

SIEMENS

Ingenuity for life

Industry Online Support

Home

PROFdrive Applikationsbeispiel AC1

Applikationsbeispiel

<https://support.industry.siemens.com/cs/ww/de/view/109751271>

Siemens
Industry
Online
Support



Gewährleistung und Haftung

Hinweis

Die Anwendungsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bei typischen Aufgabenstellungen bieten. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Dieses Anwendungsbeispiel enthebt Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieses Anwendungsbeispiels erkennen Sie an, dass wir über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden können. Wir behalten uns das Recht vor, Änderungen an diesem Anwendungsbeispiel jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Anwendungsbeispiel und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Anwendungsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z. B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Weitergabe oder Vervielfältigung dieser Anwendungsbeispiele oder Auszüge daraus sind nicht gestattet, soweit nicht ausdrücklich von der Siemens AG zugestanden.

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 <http://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 <http://www.siemens.com/industrialsecurity>.

Inhaltsverzeichnis

Gewährleistung und Haftung	2
1 Vorwort	4
1.1 Zweck des Handbuchs	4
1.2 Zielgruppe des Handbuchs	4
2 Einleitung	5
3 PROFIdrive-Modell des Applikationsbeispiels	7
3.1 PROFINET-Gerätemodell nach PROFIdrive (siehe /1/ Kapitel 8.4)	8
3.2 Zyklische Daten (IO AR).....	9
3.3 Parameterkanal (BMP Protokoll über Record Data ASE).....	10
3.4 Alarme (Alarm ASE)	11
3.5 Zustandsmaschinen	11
3.6 Synchronisierung (Isochronous Mode Application ASE)	11
3.7 Applikation	11
4 Implementierung des Applikations-beispiels auf DevKit ERTEC 200P-2	12
4.1 Übersicht	12
4.2 Dateien für App41_PROFIdrive_AC1	13
4.3 usriod_main_pdrvac1.c	14
4.4 iodapi_event_pdrvac1.c	15
4.5 pdrv_application.c	16
4.6 pdrv_diagnostics.c	16
4.7 pdrv_pardatabase.c.....	17
4.8 pdrv_parmanager.c	17
4.9 pdrv_setpointchannel.c	18
4.10 pdrv_statemachine.c	18
4.11 GSDML-Datei	19
5 Adaption des Applikationsbeispiels	20
5.1 Identifikationsdaten	20
5.2 Diagnose	20
5.3 Parameter	21
5.4 Applikation	22
5.5 Kommunikationsstack	23
6 Anhang	24
6.1 Begriffe/Abkürzungen	24
6.2 Literaturverzeichnis	24
6.3 Funktionspläne	25

1 Vorwort

1.1 Zweck des Handbuchs

Die Anwenderdokumentation beschreibt die Software-Funktionalität des PROFIdrive Applikationsbeispiels der Applikationsklasse 1 (AC1) auf Basis des Evaluation Kit EK-ERTEC 200P-2 PN IO V4.4.0. Sie baut auf dem Programming and Operating Manual „Interface description PROFINET IO Development Kits V4.4.0“ /2/ des Evaluation Kits auf. Das PROFIdrive Applikationsbeispiel ist nach PROFIdrive V4.2 /1/ realisiert.

1.2 Zielgruppe des Handbuchs

Dieses Handbuch ist für Software- und für Applikationen-Entwickler gedacht, die das Evaluation Kit für neue Produkte mit dem PROFIdrive-Profil einsetzen wollen oder die Beispielimplementierung des PROFIdrive-Profiles auf eine andere PROFINET-Hardwareplattform portieren wollen.

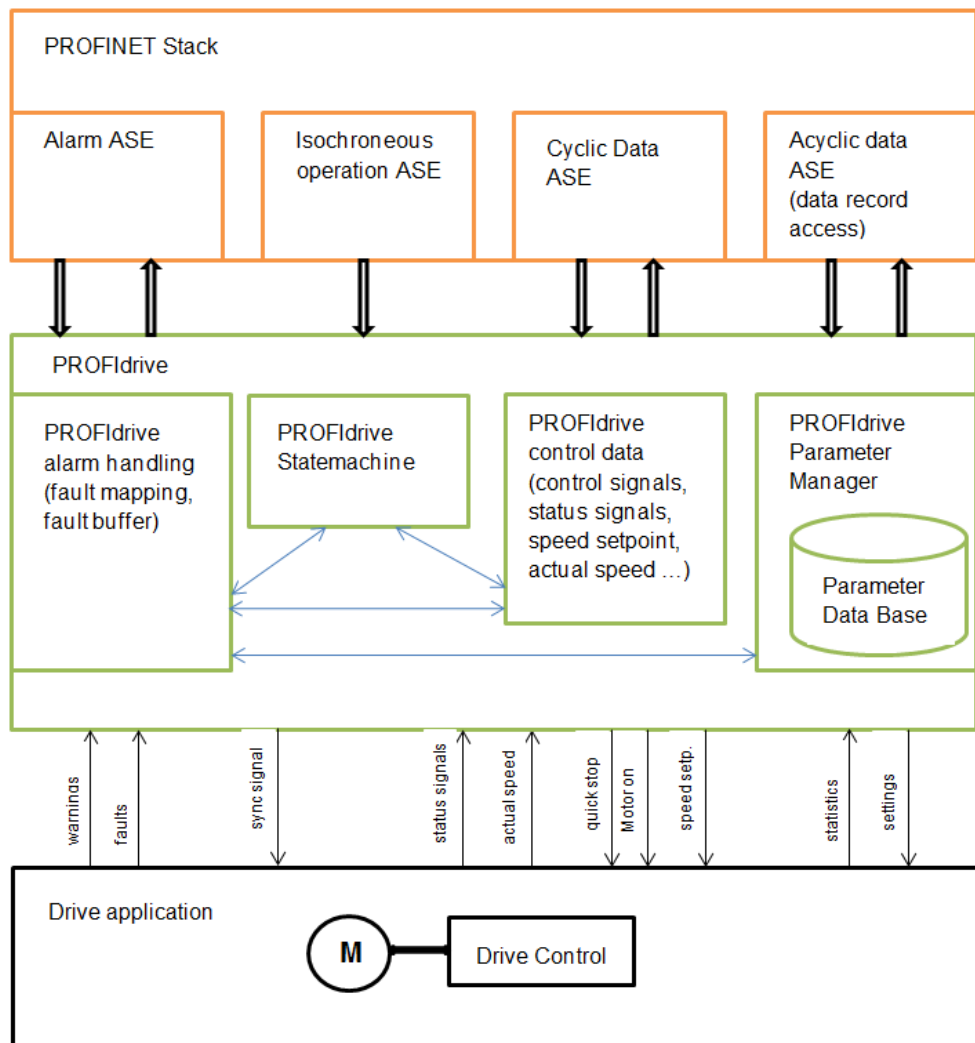
Dieses Handbuch gilt für ERTEC 200P-basierte Plattformen.

Gute PROFINET IO- und PROFIdrive-Kenntnisse werden vorausgesetzt, um den PROFINET-Firmwarestack und das PROFIdrive-Applikationsbeispiel anwenden zu können.

2 Einleitung

Das PROFIdrive-Profil ist das bewährte Geräteprofil für Antriebsgeräte auf Basis von PROFIBUS und PROFINET. Mit dem vorliegenden Applikationsbeispiel kann die Entwicklungszeit für die Realisierung einer PROFIdrive-Anbindung eines Antriebsgerätes verkürzt werden. Das Applikationsbeispiel baut auf dem PROFINET-Stack für das Evaluation Kit EK ERTEC 200P-2 auf. Es kann auch als Basis einer Portierung auf andere Plattformen und Kommunikationsstacks dienen. PROFIdrive ist eine normierte Applikationsschnittstelle für Antriebe und Encoder und in der IEC61800-7 sowie als Chinese Recommendatory National Standard GB/T 25740 standardisiert.

Abbildung 2-1 Übersicht: PROFIdrive (grün) als Anwendungsschicht zwischen PROFINET Stack (orange) und der Antriebsapplikation (schwarz) aus Antriebssicht



Der PROFINET-Stack ist der mit dem Evaluation Kit gelieferte Firmwarestack. PROFIdrive wird durch das hier vorgestellte Applikationsbeispiel implementiert. Die „Drive application“ wird im Applikationsbeispiel durch eine einfache Simulation (PT1-Glied) nachgebildet.

Die Funktionalität des PROFIdrive-Applikationsbeispiels beinhaltet:

- Realisierung eines Antriebsobjekts (ein P-Device bestehend aus genau einer Drive Unit und genau einem Drive Object) der Applikationsklasse (AC) 1 mit Drehzollsollwertschnittstelle.
- Zyklischer Datenaustausch über Standard Telegramm 1 (Steuerwort, 16-Bit-Drehzollsollwert, Zustandswort, 16-Bit-Drehzollwert) und Standard Telegramm 2 (Steuerwort, 32-Bit-Drehzollsollwert, Zustandswort, 32-Bit-Drehzollwert, kein Lebenszeichen)
- Azyklischer Datenaustausch über PROFIdrive Parameterkanal (BMP-Protokoll)
- PROFIdrive Diagnose
- Zustandsautomat PROFIdrive General State Machine
- Simulation einer Antriebsregelstrecke

Für das PROFIdrive Applikationsbeispiel existiert eine GSDML-Datei. Die GSDML-Datei basiert auf der GSDML-Datei des Evaluation Kit erweitert um Einträge für das PROFIdrive Applikationsbeispiel (PROFIdrive API).

Das Applikationsbeispiel wurde mit dem PROFIdrive PROFINET Profile Tester V50 getestet. Der PROFIdrive PROFINET Profile Tester ist das Testtool, welches die PI- Prüflabore für die PROFIdrive Zertifizierung verwenden. Der Profiltester kann auch zum entwicklungsbegleitenden Test bei der Entwicklung von PROFIdrive-Profil-Antrieben genutzt werden.

Link zum PROFIdrive PROFINET Profile Tester:

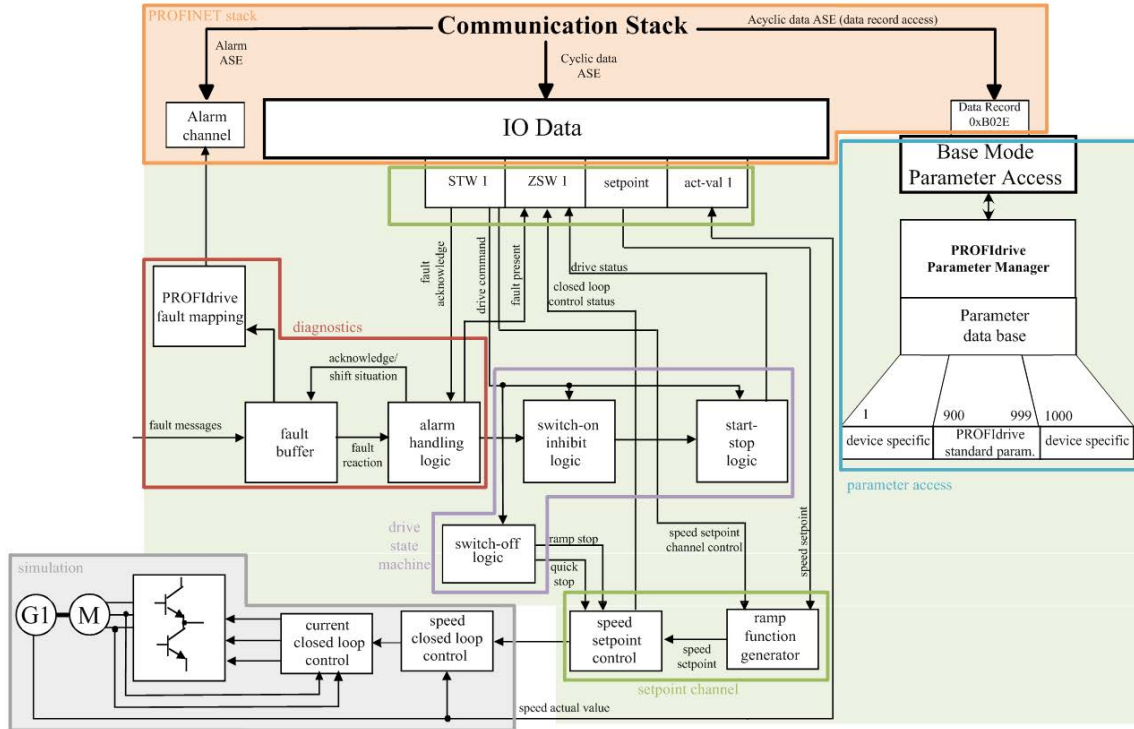
<https://www.profibus.com/download/profidrive-profinet-profile-tester/>

Als Planungshilfe für eine PROFIdrive-Implementierung kann der PROFIdrive Implementation Guide dienen.

<https://kb.hilscher.com/display/PROFIDRIVE/PROFIdrive+Implementation+Guide>

3 PROFIdrive-Modell des Applikationsbeispiels

Abbildung 3-1 Übersichtsbild der PROFIdrive Applikationsklasse AC1 „Drehzahl geregelter Antrieb“
Beispiel nach /1/ figure 136



3.1 PROFINET-Gerätemodell nach PROFIdrive (siehe /1/ Kapitel 8.4)

Das allgemeine PROFINET-Gerätemodell gliedert ein Gerät in Module und Submodule sowie Slots und Subslots. Slots und Subslots sind dabei als Platzhalter zu sehen, d.h. Module können in freie Slots bzw. Submodule in freie Subslots gesteckt werden. Die Struktur ist hierarchisch, d.h. ein Gerät beinhaltet ein oder mehrere Slots, ein Modul beinhaltet ein oder mehrere Subslots.

Slot 0 ist der Device Access Point (DAP) und beschreibt die Busanschaltung mit deren Eigenschaften. Die eigentliche Gerätefunktionalität (z.B. Eingänge/Ausgänge) wird ab Slot 1 in Subslots modelliert.

Tabelle 3-1 Logische Struktur des PROFIdrive P-Device im Applikationsbeispiel:

Slot 0 (API = 0)		Slot 1 (API = 0x3A00 PROFIdrive)		
Subslot 0		Subslot 0	Subslot 1	Subslot 2
			Module Access Point (MAP)	Standard-telegram 1 (Submodul-ID = PROFIdrive-Telegrammnummer)
			Enthält Parameter Access Point und Alarmkanal	
P-Device		Drive Object 1		

3.2 Zyklische Daten (IO AR)

Es sind die Standardtelegramme 1 und 2 nach PROFIdrive-Standard /1/ realisiert. Das Standardtelegramm 2 ist ohne Lebenszeichen implementiert, da Applikationsbeispiel noch keinen takt synchronen Betrieb unterstützt.

Tabelle 3-2 Standardtelegramm 1 nach /1/ Kapitel 6.3.4.3.2

IO Data number	Setpoint	Actual value
1	STW1	ZSW1
2	NSOLL_A	NIST_A

Das Standardtelegramm 1 hat eine Größe von 4 Octets (Bytes). STW1, NSOLL_A, ZSW1, NIST_A sind 16-Bit-Daten (Word). Input Data und Output Data sind jeweils 4 Bytes über denselben Subslot 2.

STW1 ist nach „Speed control mode“ /1/ Kapitel 6.3.2.2 realisiert. Die optionale Funktion Jog1/Jog2 ist nicht realisiert. Es sind keine gerätespezifischen Steuerbits implementiert.

ZSW1 ist wie in /1/ Kapitel 6.3.2.5 im „Speed control mode“ realisiert. Es sind keine gerätespezifischen Zustandsbits implementiert.

NSOLL_A und NIST_A haben den Datentyp N2 ($0x4000 \hat{=} 100\%$, siehe /1/ Kapitel 5.3.2). NIST_A liefert die Drehzahl der simulierten Regelstrecke (PT1-Glied) zurück.

Tabelle 3-3 Standardtelegramm 2 nach /1/ Kapitel 6.3.4.3.3

IO Data number	Setpoint	Actual value
1	STW1	ZSW1
2	NSOLL_B	NIST_B
3		
4	STW2	ZSW2

Das Standardtelegramm 2 hat eine Größe von 8 Octets (Bytes). STW1, STW2, ZSW1, ZSW2 sind 16-Bit-Daten. NSOLL_B, NIST_B sind 32-Bit-Daten. Input Data und Output Data sind jeweils 8 Bytes über denselben Subslot 2.

STW1 und ZSW1 sind wie beim Standardtelegramm 1 implementiert.

NSOLL_B und NIST_B haben den Datentyp N4 ($0x40000000 \hat{=} 100\%$, siehe /1/ Kapitel 5.3.2). NIST_B liefert die Drehzahl der simulierten Regelstrecke (PT1-Glied) zurück.

STW2 und ZSW2 sind ohne Lebenszeichen und ohne herstellerspezifische Daten implementiert. Sie sind somit derzeit komplett ungenutzt.

3.3 Parameterkanal (BMP Protokoll über Record Data ASE)

Es ist nur der mandatory PAP-Datensatz 0xB02E für den „Base Mode Parameter Access – Local“ realisiert. Der optionale Datensatz 0xB02F für den „Base Mode Parameter Access – Global“ ist nicht realisiert (wird auch nicht empfohlen). Weitere Datensätze sind von PROFIdrive nicht definiert und wurden nicht realisiert.

Es sind 3 Parameterkanalinstanzen implementiert. Damit können simultan 2 IO Controller und 1 IO Supervisor (Device Access) über jeweils separate AR zugreifen. Der Umfang von 2 IO Controller und 1 IO Supervisor ist durch das dem PROFIdrive Applikationsbeispiel zugrunde liegende PROFINET-IO-Applikationsbeispiel vorgegeben.

Tabelle 3-4 Liste realisierter Parameter:

PNU	Parameterbeschreibung	Wrt.	Impl.	Vali.
100 -200	Beispielhafte herstellerspezifische Parameter für die Parametrierung der Applikation (Parameter für Rampengenerator, Schwellwerte, Verstärkungsfaktoren der Regelstrecke)	rw	V	L
300	Beispielhafter Parameter zur Auswahl der Quelle für die Telegrammauswahl (PROFINET-Projektierung oder PROFIdrive-Parameter PNU00922)	rw	V	L
700 -819	Beispielhafte herstellerspezifischer Parameter für Tests mit dem PROFIdrive PROFINET Profile Tester	rw	V	L
900	Setpoint telegram (DO IO DATA)	ro	O	L
907	Actual value telegram (DO IO DATA)	ro	O	L
922	Telegram selection	rw	M	L
944	Fault message counter	ro	M	L
945	Fault Code	ro	O	L
947	Fault number	ro	M	L
948	Fault Time	ro	O	L
949	Fault Value	ro	O	L
950	Scaling of the fault buffer	ro	O	L
951	Fault number list with text	ro	O	L
952	Fault situation counter	rw	O	L
953	Warning parameters	ro	O	L
964	Drive Unit identification data block	ro	M	G
965	Profile Identification number	ro	M	L
974	Base Mode Parameter Access identification	ro	O	G
975	DO identification	ro	O	L
980	Number list of defined parameter	ro	M	L
60000	Velocity reference value	ro	M	L
61000	NameOfStation	ro	O	G
61001	IpOfStation	ro	O	G
61002	MacOfStation	ro	O	G
61003	DefaultGatewayOfStation	ro	O	G
61004	SubnetMaskOfStation	ro	O	G

Wrt.: ro – read only, rw – read-/writeable; Impl.: M – mandatory, O – optional, V – vendor specific; Vali.: L – local, G – Global

3.4 Alarme (Alarm ASE)

Alle Alarme sind lt. PROFIdrive optional. Es sind beispielhaft Alarme als Fehler (Fault) oder Warnungen (Warning) realisiert. Die Fehler/Warnungen werden im Applikationsbeispiel zur Demonstration über Parameter PNU00700-PNU00702 gesetzt und rückgesetzt.

3.5 Zustandsmaschinen

Es ist die Hauptzustandsmaschine nach /1/ Kapitel 6.3.3.1 realisiert.

3.6 Synchronisierung (Isochronous Mode Application ASE)

Die Synchronisierung ist nicht ein mandatory Bestandteil der Anwendungsklasse 1 (AC1) und ist deshalb im Applikationsbeispiel nicht realisiert.

3.7 Applikation

Die Regelstrecke (Regler, Wechselrichter, Motor, Geber) wird als PT1-Glied simuliert.

4 Implementierung des Applikations-beispiels auf DevKit ERTEC 200P-2

4.1 Übersicht

Das PROFIdrive_AC1-Beispiel ist nach PROFIdrive V4.2 /1/ realisiert.

Das PROFIdrive_AC1-Beispiel setzt die Installation des Firmwarestack für das Evaluation Kit voraus (siehe /2/ und /3/). Es baut auf dem Applikationsbeispiel „App1_STANDARD“ auf, welches unter

<InstallPath>\pnio_src\application\App1_STANDARD abgelegt ist (siehe /2/ Kapitel 2.4 „Application examples“).

Um das PROFIdrive_AC1-Beispiel mit dem Evaluation Kit zu nutzen, wird folgende Vorgehensweise vorgeschlagen:

1. Vollständige Inbetriebnahme des Evaluation Kit wie in /3/ Kapitel 4 „Quickstart Guide“ beschrieben.
2. Kopieren der PROFIdrive-Quelldateien aus
<SRC_PROFIdrive_AC1_example.zip>\pnio_src\application\App41_PROFIdrive_AC1
nach
<InstallPath>\pnio_src\application\App41_PROFIdrive_AC1
3. Die Auswahl des Applikationsbeispiels erfolgt in der Header-Datei
usrapp_cfg.h in
<InstallPath>\pnio_src\application\App_common
mittels folgendem Eintrag:
#define EXAMPL_DEV_CONFIG_VERSION 41
4. Neuerstellen des Projektes und Laden des PROFIdrive_AC1-Beispiles in das Evaluation Kit

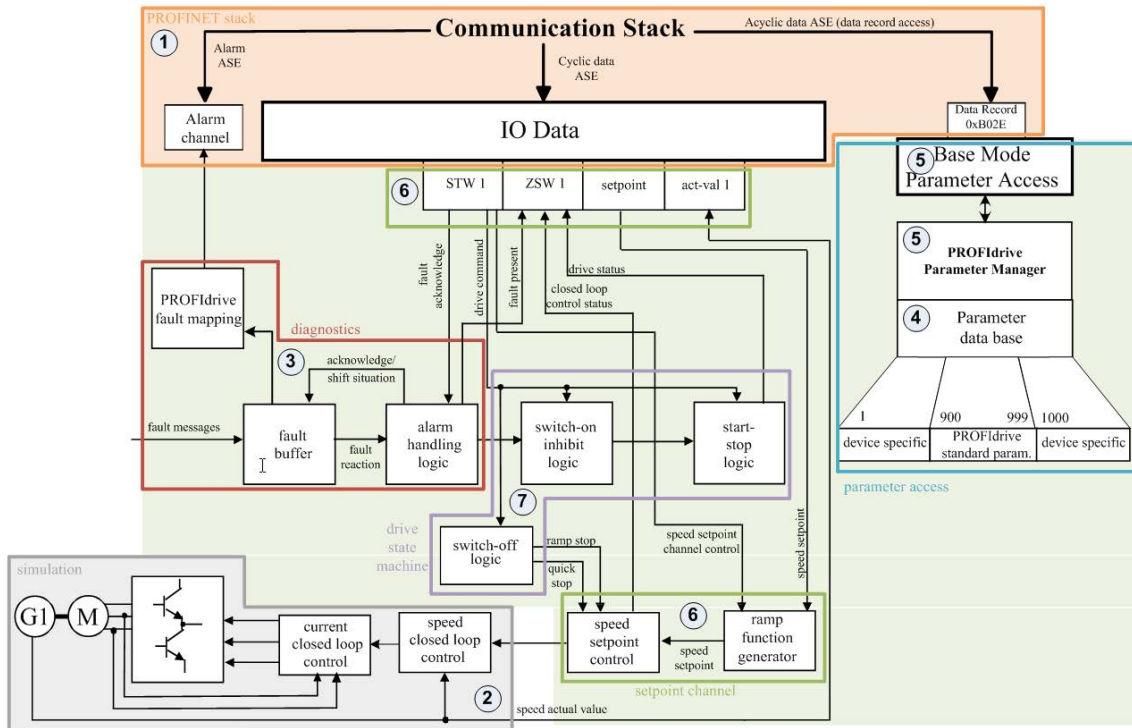
Für die Projektierung eines PROFINET-Projektes liegt dem Beispiel die modifizierte GSDML-Datei in <SRC_PROFIdrive_AC1_example.zip>\GSDML bei.

4.2 Dateien für App41_PROFIdrive_AC1

Tabelle 4-1 Kurzübersicht über die Dateien des Applikationsbeispiels:

Datei	Kurzbeschreibung	Bild-Nr.
usriod_main_pdrvac1.c	Hauptprogramm des PROFINET-Beispiels Anlauf des PROFINET-IO-Stacks, Initialisierung der PROFIdrive-Applikation, Initialisierung und Start des 1ms-Timers für die PROFIdrive-Applikation, Hauptschleife mit Start von Funktionen über Tastatur von PROFINET-Funktionen (z.B. Firmwareupdate)	①
iodapi_event_pdrvac1.c	PROFINET-Schnittstelle mit Meldungen von PROFINET-Ereignissen an die Applikation Enthält Funktionen, die der PROFINET-IO- Stack beim Auftreten von Ereignissen wie Verbindungsauf-/abbau, Alarmempfang etc. aufruft und damit der Applikation mitteilt.	①
PnUsr_Api_pdrvac1.c	PROFINET-Unterprogramme für das Anwenderbeispiel. Die enthaltenen Funktionen können bei Bedarf als Funktionsbibliothek in der Kundenapplikation verwendet werden.	①
pdrv_application.c pdrv_application.h	PROFIdrive-Applikation 1ms-Zeitscheibe mit Aufruf der anderen PROFIdrive-Module, Simulation der Regelstrecke, Terminalausgaben bei PROFIdrive-Ereignissen, Umsetzung der Applikationsparameter PNU100-PNU819	②
pdrv_diagnostics.c pdrv_diagnostics.h	PROFIdrive Diagnose mit Stör- und Warnungsbehandlung	③
pdrv_parameter.inc	Makrodatei mit Liste der realisierten PROFIdrive-Parametern	④
pdrv_pardatabase.c pdrv_pardatabase.h	PROFIdrive-Parameterdatenbank Erstellt aus pdrv_parameter.inc die Parameterdatenbank	④
pdrv_parmanager.c pdrv_parmanager.h	PROFIdrive-Parametermanager Realisiert den PROFIdrive Base Mode Parameter Access	⑤
pdrv_setpointchannel.c pdrv_setpointchannel.h	PROFIdrive-Sollwertkanal Implementiert Zugriff auf die Signale der Standard Telegramme 1 u. 2; Auswahl der Sollwertquelle abhängig vom Hauptzustand, Rampengenerator	⑥
pdrv_statemachine.c pdrv_statemachine.h	PROFIdrive-Hauptzustandsautomat	⑦
pdrv_types.h	Definitionen von PROFIdrive-Datentypen	

Abbildung 4-1 Zuordnung der Quelldateien zu den Funktionalitäten (siehe Tabelle)



4.3 usriod_main_pdrvac1.c

Das Modul ist auf Basis

<InstallPath>\pnio_src\application\App1_STANDARD\usriod_main.c

entstanden. Es beinhaltet den Anlauf des PROFIdrive-IO-Stacks, die Initialisierung der PROFIdrive-Applikation, die Initialisierung und Start des 1ms-Timers für die PROFIdrive-Applikation und Hauptschleife mit Start von PROFIdrive-Funktionen (z.B. Firmwareupdate) über Tastatur.

Außerdem sind hier die Funktionalitäten der Parameter PNU61000-PNU61004 mit Parameter-Leseroutinen `uPdrv_RfPnu61000()` bis `uPdrv_RfPnu61002()` realisiert.

4.4 iodapi_event_pdrv1.c

Das Modul ist auf Basis

```
<InstallPath>\pnio_src\application\App1_STANDARD\iodapi_event
.c
```

entstanden. Es enthält Funktionen, die der PROFINET-IO-Stack beim Auftreten von Ereignissen wie Verbindungsauf-/abbau, Alarmempfang etc. aufruft und damit der Applikation mitteilt.

In der Funktion `PNIO_cbf_data_write()` werden die Istwerte ZSW1, ZSW2, NIST_A, NIST_B aus dem PROFIdrive-Sollwertkanal in den PROFINET-Stack übergeben. Hierbei wird die unterschiedliche Byte-Reihenfolge (Endianess, ERTEC Little Endian, PROFINET Big Endian) behandelt.

In der Funktion `PNIO_cbf_data_read()` werden die Sollwerte STW1, STW2, NSOLL_A, NSOLL_B aus dem PROFINET-Stack in den PROFIdrive-Sollwertkanal übergeben. Hierbei wird die unterschiedliche Byte-Reihenfolge (Endianess, ERTEC Little Endian, PROFINET Big Endian) behandelt.

In der Funktion `PNIO_cbf_ar_connect_ind()` wird der PROFINET-AR-Verbindungsaufbau dem PROFIdrive-Parametermanager mitgeteilt, der daraufhin einen Parameterkanal für diese Verbindung reserviert.

In der Funktion `PNIO_cbf_ar_disconn_ind()` wird der PROFINET-AR-Verbindungsabbau dem PROFIdrive-Parametermanager mitgeteilt, der daraufhin den Parameterkanal für diese Verbindung schließt und den Parameterkanal freigibt.

In der Funktion `PNIO_cbf_ar_indata_ind()` wird der erstmalige Empfang von Sollwerten vom PROFINET-Stack dem PROFIdrive-Sollwertkanal mitgeteilt.

In der Funktion `PnUsr_cbf_rec_read()` wird die Leseanforderung des PROFIdrive Base Mode Parameter Access Records zum PROFIdrive-Parametermanager weitergereicht, welcher die Leseanforderung entsprechend bearbeitet und den Inhalt des Records liefert.

In der Funktion `PnUsr_cbf_rec_write()` wird die Schreibanforderung PROFIdrive Base Mode Parameter Access Records zum PROFIdrive-Parametermanager weitergereicht, welcher die Schreibanforderung entsprechend bearbeitet und eine entsprechende Antwort liefert.

Es sind die Funktionalitäten der Parameter PNU00300 „Source telegram selection“, PNU00922 „Telegram selection“, PNU00900 „Setpoint telegram“ und PNU00907 „Actual value telegram“ realisiert. Im Parameter PNU00300 wird die Quelle für die Telegrammauswahl eingestellt. Bei PNU00300=0 (Voreinstellungswert) erfolgt die Telegrammauswahl laut PROFINET-Projektierung (Expected Configuration Data) in der Funktion `PNIO_cbf_data_write()`. Bei PNU00300=1 wird die Telegrammauswahl über Parameter PNU00922 in der Parameter-Schreibroutine `uPdrv_WfPnu00922()` vorgenommen.

4.5 pdrv_application.c

Das Modul ist ein Beispiel für die Antriebsapplikation. Sie besteht aus der Funktion `PdrvApp_main()`, welche durch das Betriebssystem zyklisch alle 1ms aufgerufen wird. In dieser Funktion werden zyklisch die PROFIdrive-Funktionen wie PROFIdrive-Hauptzustandsautomat, Rampengenerator, Schwellwertberechnungen, Sollwertkanal, Regelstreckensimulation, Istwertübergabe und Parametermanager aufgerufen. Für Demonstrationszwecke werden berechnete Daten ausgegeben, welche über die serielle Schnittstelle des Evaluation Kit in einem Terminalprogramm angezeigt werden können.

Im Modul sind weiterhin applikationsabhängige PROFIdrive-Funktionen wie Stillstandserkennung `bPdrvApp_IsAxisStandstill()`, Transitionsbestätigung für den PROFIdrive-Hauptzustandsautomaten `bPdrvApp_IsTransitionCondition()` und ein 1s-Timer `uPdrvApp_GetTimer1s()` implementiert.

Des Weiteren sind hier die Parameter-Leseroutinen und Parameter-Schreibroutinen für die Applikationsparameter und für die Testparameter zum Test mit dem Profile-Tester implementiert.

4.6 pdrv_diagnostics.c

Das Modul implementiert PROFIdrive-Diagnose-Mechanismen (siehe /1/ Kapitel 6.3.8 „Diagnosis“). Es sind sowohl der PROFIdrive-„Complete Fault Buffer“ (siehe /1/ Table 114 „Fault buffer parameters“) als auch die PROFIdrive-Warnungen (siehe /1/ Kapitel 6.3.8.2 „Warning mechanism“) realisiert.

Tabelle 4-2 Liste realisierter Parameter des „Fault Buffer“ und „Warning Mechanism“:

PNU	Parameterbeschreibung	Wrt.	Impl.	Vali.
944	Fault message counter	ro	M	L
945	Fault Code	ro	O	L
947	Fault number	ro	M	L
948	Fault Time	ro	O	L
949	Fault Value	ro	O	L
950	Scaling of the fault buffer	ro	O	L
951	Fault number list with text	ro	O	L
952	Fault situation counter	rw	O	L
953	Warning parameters	ro	O	L

Wrt.: ro – read only, rw – read-/writeable; Impl.: M – mandatory, O – Optional; Vali.: L – local, G – Global

Zum Setzen eines Fehlers und Eintrag in den PROFIdrive-Fehlerpuffer wird die Funktion `PdrvDiag_SetFaultMsg()` verwendet. Dabei wird auch über den Standard-PROFINET-Alarm-Mechanismus eine entsprechende Meldung abgesetzt. Die Funktion `PdrvDiag_AckFaultSit()` quittiert einen Störfall und wird bei einer positiven Flanke des STW1 Bits 7 und beim Zurücksetzen des Parameters `PNU00952=0` aufgerufen (siehe /1/ figure 61 „Fault acknowledgement for the fault buffer mechanism“).

Die Funktionen `bPdrvDiag_IsFaultOff1()`, `bPdrvDiag_IsFaultOff2()`, `bPdrvDiag_IsFaultOff3()` liefern zurück, ob ein Fehler mit der entsprechenden Fehlerreaktion „Power down“, „Coast stop“ oder „Quick stop“ aktiv ist.

Zum Setzen und Rücksetzen einer Warnung wird die Funktion `PdrvDiag_SetWarning()` verwendet. Dabei wird auch über den Standard-PROFINET-Alarm-Mechanismus eine entsprechende Meldung abgesetzt.

Des Weiteren sind hier die Parameter-Leseroutinen und Parameter-Schreibroutinen für die oben aufgeführten Parameter implementiert.

4.7 pdrv_pardatabase.c

Dieses Modul realisiert die Parameterdatenbank, in welcher die in Datei `pdrv_parameter.inc` definierten Parameter mittels Makros in C-Strukturen angelegt werden.

Die Funktion `ptPdrvPar_GetParObj()` sucht in der Parameterdatenbank nach einem Parameterobjekt und liefert im Erfolgsfall einen Zeiger auf das Parameterobjekt.

Ein Parameterobjekt beinhaltet neben den Eigenschaften, welche in /1/ Kapitel 6.2.1.3 „Parameter description“ aufgeführt werden, jeweils 3 verschiedene Zugriffsfunktionen. Die Parameter-Leseroutine wird beim lesenden Parameter-Zugriff aufgerufen und füllt den Antwortpuffer mit den angeforderten, aktuellen Parameterwerten. Die Schreibroutine wird beim schreibenden Zugriff aufgerufen und holt aus dem Auftragspuffer die zuschreibenden Werte und trägt diese in den dem Parameter zugehörigen Speicher ein. Die Textroutine wird bei Anforderung eines zusätzlichen Textes (siehe /1/ Kapitel 6.2.1.4 „Text“) aufgerufen und liefert einen Zeiger auf den angeforderten Text zurück.

Des Weiteren sind hier die Parameter-Leseroutinen und Parameter-Schreibroutinen für verschiedene Parameter implementiert.

4.8 pdrv_parmanager.c

Dieses Modul realisiert den PROFIdrive Base Mode Parameter Access (siehe /1/ figure 131).

Beim Aufbau einer PROFINET-Verbindung wird die Funktion `bPdrvPar_EstablishCxn()` aufgerufen. Diese prüft, ob für die PROFINET-Verbindung ein PROFIdrive-Parameterkanal verfügbar ist und reserviert einen PROFIdrive-Parameterkanal für die weitere Verwendung mit dieser PROFINET-Verbindung. Dabei wird der PROFIdrive-Parameterkanal zurückgesetzt. Wird dieselbe PROFINET-Verbindung geschlossen, so wird die Funktion `bPdrvPar_DisconnCxn()` aufgerufen und der PROFIdrive-Parameterkanal ebenfalls geschlossen und für eine spätere Verwendung freigegeben. Derzeit sind simultan 3 PROFIdrive-Parameterkanäle verfügbar (einstellbar in Konstante `PARCXN_NR`). Die Beschränkung auf 3 PROFIdrive-Parameterkanäle basiert auf der Beschränkung des verwendeten PROFINET-Evaluation-Kit EK200P-2 V4.4. Damit können maximal 3 simultane PROFINET-Verbindungen gehalten werden, davon maximal 2 Controller-Verbindungen sowie eine Device-Access-Verbindung. Hinweis: Mit der Firmware V4.5 des EK 200P-2 Evaluation-Kits können bis zu 5 simultane PROFINET-Verbindungen gehalten werden, davon maximal 4 Controller-Verbindungen sowie eine Device-Access-Verbindung.

Bei Empfang von Datensatzanforderungen für den Base-Mode-Parameter-Access-Local-Datensatzes (0xB02E) werden entsprechend die Funktionen `uPdrvPar_WriteReqCxn()` oder `uPdrvPar_ReadReqCxn()` aufgerufen, welche entsprechend des Zustands des zugehörigen PROFIdrive-Parameterkanals die Antwort auf diese Datensatzanforderung erstellen (siehe /1/ table 33). Bei einem gültigen `Write.req` wird der PROFIdrive-Parameternauftrag zur weiteren Bearbeitung in den Request-Puffer übernommen. Bei einem gültigen `Read.req` wird die Antwort aus dem Response-Puffer übernommen.

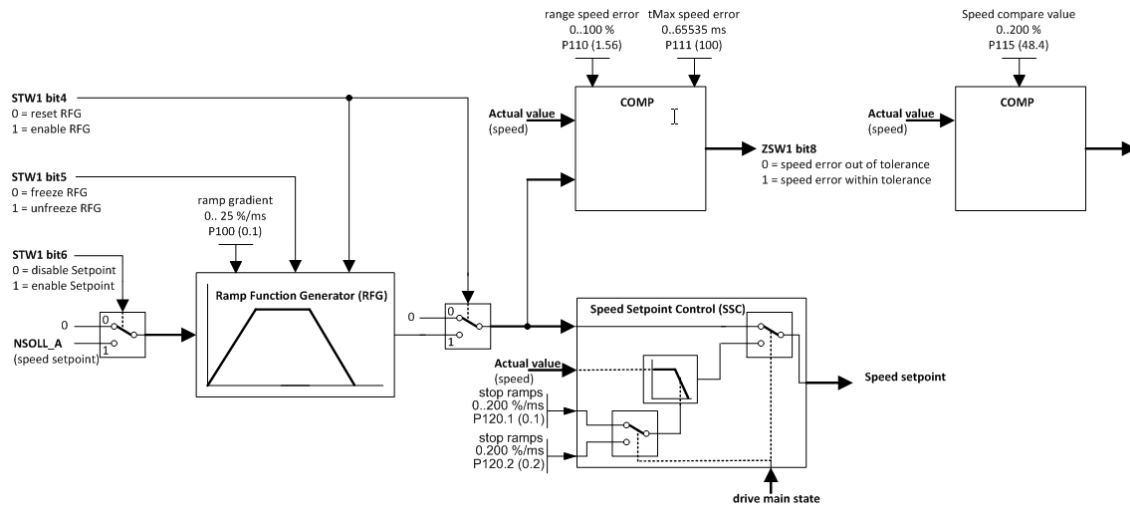
Die Funktion `PdrvPar_ProcessReq()` wird zyklisch aufgerufen. Bei jedem Aufruf überprüft die Funktion, ob in einem der Parameterkanäle ein zu bearbeitender Parameternauftrag vorliegt und der Parameternauftrag abgearbeitet und das Ergebnis in den Response-Puffer abgelegt. Liegen in mehreren Parameterkanälen zu bearbeitende Parameternaufträge vor, so werden die Parameternaufträge sequenziell nach dem Round-Robin-Verfahren abgearbeitet.

Bei der Bearbeitung der Parameternaufträge wird die unterschiedliche Byte-Reihenfolge (Endianess, ERTEC Little Endian, PROFINET Big Endian) behandelt.

4.9 pdrv_setpointchannel.c

Dieses Modul implementiert den Sollwertkanal. Es werden hier die zyklischen Sollwerte der Standardtelegramme für die weitere Verwendung gespeichert und mit get-Funktionen zur Verfügung gestellt.

Abbildung 4-2 Rampengenerator nach /1/ figure 29 (keine Jog-Funktionalität)



Der Rampengenerator (siehe /1/ figure 29) ist ohne Jogging-Funktionalität in der Funktion `nPdrvSpc_CalcRfg()` implementiert. Die Funktion `nPdrvSpc_CalcSsc()` realisiert eine Geschwindigkeitssollwertsteuerung, welche je nach Zustand des Hauptzustandsautomaten entweder den Rampengenerator-Sollwert unverändert durchreicht oder eine AUS-Rampe mit normaler oder Schnellstopp-Geschwindigkeit realisiert.

4.10 pdrv_statemachine.c

Die Funktion `ePdrvSma_AxisGeneralStateMachine()` realisiert den Hauptzustandsautomaten, welcher in /1/ figure 27 mit dem Zustandsgraphen beschrieben ist. Der aktuelle Zustand des Hauptzustandsautomaten kann mit der Funktion `ePdrvSma_GetAxisMainState()` abgefragt werden.

4.11 GSDML-Datei

Die GSDML-Datei des Evaluation Kit EK-ERTEC 200P-2 PN IO V4.4.0 wurde um das PROFIdrive-Modul erweitert. Somit erscheint das PROFIdrive-Modul in derselben Kategorie wie die anderen IO-Module des Evaluation Kits.

Abbildung 4-3 Beispiel für Hardwarekonfiguration im Programm NCM PC (Profile Tester)

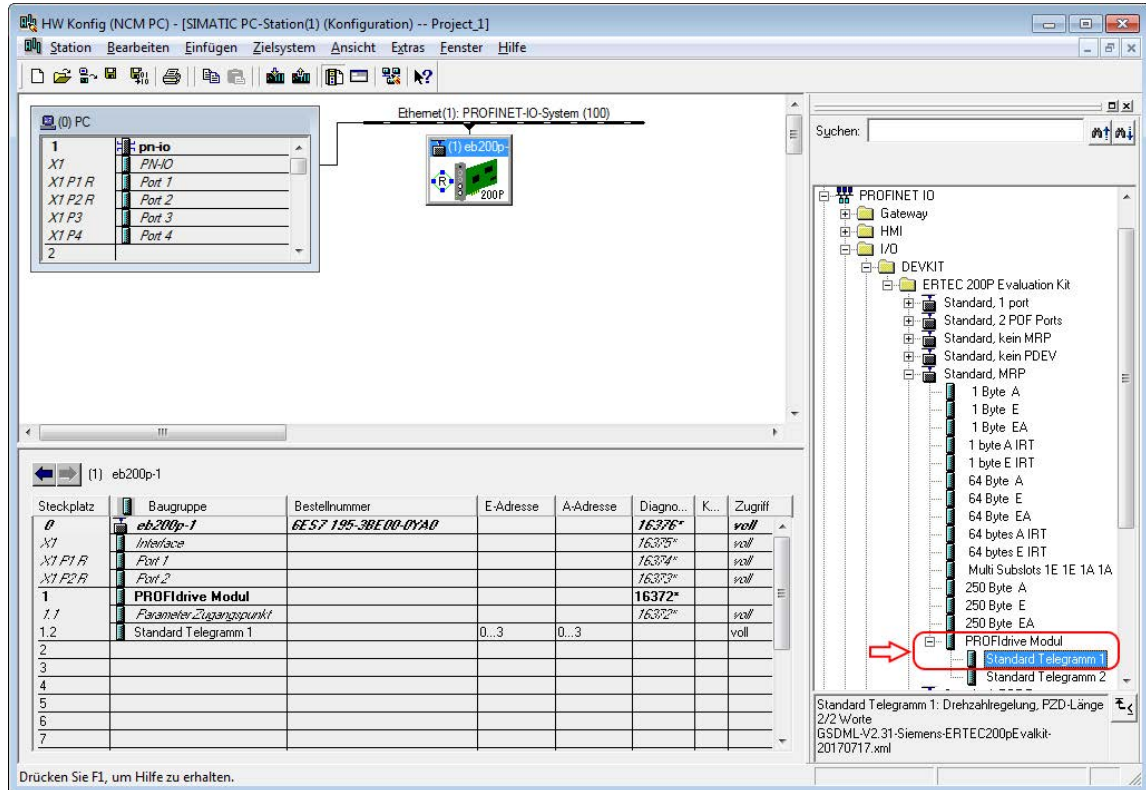
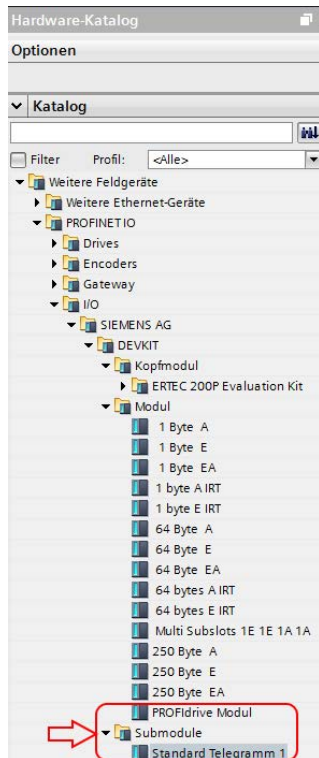


Abbildung 4-4 PROFIdrive-Beispiel im Hardwarekatalog des TIA-Portals



5 Adaption des Applikationsbeispiels

5.1 Identifikationsdaten

Zusätzlich zu den Änderungen des PROFINET-Stacks, welche in /3/ chapter 7.2.1 beschrieben sind, sind im Anwendungsbeispiel folgende Identifikationsdaten anzupassen.

Die Modul-ID `PDRV_MODULE_ID_MAP` ist in `pdrv_application.h` anzupassen (herstellerspezifisch).

Die Identifikationsdaten sind in der Datei `pdrv_application.h` anzupassen (siehe hierzu auch Parameter PNU00964, PNU00975 in /1/ table 118 und table 120)

Tabelle 5-1 Identifikationsdaten

Konstante	Beschreibung
<code>PDRV_ID_MANUFACTURER</code>	Herstellercode lt. PNO
<code>PDRV_ID_DUTYPE</code>	Drive Unit Type (herstellerspezifisch)
<code>PDRV_ID_DOTYPE</code>	Drive Object Type (herstellerspezifisch)
<code>PDRV_ID_FWVERSION</code>	Firmwareversion
<code>PDRV_ID_FWDATE_Y</code>	Jahr des Firmwaredatums
<code>PDRV_ID_FWDATE_DM</code>	Tag und Monat des Firmwaredatums

5.2 Diagnose

Die beispielhaft implementierten Fehler und Warnungen sind zu entfernen und die in der neuen Applikation vorkommenden Fehler und Warnungen zu implementieren.

Hierzu sind

- in den Dateien `pdrv_diagnostics.*` die Einträge `FAULT_USER0` bis `FAULT_USER9` und `WARNING_USER0` bis `WARNING_USER9` zu entfernen
- in Datei `pdrv_parameter.inc` die Parameter PNU00700 bis PNU00702 und in `pdrv_application.c` die dazugehörigen Implementierungen (Variablen, Funktionen) zu entfernen (Beispiel für Test mit dem Profile Tester)
- neue Warnungen sind einer Fehlerklasse der PROFINET-Standard-Alarm-Funktionalität zuzuordnen und in der Tabelle `m_tWarningAttribs[]` zu hinterlegen. Bei mehr als 16 Warnungen sind die Parameter PNU00954 bis PNU00960 zu implementieren.
- neue Fehler werden in der Tabelle `m_tFaultAttribs[]` hinterlegt. Dazu sind einem Fehler
 - * ein Fehlercode (PNU00945)
 - * eine Fehlernummer (PNU00947)
 - * eine Fehlerreaktion (OFF1 \triangleq Motor AUS, OFF2 \triangleq Impulssperre, OFF3 \triangleq Schnellhalt, keine)
 - * eine Fehlerklasse der PROFINET-Standard-Alarm-Funktionalität
 - * ein Fehlertext (16 Zeichen, PNU00951)
 zuzuordnen.
- in der neuen Applikation sind entsprechend die Funktionen `PdrvDiag_SetFaultMsg()` und `PdrvDiag_SetWarning()` aufzurufen

Es ist zu prüfen, ob bei der neuen Applikation eine Konsistenzsicherung bei der Diagnosebearbeitung erforderlich ist. In der Beispielapplikation ist keine Konsistenzsicherung erforderlich, da alle Softwareteile sequenziell mit derselben Priorität abgearbeitet werden.

5.3 Parameter

Die beispielhaft implementierten Parameter sind zu entfernen und die in der neuen Applikation vorkommenden Parameter zu implementieren.

Hierzu sind

- in Datei `pdrv_parameter.inc` die Parameter PNU00100 bis PNU00200 und in `pdrv_application.c` die dazugehörigen Implementierungen (Variablen, Funktionen) zu entfernen (Parameter der Beispielapplikation)
- in Datei `pdrv_parameter.inc` die Parameter PNU00700 bis PNU00819 und in `pdrv_application.c` die dazugehörigen Implementierungen (Variablen, Funktionen) zu entfernen (Parameter für Test mit dem Profile Tester)
- in Datei `pdrv_parameter.inc` der Parameter PNU00300 zu entfernen und in `iodapi_event_pdrvac1.c` sich für eine der beiden Varianten zur Telegrammauswahl zu entscheiden. Nach Möglichkeit ist die Telegrammauswahl lt. PROFINET-Projektierung zu wählen. Parameter PNU00922 zeigt in dem Fall das projektierte Telegramm nur noch an.
- in Datei `pdrv_parameter.inc` die Parameter PNU00900 bis PNU00999 den veränderten Bedingungen anzupassen
- für die neue Applikation sind die applikationsspezifischen Parameter zu implementieren

Jeder Parameter benötigt eine Leseroutine `pfnRead()`. Schreibbare Parameter benötigen zusätzlich eine Schreibroutine `pfnWrite()`. Wenn ein Parameter zusätzliche Texte enthält, dann ist eine Textroutine `pfnText()` erforderlich.

Hinweis: Es wird empfohlen, die Parameterdatenbasis automatisiert aus einer eigenen Parameterdatenbank zu generieren. Die in der Beispielapplikation gewählte Makro-Methode erschwert das Finden von Syntaxfehlern und formalen Fehlern in der Parameterdatenbasis, welche in der Datei `pdrv_pardatabase.c` realisiert ist.

Die Beispielapplikation hat für Parameter keine remanente Speicherung und damit auch kein explizites Zurücksetzen auf Werkseinstellung implementiert.

Folgende weitere Einstellmöglichkeiten bzgl. der Parameterbearbeitung sind vorgesehen:

Tabelle 5-2 Weitere Einstellmöglichkeiten

Konstante	Beschreibung
<code>PDRV_PAR_BLOCKSIZE</code>	Datenblockgröße einer Parameteranforderung/-antwort (siehe /1/ Kapitel 6.2.3.2)
<code>PARCXN_NR</code>	Anzahl der Parameterkanäle (simultan möglich Parameterverbindungen) , bei Änderung ist der PROFINET-Stack entsprechend anzupassen

5.4 Applikation

Die Beispielapplikation in Datei `pdrv_application.c` besteht nur aus einer sehr einfachen Simulation der gesamten Regelstrecke (Stellglied, Motor, Messwerterfassung) mittels eines PT1-Gliedes. Diese Simulation ist durch die eigene Applikation zu ersetzen.

Die Applikation hat die folgenden PROFIdrive-Funktionen in geeigneter Reihenfolge mit korrekten Parametern aufzurufen:

- Hauptzustandsautomat `ePdrvSma_AxisGeneralStateMachine()`
- Rampengenerator `nPdrvSpc_CalcRfg()`
- Sollwertkanal `nPdrvSpc_CalcSsc()`
- Toleranzberechnung für ZSW1 Bit 8
`bPdrvSpc_CalcSpeedWithinTolerance()`
- Vergleich für ZSW1 Bit 10 `bPdrvSpc_CalcSpeedReached()`
- Istwertübergabe in Standardtelegramm 1 `PdrvSpc_SetNistA()`
- Parametermanager `PdrvPar_ProcessReq()`

Typischerweise werden mit Ausnahme des Parametermanagers die Funktionen äquidistant mit der gleichen Priorität wie die eigentliche Antriebsapplikation abgearbeitet. Der Parametermanager wird oft mit einer niedrigeren Priorität und nicht unbedingt äquidistant abgearbeitet. Im vorliegenden Beispiel werden alle Funktionen mit derselben Priorität alle 1ms abgearbeitet, weshalb im Beispiel auf Konsistenzsicherungen verzichtet wurde. Wenn eine Applikation mit unterschiedlichen Ablaufprioritäten realisiert wird, so sind an erforderlichen Stellen geeignete Maßnahmen (Interruptsperrern, Wechsellpuffer) zur Konsistenzsicherung zu implementieren.

Weiterhin müssen folgende Funktionen implementiert sein:

- Stillstandserkennung `bPdrvApp_IsAxisStandstill()`
(verwendet im Hauptzustandsautomaten)
- Transitionsbestätigung `bPdrvApp_IsTransitionCondition()`
(verwendet im Hauptzustandsautomaten)
- 1s-Timer `uPdrvApp_GetTimer1s()`
(verwendet beim Fehlerpuffer PNU00948, optional)

Die Transitionsbestätigung `bPdrvApp_IsTransitionCondition()` dient dazu, die Zustandsweitschaltung des Hauptzustandsautomaten durch die Applikation zu verzögern, falls die Applikation noch nicht für den Folgezustand bereit ist. Dies ist jeweils für die Hauptzustandswechsel von S1 Einschaltsperr nach S2 Einschaltbereit, von S2 Einschaltbereit nach S3 Betriebsbereit, von S3 Betriebsbereit nach S4 Betrieb möglich (siehe /1/ figure 27 „General State Diagram“).

5.5 Kommunikationsstack

Beim Wechsel des Kommunikationsstack sind die Module `iodapi_event_pdrvac1.c` und `usriod_main_pdrvac1.c` anzupassen bzw. auszutauschen.

PROFIdrive nutzt/benötigt folgende Funktionen:

- Kommunikationsaufbau `PNIO_cbf_ar_connect_ind()`
- Kommunikationstrennung `PNIO_cbf_ar_disconn_ind()`
- Erkennung des ersten erfolgreichen Kommunikationsaustausches `PNIO_cbf_ar_indata_ind()`
- Senden zyklischer Daten `PNIO_cbf_data_write()`
- Empfang zyklischer Daten `PNIO_cbf_data_read()`
- Datensatzlesen `PnUsr_cbf_rec_read()`
- Datensatzschreiben `PnUsr_cbf_rec_write()`
- Kanaldiagnose `uPdrvUsr_ChanDiag()`
- Ziehen und Stecken eines Submoduls `uPdrv_WfPnu00922()` (bei Verwendung mehrere Standardtelegramme, optional)
- PROFINET und TCP/IP-Stationsdaten `uPdrv_RfPnu61000()` - `uPdrv_RfPnu61002()` (optional)

In den Modulen `iodapi_event_pdrvac1.c` und `usriod_main_pdrvac1.c` lassen sich die zu für PROFIdrive übernehmenden Codestellen einfach mit einer Suche nach der Zeichenkette „Pdrv“ finden.

Bei Änderungen der Prozessor-Plattform oder des Kommunikationsstacks ist zu beachten, dass der ERTEC 200P vom Evaluation Kit im LittleEndian-Format arbeitet und PROFINET das BigEndian-Format verwendet.

6 Anhang

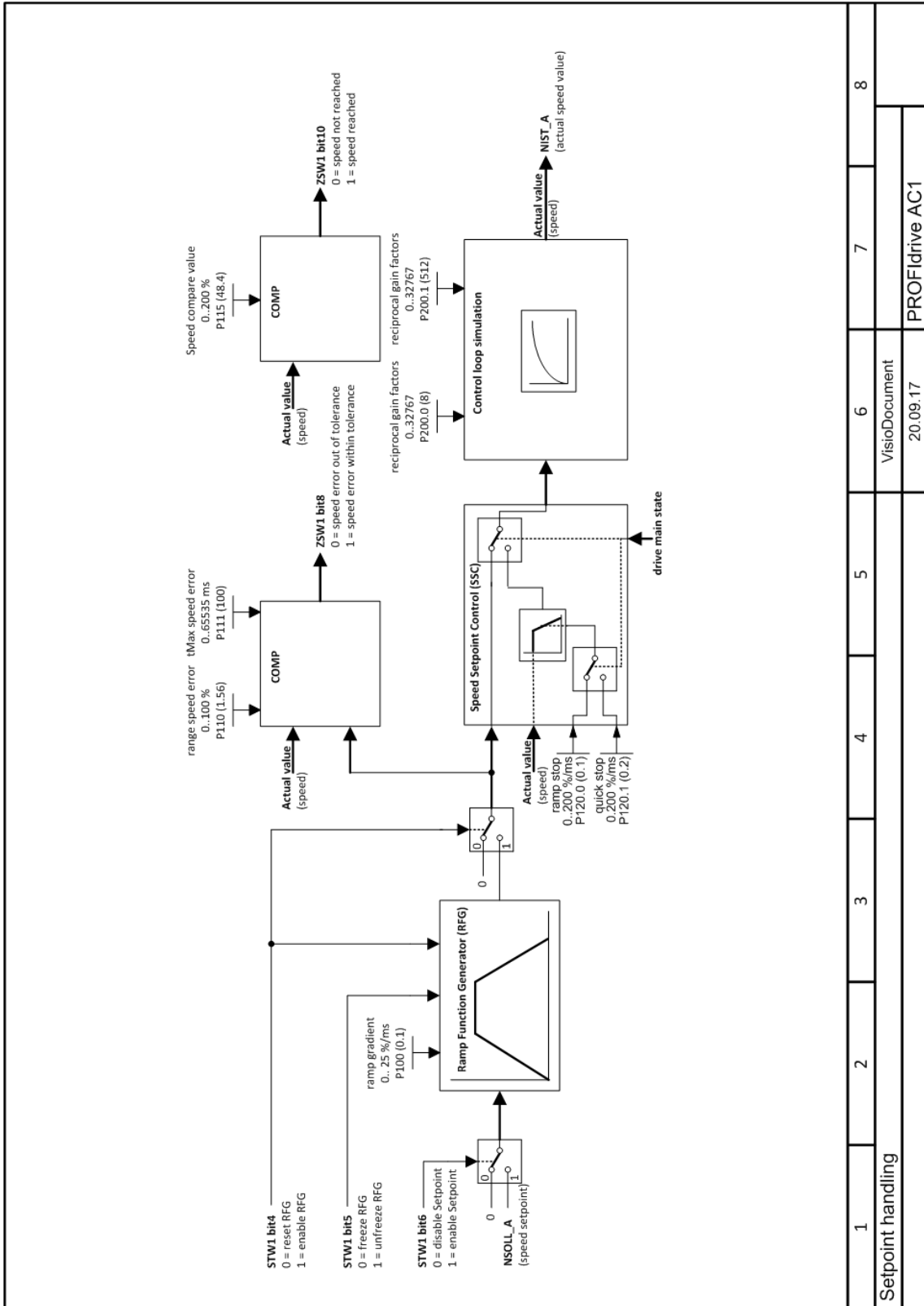
6.1 Begriffe/Abkürzungen

API	Application Interface
API	Application Process Identifier
AC	Applikationsklasse (PROFIdrive)
ASE	Application Service Element (PROFIdrive)
DAP	Drive Access Point (PROFINET)
DO	Drive Object (PROFIdrive)
DU	Drive Unit (PROFIdrive)
MAP	Module Access Point (PROFIdrive)
PAP	Parameter Access Point (PROFIdrive)
P-Device	Peripheral device (PROFIdrive)
PNU	Parameter NUmmer (PROFIdrive)

6.2 Literaturverzeichnis

/Nr./	"Autor": "Titel", "Verlag", "Erscheinungsort", "Jahr", "Seitenangabe"
/1/	PROFIBUS Nutzerorganisation e.V.: Profile Drive Technology PROFIdrive Version 4.2 October 2015
/2/	Siemens AG: „Interface description PROFINET IO Development Kits V4.4.0“ Programming and Operating Manual, A5E33638878-AC, 11/2016
/3/	Siemens AG: „Guidelines for Evaluation Kit ERTEC 200P-2 V4.4.0“ Programming and Operating Manual, A5E03855331-AC, 11/2016

6.3 Funktionspläne



1	2	3	4	5	6	7	8
Setpoint handling							
VisioDocument						PROFIdrive AC1	
20.09.17							

