

Magelis SCU

HMI-Steuerung

PLCSystem-Bibliothekshandbuch

12/2016



EIO0000001248.03

www.schneider-electric.com

Schneider
Electric

Die Informationen in der vorliegenden Dokumentation enthalten allgemeine Beschreibungen und/oder technische Leistungsmerkmale der hier erwähnten Produkte. Diese Dokumentation dient keinesfalls als Ersatz für die Ermittlung der Eignung oder Verlässlichkeit dieser Produkte für bestimmte Verwendungsbereiche des Benutzers und darf nicht zu diesem Zweck verwendet werden. Jeder Benutzer oder Integrator ist verpflichtet, angemessene und vollständige Risikoanalysen, Bewertungen und Tests der Produkte im Hinblick auf deren jeweils spezifischen Verwendungszweck vorzunehmen. Weder Schneider Electric noch deren Tochtergesellschaften oder verbundene Unternehmen sind für einen Missbrauch der Informationen in der vorliegenden Dokumentation verantwortlich oder können diesbezüglich haftbar gemacht werden. Verbesserungs- und Änderungsvorschläge sowie Hinweise auf angetroffene Fehler werden jederzeit gern entgegengenommen.

Dieses Dokument darf ohne entsprechende vorhergehende, ausdrückliche und schriftliche Genehmigung durch Schneider Electric weder in Teilen noch als Ganzes in keiner Form und auf keine Weise, weder anhand elektronischer noch mechanischer Hilfsmittel, reproduziert oder fotokopiert werden.

Bei der Montage und Verwendung dieses Produkts sind alle zutreffenden staatlichen, landesspezifischen, regionalen und lokalen Sicherheitsbestimmungen zu beachten. Aus Sicherheitsgründen und um die Übereinstimmung mit dokumentierten Systemdaten besser zu gewährleisten, sollten Reparaturen an Komponenten nur vom Hersteller vorgenommen werden.

Beim Einsatz von Geräten für Anwendungen mit technischen Sicherheitsanforderungen sind die relevanten Anweisungen zu beachten.

Die Verwendung anderer Software als der Schneider Electric-eigenen bzw. einer von Schneider Electric genehmigten Software in Verbindung mit den Hardwareprodukten von Schneider Electric kann Körperverletzung, Schäden oder einen fehlerhaften Betrieb zur Folge haben.

Die Nichtbeachtung dieser Informationen kann Verletzungen oder Materialschäden zur Folge haben!

© 2016 Schneider Electric. Alle Rechte vorbehalten.



	Sicherheitshinweise	5
	Über dieses Buch	7
Kapitel 1	HMI SCU - Systemvariablen	13
1.1	Systemvariablen: Definition und Verwendung	14
	Grundlegendes zu Systemvariablen	15
	Verwenden von Systemvariablen	16
1.2	PLC_R- und PLC_W-Strukturen	18
	PLC_RPLC_R: Schreibgeschützte Steuerungssystemvariablen	19
	PLC_W: Steuerungssystemvariablen mit Lese- und Schreibzugriff ...	21
1.3	SERIAL_W-Strukturen	22
	SERIAL_W[0]: Systemvariablen für serielle Leitung mit Lese- und Schreibzugriff	22
Kapitel 2	HMI SCU-Systemfunktionen	23
2.1	HMI SCU-Funktionen für den Lesezugriff	24
	GetBatteryLevel: Gibt die verbleibende Batterieladung zurück ..	25
	GetLocalAIOSStatus: Gibt den Status der integrierten Analog-E/A zurück	26
	GetLocalIOSStatus: Gibt den integrierten E/A-Status aus	27
	GetShortCutStatus: Gibt den Kurzschlussstatus integrierter Ausgänge zurück.	29
	GetTempStatus: Die Funktion gibt den Temperatureingangsstatus aus.	30
	IsFirstMastColdCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Kaltstartzyklus handelt	31
	IsFirstMastCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Zyklus handelt	32
	IsFirstMastWarmCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Warmstartzyklus handelt	33

Kapitel 3	HMI SCU PLCSystem-Bibliothek - Datentypen	35
3.1	Datentypen der Systemvariablen in PLC_R/W	36
	PLC_R_APPLICATION_ERROR: Fehlerstatuscodes erkannter Anwendungsfehler	37
	PLC_R_BOOT_PROJECT_STATUS: Statuscodes für Boot-Projekt	38
	PLC_R_IO_STATUS: E/A-Statuscodes	39
	PLC_R_STATUS: Codes für den Steuerungsstatus	40
	PLC_R_STOP_CAUSE: Codes für den Übergang vom RUN-Status in anderen Status	41
	PLC_W_COMMAND: Codes für Steuerbefehle	42
3.2	Daten vom Typ Systemfunktion	43
	FIRMWARE_VERSION: Funktion GetFirmwareVersion Ausgangstyp	44
	BOOT_PROJECT_STATUS: GetBootProjectStatus Funktionsausgangscodes	45
	STOP_WHY: GetLastStopCause Funktionsausgangscodes	46
	LOCAL_IO_GET_STATUS: GetLocalIOStatus Parametercodes der Funktion	47
	LOCAL_IO_GEN_STATUS: Ausgangscodes der Funktion GetLocalIOStatus	48
Anhang		49
Anhang A	Darstellung von Funktionen und Funktionsbausteinen	51
	Unterschiede zwischen einer Funktion und einem Funktionsbaustein	52
	Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache AWL	53
	Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache ST	57
Glossar		61
Index		65



Wichtige Informationen

HINWEISE

Lesen Sie sich diese Anweisungen sorgfältig durch und machen Sie sich vor Installation, Betrieb, Bedienung und Wartung mit dem Gerät vertraut. Die nachstehend aufgeführten Warnhinweise sind in der gesamten Dokumentation sowie auf dem Gerät selbst zu finden und weisen auf potenzielle Risiken und Gefahren oder bestimmte Informationen hin, die eine Vorgehensweise verdeutlichen oder vereinfachen.



Wird dieses Symbol zusätzlich zu einem Sicherheitshinweis des Typs „Gefahr“ oder „Warnung“ angezeigt, bedeutet das, dass die Gefahr eines elektrischen Schlags besteht und die Nichtbeachtung der Anweisungen unweigerlich Verletzung zur Folge hat.



Dies ist ein allgemeines Warnsymbol. Es macht Sie auf mögliche Verletzungsgefahren aufmerksam. Beachten Sie alle unter diesem Symbol aufgeführten Hinweise, um Verletzungen oder Unfälle mit Todesfälle zu vermeiden.



GEFAHR

GEFAHR macht auf eine gefährliche Situation aufmerksam, die, wenn sie nicht vermieden wird, Tod oder schwere Verletzungen **zur Folge hat**.



WARNUNG

WARNUNG macht auf eine gefährliche Situation aufmerksam, die, wenn sie nicht vermieden wird, Tod oder schwere Verletzungen **zur Folge haben kann**.



VORSICHT

VORSICHT macht auf eine gefährliche Situation aufmerksam, die, wenn sie nicht vermieden wird, leichte Verletzungen **zur Folge haben kann**.

HINWEIS

HINWEIS gibt Auskunft über Vorgehensweisen, bei denen keine Verletzungen drohen.

BITTE BEACHTEN

Elektrische Geräte dürfen nur von Fachpersonal installiert, betrieben, bedient und gewartet werden. Schneider Electric haftet nicht für Schäden, die durch die Verwendung dieses Materials entstehen.

Als qualifiziertes Fachpersonal gelten Mitarbeiter, die über Fähigkeiten und Kenntnisse hinsichtlich der Konstruktion und des Betriebs elektrischer Geräte und deren Installation verfügen und eine Schulung zur Erkennung und Vermeidung möglicher Gefahren absolviert haben.

Über dieses Buch



Auf einen Blick

Ziel dieses Dokuments

In dieser Dokumentation werden die in der HMI SCU-Steuerung enthaltenen Funktionen und Variablen beschrieben. Die HMI SCU PLCSystem-Bibliothek enthält Funktionen und Variablen, um Informationen abzurufen und Befehle an das Steuerungssystem zu senden.

In dieser Dokumentation werden die Datentypen, Funktionen und Variablen der HMI SCU PLCSystem-Bibliothek beschrieben.

Wir setzen folgendes Grundwissen voraus:

- Grundlegende Kenntnisse der Funktionalität, Struktur und Konfiguration des HMI SCU
- Kenntnisse der Programmiersprachen FBD, LD, ST, AWL (IL) oder CFC
- Systemvariablen (globale Variablen)

Gültigkeitsbereich

Diese Dokumentation wurde für die SoMachine-Version V4.2 aktualisiert.

