Hinweise zu den Standartfunktionsbausteinen der IEC61131stdfb

Verwendungshinweise
Version 29. April 2011

Dipl.-Ing. Lars Evertz

Lehrstuhl für Prozessleittechnik
Prof. Dr.-Ing. Ulrich Epple
RWTH Aachen
D-52064 Aachen, Deutschland
Telefon +49 (0) 241 80 94339
Fax +49 (0) 241 80 92238
www.plt.rwth-aachen.de

Inhaltsverzeichnis

1 Einleitung

Die Bibliothek **iec61131stdfb** stellt eine Reihe standardisierter Funktionsbausteine zur Verfügung. Diese Bausteine sind kompatibel zur gleichnamigen Norm (IEC61131-3 Stand 28.10.2010) implementiert. Unter den Bausteinen sind einfache Rechenoperationen, trigonometrische Funktionen, Potenzieren zur Euler'schen Zahl oder einer beliebigen Basis, natürlicher und 10er-Logarithmus vorhanden. Des Weiteren gibt es Bausteine zum Vergleichen von Werten, zu deren Demultiplexen oder Auswählen nach Maximum oder Minimum und zur Begrenzung eines Wertes nach oben und unten. Boolsche Logiken sind ebenso vorhanden. Außerdem gibt es Flankenerkennungen, FlipFlops, Zähler und Timer-Bausteine.
Mit dieser Auswahl an Bausteinen lassen sich die nahezu alle Funktionen im Leitsystem umsetzen. Zusätzliche Bausteine wie Integratoren, Differetiatoren und Regler sowie Totzeitglieder werden zukünftig noch hinzugefügt.

Dieses Dokument gibt Hinweise zu den Standartfunktionsbausteinen der iec61131stdfb Bibliothek. Insbesondere wird auf die Verwendung des Variablentyps *OV_ANY* und die Verbindungen zu anderen Funktionsblöcken eingegangen. Außerdem wird das Verhalten der mathematischen Funktionsbausteine bei undefinierten Rechenoperationen (z. B. Division durch 0) und bei Werteüberlauf beschrieben.

2 Erklärung vorhandener Funktionsblöcke

In der folgenden Tabelle 2.1 sind die vorhandenen Funktionsbausteine nach Verwendung aufgelistet. Nachfolgend wird auf die einzelnen Gruppen kurz eingegangen. Die einzelnen Funktionsblöcke sind weitestgehend selbsterklärend.

Angesehen vom Block *ABS* geben die Blöcke der Numerischen Funktionen ihr Ergebnis als SINGLE oder, wenn der Input DOUBLE ist, als DOUBLE zurück. Eine Ausgabe als Ganzzahl macht hier in den meisten Fällen keinen Sinn. Da die Berechnungen ohnehin mit Fließkommazahlen ausgeführt werden (müssen), würde sich aus Integern auch kein Vorteil in der Rechengeschwindigkeit ergeben.
Blöcke der Bereiche Grundarithmetik und numerische Funktionen können Wertebereichsüberschreitungen oder undefinierte Operationen teilweise abfangen. Sie reagieren auf diese Fälle durch setzen des Statusflags *Bad* der Ausgangsvariable. Genaueres hierzu findet sich in Kapitel 2.2.

Die Bitweise Logik arbeitet innerhalb der ANY-Variablen nur mit UINT, BYTES und BOOLs. Da die Verknüpfungen bitweise erfolgen, sind diese Typen ausreichend. BOOL wird zwar im Speicher wie eine unsigned int-Variable abgelegt, trotzdem wird nur zwischen zwei Werten (TRUE oder FALSE) unterschieden.

Auswahlfunktionen sind für alle Datentypen gültig. Bei Minimums- und Maximumsauswahl wird für BOOL'sche Variablen $TRUE > FALSE$ angenommen. Zeichenketten werden mit ov_string_compare(...) verglichen. Die selben Vergleichskriterien sind für die Vergleichsfunktionen gültig. Auch hier sind alle Typen einsetzbar. Lediglich bei Vektoren wird nur ihre Länge verglichen (ein Vergleich der einzelnen Elemente würde einen BOOL'schen Vektor als Ausgang erfordern).

Die Variable ET der Timer-Blöcke gibt die verstrichene Zeit seit Auslösung an. Ist die über PT eingestellte Zeit erreicht wird jedoch nicht weiter gezählt. TP und TON werden durch eine steigende Flanke ausgelöst, TOFF durch eine fallende.
Bei Einschaltverzögerungen wird auf einer fallenden Flanke ET sofort sofort zurückgesetzt. Das gleiche gilt für steigende Flanken am Eingang einer Ausschaltverzögerung. Erreicht ET PT, so wird ein- bzw ausgeschaltet.

