA Portal konfigurieren

Sie können die Eingabe- und Ausgabeparameter von "TCON" und "TDISCON" entweder manuell oder mithilfe des TIA Portal Konfigurationswizards einstellen.



11.3 Datenaustausch

Daten lassen sich bei aktivierter Verbindung in beide Richtungen senden. Sie können also sowohl gleichzeitig gesendet und empfangen werden.

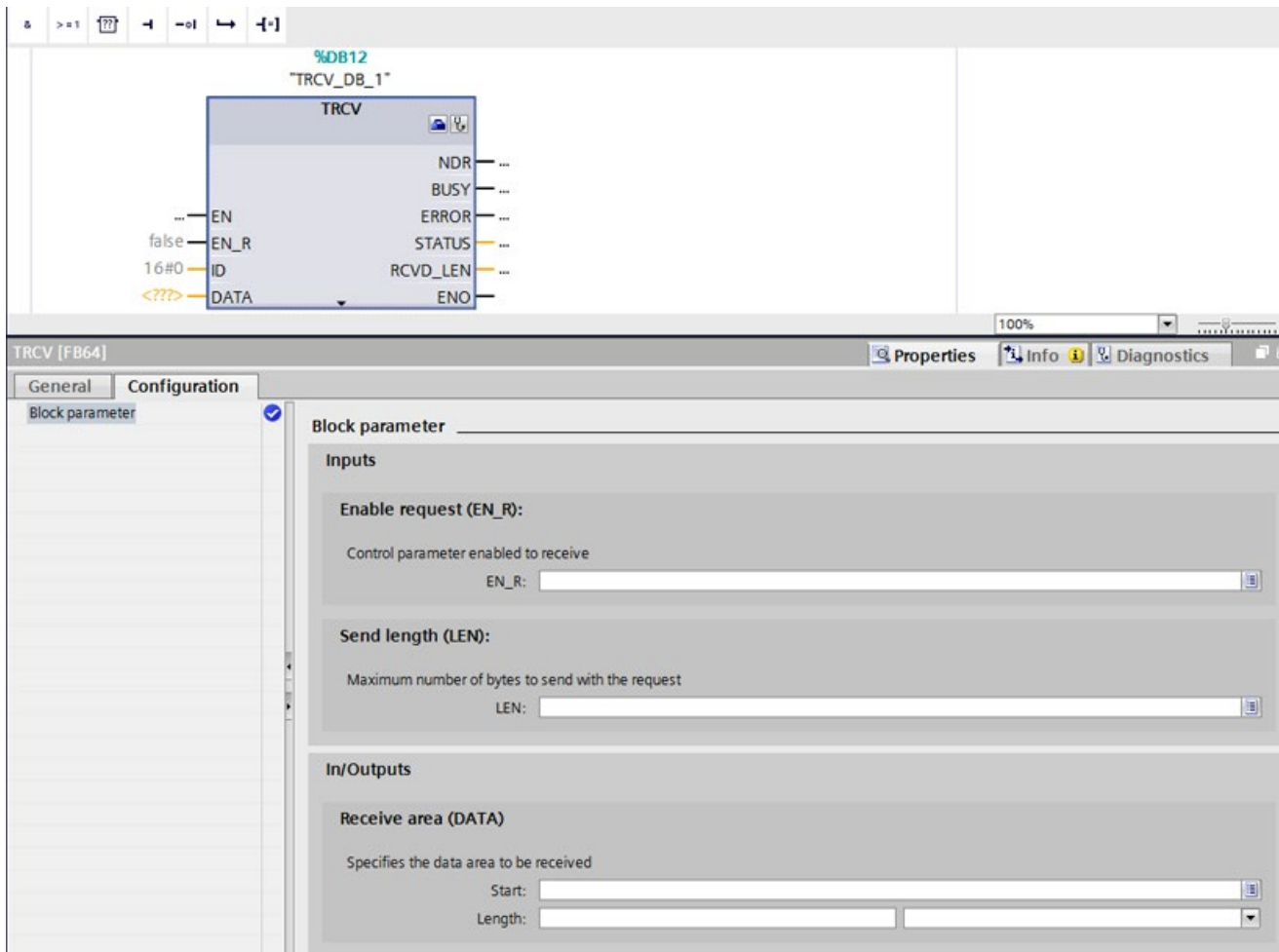
Folgende Kommunikationsanweisungen stehen Ihnen für den Datenaustausch zur Verfügung:

- "TSEND" (SFC 131)
- "TRCV" (SFC 132)
- "TUSEND" (SFC 135)
- "TURCV" (SFC 136)

Communication		
Name	Description	Version
▶ S7 communication		V1.3
▼ Open user communication		V5.1
■ TSEND	Send data via communication connection	V4.0
■ TRCV	Receive data via communication connection	V4.0
■ TUSEND	Send data via Ethernet (UDP)	V4.0
■ TURCV	Receive data via Ethernet (UDP)	V4.0

Diese Anweisungen finden Sie im TIA Portal in der Task Card "Anweisungen" unter "Kommunikation > Open User Communication".