Weiterführende Dokumentation

Titel der Dokumentation	Referenz-Nummer
Magelis SCU HMI Controller - Programmierhandbuch	EIO0000001240 (ENG), EIO0000001241 (FRE), EIO0000001242 (GER), EIO0000001243 (SPA), EIO0000001244 (ITA), EIO0000001245 (CHS)
Magelis SCU - HMI-Steuerung - HSC-Bibliothekshandbuch	EIO0000001512 (ENG), EIO0000001513 (FRE), EIO0000001514 (GER), EIO0000001515 (SPA), EIO0000001516 (ITA), EIO0000001517 (CHS)
Magelis SCU HMI Controller - PTO/PWM-Bibliothekshandbuch	EIO0000001518 (ENG), EIO0000001519 (FRE), EIO0000001520 (GER), EIO0000001521 (SPA), EIO0000001522 (ITA), EIO0000001523 (CHS)

Titel der Dokumentation	Referenz-Nummer
PLCCommunication-Bibliothekshandbuch	EIO0000000361 (ENG), EIO0000000742 (FRE), EIO0000000743 (GER), EIO0000000744 (SPA), EIO0000000745 (ITA), EIO0000000746 (CHS)
Magelis SCU HMI Controller - Hardwarehandbuch	EIO0000001232 (ENG), EIO0000001233 (FRE), EIO0000001234 (GER), EIO0000001235 (SPA), EIO0000001236 (ITA), EIO0000001237 (CHS), EIO0000001238 (POR)

Diese technischen Veröffentlichungen sowie andere technische Informationen stehen auf unserer Website <http://www.schneider-electric.com/ww/en/download> zum Download bereit.

WARNUNG

STEUERUNGS AUSFALL

- Bei der Konzeption von Steuerungsstrategien müssen mögliche Störungen auf den Steuerpfaden berücksichtigt werden, und bei bestimmten kritischen Steuerungsfunktionen ist dafür zu sorgen, dass während und nach einem Pfadfehler ein sicherer Zustand erreicht wird. Beispiele kritischer Steuerungsfunktionen sind die Notabschaltung (Not-Aus) und der Nachlauf-Stopp, Stromausfall und Neustart.
- Für kritische Steuerungsfunktionen müssen separate oder redundante Steuerpfade bereitgestellt werden.
- Systemsteuerungspfade können Kommunikationsverbindungen umfassen. Dabei müssen die Auswirkungen unerwarteter Sendeverzögerungen und Verbindungsstörungen berücksichtigt werden.
- Sämtliche Unfallverhütungsvorschriften und lokale Sicherheitsrichtlinien sind zu beachten.¹
- Jede Implementierung des Geräts muss individuell und sorgfältig auf einen einwandfreien Betrieb geprüft werden, bevor das Gerät an Ort und Stelle in Betrieb gesetzt wird.

Die Nichtbeachtung dieser Anweisungen kann Tod, schwere Verletzungen oder Sachschäden zur Folge haben.

¹ Weitere Informationen finden Sie in den aktuellen Versionen von NEMA ICS 1.1 „Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control“ sowie von NEMA ICS 7.1, „Safety Standards for Construction and Guide for Selection, Installation, and Operation of Adjustable-Speed Drive Systems“ oder den entsprechenden, vor Ort geltenden Vorschriften.

WARNUNG

UNBEABSICHTIGTER GERÄTEBETRIEB

- Verwenden Sie mit diesem Gerät nur von Schneider Electric genehmigte Software.
- Aktualisieren Sie Ihr Anwendungsprogramm jedes Mal, wenn Sie die physische Hardwarekonfiguration ändern.

Die Nichtbeachtung dieser Anweisungen kann Tod, schwere Verletzungen oder Sachschäden zur Folge haben.

Terminologie gemäß den geltenden Standards

Die technischen Begriffe, Terminologien, Symbole und zugehörigen Beschreibungen, die in diesem Handbuch oder auf dem Produkt selbst verwendet werden, werden im Allgemeinen von den Begriffen oder Definitionen internationaler Standards abgeleitet.

Im Bereich der funktionalen Sicherheitssysteme, Antriebe und allgemeinen Automatisierungssysteme betrifft das unter anderem Begriffe wie *Sicherheit, Sicherheitsfunktion, Sicherer Zustand, Fehler, Fehlerreset/Zurücksetzen bei Fehler, Ausfall, Störung, Warnung/Warmmeldung, Fehlermeldung, gefährlich/gefahrbringend* usw.

Nachstehend einige der geltenden Standards:

Norm	Beschreibung
EN 61131-2:2007	Speicherprogrammierbare Steuerungen, Teil 2: Betriebsmittelanforderungen und Prüfungen.
ISO 13849-1:2008	Sicherheit von Maschinen: Sicherheitsbezogene Teile von Steuerungen. Allgemeine Gestaltungsleitsätze
EN 61496-1:2013	Sicherheit von Maschinen: Berührungslos wirkende Schutzeinrichtungen. Teil 1: Allgemeine Anforderungen und Prüfungen
ISO 12100:2010	Sicherheit von Maschinen – Allgemeine Gestaltungsleitsätze – Risikobeurteilung und Risikominderung
EN 60204-1:2006	Sicherheit von Maschinen – Elektrische Ausrüstungen von Maschinen – Teil 1: Allgemeine Anforderungen
EN 1088:2008 ISO 14119:2013	Sicherheit von Maschinen – Verriegelungseinrichtungen in Verbindung mit trennenden Schutzeinrichtungen – Leitsätze für Gestaltung und Auswahl
ISO 13850:2006	Sicherheit von Maschinen – Not-Halt – Gestaltungsleitsätze
EN/IEC 62061:2005	Sicherheit von Maschinen – Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbar elektronischer Steuerungssysteme
IEC 61508-1:2010	Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme: Allgemeine Anforderungen
IEC 61508-2:2010	Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme: Anforderungen an sicherheitsbezogene elektrische/elektronische/programmierbare elektronische Systeme
IEC 61508-3:2010	Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme: Anforderungen an Software
IEC 61784-3:2008	Industrielle Kommunikationsnetze – Profile – Teil 3: Funktional sichere Übertragung bei Feldbussen
2006/42/EC	Maschinenrichtlinie
2014/30/EU	EMV-Richtlinie (Elektromagnetische Verträglichkeit)
2014/35/EU	Niederspannungsrichtlinie

Darüber hinaus wurden einige der in diesem Dokument verwendeten Begriffe unter Umständen auch anderen Normen entnommen, u. a.:

Norm	Beschreibung
Normenreihe IEC 60034	Rotierende elektrische Geräte
Normenreihe IEC 61800	Elektrische Leistungsantriebssysteme mit einstellbarer Drehzahl
Normenreihe IEC 61158	Industrielle Kommunikationsnetze – Feldbus für industrielle Steuerungssysteme

Bei einer Verwendung des Begriffs *Betriebsumgebung/Betriebsbereich* in Verbindung mit der Beschreibung bestimmter Gefahren und Risiken entspricht der Begriff der Definition von *Gefahrenbereich* oder *Gefahrenzone* in der *Maschinenrichtlinie (2006/42/EC)* der Norm *ISO 12100:2010*.

HINWEIS: Die vorherig erwähnten Standards können auf die spezifischen Produkte in der vorliegenden Dokumentation zutreffen oder nicht. Für weitere Informationen hinsichtlich individueller Standards, die auf hier beschriebene Produkte zutreffen, siehe die Eigenschaftstabellen der hier erwähnten Produkte.

Kapitel 1

HMI SCU - Systemvariablen

Überblick

Dieses Kapitel:

- bietet eine Einführung in die Systemvariablen (*siehe Seite 14*).
- beschreibt die Systemvariablen (*siehe Seite 22*), die in der HMI SCU PLCSystem-Bibliothek enthalten sind.

Inhalt dieses Kapitels

Dieses Kapitel enthält die folgenden Abschnitte:

Abschnitt	Thema	Seite
1.1	Systemvariablen: Definition und Verwendung	14
1.2	PLC_R- und PLC_W-Strukturen	18
1.3	SERIAL_W-Strukturen	22

Abschnitt 1.1

Systemvariablen: Definition und Verwendung

Überblick

Dieser Abschnitt enthält eine Definition der Systemvariablen und beschreibt, wie diese im Magelis SCU HMI Controller implementiert werden.

Inhalt dieses Abschnitts

Dieser Abschnitt enthält die folgenden Themen:

Thema	Seite
Grundlegendes zu Systemvariablen	15
Verwenden von Systemvariablen	16

Grundlegendes zu Systemvariablen

Einführung

In diesem Abschnitt wird die Implementierung von Systemvariablen für die Steuerung beschrieben. Diese Variablen haben folgenden Attribute:

- Systemvariablen erlauben Ihnen den Zugriff auf allgemeine Systeminformationen, das Durchführen einer Systemdiagnose und das Befehlen einfacher Aktionen.
- Bei den Systemvariablen handelt es sich um strukturierte Variablen mit IEC 61131-3-konformen Definitionen und Namenskonventionen. Sie können mit dem symbolischen IEC-Namen `PLC_GVL` auf die Systemvariablen zugreifen.
- Einige `PLC_GVL`-Variablen sind schreibgeschützt (z. B. `PLC_R`) und für andere besteht Lese- und Schreibzugriff (z. B. `PLC_W`).
- Systemvariablen werden automatisch als globale Variablen deklariert. Sie wirken sich systemweit aus und müssen mit Umsicht verwendet werden, da von jeder POU (Program Organization Unit) in einem beliebigen Task darauf zugegriffen werden kann.