Tabelle 2.1: Liste der Funktionsblöcke der iec61131stdfb-Bibliothek.

Name	Beschreibung	Eingänge		Ausgänge	
Typumwandlungen					
ANYtoANY	Typumwanlung von beliebigem Typ in den durch K spezifizierten (sofern Sinnvoll)	K IN	UINT ANY	OUT	ANY
Grundarithmetik					
ADD	Addition zweier Variablen	IN1 IN2	ANY ANY	OUT	ANY
SUB	Subtrahiert IN2 von IN1	IN1 IN2	ANY ANY	OUT	ANY
MUL	Multiplikation zweier Variablen	IN1 IN2	ANY ANY	OUT	ANY
DIV	Dividiert IN1 durch IN2	IN1 IN2	ANY ANY	OUT	ANY
EXPT	Potenziert IN1 hoch IN2	IN1 IN2	ANY ANY	OUT	ANY
MOD	Gibt den Rest der operation IN1 / IN2 an, nur gültig für Integer-Datentypen)	IN1 IN2	ANY ANY	OUT	ANY
MOVE	Verschiebt IN nach OUT (reicht durch)	IN	ANY	OUT	ANY
Numerische Funktionen					
ABS	Bildet den Absolutbetrag von IN	IN	ANY	OUT	ANY
SQRT	Bildet die Quadratwurzel von IN	IN	ANY	OUT	ANY
LN	Bildet den natürlichen Logarithmus von IN	IN	ANY	OUT	ANY
LOG	Bildet den Logarithmus zur Basis 10 von IN	IN	ANY	OUT	ANY
EXP	Potenziert zur Basis e (natürliche Exponentiation)	IN	ANY	OUT	ANY
SIN	Bildet den Sinus von IN (in Radian)	IN	ANY	OUT	ANY
COS	Bildet den Kosinus von IN (in Radian)	IN	ANY	OUT	ANY
TAN	Bildet den Tangens von IN (in Radian)	IN	ANY	OUT	ANY
ASIN	Bildet den Arkussinus (in Radian) von IN	IN	ANY	OUT	ANY
ACOS	Bildet den Arkuskosinus (in Radian) von IN	IN	ANY	OUT	ANY
ATAN	Bildet den Arkustangens (in Radian) von IN	IN	ANY	OUT	ANY
ATAN2	Bildet den Arkustangens (in Radian) von IN1 / IN2 (Winkel des Vektors (IN1 | IN2) zur X-Achse)	IN1 IN2	ANY ANY	OUT	ANY

Bitverschiebungen			
SHL	Verschiebt die Bitfolge in IN um N Bits nach links. Füllt dabei rechts mit 0en auf.	IN UINT N UINT	OUT UINT
SHR	Verschiebt die Bitfolge in IN um N Bits nach rechts. Füllt dabei links mit 0en auf.	IN UINT N UINT	OUT UINT
ROL	Rotiert die Bitfolge in IN um N Bits nach links. Die links überlaufenden Bits werden rechts nachgeschoben.	IN UINT N UINT	OUT UINT
ROR	Rotiert die Bitfolge in IN um N Bits nach rechts. Die rechts überlaufenden Bits werden links nachgeschoben.	IN UINT N UINT	OUT UINT
Bitweise Logik			
AND	VerUNDet IN1 und IN2 bitweise.	IN1 ANY IN2 ANY	OUT ANY
OR	VerODERt IN1 und IN2 bitweise.	IN1 ANY IN2 ANY	OUT ANY
XOR	Bitweise Exklusiv-ODER Verknüpfung von IN1 und IN2	IN1 ANY IN2 ANY	OUT ANY
NOT	Negiert IN bitweise	IN ANY	OUT ANY
Auswahlfunktionen			
SEL	Reicht IN0 an OUT weiter, wenn G == 0. Andernfalls wird IN1 weitergereicht.	IN0 ANY IN1 ANY G BOOL	OUT ANY
MAX	Gibt den größeren Wert aus IN1 und IN2 an OUT weiter.	IN1 ANY IN2 ANY	OUT ANY
MIN	Gibt den kleineren Wert aus IN1 und IN2 an OUT weiter.	IN1 ANY IN2 ANY	OUT ANY
LIMIT	Begrenzt den Wertebreich von IN auf das Intervall [MN | MX].	IN ANY MN ANY MX ANY	OUT ANY
MUX	Demultiplext IN1 bis IN8 auf OUT. K gibt die Nummer des Durchgereichten INx an.	IN1 ANY - IN8 ANY K UINT	OUT ANY