11.4 TSEND, TRCV und TUSEND, TURCV im TIA Portal konfigurieren



"TSEND" und "TRCV" sind Verbindungsanweisungen, die nur mit TCP-basierten OUC-Verbindungen funktionieren.

"TUSEND" und "TURCV" sind verbindungslose Anweisungen, die nur mit UDP-basierten OUC-Verbindungen funktionieren.

Hinweis

Die IOT2000EDU unterstützt bis zu 4 OUC-Verbindungen gleichzeitig. Wählen Sie die von OUC verwendeten Ports sorgfältig aus, da diese Ports nicht von anderen Netzwerkprogrammen verwendet werden sollten.

Sicherstellen der Kompatibilität des Codes für IOT2000EDU und S7-1500

Stellen Sie in der Programmierung ihres Codes sicher, das Ihr Programm sowohl auf einer IOT2000EDU als auch einer S7-1500 Steuerung lauffähig ist. Beachten Sie dazu den folgenden Hinweis:

Hinweis

Alle 4 genannten Bausteine geben auch bei Erstaufruf mit REQ=1 und erfolgreicher Ausführung im RET_VAL den Code W#16#0000 zurück.

Wenn das Programm auf einer S7-1500 Steuerung ausgeführt werden soll, dann prüfen Sie in Ihrem Code auch auf den RET_VAL=W#16#7001.

Weitere Informationen finden Sie dazu in der TIA Portal Hilfe unter "PLC programmieren > Anweisungen > Anweisungen (S7-1200, S7-1500) > Asynchron arbeitende Anweisungen (S7-1200, S7-1500) > Unterschied zwischen synchron und asynchron arbeitende Anweisungen".

Technische Daten

A.1 Technische Daten

Technische Daten der IOT2000EDU

Folgende technische Daten gelten für die IOT2000EDU:

Artikelnummer	6ES7671-0LE00-0YB0
Allgemeine Informationen	
Produkttyp-Bezeichnung	IOT2000EDU
Firmware-Version	V1.1
Engineering mit	
• Programmierpaket	STEP 7 im TIA Portal ab V15
Speicher	
Arbeitsspeicher	
• integriert (für Programm)	128 kbyte
• integriert (für Daten)	256 kbyte
CPU-Bausteine	
DB	
• Anzahl, max.	200; Begrenzung durch Arbeitsspeicher für Daten
• Größe, max.	64 kbyte
FB	
• Anzahl, max.	20; Begrenzung durch Arbeitsspeicher für Code
• Größe, max.	64 kbyte
FC	
• Anzahl, max.	20; Begrenzung durch Arbeitsspeicher für Code
• Größe, max.	64 kbyte

Artikelnummer	6ES7671-0LE00-0YB0
OB	
<ul style="list-style-type: none"> Anzahl, max. Anzahl Freie-Zyklus-OBs Anzahl Uhrzeitalarm-OBs Anzahl Verzögerungsalarm-OBs Anzahl Weckalarm-OBs Anzahl Anlauf-OBs Anzahl Asynchron-Fehler-OBs Anzahl Synchron-Fehler-OBs 	Begrenzung nur durch Arbeitsspeicher für Code 1; OB 1 1; OB 10 1; OB 20 9; OB 30 -38 2; OB 100, 102 3; OB 80, 84, 85 1; OB 121
Schachtelungstiefe	
<ul style="list-style-type: none"> je Prioritätsklasse zusätzliche innerhalb eines Fehler-OBs 	16 16
Zähler, Zeiten und deren Remanenz	
S7-Zähler	
<ul style="list-style-type: none"> Anzahl 	2 048
IEC-Counter	
<ul style="list-style-type: none"> vorhanden Art Anzahl 	Ja SFB Begrenzung durch Arbeitsspeicher für Daten/DB Zähler
S7-Zeiten	
<ul style="list-style-type: none"> Anzahl 	2 048
IEC-Timer	
<ul style="list-style-type: none"> vorhanden Art Anzahl 	Ja SFB Begrenzung durch Arbeitsspeicher für Daten/DB Zähler
Datenbereiche und deren Remanenz	
Merker	
<ul style="list-style-type: none"> Anzahl, max. Anzahl Taktmerker 	16 kbyte 8; es sind 8 Taktmerkerbits, zusammengefasst in einem Taktmerkerbyte
Lokaldaten	
<ul style="list-style-type: none"> voreingestellt 	32 kbyte
Digitale Kanäle	
<ul style="list-style-type: none"> Eingänge Ausgänge 	20; Über Arduino UNO Shields 20; Über Arduino UNO Shields; davon 6 PWM Ausgänge

Artikelnummer	6ES7671-0LE00-0YB0
Analoge Kanäle	
<ul style="list-style-type: none"> Eingänge 	6; Über Arduino UNO Shields
Uhrzeitsynchronisation	
<ul style="list-style-type: none"> unterstützt 	Ja; Synchronisiert mit Systemuhr von I-oT2020/2040
Protokolle	
Offene IE-Kommunikation	
<ul style="list-style-type: none"> TCP/IP 	Ja
<ul style="list-style-type: none"> <ul style="list-style-type: none"> Anzahl Verbindungen, max. 	4; Von insgesamt 4 TCP/UDP Verbindungen
<ul style="list-style-type: none"> <ul style="list-style-type: none"> Datenlänge, max. 	16 kbyte
<ul style="list-style-type: none"> UDP 	Ja
<ul style="list-style-type: none"> <ul style="list-style-type: none"> Anzahl Verbindungen, max. 	4; Von insgesamt 4 TCP/UDP Verbindungen
<ul style="list-style-type: none"> <ul style="list-style-type: none"> Datenlänge, max. 	1 472 byte
Webserver	
<ul style="list-style-type: none"> anwenderdefinierte Webseiten 	Nein
<ul style="list-style-type: none"> Anzahl HTTP-Clients 	2; Von insgesamt 4 TCP/UDP Verbindungen
Kommunikationsfunktionen	
PG/OP-Kommunikation	Ja
S7-Kommunikation	
<ul style="list-style-type: none"> unterstützt 	Ja; Für Engineering, HMI
Test- Inbetriebnahmefunktionen	
Status Baustein	Ja
Einzelschritt	Ja
Anzahl Haltepunkte	16
Status/Steuern	
<ul style="list-style-type: none"> Status/Steuern Variable 	Ja
Diagnosepuffer	
<ul style="list-style-type: none"> vorhanden 	Ja
<ul style="list-style-type: none"> Anzahl Einträge, max. 	120
Hardware-Voraussetzung	
benötigte Hardware	IoT2020, IoT2040
Programmierung	
<ul style="list-style-type: none"> Klammerebenen 	8
Programmiersprache	
<ul style="list-style-type: none"> KOP 	Ja
<ul style="list-style-type: none"> FUP 	Ja
<ul style="list-style-type: none"> AWL 	Ja
<ul style="list-style-type: none"> SCL 	Ja
<ul style="list-style-type: none"> GRAPH 	Ja

Weiterführende Informationen

Unter folgenden Links finden Sie weiterführende Informationen:

- Gerätehandbuch der IOT20x0:
<https://support.industry.siemens.com/cs/document/109741658/simatic-iot2020-simatic-iot2040?dti=0&lc=de-WW>
<https://support.industry.siemens.com/cs/document/109741658/simatic-iot2020-simatic-iot2040?dti=0&lc=de-WW>
- Inbetriebnahme der IOT20x0:
<https://support.industry.siemens.com/tf/ww/en/posts/setting-up-the-simatic-iot2000/155642/?page=0&pageSize=10>
<https://support.industry.siemens.com/tf/ww/en/posts/setting-up-the-simatic-iot2000/155642/?page=0&pageSize=10>
- IOT20x0-Bereich im Siemens Industry Support Forum: <http://www.siemens.com/iot2000-forum> (<http://www.siemens.com/iot2000-forum>)
- IOT2020 Informationsseite: <https://siemens.com/iot2020> (<https://siemens.de/iot2020>)
- SD-Card Beispielimage:
<https://support.industry.siemens.com/cs/document/109741799/simatic-iot2000-sd-card-example-image> (<https://support.industry.siemens.com/cs/document/109741799/simatic-iot2000-sd-card-beispielimage?dti=0&lc=de-WW>)
- Anleitung zur Erstellung eines eigenen Images: <https://github.com/siemens/meta-iot2000> (<https://github.com/siemens/meta-iot2000>)
- Informationen zu den Yocto Variablen:
 - EXTRA_IMAGE_FEATURES: <http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#ref-features-image> (<http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#ref-features-image>)
 - PACKAGE_CLASSES: http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#var-PACKAGE_CLASSES (http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html#var-PACKAGE_CLASSES)
- MRAA library (Version 1.6.1): <https://github.com/intel-iot-devkit/mraa#installing-on-intel-32bit-yocto-based-opkg-image> (<https://github.com/intel-iot-devkit/mraa#installing-on-intel-32bit-yocto-based-opkg-image>)
- Detailinformationen zur Zuordnung der GPIOs:
<https://support.industry.siemens.com/tf/WW/en/posts/how-is-the-assignment-of-the-gpios/155609?page=0&pageSize=10>
<https://support.industry.siemens.com/tf/WW/en/posts/how-is-the-assignment-of-the-gpios/155609?page=0&pageSize=10>
- Informationen zum SIEMENS IOT2000 IO, Input/Output Module:
<https://support.industry.siemens.com/cs/document/109745681/iot2000-io-input-output-module?dti=0&lc=de-DE>
<https://support.industry.siemens.com/cs/document/109745681/iot2000-io-input-output-module?dti=0&lc=de-DE>

Abkürzungen

CPU	Central Processing Unit
GPIO	General Purpose Input/Output
HSP	Hardware Support Package
IOT	Internet of Things
IOT2000EDU	SIMATIC S7 Software Controller, IOT2000EDU (EDU für education)
Opkg	Open PacKaGe management
OS	Operating System
PLC	Programmable Logic Controller
PWM	Pulse Width Modulation
SSH	Secure Shell
TIA	Totally Integrated Automation