Namenskonvention für Systemvariablen

Systemvariablen sind an folgenden Eigenschaften erkennbar:

- An einem Strukturnamen, der die Kategorie der Systemvariablen darstellt (z. B. ist `PLC_R` ein Strukturname für schreibgeschützte Variablen, die zur Diagnose der Steuerung verwendet werden).
- An einem Satz von Komponentennamen, die auf den Zweck der Variablen verweisen (z. B. steht `i_wVendorID` für die Hersteller-ID (Vendor-ID) der Steuerung).

Um auf die Variablen zuzugreifen, geben Sie den Strukturnamen der Variablen gefolgt von dem Komponentennamen ein.

Im Folgenden finden Sie ein Beispiel für die Implementierung der Systemvariablen:

```
VAR myCtr_Serial : DWORD; myCtr_ID : DWORD; myCtr_FramesRx : UDINT;
END_VAR
```

```
myCtr_Serial := PLC_R.i_dwSerialNumber; myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

HINWEIS: Der vollständig qualifizierte Name der Systemvariablen im obigen Beispiel lautet `PLC_GVL.PLC_R.i_wVendorID`. Der Teil `PLC_GVL` ist implizit, wenn eine Variable mittels der **Eingabehilfe** deklariert wird, kann aber auch vollständig eingegeben werden.

Speicherort der Systemvariablen

Eine Art Systemvariable, die für die Verwendung definiert wird, wenn die Steuerung mit nicht lokalisierten Variablen programmiert wird.

Diese nicht lokalisierten Variablen sind nur über eine Anmeldung oder die Freigabe des Symbols **Symbolkonfiguration** für die HMI zugänglich.

Verwenden von Systemvariablen

Einführung

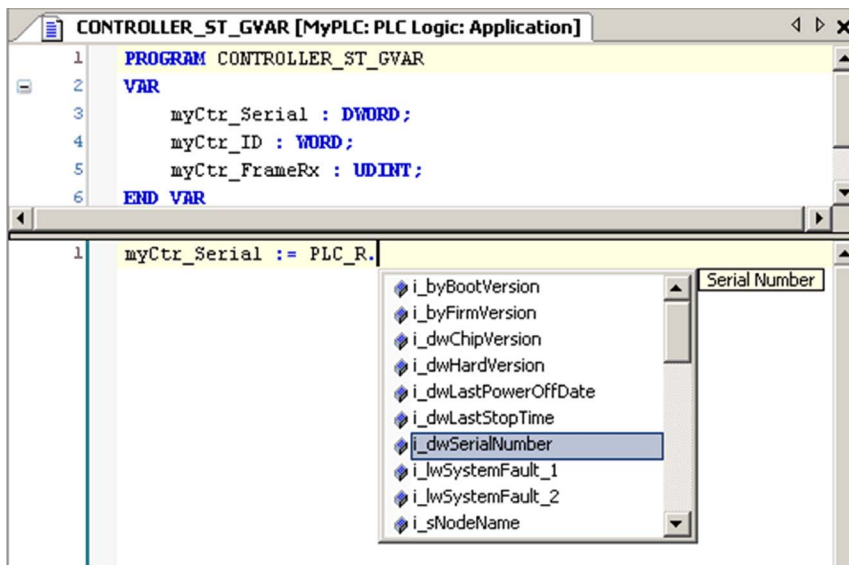
In diesem Abschnitt werden die verschiedenen Schritte zur Programmierung und Verwendung von Systemvariablen in SoMachine beschrieben.

Systemvariablen wirken sich global aus und können in allen POU (Program Organization Units) der Anwendung verwendet werden.

Systemvariablen müssen nicht in der GVL (Globale Variablenliste) deklariert werden. Sie werden automatisch über die Systembibliothek der Steuerung deklariert.

Verwenden von Systemvariablen in einer POU

SoMachine enthält eine Funktion zur automatischen Vervollständigung. Geben Sie in der **POU** zunächst den Strukturnamen der Systemvariablen (PLC_R, PLC_W usw.), gefolgt von einem Punkt ein. Die Systemvariablen erscheinen in der **Eingabehilfe**. Sie können die gewünschte Variable auswählen oder den vollen Namen manuell eingeben.



HINWEIS: Sobald Sie im Beispiel oben den Strukturnamen PLC_R. eingegeben haben, zeigt SoMachine ein Popup-Menü mit möglichen Komponentennamen/Variablen an.

Beispiel

Das folgende Beispiel veranschaulicht die Verwendung einiger Systemvariablen:

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : WORD;
    myCtr_FramesRx : UDINT;
END_VAR

myCtr_Serial := PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

Abschnitt 1.2

PLC_R- und PLC_W-Strukturen

Überblick

In diesem Abschnitt werden die verschiedenen Systemvariablen aufgelistet und beschrieben, die in den Strukturen `PLC_R` und `PLC_W` enthalten sind.

Inhalt dieses Abschnitts

Dieser Abschnitt enthält die folgenden Themen:

Thema	Seite
<code>PLC_R</code> / <code>PLC_R</code> : Schreibgeschützte Steuerungssystemvariablen	19
<code>PLC_W</code> : Steuerungssystemvariablen mit Lese- und Schreibzugriff	21

PLC_RPLC_R: Schreibgeschützte Steuerungssystemvariablen

Variablenstruktur

In der folgenden Tabelle werden die Parameter der Systemvariablen PLC_R (Typ PLC_R_STRUCT) beschrieben:

Variablenname	Typ	Kommentar
i_wVendorID	WORD	Steuerungshersteller-ID 101A hex = Schneider Electric
i_wProductID	WORD	Steuerungsreferenz-ID HINWEIS: Hersteller-ID und Referenz-ID sind Teil der Ziel-ID der Steuerung, die in der Ansicht mit den Kommunikationseinstellungen angezeigt wird (Ziel-ID = 101A XXXX hex).
i_dwSerialNumber	DWORD	Seriennummer der Steuerung (Rückgabe der letzten fünf Stellen).
i_byFirmVersion[0..3]	ARRAY[0..3] OF BYTE	Firmware-Version der Steuerung [aa.bb.cc.dd]: • i_byFirmVersion[0]= aa • ... • i_byFirmVersion[3]= dd
i_byBootVersion[0..3]	ARRAY[0..3] OF BYTE	Boot-Version der Steuerung [aa.bb.cc.dd]: • i_byBootVersion[0]= aa • ... • i_byBootVersion[3]= dd
i_dwHardVersion	DWORD	Steuerungshardware-Version
i_dwChipVersion	DWORD	Steuerungsverarbeitungsversion.
i_wStatus	PLC_R_STATUS (siehe Seite 40)	Status der Steuerung
i_wBootProjectStatus	PLC_R_BOOT_PROJECT_STATUS (siehe Seite 38)	Gibt die Informationen über die Boot-Anwendung zurück, die im FLASH-Speicher gespeichert ist:
i_wLastStopCause	PLC_R_STOP_CAUSE (siehe Seite 41)	Ursache für den letzten Übergang von RUN in einen anderen Status
i_wLastApplicationError	PLC_R_APPLICATION_ERROR (siehe Seite 37)	Ursache der letzten Steuerungsausnahme.
i_lwSystemFault_1	LWORD	Nicht verwendet. gibt immer FFFF FFFF FFFF FFFF hex aus.
i_lwSystemFault_2	LWORD	Nicht verwendet.

Variablenname	Typ	Kommentar
i_wIOStatus1	PLC_R_IO_STATUS (siehe Seite 39)	Status der integrierten E/A
i_wIOStatus2	PLC_R_IO_STATUS	Reserviert.
i_wBatteryStatus	WORD	Verbleibende Batterieladung. Diese Systemvariable kann die folgenden signifikanten Werte ausgeben: ● 0064 hex = 100 % = 3 V ● 0032 hex = 50 % = 2.5 V ● 0000 hex = 0 % = 2 V HINWEIS: Der HMI-Steuerungsbildschirm gibt auch an, dass die Batterieladung 50 % oder weniger beträgt.
i_dwAppliSignature1	DWORD	1. Doppelwort (DWORD) einer Signatur mit vier DWORDs (16 Byte insg.). Die Anwendungssignatur wird automatisch beim Software-Build generiert.
i_dwAppliSignature2	DWORD	2. Doppelwort (DWORD) einer Signatur mit vier DWORDs (16 Byte insg.). Die Anwendungssignatur wird automatisch beim Software-Build generiert.
i_dwAppliSignature3	DWORD	3. Doppelwort (DWORD) einer Signatur mit vier DWORDs (16 Byte insg.). Die Anwendungssignatur wird automatisch beim Software-Build generiert.
i_dwAppliSignature4	DWORD	4. Doppelwort (DWORD) einer Signatur mit vier DWORDs (16 Byte insg.). Die Anwendungssignatur wird automatisch beim Software-Build generiert.

PLC_W: Steuerungssystemvariablen mit Lese- und Schreibzugriff

Variablenstruktur

In der folgenden Tabelle werden die Parameter der Systemvariablen PLC_W (Typ PLC_W_STRUCT) beschrieben:

Variablenname	Typ	Kommentar
q_uiOpenPLCControl	UINT	Mit dieser Variable können Sie einen Befehl vom Typ STOP -, RUN -, Kaltstart - oder Warmstart auslösen.
q_wPLCControl	PLC_W_COMMAND (siehe Seite 42)	Diese Variable enthält den Wert für einen Statusänderungsbefehl, der vom Benutzer gewünscht wird. STOP , RUN , Kaltstart und Warmstart der Steuerung werden im Datentyp PLC_W_COMMAND aufgeführt. Diese erwünschte Statusänderung wird ausgelöst, wenn der Wert für PLC_W.q_uiOpenPLCControl von 0 zu 6699 geändert wird.

Abschnitt 1.3

SERIAL_W-Strukturen

SERIAL_W[0]: Systemvariablen für serielle Leitung mit Lese- und Schreibzugriff

Einführung

SERIAL_W ist eine Systemvariable vom Typ SERIAL_W_STRUCT. Ein Element zwingt die SERIAL_R-Systemvariablen für die entsprechende serielle Leistung zum Rücksetzen.

Für HMI SCU verweist Serial_W[0] auf COM1.

Variablenstruktur

Die folgende Tabelle beschreibt die Parameter der Systemvariable SERIAL_W[0]:

Variablenname	Typ	Kommentar
q_wResetCounter	WORD	Bei einem Übergang von 0 auf 1 werden alle SERIAL_R[0]-Zähler zurückgesetzt. Um die Zähler erneut zurückzusetzen, müssen Sie in dieses Register eine 0 schreiben, bevor ein weiterer Übergang von 0 auf 1 erfolgen kann.

HINWEIS: Es gibt einen definierten Kommunikationsport für HMI-SCU.

Kapitel 2

HMI SCU-Systemfunktionen

Abschnitt 2.1

HMI SCU-Funktionen für den Lesezugriff

Überblick

In diesem Abschnitt werden die in der HMI SCU PLCSystem-Bibliothek enthaltenen Funktionen für einen Lesezugriff beschrieben.

Inhalt dieses Abschnitts

Dieser Abschnitt enthält die folgenden Themen:

Thema	Seite
GetBatteryLevel: Gibt die verbleibende Batterieladung zurück	25
GetLocalAIIOStatus: Gibt den Status der integrierten Analog-E/A zurück	26
GetLocalIOStatus: Gibt den integrierten E/A-Status aus	27
GetShortCutStatus: Gibt den Kurzschlussstatus integrierter Ausgänge zurück.	29
GetTempStatus: Die Funktion gibt den Temperatureingangsstatus aus.	30
IsFirstMastColdCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Kaltstartzyklus handelt	31
IsFirstMastCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Zyklus handelt	32
IsFirstMastWarmCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Warmstartzyklus handelt	33

GetBatteryLevel: Gibt die verbleibende Batterieladung zurück

Beschreibung der Funktion

Diese Funktion gibt den Ladezustand der externen Reservebatterie zurück (in Prozent).

Weitere Informationen über interne und externe Batterien finden Sie imHMI SCU Hardwarehandbuch (siehe Magelis SCU, HMI Controller –, Hardwarehandbuch).

HINWEIS: Ähnliche Informationen sind auch über die Systemvariable PLC_R.i_wBootProjectStatus (siehe Seite 19) erhältlich.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel Darstellung von Funktionen und Funktionsbausteinen (siehe Seite 51).

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Ausgangsvariable:

Ausgang	Typ	Kommentar
GetBatteryLevel	WORD	Verbleibende Batteriekapazität in Prozent. Bereich: 0 - 100: ● 100 % = 3 V ● 50 % = 2,5 V ● 0 % = 2 V HINWEIS: Auf der HMI-Steuerung wird die Meldung Battery Level is low angezeigt, wenn die Spannung unter 2,5 Volt liegt.

GetLocalAIOSStatus: Gibt den Status der integrierten Analog-E/A zurück

Beschreibung der Funktion

Diese Funktion gibt den Status der lokalen Ein- und Ausgänge zurück.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen (siehe Seite 51)*.

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Ausgangsvariable:

Bit	Beschreibung
0	Analogeingang 0 mit Konfigurationsfehler erkannt.
1	Analogeingang 0 mit Spannung außerhalb des Gültigkeitsbereichs.
2	Analogeingang 0 mit ungültigen Daten.
3	Analogeingang 0 mit defektem Kabel (nur verfügbar, wenn der Eingangsbereich zwischen 4 und 20 mA liegt).
4	Analogeingang 1 mit Konfigurationsfehler erkannt.
5	Analogeingang 1 mit Spannung außerhalb des Gültigkeitsbereichs.
6	Analogeingang 1 mit ungültigen Daten.
7	Analogeingang 1 mit defektem Kabel (nur verfügbar, wenn der Eingangsbereich zwischen 4 und 20 mA liegt).
8	Analogausgang 0 mit Konfigurationsfehler erkannt.
9	Analogausgang 0 mit Spannung oder Strom außerhalb des Gültigkeitsbereichs.
10	Analogausgang 0 mit ungültigen Daten.
11	Analogausgang 0 mit Fehler erkannt.
12	Analogausgang 1 mit Konfigurationsfehler erkannt.
13	Analogausgang 1 mit Spannung oder Strom außerhalb des Gültigkeitsbereichs.
14	Analogausgang 1 mit ungültigen Daten.
15	Analogausgang 1 mit Fehler erkannt.

GetLocalIOStatus: Gibt den integrierten E/A-Status aus

Beschreibung der Funktion

Diese Funktion gibt den Status der integrierten E/A zurück.

HINWEIS: Ähnliche Informationen sind auch über die Systemvariable PLC_R.i_wLocalIO-Status (siehe Seite 19) erhältlich.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (siehe Seite 51).

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Eingangsparameter:

Eingang	Typ	Kommentar
Betriebsart	LOCAL_IO_GET_STATUS (siehe Seite 47)	Parameter der Funktion: derzeit ist nur LOCAL_IO_GET_GEN_STATUS (00 hex) (siehe Seite 47) verfügbar.

Die folgende Tabelle beschreibt die Ausgangsvariable:

Ausgang	Typ	Kommentar
GetLocalIOStatus	LOCAL_IO_GEN_STATUS (siehe Seite 48)	Status der integrierten E/A

Beispiel 1

Dieses Beispiel zeigt die direkte Verwendung des Enumerators `LOCAL_IO_GET_GEN_STATUS` des Enumerationstyps `LOCAL_IO_GET_STATUS` für den Eingangsparameter "Modus":

```
VAR
    MyLocalIOStatus : LOCAL_IO_GEN_STATUS;
END_VAR

MyLocalIOStatus := GetLocalIOStatus(LOCAL_IO_GET_GEN_STATUS);
```

Beispiel 2

Dieses Beispiel illustriert die Verwendung einer intermediären Variablen für den Eingangsparameter "Modus".

```
VAR
    MyLocalIOStatus : LOCAL_IO_GEN_STATUS;
    MyMode : LOCAL_IO_GET_STATUS;
END_VAR

MyMode := LOCAL_IO_GET_GEN_STATUS;
MyLocalIOStatus := GetLocalIOStatus(MyMode);
```

GetShortCutStatus: Gibt den Kurzschlussstatus integrierter Ausgänge zurück.

Beschreibung der Funktion

Diese Funktion gibt eine Kurzschluss- oder Überlastdiagnose für integrierte Ausgänge zurück.

HINWEIS: Weitere Informationen über die Verwaltung integrierter Ausgänge finden Sie im HMI SCU Hardwarehandbuch (*siehe Magelis SCU, HMI Controller –, Hardwarehandbuch*).

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (*siehe Seite 51*).

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Ausgangsvariable:

Parameter	Typ	Kommentar
GetShortCutStatus	WORD	Siehe Bitfeld-Beschreibung weiter unten.

Die folgende Tabelle beschreibt das Bitfeld für die Steuerung:

Bit	Beschreibung
0	TRUE = Kurzschluss auf Ausgängen (Q0 und Q1).

GetTempStatus: Die Funktion gibt den Temperatureingangsstatus aus.

Beschreibung der Funktion

Diese Funktion gibt den Status der Temperatureingänge aus.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (siehe Seite 51).

Beschreibung der E/A-Variablen

Parameter	Typ	Kommentar
GetTempStatus	WORD	Siehe Bitfeld-Beschreibung weiter unten.

Die folgende Tabelle beschreibt die Eingangsvariable:

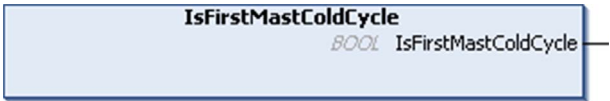
Bit	Kommentar
0	Temperatureingang 0 mit Konfigurationsfehler erkannt.
1	Temperatureingang 1 mit Konfigurationsfehler erkannt.
2	Temperatureingang 0 mit Bereichsfehler erkannt.
3	Temperatureingang 1 mit Bereichsfehler erkannt.
4	Temperatureingang 0 mit ungültigen Daten.
5	Temperatureingang 1 mit ungültigen Daten.
6	Temperatureingang 0 mit defektem Kabel erkannt.
7	Temperatureingang 1 mit defektem Kabel erkannt.
8 - 15	Reserviert.

IsFirstMastColdCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Kaltstartzyklus handelt

Beschreibung der Funktion

Diese Funktion gibt den Wert `TRUE` während des ersten MAST-Zyklus nach einem Kaltstart zurück (erster Zyklus nach dem Herunterladen oder nach einem Kaltstart).

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (*siehe Seite 51*).

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Ausgangsvariable:

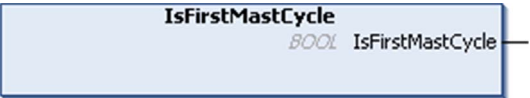
Ausgang	Typ	Kommentar
IsFirstMastColdCycle	BOOL	TRUE während des ersten MAST-Task-Zyklus nach einem Kaltstart.

IsFirstMastCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Zyklus handelt

Beschreibung der Funktion

Diese Funktion gibt während des ersten MAST-Zyklus nach einem Start den Wert `TRUE` zurück.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (*siehe Seite 51*).

Beschreibung der E/A-Variablen

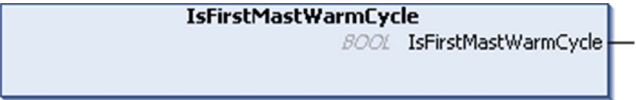
Ausgang	Typ	Kommentar
IsFirstMastCycle	BOOL	TRUE während des ersten MAST-Task-Zyklus nach einem Kaltstart.

IsFirstMastWarmCycle: Gibt an, ob es sich bei dem Zyklus um den ersten MAST-Warmstartzyklus handelt

Beschreibung der Funktion

Diese Funktion gibt während des ersten MAST-Zyklus nach einem Warmstart den Wert `TRUE` zurück.

Grafische Darstellung



Darstellung in AWL (IL) und ST

Sie finden eine allgemeine Darstellung in AWL (IL) oder ST im Kapitel *Darstellung von Funktionen und Funktionsbausteinen* (*siehe Seite 51*).

Beschreibung der E/A-Variablen

Die folgende Tabelle beschreibt die Ausgangsvariable:

Ausgang	Typ	Kommentar
IsFirstMastWarmCycle	BOOL	TRUE während des ersten MAST-Task-Zyklus nach einem Warmstart.

Kapitel 3

HMI SCU PLCSystem-Bibliothek - Datentypen

Überblick

Dieses Kapitel beschreibt die **Datentypen** der HMI SCU PLCSystem-Bibliothek.

Es gibt zwei Arten von **Datentypen**:

- **System-Variablendatentypen** werden von den **Systemvariablen** der PLCSystem-Bibliothek (PLC_R, PLC_W ...) verwendet. (siehe Seite 19)HMI SCU
- **Daten vom Typ Systemfunktion** werden von den **Systemfunktionen** (siehe Seite 23) mit einem Lese-/Schreibzugriff der HMI SCU PLCSystem-Bibliothek verwendet.

Inhalt dieses Kapitels

Dieses Kapitel enthält die folgenden Abschnitte:

Abschnitt	Thema	Seite
3.1	Datentypen der Systemvariablen in PLC_R/W	36
3.2	Daten vom Typ Systemfunktion	43

Abschnitt 3.1

Datentypen der Systemvariablen in PLC_R/W

Überblick

In diesem Abschnitt werden die verschiedenen Systemvariablen aufgelistet und beschrieben, die in den Strukturen `PLC_R` und `PLC_W` enthalten sind.

Inhalt dieses Abschnitts

Dieser Abschnitt enthält die folgenden Themen:

Thema	Seite
<code>PLC_R_APPLICATION_ERROR</code> : Fehlerstatuscodes erkannter Anwendungsfehler	37
<code>PLC_R_BOOT_PROJECT_STATUS</code> : Statuscodes für Boot-Projekt	38
<code>PLC_R_IO_STATUS</code> : E/A-Statuscodes	39
<code>PLC_R_STATUS</code> : Codes für den Steuerungsstatus	40
<code>PLC_R_STOP_CAUSE</code> : Codes für den Übergang vom RUN-Status in anderen Status	41
<code>PLC_W_COMMAND</code> : Codes für Steuerbefehle	42

PLC_R_APPLICATION_ERROR: Fehlerstatuscodes erkannter Anwendungsfehler

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdattentyp PLC_R_APPLICATION_ERROR enthält die folgenden Werte:

Enumerator	Wert	Kommentar
PLC_R_APP_ERR_UNKNOWN	FFFF hex	Nicht definierter Fehler.
PLC_R_APP_ERR_NOEXCEPTION	0000 hex	Es wurde kein Fehler festgestellt.
PLC_R_APP_ERR_WATCHDOG	0010 hex	Anwendungs-Watchdog abgelaufen.
PLC_R_APP_ERR_HARDWAREWATCHDOG	0011 hex	Hardware-Watchdog ist abgelaufen.
PLC_R_APP_ERR_IO_CONFIG_ERROR	0012 hex	Es wurden falsche E/A-Konfigurationsparameter erkannt.
PLC_R_APP_ERR_UNRESOLVED_EXTREFS	0018 hex	Nicht definierte Funktionen erkannt.
PLC_R_APP_ERR_IEC_TASK_CONFIG_ERROR	0025 hex	Falsche Parameter für die Taskkonfiguration erkannt.
PLC_R_APP_ERR_ILLEGAL_INSTRUCTION	0050 hex	Nicht definierte Anweisung erkannt.
PLC_R_APP_ERR_ACCESS_VIOLATION	0051 hex	Zugriffsversuch auf reservierten Speicherbereich.
PLC_R_APP_ERR_DIVIDE_BY_ZERO	0102 hex	Es wurde die Division einer Ganzzahl durch Null erkannt.
PLC_R_APP_ERR_DIVIDE_REAL_BY_ZERO	0152 hex	Es wurde die Division einer Realzahl durch Null erkannt.

PLC_R_BOOT_PROJECT_STATUS: Statuscodes für Boot-Projekt

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdantentyp PLC_R_BOOT_PROJECT_STATUS enthält die folgenden Werte:

Enumerator	Wert	Kommentar
PLC_R_NO_BOOT_PROJECT	0000 hex	Nicht verwendet
PLC_R_BOOT_PROJECT_CREATION_IN_PROGRESS	0001 hex	Boot-Projekt wurde erstellt.
PLC_R_DIFFERENT_BOOT_PROJECT	0002 hex	Das Boot-Projekt im Flash-Speicher entspricht nicht dem im RAM geladenen Projekt oder existiert nicht.
PLC_R_VALID_BOOT_PROJECT	FFFF hex	Das Boot-Projekt im Flash-Speicher und das in den Arbeitsspeicher geladene Projekt sind identisch.

PLC_R_IO_STATUS: E/A-Statuscodes

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdattentyp `PLC_R_IO_STATUS` enthält die folgenden Werte:

Enumerator	Wert	Kommentar
<code>PLC_R_IO_OK</code>	FFFF hex	Eingänge / Ausgänge sind funktionsfähig.
<code>PLC_R_IO_NO_INIT</code>	0001 hex	Eingänge / Ausgänge sind nicht initialisiert.
<code>PLC_R_IO_CONF_FAULT</code>	0002 hex	Es wurden falsche E/A-Konfigurationsparameter erkannt.
<code>PLC_R_IO_SHORTCUT_FAULT</code>	0003 hex	An Eingängen / Ausgängen wurde ein Kurzschluss erkannt.
<code>RESERVED</code>	0004 hex	Nicht verwendet
<code>PLC_R_IO_COM_LOST</code>	0005 hex	Kommunikationsfehler mit einem Prozessor erkannt. Die integrierten E/As werden nicht aktualisiert.

PLC_R_STATUS: Codes für den Steuerungsstatus

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdatentyp `PLC_R_STATUS` enthält folgende Werte:

Enumerator	Wert	Kommentar
<code>PLC_R_EMPTY</code>	0000 hex.	Steuerung enthält keine Anwendung.
<code>PLC_R_STOPPED</code>	0001 hex.	Steuerung wurde gestoppt.
<code>PLC_R_RUNNING</code>	0002 hex.	Steuerung ist in Betrieb.
<code>PLC_R_HALT</code>	0004 hex.	Steuerung ist in einem HALT-Status. (Weitere Informationen können Sie dem Statusdiagramm der Steuerung im entsprechenden <i>Programmierhandbuch</i> entnehmen).
<code>PLC_R_BREAKPOINT</code>	0008 hex.	Steuerung hat am Haltepunkt angehalten.

PLC_R_STOP_CAUSE: Codes für den Übergang vom RUN-Status in anderen Status

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdattentyp `PLC_R_STOP_CAUSE` enthält die folgenden Werte:

Aufzähler	Wert	Kommentar
<code>PLC_R_STOP_REASON_UNKNOWN</code>	00 hex	Initialwert oder Stopp-Ursache ist nicht definiert.
<code>PLC_R_STOP_REASON_HW_WATCHDOG</code>	01 hex	Nach Hardware-Watchdog-Timeout (Systemhardware- oder Softwaretask) gestoppt.
<code>PLC_R_STOP_REASON_RESET</code>	02 hex	Gestoppt nach Zurücksetzen.
<code>PLC_R_STOP_REASON_EXCEPTION</code>	03 hex	Gestoppt nach Ausnahme.
<code>PLC_R_STOP_REASON_USER</code>	04 hex	Gestoppt nach einem Anwender-Request.
<code>PLC_R_STOP_REASON_IECPROGRAM</code>	05 hex	Nach einem Request für einen Programmbefehl gestoppt.
<code>PLC_R_STOP_REASON_DELETE</code>	06 hex	Gestoppt nach einem Befehl zum Entfernen einer Anwendung.
<code>PLC_R_STOP_REASON_DEBUGGING</code>	07 hex	Gestoppt nach der Aktivierung des Debug-Modus.
<code>PLC_R_STOP_FROM_NETWORK_REQUEST</code>	0A hex	Nach einem Request des Netzwerks (<code>PLC_W.q_wPLCControl:=PLC_W_COMMAND.PLC_W_STOP;</code>) . gestoppt.
<code>PLC_R_STOP_FROM_INPUT</code>	0B hex	Stopp wurde von einem Steuerungseingang gefordert.

Weitere Informationen zu Gründen, aus denen die Steuerung gestoppt wird, finden Sie unter Beschreibung der Steuerungszustände (*siehe Magelis SCU, HMI Controller, Programmierhandbuch*).

PLC_W_COMMAND: Codes für Steuerbefehle

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdantentyp `PLC_W_COMMAND` enthält folgende Werte:

Enumerator	Wert	Kommentar
<code>PLC_W_STOP</code>	0001 hex.	Befehl zum Stopp der Steuerung.
<code>PLC_W_RUN</code>	0002 hex.	Befehl zur Ausführung der Steuerung.
<code>PLC_W_RESET_COLD</code>	0004 hex.	Befehl zur Initialisierung eines Kalt-Resets der Steuerung.
<code>PLC_W_RESET_WARM</code>	0008 hex.	Befehl zur Initialisierung eines Warm-Resets der Steuerung.

Abschnitt 3.2

Daten vom Typ Systemfunktion

Überblick

In diesem Abschnitt werden die verschiedenen Daten vom Typ Systemfunktion in der HMI SCU PLCSystem-Bibliothek beschrieben.

Inhalt dieses Abschnitts

Dieser Abschnitt enthält die folgenden Themen:

Thema	Seite
FIRMWARE_VERSION: Funktion <code>GetFirmwareVersion</code> Ausgangstyp	44
BOOT_PROJECT_STATUS: <code>GetBootProjectStatus</code> Funktionsausgangscodes	45
STOP_WHY: <code>GetLastStopCause</code> Funktionsausgangscodes	46
LOCAL_IO_GET_STATUS: <code>GetLocalIOStatus</code> Parametercodes der Funktion	47
LOCAL_IO_GEN_STATUS: Ausgangscodes der Funktion <code>GetLocalIOStatus</code>	48

FIRMWARE_VERSION: Funktion `GetFirmwareVersion` Ausgangstyp

Strukturbeschreibung

Die Datenstruktur enthält folgende Variablen:

Variable	Typ	Beschreibung
FwVersion	DWORD	Enthält die Firmware-Version (z. B. 2.0.20.4 = 02001404 hex).
BootVersion	WORD	Enthält die Boot-Version (z. B., 1.4 = 0104 hex).
AsicVersion	WORD	Enthält die Koprozessor-Version.

BOOT_PROJECT_STATUS: GetBootProjectStatus Funktionsausgangscodes

Beschreibung des Datentyps Aufzählung

Der Aufzählungsdattentyp enthält die folgenden Werte:

Enumerator	Wert	Beschreibung
NO_BOOT_PROJECT	0000 hex	Nicht verwendet.
BOOT_PROJECT_CREATION_IN_PROGRESS	0001 hex	Boot-Projekt wurde erstellt.
DIFFERENT_BOOT_PROJECT	0002 hex	Das Boot-Projekt im Flash-Speicher entspricht nicht dem im RAM geladenen Projekt oder existiert nicht.
VALID_BOOT_PROJECT	FFFF hex	Das Boot-Projekt im Flash-Speicher und das in den Arbeitsspeicher geladene Projekt sind identisch.

STOP_WHY: GetLastStopCause Funktionsausgangscodes

Beschreibung des Datentyps Aufzählung

Der Enumerationsdatentyp enthält den Status mit den folgenden Werten:

Enumerator	Wert	Beschreibung
STOP_REASON_UNKNOWN	00 hex	Initialwert oder Stopp-Ursache ist nicht definiert.
STOP_REASON_HW_WATCHDOG	01 hex.	Nach Software-Watchdog-Timeout gestoppt.
STOP_REASON_RESET	02 hex	Gestoppt nach Zurücksetzen.
STOP_REASON_EXCEPTION	03 hex	Gestoppt nach Ausnahme.
STOP_REASON_USER	04 hex	Gestoppt nach einem Anwender-Request.
STOP_REASON_IECPROGRAM	05 hex	Gestoppt nach Anforderung durch einen Programmbefehl (zum Beispiel Steuerungsbefehl mit dem Parameter PLC_W.q_wPLCControl := PLC_W.COMMAND.PLC_W_STOP;).
STOP_REASON_DELETE	06 hex	Gestoppt nach einem Befehl zum Entfernen einer Anwendung.
STOP_REASON_DEBUGGING	07 hex	Gestoppt nach der Aktivierung des Debug-Modus.
STOP_FROM_NETWORK_REQUEST	0A hex	Nach einem Request aus dem Netzwerk (PLC_W-Befehl) gestoppt.
STOP_FROM_INPUT	0B hex	Stopp wurde von einem Steuerungseingang gefordert.

LOCAL_IO_GET_STATUS: GetLocalIOStatus Parametercodes der Funktion

Beschreibung des Datentyps Enumeration

Der Enumerationsdatentyp `LOCAL_IO_GET_STATUS` enthält den folgenden Wert:

Enumerator	Wert	Beschreibung
<code>LOCAL_IO_GET_GEN_STATUS</code>	00 hex.	Wert, der zur Anforderung des allgemeinen Status integrierter E/A dient.

LOCAL_IO_GEN_STATUS: Ausgangscodes der Funktion GetLocalIOStatus

Beschreibung des Enumerationstyps

Der Enumerationsdatentyp `LOCAL_IO_GEN_STATUS` enthält den Status lokaler E/A mit den folgenden Werten:

Enumerator	Wert	Beschreibung
<code>LOCAL_IO_OK</code>	00 hex	Eingänge/Ausgänge sind in Betrieb.
<code>LOCAL_IO_NO_INIT</code>	01 hex	Falsche E/A-Konfigurationsparameter festgestellt.
<code>LOCAL_IO_COM_LOST</code>	02 hex	Fehler in der Kommunikation mit Koprozessor. Die integrierten E/A sind nicht aktualisiert.
<code>LOCAL_IO_CONF_FAULT</code>	03 hex	Falsche E/A-Konfigurationsparameter festgestellt.

Anhang



Anhang A

Darstellung von Funktionen und Funktionsbausteinen

Übersicht

Jede Funktion kann in den folgenden Sprachen dargestellt werden.

- AWL: Anweisungsliste
- ST: Strukturierter Text
- KOP: Kontaktplan
- FBD: Funktionsbausteindiagramm
- CFC: Continuous Function Chart

Dieses Kapitel enthält Darstellungen von Funktionen und Funktionsbausteinen und erläutert deren Verwendung in den Sprachen AWL und ST.

Inhalt dieses Kapitels

Dieses Kapitel enthält die folgenden Themen:

Thema	Seite
Unterschiede zwischen einer Funktion und einem Funktionsbaustein	52
Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache AWL	53
Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache ST	57

Unterschiede zwischen einer Funktion und einem Funktionsbaustein

Funktion

Eine Funktion hat die folgenden Eigenschaften:

- Ist eine POU (Program Organization Unit), die ein einzelnes direktes Ergebnis zurückgibt
- Wird direkt über ihren Namen aufgerufen (nicht über eine Instanz)
- Ist nicht instanziiert
- Kann als Operand in anderen Ausdrücken verwendet werden

Beispiele: Boolesche Operatoren (AND), Berechnungen, Konvertierung (BYTE_TO_INT)

Funktionsbaustein

Ein Funktionsbaustein hat die folgenden Eigenschaften:

- Ist eine POU (Program Organization Unit), die ein oder mehrere direkte Ausgänge zurückgibt
- Muss von einer Instanz aufgerufen werden (Funktionsbausteinkopie mit dediziertem Namen und Variablen)
- Hat für jede Instanz einen persistenten Status (Ausgänge und interne Variablen) von einem Aufruf zum anderen aus einem Funktionsbaustein oder Programm

Beispiele: Zeitgeber, Zähler

In dem nachstehenden Beispiel ist `Timer_ON` eine Instanz des Funktionsbausteins `TON`:

```

1  PROGRAM MyProgram_ST
2  VAR
3      Timer_ON: TON; // Function Block Instance
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR

```

```

1  Timer_ON(
2      IN:=Timer_RunCd,
3      PT:=Timer_PresetValue,
4      Q=>Timer_Output,
5      ET=>Timer_ElapsedTime);

```

Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache AWL

Allgemeine Informationen

In diesem Abschnitt wird das Implementieren einer Funktion und eines Funktionsbausteins in der Sprache AWL beschrieben.

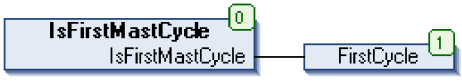
Die Funktionen `IsFirstMastCycle` und `SetRTCDrift` und der Funktionsbaustein `TON` werden als Implementierungsbeispiele verwendet.

Verwenden einer Funktion in der AWL-Sprache

Im Folgenden wird das Einfügen einer Funktion in der AWL-Sprache beschrieben:

Schritt	Aktion
1	Öffnen oder erstellen Sie eine neue POU in der AWL-Sprache. HINWEIS: Die Vorgehensweise zum Erstellen einer POU wird hier nicht erläutert. Weitere Informationen finden Sie unter Hinzufügen und Aufrufen von POU's (<i>siehe SoMachine, Programmierhandbuch</i>).
2	Erstellen Sie die Variablen, die für die Funktion erforderlich sind.
3	Wenn die Funktion über mindestens einen Eingang verfügt, beginnen Sie mit dem Laden des ersten Eingangs mithilfe der LD-Anweisung.
4	Fügen Sie unten eine neue Zeile ein, und gehen Sie wie folgt vor: • Geben Sie den Namen der Funktion in der Operator-Spalte (linkes Feld) ein. • Oder verwenden Sie die Eingabehilfe, um die Funktion auszuwählen. (Wählen Sie im Kontextmenü Bausteinaufruf einfügen.)
5	Wenn die Funktion über mehr als einen Eingang verfügt und die Eingabehilfe verwendet wird, wird die erforderliche Anzahl von Zeilen automatisch mit ??? in den Feldern rechts erstellt. Ersetzen Sie ??? durch den geeigneten Wert oder die Variable, die der Reihenfolge der Eingänge entspricht.
6	Fügen Sie eine neue Zeile ein, um das Ergebnis der Funktion in der entsprechenden Variable zu speichern: Geben Sie die ST-Anweisung in die Operator-Spalte (linkes Feld) und einen Variablennamen in das rechte Feld ein.

Die Funktionen `IsFirstMastCycle` (ohne Eingangsparameter) und `SetRTCDrift` (mit Eingangsparametern) werden im Folgenden grafisch dargestellt:

Funktion	Grafische Darstellung
ohne Eingangsparameter: <code>IsFirstMastCycle</code>	
mit Eingangsparametern: <code>SetRTCDrift</code>	

In der AWL-Sprache wird der Funktionsname direkt in der Operator-Spalte verwendet:

Funktion	Darstellung im SoMachine POU-Editor in AWL
Beispiel einer Funktion ohne Eingangsparameter in der AWL-Sprache: <code>IsFirstMastCycle</code>	<pre>1 PROGRAM MyProgram_IL 2 VAR 3 FirstCycle: BOOL; 4 END_VAR 5 1 IsFirstMastCycle ST FirstCycle</pre>

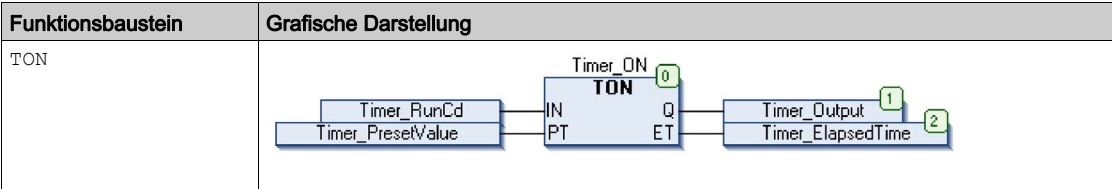
Funktion	Darstellung im SoMachine POU-Editor in AWL
Beispiel einer Funktion mit Eingangsparametern in der AWL-Sprache: SetRTCDrift	1PROGRAM MyProgram_IL 2VAR 3myDrift: SINT (-29..29) := 5; 4myDay: DAY_OF_WEEK := SUNDAY; 5myHour: HOUR := 12; 6myMinute: MINUTE; 7myDiag: RTCSETDRIFT_ERROR; 8END_VAR 9 1LDmyDrift SetRTCDriftmyDay myHour myMinute STmyDiag

Verwenden eines Funktionsbausteins in der AWL-Sprache

Im Folgenden wird das Einfügen eines Funktionsbausteins in der AWL-Sprache beschrieben:

Schritt	Aktion
1	Erstellen Sie eine neue POU in der AWL-Sprache. HINWEIS: Die Vorgehensweise zum Erstellen einer POU wird hier nicht erläutert. Weitere Informationen finden Sie unter Hinzufügen und Aufrufen von POU's (<i>siehe SoMachine, Programmierhandbuch</i>).
2	Erstellen Sie die Variablen, die für den Funktionsbaustein erforderlich sind, einschließlich des Instanznamens.
3	Funktionsbausteine werden mithilfe einer CAL-Anweisung aufgerufen: - Verwenden Sie die Eingabehilfe, um den FB auszuwählen. (Klicken Sie mit der rechten Maustaste, und wählen Sie im Kontextmenü Bausteineinruf einfügen aus.) - Die CAL-Anweisung und der entsprechende E/A werden erstellt. Jeder Parameter (E/A) ist eine Anweisung: - Werte für Eingänge werden mit ":"=" festgelegt. - Werte für Ausgänge werden mit "=>" festgelegt.
4	Ersetzen Sie im rechten CAL-Feld die ??? durch den Instanznamen.
5	Ersetzen Sie weitere ??? durch eine geeignete Variable oder einen direkten Wert.

Der grafisch dargestellte Funktionsbaustein TON dient in diesem Beispiel zur Veranschaulichung:



In der AWL-Sprache wird der Name des Funktionsbausteins direkt in der Operator-Spalte verwendet:

Funktionsbaustein	Darstellung im SoMachine POU-Editor in AWL
TON	<pre>1 PROGRAM MyProgram_IL 2 VAR 3 Timer_ON: TON; // Function Block instance declaration 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000</pre>

Verwenden einer Funktion oder eines Funktionsbausteins in der Sprache ST

Allgemeine Informationen

In diesem Teil wird die Implementierung einer Funktion oder eines Funktionsbausteins in der ST-Sprache erläutert.

Dabei werden die Funktion `SetRTCDrift` und der Funktionsbaustein `TON` als Beispiele verwendet.

Verwenden einer Funktion in der ST-Sprache

Im Folgenden wird das Einfügen einer Funktion in der ST-Sprache beschrieben:

Schritt	Aktion
1	Erstellen Sie eine neue POU in der ST-Sprache. HINWEIS: Die Vorgehensweise zum Erstellen einer POU wird hier nicht erläutert. Weitere Informationen finden Sie unter Hinzufügen und Aufrufen von POUs (<i>siehe SoMachine, Programmierhandbuch</i>).
2	Erstellen Sie die Variablen, die für die Funktion erforderlich sind.
3	Verwenden Sie im POU-ST-Editor die allgemeine Syntax zur Darstellung einer Funktion in der ST-Sprache. Die allgemeine Syntax lautet: Funktionsergebnis:= Funktionsname(VarEingang1, VarEingang2,.. VarEingangx);

Zur Veranschaulichung dieses Verfahrens betrachten wir die grafisch dargestellte Funktion `SetRTCDrift`:

Funktion	Grafische Darstellung
<code>SetRTCDrift</code>	

In der ST-Sprache wird diese Funktion folgendermaßen dargestellt:

Funktion	Darstellung im SoMachine POU-Editor in der ST-Sprache
SetRTCDrift	<pre>PROGRAM MyProgram_ST VAR myDrift: SINT(-29..29) := 5; myDay: DAY_OF_WEEK := SUNDAY; myHour: HOUR := 12; myMinute: MINUTE; myRTCAjust: RTCDRIFT_ERROR; END_VAR myRTCAjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);</pre>

Verwenden eines Funktionsbausteins in der ST-Sprache

Im Folgenden wird das Einfügen eines Funktionsbausteins in der ST-Sprache beschrieben:

Schritt	Aktion
1	Erstellen Sie eine neue POU in der ST-Sprache. HINWEIS: Die Vorgehensweise zum Erstellen einer POU wird hier nicht erläutert. Weitere Informationen zum Hinzufügen, Deklarieren und Aufrufen von POU's finden Sie in der entsprechenden Dokumentation (<i>siehe SoMachine, Programmierhandbuch</i>).
2	Erstellen Sie die Eingangs- und Ausgangsvariablen und die Instanzen, die für den Funktionsbaustein erforderlich sind: - Die Eingangsvariablen sind die für den Funktionsbaustein erforderlichen Eingangsparameter. - Die Ausgangsvariablen erhalten den vom Funktionsbaustein zurückgegebenen Wert.