Vergleichsfunktionen			
GT	Ergibt TRUE, wenn IN1 größer als IN2	IN1 ANY IN2 ANY	OUT BOOL
GE	Ergibt TRUE, wenn IN1 größer oder gleich IN2	IN1 ANY IN2 ANY	OUT BOOL
EQ	Ergibt TRUE, wenn IN1 gleich IN2	IN1 ANY IN2 ANY	OUT BOOL
LE	Ergibt TRUE, wenn IN1 kleiner oder gleich IN2	IN1 ANY IN2 ANY	OUT BOOL
LT	Ergibt TRUE, wenn IN1 kleiner als IN2	IN1 ANY IN2 ANY	OUT BOOL
NE	Ergibt TRUE, wenn IN1 ungleich IN2	IN1 ANY IN2 ANY	OUT BOOL
Zähler			
CTU	Zählt CV mit jeder steigenden Flanke in CU um 1 hoch. Wird PV überschritten wird zusätzlich Q auf TRUE gesetzt. Erreicht der Zähler PVmax, wird nicht weiter gezählt. TRUE an R setzt den zähler auf 0 zurück.	CU BOOL R BOOL PV INT PVmax INT	Q BOOL CV INT
CTD	Zählt CV mit jeder steigenden Flanke in CD um 1 herunter. Wird 0 unterschritten wird zusätzlich Q auf TRUE gesetzt. Erreicht der Zähler PVmin, wird nicht weiter gezählt. TRUE an LD setzt den zähler auf PV zurück.	CD BOOL LD BOOL PV INT PVmin INT	Q BOOL CV INT
CTUD	Kombiniert CTU und CTD. Steigende Flanken an CU erhöhen CV um 1, an CD verringern CV um 1. QU und QD werden beim Überschreiten von PV bzw. beim unterschreiten von 0 gesetzt. Bei Über- bzw Unterschreiten von PVmax bzw. PVmin wird nicht weiter gezählt. TRUE an R setzt CV auf 0, an LD auf PV zurück.	CU BOOL CD BOOL R BOOL LD BOOL PV INT PVmax INT PVmin INT	QU BOOL QD BOOL CV INT
Timer			
TP	Steigende Flanke an IN setzt Q für die in PT angegebene Zeitspanne.	IN BOOL PT TS(*)	Q BOOL ET TS(*)
TON	Einschaltverzögerung	IN BOOL PT TS(*)	Q BOOL ET TS(*)
TOFF	Ausschaltverzögerung	IN BOOL PT TS(*)	Q BOOL ET TS(*)

(*) TS: TIME_SPAN: Variable für Zeitspannen

Bistabile Blöcke			
SR	FlipFlop mit Dominanz auf Set	S1 BOOL R BOOL	Q1 BOOL
RS	FlipFlop mit Dominanz auf Reset	S BOOL R1 BOOL	Q1 BOOL
Flankenerkennungen			
RTRIG	Setzt Q für einen Zyklus auf TRUE, wenn eine steigende Flanke er in CLK erkannt wird.	CLK BOOL	Q BOOL
FTRIG	Setzt Q für einen Zyklus auf TRUE, wenn eine fallende Flanke er in CLK erkannt wird.	CLK BOOL	Q BOOL
Funktionen für Zeichenketten			
LEN	Gibt die Länge einer Zeichenkette zurück.	IN1 STRING	OUT UINT
LEFT	Gibt L Zeichen von der linken Seite von IN an OUT weiter.	IN STRING L UINT	OUT STRING
RIGHT	Gibt L Zeichen von der rechten Seite von IN an OUT weiter.	IN STRING L UINT	OUT STRING
MID	Gibt L Zeichen ab dem P-ten Zeichen von IN an OUT weiter.	IN STRING L UINT P UINT	OUT STRING
CONCAT	Konkateniert IN1 und IN2.	IN1 STRING IN2 STRING	OUT STRING
INSERT	Setzt IN2 ab der Position des P-ten Zeichens in IN1 ein	IN1 STRING IN2 STRING P UINT	OUT STRING
DELETE	Löscht L Zeichen ab dem P-ten Zeichen von IN.	IN STRING L UINT P UINT	OUT STRING
DELETE	Ersetzt L Zeichen ab der Position des P-ten zeichens in IN1 durch IN2.	IN1 STRING IN2 STRING P UINT L UINT	OUT STRING
FIND	Gibt die Position des ersten Auftretens von IN2 in IN1 an.	IN1 STRING IN2 STRING	OUT UINT