3	Verwenden Sie im POU-ST-Editor die allgemeine Syntax zur Darstellung eines Funktionsbausteins in der ST-Sprache. Die allgemeine Syntax lautet: Funktionsbaustein_InstanceName (Eingang1:=VarEingang1, Eingang2:=VarEingang2, ... Ausgang1=>VarAusgang1, Ausgang2=>VarAusgang2, ...);

Der grafisch dargestellte Funktionsbaustein TON dient in diesem Beispiel zur Veranschaulichung:

Funktionsbaustein	Grafische Darstellung
TON	

Die folgende Tabelle zeigt Beispiele für den Aufruf eines Funktionsbausteins in der ST-Sprache:

Funktionsbaustein	Darstellung im SoMachine POU-Editor in der ST-Sprache
TON	<pre> 1 PROGRAM MyProgram_ST 2 VAR 3 Timer_ON: TON; // Function Block Instance 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 1 Timer_ON(2 IN:=Timer_RunCd, 3 PT:=Timer_PresetValue, 4 Q=>Timer_Output, 5 ET=>Timer_ElapsedTime); </pre>



A

Anwendung

Programm mit Konfigurationsdaten, Symbolen und Dokumentation.

B

Byte

In einem 8-Bit-Format codierter Typ. Gültiger Wertebereich: 00 hex bis FF hex.

C

CFC

(*Continuous Function Chart*) Grafische Programmiersprache (Erweiterung des Standards IEC 61131-3) auf der Grundlage der FBD-Sprache (Funktionsbausteindiagramm), die wie ein Flussdiagramm aufgebaut ist. Grafische Elemente werden allerdings, sofern möglich, ohne die Verwendung von Netzwerken frei positioniert, sodass Rückkopplungsschleifen möglich sind. Bei jedem Baustein befinden sich die Eingänge links und die Ausgänge rechts. Sie können die Bausteinausgänge mit den Eingängen anderer Bausteine verbinden, um komplexe Ausdrücke zu erstellen.

E

E/A

Eingang/Ausgang

Erweiterungsbuss

Elektronischer Kommunikationsbus zwischen E/A-Erweiterungsmodulen und einer Steuerung.

F

FB

(*Function Block: Funktionsbaustein*) Nützlicher Programmiermechanismus, der eine Gruppe von Programmieranweisungen zur Durchführung eines spezifischen und normierten Vorgangs konsolidiert, z. B. Drehzahlregelung, Intervallkontrolle oder Zählen. Ein Funktionsbaustein kann Konfigurationsdaten, eine Gruppe interner oder externer Betriebsparameter und in der Regel 1 oder mehrere Dateneingänge und -ausgänge umfassen.

Firmware

Umfasst das BIOS, Datenparameter und Programmieranweisungen, aus denen das Betriebssystem einer Steuerung besteht. Die Firmware wird in einem nicht flüchtigen Speicher in der Steuerung abgelegt.

Funktionsbausteindiagramm (Programmiersprache)

Eine von 5 Sprachen für die Logik oder Steuerung, die von dem Standard IEC 61131-3 für Steuerungssysteme unterstützt wird. Es handelt sich hierbei um eine grafisch orientierte Programmiersprache. Sie arbeitet mit einer Liste von Netzwerken, wobei jedes Netzwerk eine grafische Struktur von Feldern und Verbindungslinien enthält, die entweder einen logischen oder einen arithmetischen Ausdruck, den Aufruf eines Funktionsbausteins, einen Sprung oder einen Rückkehrbefehl darstellen.

G

GVL

(*Globale Variablenliste*) Verwaltet globale Variablen innerhalb eines SoMachine-Projekts.

I

IL

(*Instruction List: Anweisungsliste (AWL)*) Ein in Anweisungsliste geschriebenes Programm besteht aus einer Abfolge textbasierter Anweisungen, die von der Steuerung der Reihe nach ausgeführt werden. Jede Anweisung besteht aus einer Zeilennummer, einem Anweisungscode und einem Operanden (siehe IEC 61131-3).

INT

(*Integer: Ganzzahl*) Über 16 Bits codierte Ganzzahl.

K

Konfiguration

Die Anordnung und Vernetzung von Hardwarekomponenten innerhalb eines Systems und die Hardware- und Softwareparameter, die die Betriebsmerkmale des Systems bestimmen.

L

LD

(*Ladder Diagramm: Kontaktplan (KOP)*) Grafische Darstellung der Anweisungen eines Steuerungsprogramms mit Symbolen für Kontakte, Spulen und Bausteine in einer Abfolge von Programmbausteinen, die von der Steuerung der Reihe nach ausgeführt werden (siehe IEC 61131-3).

M

MAST

Prozessortask, die über die zugehörige Programmiersoftware ausgeführt wird. Die MAST-Task besteht aus zwei Sections:

- **IN:** Vor der Ausführung der MAST-Task werden die Eingänge in die IN-Section kopiert.
- **OUT:** Nach der Ausführung der MAST-Task werden die Ausgänge in die OUT-Section kopiert.

P

POU

(*Program Organization Unit: Programmierorganisationseinheit*) Variablendeklaration im Quellcode und der entsprechende Anweisungssatz. POU's ermöglichen die modulare Wiederverwendung von Softwareprogrammen, Funktionen und Funktionsbausteinen. Sobald POU's deklariert sind, stehen sie sich gegenseitig zur Verfügung.

Programm

Komponente einer Anwendung, die aus kompiliertem Quellcode besteht und im Speicher einer programmierbaren Steuerung installiert werden kann.

S

ST

(*Structured Text: Strukturierter Text*) Programmiersprache, die komplexe und verschachtelte Anweisungen umfasst (z. B. Iterationsschleifen, bedingte Ausführungen oder Funktionen). ST ist IEC 61131-3-kompatibel.

Steuerung

Ermöglicht die Automatisierung industrieller Prozesse (auch als speicherprogrammierbare Steuerung oder SPS bezeichnet).

Steuerungsnetzwerk

Ein Netzwerk mit Logic Controllern, SCADA-Systemen, PCs, HMI, Switches usw.

Es werden zwei Arten von Topologien unterstützt:

- **Flach:** Alle Module und Geräte in diesem Netzwerk gehören demselben Teilnetz an.
- **2-stufig:** Das Netzwerk ist in ein Betriebsnetzwerk und ein Steuerungsnetzwerk unterteilt.

Diese beiden Netzwerke sind zwar physisch voneinander unabhängig, in der Regel jedoch über ein Routing-Gerät miteinander verbunden.

V

Variable

Speichereinheit, die von einem Programm adressiert und geändert werden kann.



B

BOOT_PROJECT_STATUS

Datentypen, 45

D

Datentyp

FIRMWARE_VERSION, 44

Datentypen

BOOT_PROJECT_STATUS, 45

LOCAL_IO_GEN_STATUS, 48

LOCAL_IO_GET_STATUS, 47

PLC_R_APPLICATION_ERROR, 37

PLC_R_BOOT_PROJECT_STATUS, 38

PLC_R_STATUS, 40

PLC_R_STOP_CAUSE, 41

PLC_W_COMMAND, 42

SPS_R_IO_STATUS, 39

STOP_WHY, 46

F

FIRMWARE_VERSION

Datentyp, 44

Funktionen

GetBatteryLevel, 25

GetLocalIOStatus, 27

GetRunStopSwitch, 26

GetShortCutStatus, 29

IsFirstMastColdCycle, 31

IsFirstMastCycle, 32

IsFirstMastWarmCycle, 33

Unterschiede zwischen einer Funktion
und einem Funktionsbaustein, 52

Verwenden einer Funktion oder eines
Funktionsbausteins in der Sprache AWL,
53

Verwenden einer Funktion oder eines
Funktionsbausteins in der Sprache ST, 57

G

GetBatteryLevel

Funktionen, 25

GetLocalIOStatus

Funktionen, 27

GetRunStopSwitch

Funktionen, 26

GetShortCutStatus

Functions, 29

I

IsFirstMastColdCycle

Funktionen, 31

IsFirstMastCycle

Funktionen, 32

IsFirstMastWarmCycle

Funktionen, 33

L

LOCAL_IO_GEN_STATUS

Datentypen, 48

LOCAL_IO_GET_STATUS

Datentypen, 47

P

PLC_R

Systemvariable, 19

PLC_R_APPLICATION_ERROR

Datentypen, 37

PLC_R_BOOT_PROJECT_STATUS

Datentypen, 38

PLC_R_IO_STATUS

Datentypen, 39

PLC_R_STATUS

Datentypen, 40

PLC_R_STOP_CAUSE

Datentypen, 41

PLC_W

Systemvariable, *21*

PLC_W_COMMAND

Datentypen, *42*

S

SERIAL_W

Systemvariable, *22*

STOP_WHY

Datentypen, *46*

Systemvariable, *19*

PLC_W, *21*

SERIAL_W, *22*

Systemvariablen

Definition, *15*

Verwenden, *16*