Sonstige Funktionsblöcke			
StateWatch	Überwacht den Status der Variable IN. Der aktuelle Status wird in der Variable CurState ausgegeben. In HasState wird angegeben, ob IN überhaupt ein Status-Flag hat. Die Ausgänge CXxx geben die Aktuellen Statusflags wieder. Die Ausgänge HXxx bleiben bei Auftreten so lange gesetzt, bis sie über Reset zurückgesetzt werden. Der Ausgang CurState ist vom Typ UINT, OUT von Typ ANY. Alle anderen Ausgänge sind vom Typ BOOL. Aus Platzgründen sind die Typen rechts nicht angegeben.	IN ANY Reset BOOL	OUT ANY CurState HasState CBad CGood CQuestionable CUnknown HBad HGood HQuestionable HUnknown

Der Block *StateWatch* dient der Überwachung der Status-Flags von ANY-Variablen. Über die Ausgänge CXxx kann der Status der Variable im aktuellen Zyklus ausgelesen werden, zum Beispiel um in folgenden Blöcken auf eine ungültige (CBad == TRUE) Eingabe zu reagieren. Die Ausgänge HXxxx halten einen Aufgetretenen Status fest, bis sie über Reset zurückgesetzt werden. Sie können so als Rückmeldung an den operator dienen.

2.1 Benutzung des Datentyps OV_ANY

Der Datentyp OV_ANY ist in der Lage jeden im OV-System definierten Datentyp zu repräsentieren. Da die meisten Blöcke mit vielen verschiedenen Typen arbeiten können bietet sich der Einsatz des Typs OV_ANY hier an. Trotzdem unterstützt nicht jeder Block jeden Typ. Die Ausnahmen sind bereits in der Erklärung der Blöcke dargestellt, daher wird hier aus Wiederholung verzichtet.

Um Typensicherheit zu gewährleisten ist eine „Umschaltung“der Datentypen an den Eingängen der Bausteine nicht immer erlaubt. Um zu vermeiden, dass sich während der Laufzeit eines Programms Typenkonflikte ergeben, kann der Datentyp innerhalb der ANY-Struktur am Eingang eines Funktionsbausteins nur geändert werden, wenn der Funktionsbaustein über keine Verbindung zu anderen Funktionsbausteinen (fb_connection, oder davon abgeleitet) verfügt.
Ist der Baustein verbunden, so führt ein Versuch der Änderung des Datentyps zu einer Fehlermeldung `OV_ERR_NOACCESS`. Das bedeutet, dass der Typ, mit dem ein Funktionsbaustein arbeiten soll festgelegt werden muss, bevor der Baustein mit anderen Verbunden wird. Dies kann durch einmalige Zuweisung eines Wertes umgesetzt werden. Bei Instantiierung ist für arithmetische und numerische Funktionsbausteine sowie für Vergleichsoperationen der Typ OV_SINGLE voreingestellt. Bitweise Logikbausteine sind auf OV_BOOL eingestellt.

2.2 Umgang mit Überläufen und undefinierten Operationen

Bei den mathematischen Funktionen kann es je nach Eingangsparametern zu undefinierten Operationen (z. B. Division durch 0) kommen. Solche Zustände werden erkannt und die Funktionsbausteine reagieren entsprechend.

Bei einer Division durch 0 von Realzahlen wird das Ergebnis für positive Dividenden auf `+INF` (positive Unendlichkeit) und für negative Dividenden auf `-INF` gesetzt. Wird 0 durch 0 geteilt, so ist das Ergebnis ebenfalls 0. In allen drei Fällen wird der Status der Ergebnisvariablen auf `OV_ST_BAD` gesetzt.
Eine Division durch 0 von Integern wird gar nicht erst ausgeführt, da es hierdurch, je nach FB-System zu einem Server-Absturz kommen kann. Das Ergenis wird, analog zu Realzahlen, auf den größtmöglichen bzw. kleinstmöglichen Integer-Wert oder 0 gesetzt. Auch hier wird der Status auf `OV_ST_BAD` gesetzt.

Wird versucht, einer numerische Funktion ein Parameter außerhalb des Definitionsbereiches zu übergeben (z. B. `asin(2)`) so reagiert die Funktion mit der Rückgabe des Wertes `NaN` (Not a Number) und Setzen des Status `OV_ST_BAD`.

Kommt es bei einer Rechenoperation zu einem Werteüber- oder -unterlauf, so wird dieser bei Realzahlen erkannt. Die Ausgangsvariable führt dann den Wert `HUGE_VAL` (`+INF`) bzw. `-HUGE_VAL` (`-INF`). Auch hier wird der Status auf `OV_ST_BAD` gesetzt.
Bei einer Rechnung mit Integern kann ein Über- oder Unterlauf nicht erkannt werden. Die überlaufenden Bits werden einfach verworfen. Eine Erkennung der Über- / Unterläufe wäre möglich durch die Berechnung als Fließkommazahl und anschließende Rückkonvertierung. Dies würde jedoch viel Rechenzeit erfordern, so dass die Rechnung nicht mehr schneller wäre, als der Umgang mit Realzahlen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis