

1 Wsp Softwaredokumentation

Autor: Michael Münch

2 Revision History

Autor: Primove SW

2.1 Revions

Autor	Datum	ID	Modul	Version	Änderungen
Michael Münch	25.11.2017	Init	Alle	x420 Pre20	Initial Version
Michael Münch	08.12.2017	Init	Protec	x420Pre30	Initial Version
Michael Münch	27.12.2017	Init	WspStMgr	x420Pre40	Initial Version
Michael Münch	28.11.2017	Init	Alle	x420Pre40	Überarbeitung der Struktur
Michael Münch	05.01.2018	165204	WspSfty	x420Pre40	Implementierung Eingangssignalqualifikation
Michael Münch	06.01.2017	80527	WspStMgr	x420Pre40	Implementierung Z-Mover AWC Statusmessages
Michael Münch	06.03.2018	Init	WspLedMgr	x430Pre10	Prepare Docu of LedMgr
Michael Münch	13.03.2018	Init	PwrTrf	x430Pre20	Prepare Docu of PwrTrf
Michael Münch	09.04.2018	-	PwrTrf	x430Pre30	Optimization of Charge Control
Michael Münch	23.04.2018	-	WspStMgr	x430Pre40	Adapt Docu to Changes in WspStMgr
Michael Münch	04.05.2018	190644	WspSafety, HwloAbs, Protect	x430Pre40	Implementation of RCD Selftest
Michael Münch	11.05.2018	205575	WspSafety	x430Pre50	BugFix DTC bei ASC

3 Namenskonventionen

Autor: Michael Münch

3.1 Allgemein

Im Nachfolgenden werden die gemäß der Autosar Requirements verwendeten Abkürzungen und Namenskonventionen für (Mess-)Variablen, Parameter und Funktionen beschrieben.

Siehe hierzu auch: „Guidelines model based Development.doc“ V1.6

3.2 Abkürzungen

3.2.1 Abkürzungen für Variablenpräfix der physikalische / Logischen Größen

(werden immer klein geschrieben)

Bedeutung	Phys. / Präfix
Acceleration [m/s ²]	a
Angle [°Crank]	phi
Counter [-]	cnt
Current [A]	i
Cycles [-]	cyc
Engine speed [rpm]	n
Factor [-]	fac
Force [N]	f
Height [m, cm, mm]	h
Index [-] (integer Value)	idx
Lenght [m, cm, mm]	l
Mass [kg]	m
Number [-]	num
Power [W]	pwr
Pressure [bar]	p
Ratio [%]	r

Record (complex Datatype / Structure)	rec
State [-]	st
Switch [-]	swt
Temperature [°C]	t
Time [s,ms, μs]	ti
Torque [Nm]	trq
Velocity [km/h]	v
Voltage [V]	u

3.2.2 Definierte Abkürzungen

Begriff	Abkürzung
Absolute	Abs
Maximum	Max
Minimum	Min
Value	Val
Average	Avrg
Signal Range Check	SRC

3.3 Namenskonventionen

Alle Namen sollten in Englisch gehalten werden. Die Gesamtlänge eines Namens sollte 40 Buchstaben nicht überschreiten.

3.3.1 Modulnamen (Dateinamen)

Sinnvolle Modulbezeichnung aus 3 bis 9 Buchstaben. Erster Buchstabe immer groß. Bei Modulnamen die aus Abkürzungen mehrerer Wörter besteht jeweils der Anfangsbuchstabe des konstituierenden Wortes groß.

Beispiel

OruChrgMgr (OruChargingManager)

3.3.2 (Modul)Lokale Variablen

Beginnen immer mit der physikalisch / Logischen Einheit.

Beispiel

stInternal

3.3.3 Mess- und globale Variablen

Messvariablen sind globale Variablen die im Applikationstool angezeigt werden sollen. Sie beginnen immer mit dem Modulnamen gefolgt von einem Unterstrich.

ModName_xMeaning

Beispiel: Aktueller Status der Oru Charging Manager Statemachine

OruChrgMgr_stChrgMode

3.3.4 Applikationsgrößen

Sind aufgebaut wie Messvariablen, mit dem Unterschied dass ihnen noch ein die Art der Applikationsgröße bezeichnendes Postfix nachgestellt wird; mit:

_C	für Kennwerte
_Cur	für Kennlinien
_Map	für Kennfelder
_CA	für Arrays

Beispiel

OruChrgMgr_uDCUpLim_C

4 Bibliotheksfunktionen

Autor: Michael Münch

4.1 Allgemein

Zum einfacheren Handling und um die Modellbasierte Software besser nachvollziehbar zu gestalten, werden die im folgenden beschriebenen TargetLink Bibliotheksfunktionen zur Verfügung gestellt. Alle Komponenten der Bibliotheksfunktionen befinden sich in: trunk\Au-di\07_Development\40_SW\30_Implementation\08_LIB\15_TMPL\LIB_TMPL.

Alle hier beschriebenen Bibliotheksfunktionen werden genau wie Targetlink Funktionen inline in den generierten Code geschrieben, so daß zum einen die Laufzeit optimiert ist und zum anderen keine Wechselwirkungen bei mehrfacher Verwendung in einem Modul möglich sind.

4.2 (Mess-) Variablen, Applikationskonstanten

4.2.1 Measurement Variable

Die Messvariable ist eigentlich ein Rescaler. Sie dient zur Aufnahme von Messvariablen, welche im Data Dictionary definiert sind. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben.



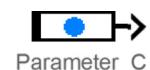
4.2.2 Globale Variable

Die globale Variable ist eigentlich ein DataStore Block. Sie dient zum Verwenden von globalen Variablen, welche im Data Dictionary definiert sind. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben. Das Template *GlobVar* dient hierbei zur Definition bzw. Deklaration der globalen Variable. Mit *GetVal* kann diese gelesen und mit *SetVal* geschrieben werden.



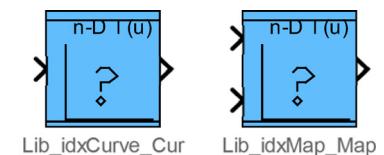
4.2.3 Parameter (Applikationskonstante)

Der Parameter ist eigentlich ein Konstante. Sie dient zur Aufnahme von Applikationskonstanten, welche im Data Dictionary definiert sind. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben.



4.2.4 Map, Curve (Applikationskonstante)

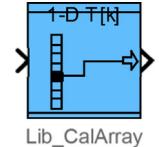
Der Parameter Map ist ein Kennfeld und Curve eine Kennlinie. Sie dienen zur Aufnahme von Kennlinien und Kennfeldern, welche im Data Dictionary definiert sind. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben.



4.2.5 Array

Softwaredokumentation

Das Array ist ein TL-Array Block. Es dient zur Aufnahme von Arrays, welche im Data Dictionary definiert sind. Der Eingang ist der Null basierte Index des auszugebenden Elements. Das DataDictionary Array muss in dem Output-Tab als Variable selektiert werden. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben.



4.2.6 Define

Der Define ist eigentlich ein Konstante. Er dient zur Aufnahme von Defines, welche im Data Dictionary definiert sind. Es sind Callback Funktionen implementiert, welche den Variablenamen bei Bewegung des Blocks automatisch unter den Block schreiben.



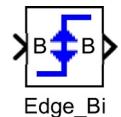
4.3 Logische Hilfsfunktionen

4.3.1 Trigger

Die Triggerfunktionen reagieren auf einen Flankenwechsel des Bool'schen Eingangs. Bei entsprechender Veränderung wird der Bool'sche Ausgang für einen Taskzyklus auf *true* gesetzt.

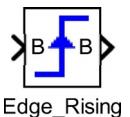
4.3.1.1 Trigger beide Flanken

Bei einem Wechsel des Eingangssignals von *true* auf *false* oder umgekehrt wird der Ausgang für einen TaskZyklus auf *true* gesetzt.



4.3.1.2 Trigger steigende Flanke

Bei einem Wechsel des Eingangssignals von *false* auf *true* wird der Ausgang für einen TaskZyklus auf *true* gesetzt.



4.3.1.3 Trigger fallende Flanke

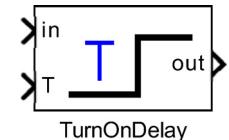
Bei einem Wechsel des Eingangssignals von *true* auf *false* wird der Ausgang für einen TaskZyklus auf *true* gesetzt.



4.3.2 Turn On Delay

Softwaredokumentation

Das Turn On Delay gibt am Ausgang *out* ein *true* aus, wenn der Eingang *in* seit mindestens der Zeit *T* konstant auf *true* ist. Ist der Eingang *false* wird der Ausgang sofort auf *false* gesetzt und die Delayzeit resetet,

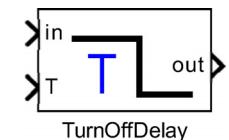


4.3.2.1 Delayzeit T

Die Delayzeit *T* ist ein unsigned integer Datentyp mit einem Scaling. Das Scaling skaliert den Zeitwert mit der Einheit Sekunde bezüglich der Task Periodenzeit des Moduls, in welchem der Delayblock verwendet wird. Für einen 10ms Task muss also das Scaling *Cnv_TimeConst10ms* mit LSB 0.01 verwendet werden.

4.3.3 Turn Off Delay

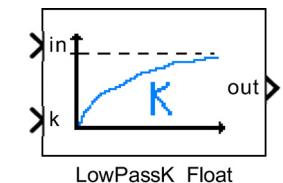
Das Turn Off Delay gibt am Ausgang *out* ein *false* aus, wenn der Eingang *in* seit mindestens der Zeit *T* konstant auf *false* ist. Ist der Eingang *true* wird der Ausgang sofort auf *true* gesetzt und die Delayzeit resetet. Die Delayzeit *T* entspricht der Beschreibung beim Turn On Delay.



4.4 Regelungstechnische Funktionen

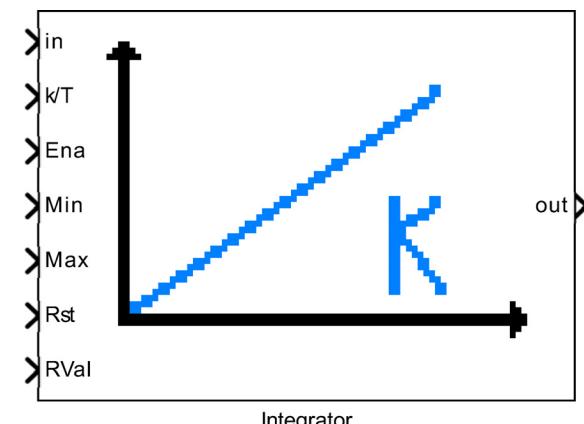
4.4.1 Lowpassfilter (PT1)

Hiermit wird ein Lowpassfilter mit PT1 Charakteristik implementiert. Das Gain *k* entspricht hierbei dem Produkt aus der Zeitkonstanten mit der Samplerate. Das Gain *k* hat hierbei folgende Eigenschaften: Beim Maximalwert 1 findet keine Filterung statt, beim Minimalwert 0 sperrt der Filter. Im Intervall [0, 1] wird die Filterung um so stärker, je kleiner der Wert *k* ist. Standardmäßig kann die float Implementierung verwendet werden. Für zeitkritische Anwendungen stehen aber noch 16 Bit Implementierungen zur Verfügung.



4.4.2 Integrator

Beim Integrator wird zunächst der Eingang *in* mit dem Gain *k/T* multipliziert. Dieses Produkt wird bei jedem Taskdurchlauf auf den internen Speicher (Akkumulator) des Integrators aufaddiert. Der Wert des Akkumulators wird hier nach oben hin durch den Wert am Eingang *Max* und nach unten hin durch den Wert am Eingang *Min* begrenzt. Für den Fall, daß der Enable-Eingang *Ena* den Wert *false* hat, wird der Integrator angehalten und der Wert des Akkumulators eingefroren. Für den Fall, daß der Reset-Eingang *Rst* den Wert *true* hat, wird der Akkumulator auf den Wert des Eingangs *RVal* gesetzt. Schließlich wird der Wert des Akkumulators am Ausgang *out* ausgegeben.



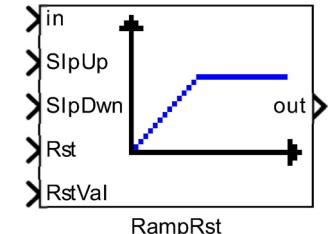
4.4.3 Rampe

Die Ramp - Funktion stellt einen Gradientenbegrenzer dar. Hierbei wird der Ausgang *out* mit dem Gradienten *SlpUp* in positiver Richtung hin zum Eingang *in* gerampt bzw. in negativer Richtung mit dem Gradienten *SlpDwn*. Für den Fall, daß der Reset - Eingang *Rst* den Wert *true* hat, wird der interne Rampwert auf den Wert des Eingangs *RstVal* gesetzt..

Interface: Bei den Interfaceeinstellungen ist folgendes zu Beachten:

Die Inputs *in*, *RstVal*, der Output *out* sowie die Work Variables *oldVal* und *outVal* müssen vorzeichenbehaftete Integerwerte mit der doppelten Bitbreite des ursprünglich zu rampenden Wertes sein. Das Scaling dieser Werte ergibt sich aus dem Scaling des ursprünglich zu rampenden Wertes multipliziert mit dem Zeitkonstantenwert des Tasks, in welchem die Rampfunktion aufgerufen wird. Die Eingänge *SlpUp* sowie *SlpDwn* haben den selben Datentyp und Scaling wie der Ursprünglich zu Rampende Wert, wobei die Einheit [Phys/s] ist.

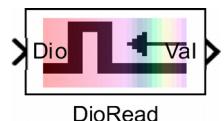
Beispiel: Zu Rampender Wert hat die Einheit [A] mit dem Scaling *Cnv_Current*, LSB = 0.1, uint16. Die Funktion wird im 10ms Task aufgerufen, also ist die entsprechende Zeitkonstante 0.01(s). Daraus ergibt sich für die Interfaceeinstellungen für *in*, *RstVal*, *out*, *oldVal* und *outVal* der Datentyp sint32 mit Scaling 0.001 (*Cnv_GradCurrent_Ramp10ms*). *SlpUp* sowie *SlpDwn* sind dann uint16 mit Scaling *Cnv_GradCurrent*, uint16 LSB = 0.1 [A/s].



4.5 HW Zugriffsfunktionen

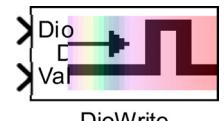
4.5.1 DioRead

Digitalen Eingangswert lesen. Es wird der Boolesche Wert des digitalen Eingangs *Dio* ausgegeben. Der Eingang *Dio* der Funktion ist hierbei der Tresos - Define des entsprechenden HW Ports.



4.5.2 DioWrite

Digitalen Ausgangswert schreiben. Es wird der Boolesche Wert *Val* des digitalen Ausgangs *Dio* ausgegeben. Der Eingang *Dio* der Funktion ist hierbei der Tresos - Define des entsprechenden HW Ports.



4.5.3 PwmWrite

PWM Ausgangswert schreiben. Es wird die Periodendauer *Per* und der DutyCycle *Dty* des PWM Ausgangs *Chn* ausgegeben. Der Eingang *Chn* der Funktion ist hierbei der Tresos - Define des entsprechenden HW Ports.

5 Modul ErrHndl (Error Handler)

Autor: Michael Münch

Software-Struktur-Position: 30_Implementation/10_WSP/08_ERRHNDLR

5.1 Allgemein

Der Error Handler verwaltet alle vom der Software festgestellten Fehlerzustände. Hier findet auch die Entprellung der Fehler statt. Die Fehlerreaktionen können hier jedem einzelnen Fehler zugeordnet werden. Ebenso ist hier das setzen der jeweiligen DTC's implementiert.

5.2 Komponente Konfiguration der Fehler

The screenshot shows the DiagConfig software interface for configuring fault components. The main window title is "DiagConfig". The left sidebar lists "Alle Access-Objekte" (All Access Objects) under "Tabellen" (Tables), with "DSListe" selected. The main panel displays the configuration details for the selected object:

- Bezeichner:** P3V3BTVoltageSensorOverVol
- Defekt Bedingung:** OruloHwAbs_uAdcP3V3BT
 \geq
OruloHwAbs_uP3V3BTHi_C
- Heil Bedingung:** Defektbedingung liegt nicht mehr an
- Name des Fehlerspeicherintrags:** ID: 1
- Prosa:** Beschreibung, wann Defektmeldung an System erfolgt.
- Zeit bis endgültig Defekt:** 100 ms
- Zeit bis endgültig Geheilt:** 10 ms
- Modul:** OruHwloAbs
- DTC:** 0x021019
- Fehlerklasse:**
 - Aktiv
 - Deb Up/Down
 - zMoverDwn
 - Safe State
 - MIL
 - Inhibit Charging
 - Temperature Delay
 - Freeze Frame

Buttons at the bottom include "Datensatz hinzufügen" (Add Record), "ProjektPfad" (Project Path), "Übernehmen" (Accept), "ParameterListe" (Parameter List), and "Code generieren" (Generate Code). The status bar at the bottom shows "Formularansicht" (Form View), "Datensatz: 1 von 1", "Kein Filter" (No Filter), "Suchen" (Search), and various navigation icons.

Der Error Handler wird über das Formular der Access Datenbank ErrHndlConfig.mdb konfiguriert. Diese befindet sich jeweils im Verzeichnis '\08_ERRHNDLR' des Projekts. Hier muss in obiger Eingabemaske der Bezeichner des Fehlers eingetragen werden. Aus diesem Bezeichner werden später die Namen der jeweiligen Statusvariablen sowie der Applikationsparameter.

Für die Defekt und Heilungs-Entprellzeiten sowie der Fehlerklasse generiert. Daher muss der Bezeichner die Einschränkungen wie bei C Variablen Namen einhalten. Des Weiteren sollte jeweils eine Beschreibung der Fehlerbedingung, also unter welchen Umständen dem Diagnosesystem ein Fehler gemeldet wird, sowie der Heilbedingung, d.h. wann an das Diagnosesystem OK gemeldet wird eingetragen werden. Im Folgenden müssen die Voreinstellungen für die Defekt und Heil- Entprellzeiten eingegeben werden (Auflösung 10ms) sowie die im Folgenden beschriebene Fehlerklasse.

5.2.1 Bedienung der Benutzeroberfläche

Button Datensatz hinzufügen:

Durch betätigen dieses Buttons wird mit den aktuell Angezeigten bzw. Eingegebenen Werten ein neuer Datensatz erstellt, wobei die letzte ID inkrementiert wird.

Button Übernehmen:

Durch betätigen dieses Buttons wird der Datensatz mit der Angezeigten ID mit den aktuell Angezeigten bzw. Editierten Werten überschrieben.

Button Code generieren:

Durch betätigen dieses Buttons werden die folgenden Dateien erzeugt:

- ErrHndlDef.c: Hier werden die unten beschriebenen Variablen und Applikationsgrößen definiert
- ErrHndlDef.h: Hier werden die Variablen und Applikationsgrößen deklariert, die benötigten Typen angelegt sowie die Indices der einzelnen Fehlerpfade definiert.
- ErrHndl.a2l: Die a2l Beschreibungsdatei für INCA oder CanApe.
- Eine textuelle Zusammenfassung der Konfiguration mit der Bezeichnung „DiagDef.txt“ im Doc - Verzeichnis des jeweiligen Projektpfades.

Der Codegenerator entnimmt im Projektpfad im Tresos Output Ordner aus der DEM Config- Headerdatei die Indices der für die jeweiligen Fehler Eingetragenen DTC's und speichert diese in seiner eigenen Konfiguration ab. Wird ein DTC nicht gefunden, wird der Fehler auf den DTC mit index 0 gemappt und es erfolgt ein diesbezüglicher Hinweis.

Button Projektpfad:

Mittels dieses Buttons kann der Projektpfad in welchen die Dateien generiert werden ausgewählt werden.

Beispiel: D:\br_CR_B43_DEV_CRQ\Audi\07_Development\40_SW\30_Implementation\10_WSP

Der ausgewählte Projektpfad wird in der Datenbank gespeichert.

5.2.2 Statusvariable (XCP Messgröße)

Der Name der Statusvariablen des jeweiligen Fehlers lautet *ErrHndl_stBezeichner*. Diese Statusvariable ist ein 8-Bit Wert mit folgender Bit-Belegung:

Bit	Bedeutung
0	25% defekt
1	50% defekt
2	75% defekt
3	100% defekt
4	Fehler wurde nach Start des Steuergerätes mindestens 1 mal entprellt (Readynes Bit)
5	Fehler war mindestens 1 mal auf endgültig defekt
6	Fehler aktuell im Zustand endgültig defekt - wird erst bei endgültig geheilt wieder gelöscht

5.2.3 Fehlerklasse (Applikationsparameter)

Der Name der Fehlerklasse lautet: *ErrHndl_Class_Bezeichner_C*. Dieser Parameter ist ein 8-Bit Wert mit folgender Bitbelegung:

Bit	Bedeutung
0	Aktiviert - 1 = Fehler wird bearbeitet, 0 = Fehlerbehandlung ist abgeschaltet
1	Entprellart - 1: Endzustand, 0: Up/Down (siehe unten)
2	Z-Mover Down, bei endgültig Defekt wird der Z-Mover runtergefahren
3	Safe State bei endgültig Defekt wird der Sichere Zustand geschaltet
4	MIL - wenn gesetzt, wird beim Erreichen des endgültig defekt Zustandes die MIL - Lampe eingeschaltet
5	Inhibit Charging, bei endgültig Defekt wird das Laden verhindert / abgebrochen
6	Ladeverzögerung Temperatur
7	tbd.

5.2.3.1 Entprellart

1 - Endzustand; Der Fehlerzähler wird beim Anliegen des Defektzustandes bis zum Erreichen des *ErrHndl_tIDebDef_Bezeichner_C* Wertes hochgezählt. Erst im Zustand endgültig defekt wird der Fehlerzähler beim Anliegen des Ok Zustandes wieder herunter gezählt und ist nach *ErrHndl_tIDebOk_Bezeichner_C* ms wieder geheilt. HINWEIS: Ist der Wert für *ErrHndl_tIDebOk_Bezeichner_C* auf 0xFFFF appliziert, kann ein einmal als endgültig defekt entprellter Fehler nur noch durch eine *ErrHndl_Clearxxx* geheilt werden. Eine selbstständige Heilung bei nicht mehr anliegender Defektbedingung erfolgt dann nicht.

0 - Up/Down; Der Fehlerzähler wird beim Anliegen des Defektzustandes immer hochgezählt und beim Anliegen des Ok Zustandes immer herunter gezählt. Überschreitet der jeweilige Fehlerzähler den Applikationswert *ErrHndl_tIDebDef_Bezeichner_C*, gilt der Fehler als endgültig defekt, unterschreitet er den Wert *ErrHndl_tIDebOk_Bezeichner_C* gilt er als geheilt. Ein Wert für *ErrHndl_tIDebOk_Bezeichner_C* kleiner als 10ms bedeutet, dass der Fehler nicht geheilt werden kann.

5.2.4 Globale Statusvariable (XCP Messgröße)

Es werden folgende globale Statusvariablen angelegt:

- *ErrHndl_LastError* - enthält die Indexnummer des letzten aktiven Fehlers, wobei im Falle mehrerer anliegender Fehler immer der mit der Höchsten Indexnummer angezeigt wird.
- *ErrHndl_FirstShOffErr* - enthält die Indexnummer des ersten aktiven Fehlers, welcher ein Inhibit Charging auslöst.
- *ErrHndl_numErrors* - enthält die Anzahl der momentan endgültig defekten Fehler.
- *ErrHndl_stGlobal* - Globaler Fehlerstatus. Der globale Fehlerstatus ist bitcodiert und hat folgende Belegung:

Bit	Bedeutung
0	MIL - mindestens ein Fehlerpfad mit Fehlerklasse MIL aktivieren ist endgültig defekt
1	tbd.
2	Ladeverzögerung Temperatur aktiv
3	tbd.
4	tbd.
5	Inhibit Charging aktiv
6 - 15	tbd.

5.2.5 Schnittstellen

5.2.5.1 Fehlerzustand melden

Um den aktuellen Zustand eines Fehlers dem Errorhandler zu melden steht jedem Runnable folgender Event zur Verfügung: *rpSWCNameError* mit der Operation SetError-State. Der Event hat folgende Parameter:

ErrorId: (uint16) Index des Fehlers

ErrorState: (uint8) aktueller Fehlerzustand. 0 – kein Fehler, 1 – Fehler

5.2.5.2 Abfrage Inhibit Charging

Zur Abfrage, ob ein Fehler den Inhibit Charging Zustand (Laden verhindern, bzw. Abbrechen) ausgelöst hat, steht der Event *rpEHInhibitCharging* mit der Operation Inhibit-Charging zur Verfügung. Dieser hat den Rückgabewert YesNo (bool). Hierbei bedeutet *True* = Ladeverhinderung aktiv, *False* = alles ok, keine Ladeverhinderung.

5.2.5.3 Abfrage Ladeverzögerung Temperatur

Zur Abfrage, ob ein Fehler den Laderverzögerung Temperatur Zustand (Laden verhindern, bzw. Abbrechen wegen Übertemperatur) ausgelöst hat, steht der Event *rpEHTempDly* mit der Operation *TempDly* zur Verfügung. Dieser hat den Rückgabewert YesNo (bool). Hierbei bedeutet *True* = Ladeverhinderung Temperatur aktiv, *False* = alles ok, keine Ladeverhinderung Temperatur.

5.2.5.4 Errorhandler Main-Funktion

Die Funktion void *ErrorHandler_reErrorHandlerMain10ms(void)* wird im 10ms Task aufgerufen. Hier werden alle konfigurierten Fehler zyklisch entprellt und die oben beschriebenen Stati gesetzt. Ebenso wird das aktuell entprellte Ergebnis zyklisch dem DEM mitgeteilt.

5.2.5.5 void ErrHndl_ClearOne(uint16 ErrNr)

Diese Funktion dient zum zurücksetzen eines einzelnen Fehlers. Sie kann ausgelöst werden, indem in den XCP Parameter *ErrHndl_numClrOne_C* der Gewünschte Fehlerindex geschrieben wird. Die Funktion wird dann einmal mit dem aktuellen Wert von *ErrHndl_numClrOne_C* aufgerufen, wenn sich dieser Wert seit dem letzten Aufruf geändert hat.

5.2.5.6 void ErrHndl_ClearAll(uint16 ErrNr)

Diese Funktion dient zum löschen aller Fehler. Sie wird aufgerufen, wenn die der Wert des XCP Parameters *ErrHndl_stClrAll_C* seit dem letzten Aufruf geändert hat.

6 Modul PwrTrf (WaySide PowerTransfer)

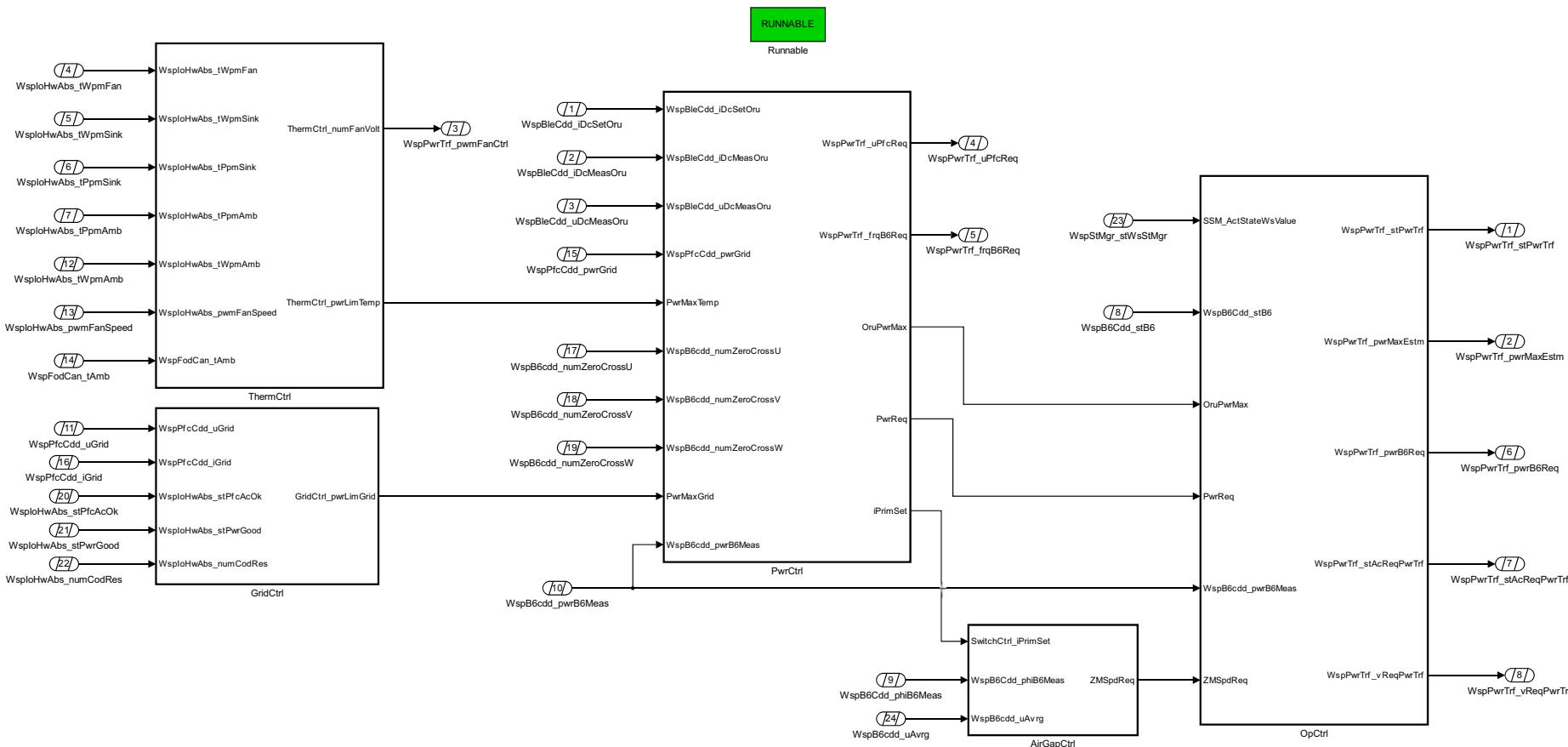
Autor: Bogdan Marcu

Software-Struktur-Position: 30_Implementation/10_WSP/10_PWRTRF

Softwaredokumentation

primove
true e-mobility

6.1 Allgemein



Softwaredokumentation

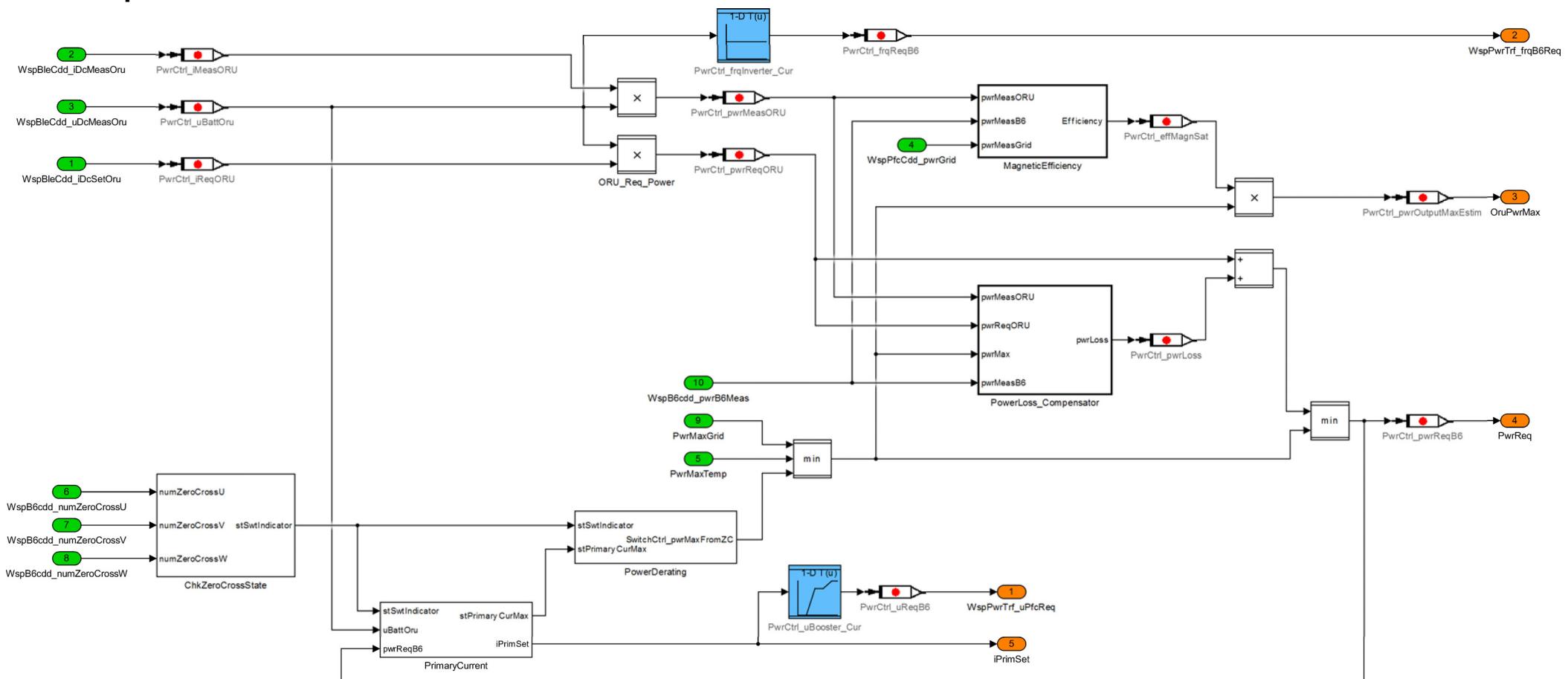
Das Modul *PwrTrf* implementiert die Energieübertragungsfunktionen der Wayside für das induktive Laden.

Das Modul *PwrTrf* enthält folgende Komponenten:

- *PwrCtrl* - Berechnung der aktuellen Soll und Max Ladeleistung
- *ThermCtrl* - Temperatur Derating und Lüfterkontrolle
- *GridCtrl* - Derating bezüglich Netz Eingangsstrom und Spannung
- *AirGapCtrl* - Regulierung der Phasenverschiebung der Primärströme mittels Z-Mover Höhe
- *OpCtrl* - Regelung und Überwachung der B6-Brücke

Softwaredokumentation

6.2 Komponente PwrCtrl



In der Komponente *PwrCtrl* wird die aktuell einzuregelnde Ladeleistung *PwrCtrl_pwrReqB6* (*PwrReq*) für den Komponententreiber der B6 Brücke sowie die maximal erreichbare Ladeleistung *PwrCtrl_pwrOutputMaxEstim* (*OruPwrMax*), welche der Oru übermittelt wird, bestimmt.

Hier wird auch im Block *PrimaryCurrent* der einzuregelnde Primärstrom *SwitchCtrl_iPrimSet* (*iPrimSet*) bestimmt, welcher über die Boostspannung und den Luftspalt eingeregt wird.

Es besteht hier zusätzlich die Möglichkeit, in Abhängigkeit der HV Batteriespannung, mittels der Kennlinie *PwrCtrl_fqInverter_Cur* die Inverterfrequenz anzupassen (Standardmäßig 85.5kHz).

Softwaredokumentation

Die an die Oru zu übermittelnde maximale Ladeleistung *PwrCtrl_pwrOutputMaxEstim* wird berechnet, indem das Minimum der in den jeweiligen Komponenten bestimmten Leistungsbegrenzungen *PwrMaxGrid*, *SwitchCtrl_pwrMaxFromZC* (vom Block *PowerDerating*) sowie *PwrMaxTemp* mit der im Block *MagneticEfficiency* berechneten Übertragungseffizienz *PwrCtrl_effMagnSat* multipliziert wird.

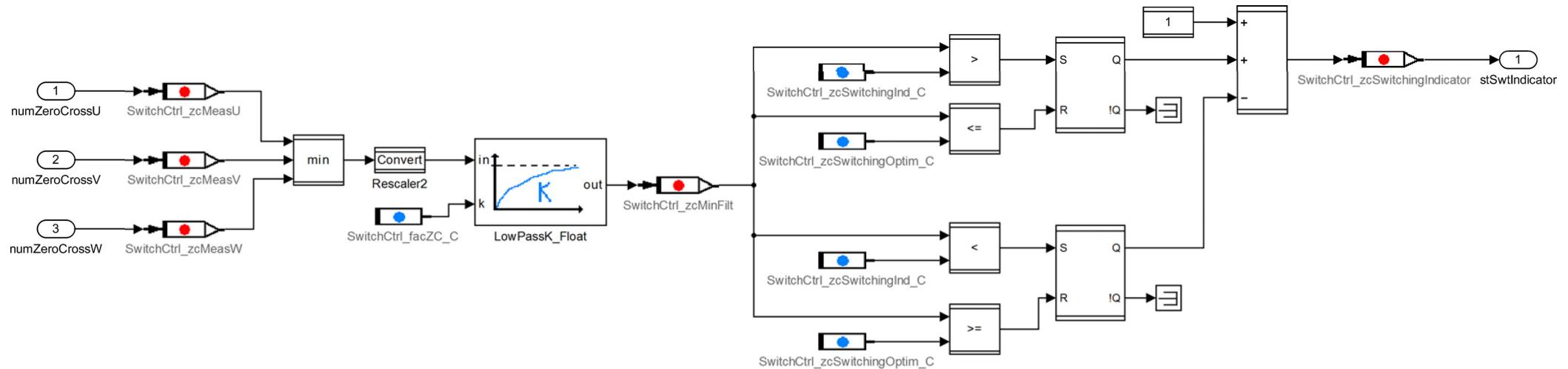
Zur Bestimmung der Soll - Ladeleistung *PwrCtrl_pwrReqB6* wird zunächst die angeforderte Ladeleistung *PwrCtrl_pwrReqORU* als Produkt aus der HV Batteriespannung *PwrCtrl_uMeasORU* und dem von der Oru geforderten Ladestrom *PwrCtrl_iReqORU* berechnet. Auf diese wird dann die im Block *PowerLoss_Compensator* bestimmte Verlustleistung *PwrCtrl_pwrLoss* addiert. Die Soll - Ladeleistung *PwrCtrl_pwrReqB6* ist dann die durch das Minimum der zuvor genannten drei Maximalbegrenzungen limitierte angeforderte und verlustkompensierte Ladeleistung *PwrCtrl_pwrReqORU*.

Dann wird die für die Effizienzberechnung sowie für die Berechnung der Verlustleistung benötigte Ist - Ladeleistung *PwrCtrl_pwrMeasORU* berechnet. Diese ist das Produkt aus dem aktuell gemessenen Ladestrom *PwrCtrl_iMeasORU* sowie der aktuellen HV Batteriespannung *PwrCtrl_uMeasORU*.

Der für das Powerderating sowie zur Berechnung des Soll - Primärstroms benötigte Schaltzustand *stSwtIndicator* (kapazitiv, induktiv oder optimal) wird zuvor im Block *Chk-ZeroCrossState* bestimmt.

Die Soll - Boostspannung *PwrCtrl_uReqB6* (*WspPwrTrf_uPfcReq*) wird in Abhängigkeit des Soll Primärstroms *iPrimSet* mittels der Kennlinie *PwrCtrl_uBooster_Cur* bestimmt.

6.2.1 Block ChkZeroCrossState



Hier wird der Status [SwitchCtrl_zcSwitchingIndicator](#) ([stSwtIndicator](#)) bestimmt. Dieser beschreibt die aktuelle Phasenlage zwischen Strom und Spannung der drei Phasen U,V und W der B6 Brücke hinsichtlich des ungünstigsten Falles (Kapazitives Schalten).

Zur Bestimmung der Spannungs - Strom- Phasenlage werden die Timerticks [SwitchCtrl_zcMeasU](#), [SwitchCtrl_zcMeasV](#) und [SwitchCtrl_zcMeasW](#) der drei Phasen erfasst.

Diese werden im B6 Brückentreiber ermittelt. Hierbei wird bei jedem Einschalten einer Phasenspannung ein Timer gestartet, welcher die Prozessor - Ticks bis zum Strom - Nulldurchgang der jeweiligen Phase misst.

Das Minimum dieser drei Zähler wird dann mittels der Filterkonstanten [SwitchCtrl_facZC_C](#) tiefpassgefiltert und ergibt den so im nachfolgenden betrachteten Timerwert [SwitchCtrl_zcMinFilt](#). Überschreitet dieser Timerwert die Schwelle [SwitchCtrl_zcSwitchingInd_C](#), liegt induktives Schalten vor und der Status [SwitchCtrl_zcSwitchingIndicator](#) wird auf 2 gesetzt. Unterschreitet der Timerwert diese Schwelle, liegt kapazitives Schalten vor und der Status [SwitchCtrl_zcSwitchingIndicator](#) wird auf 0 gesetzt. Hat der Timer der Wert [SwitchCtrl_zcSwitchingOptim_C](#) liegt optimales Schalten vor und der Status [SwitchCtrl_zcSwitchingIndicator](#) wird auf 1 gesetzt.

Der Status [SwitchCtrl_zcSwitchingIndicator](#) kann hierbei also folgende Werte annehmen:

6.2.1.1 Status [SwitchCtrl_zcSwitchingIndicator](#)

Staus	Bedeutung
0	kapazitives Schalten - Beim Einschalten der Spannung hat der Strom bereits die Nulllinie überschritten -> B6 Brücke kann zerstört werden.
1	ideales Schalten - Beim Einschalten der Spannung ist der Strom nahezu Null -> idealer Betriebszustand.
2	induktives Schalten - Beim Einschalten der Spannung hat der Strom die Nulllinie noch nicht erreicht -> akzeptabler Betriebszustand

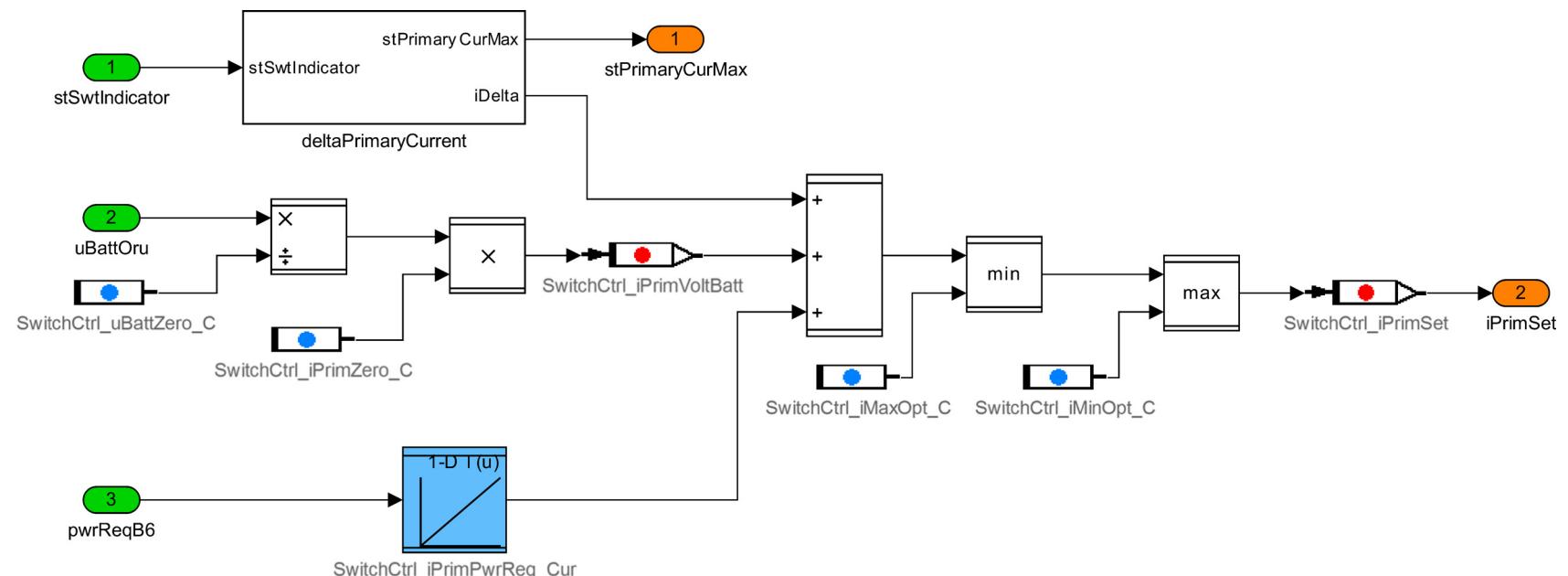
6.2.2 Block Primary-Current

Hier wird in Abhängigkeit von der Angeforderten Ladeleistung, der HV Batteriespannung, der Phaseshift sowie der Phasenverschiebung zwischen Spannung und Strom der drei Schaltströme der Soll - Primästrom $i_{PrimSet}$ berechnet.

Der Soll - Primästrom $SwitchCtrl_iPrimSet$ ($i_{PrimSet}$) ist die Summe aus 3 Anteilen:

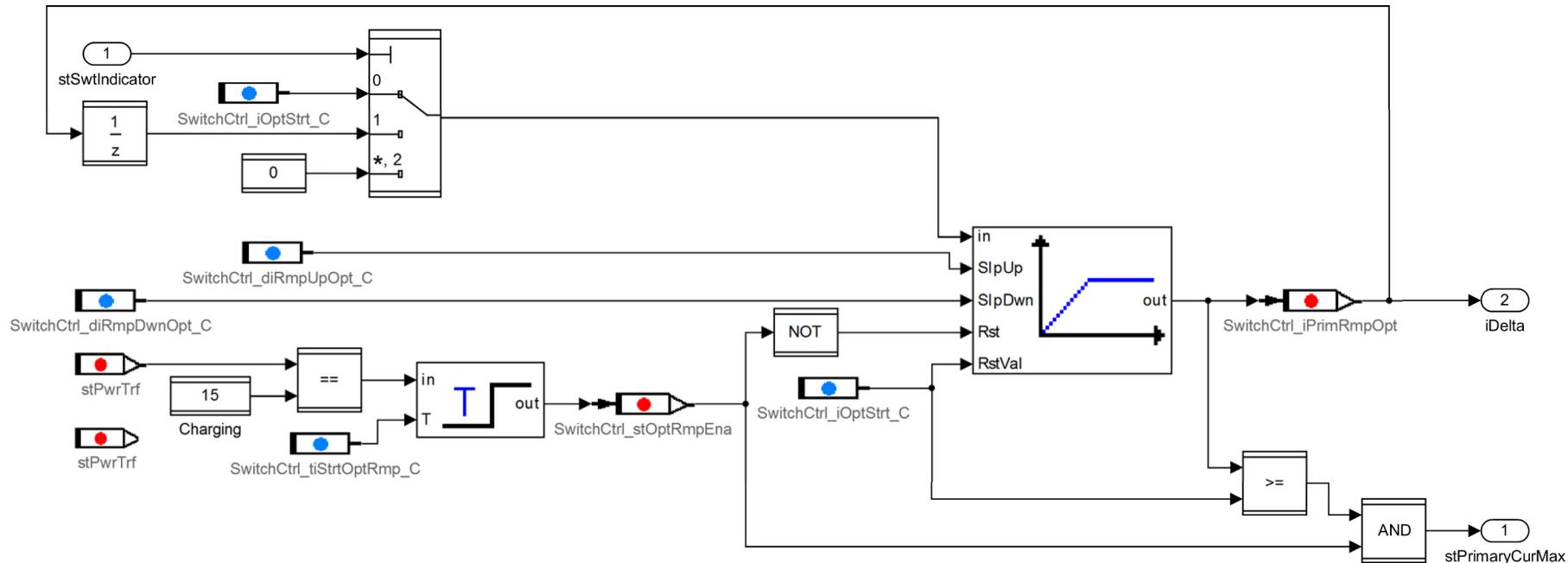
- 1) Einem für die geforderte Ladeleistungen, welcher unterhalb der maximal möglichen einen negativen Anteil hat. Dieser wird in Abhängigkeit der angeforderten Ladeleistung $PwrTrfStMgr_pwrReqB6$ über die Kennlinie $SwitchCtrl_iPrimPwrReq_Cur$ bestimmt.
- 2) Dem von der HV Batteriespannung $PwrCtrl_uBattOru$ abhängigen Anteil $SwitchCtrl_iPrimVoltBatt$. Dieser errechnet sich als Produkt aus dem Quotienten der gemessenen HV Batteriespannung $PwrCtrl_uBattOru$ zu der minimal möglichen (zulässigen) Batteriespannung $SwitchCtrl_uBattZero_C$ und dem zugehörigen minimal Primärstrom $SwitchCtrl_iPrimZero_C$.
- 3) Dem Optimierungsstrom $SwitchCtrl_iPrimRmpOpt$ ($iDelta$ vom Block $deltaPrimaryCurrent$).

Die Summe aus obigen drei Soll - Primärstromanteilen wird nach oben hin mittels $SwitchCtrl_iMaxOpt_C$ und nach unten hin mittels $SwitchCtrl_iMinOpt_C$ begrenzt und ergibt so den letztendlichen Soll - Primärstrom $SwitchCtrl_iPrimSet$. Das Einregeln dieses Stroms erfolgt in der Komponente $AirGapCtrl$ durch Variation des Luftspalts.



Softwaredokumentation

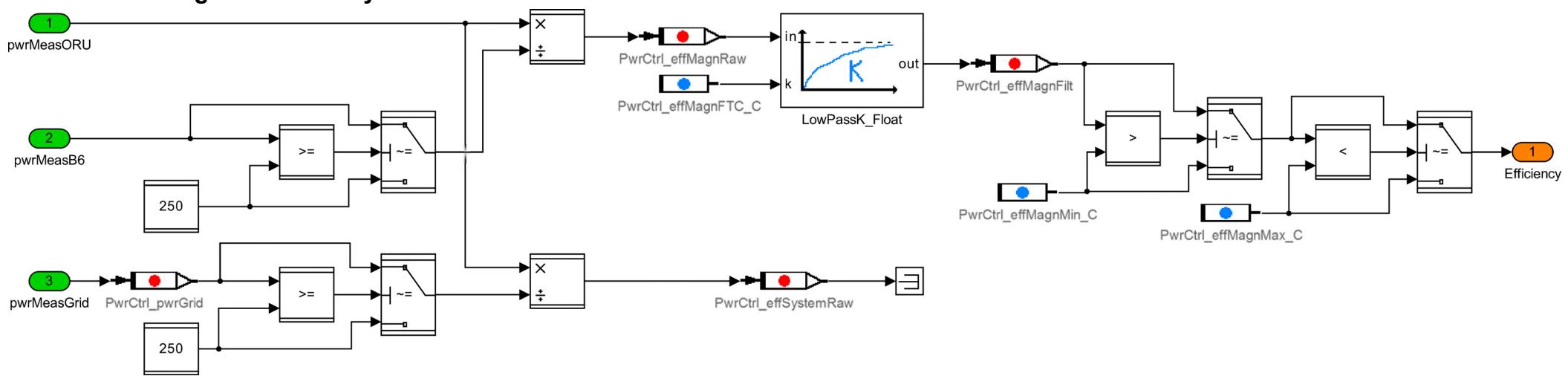
6.2.2.1 Block deltaPrimaryCurrent



Der Optimierungsstrom hängt vom Zustand des Zero Switchindikators ab. Im Falle des kapazitiven Schaltens (`stSwtIndicator` = 0) wird der Optimierungsstrom `SwitchCtrl_iPrimRmpOpt` mit der Geschwindigkeit `SwitchCtrl_diRmpUpOpt_C` auf den Wert `SwitchCtrl_iOptStrt_C` hochgerampt. Beim induktiven Schalten (`stSwtIndicator` = 2) wird er mit der Geschwindigkeit `SwitchCtrl_diRmpDwnOpt_C` auf 0 runtergrampt. Im Falle des optimalen Schaltens (`stSwtIndicator` = 1) wird die Rampe auf dem aktuellen Wert eingefroren. Die Rampe ist nur während des Ladens aktiv (`stPwrTrf` = 15). Wird nicht geladen wird der Rampewert und damit auch der Optimierungsstrom auf den Startwert `SwitchCtrl_iOptStrt_C` gesetzt.

Softwaredokumentation

6.2.3 Block MagneticEfficiency

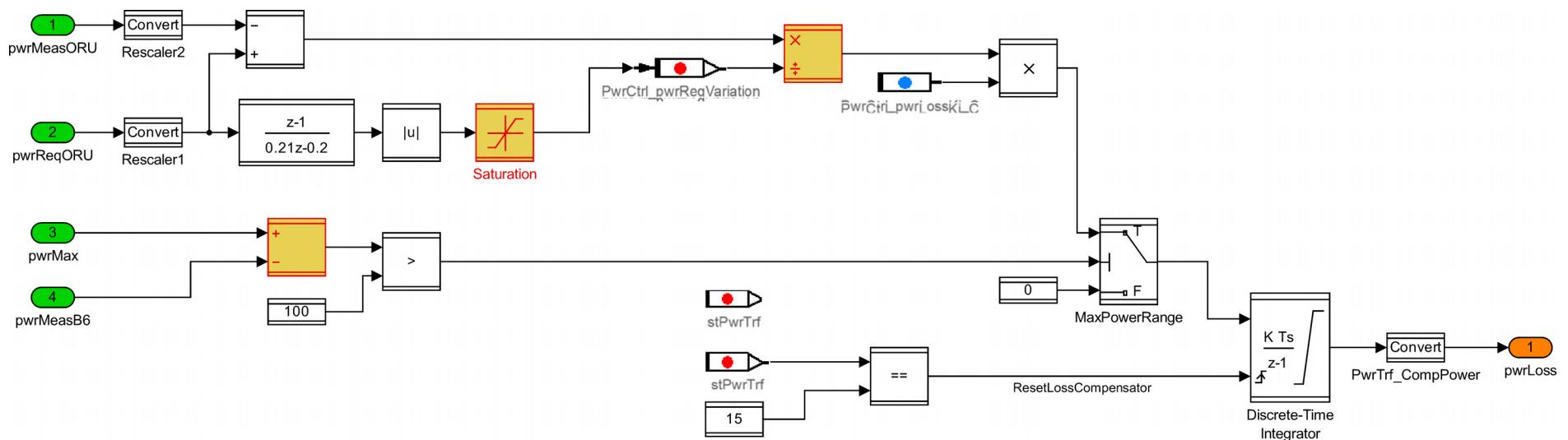


Hier wird die Übertragungseffizienz (*Efficiency*) berechnet. Dazu wird zunächst der Effizienz - Rohwert [PwrCtrl_effMagnRaw](#) als Quotient der aktuellen Ladeleistung (*pwrMeasORU*) der Oru und der gemessenen Ausgangsleistung (*pwrMeasB6*) der B6 Brücke berechnet, wobei *pwrMeasB6* nach unten hin mit 250W limitiert wird. Der Rohwert [PwrCtrl_effMagnRaw](#) wird anschließend mittels der Filterkonstanten [PwrCtrl_effMagnFTC_C](#) tiefpassgefiltert und ergibt so der Effizienzwert [PwrCtrl_effMagnFilt](#).

Dieser wird nach unten mittels [PwrCtrl_effMagnMin_C](#) und nach oben mittels [PwrCtrl_effMagnMax_C](#) begrenzt und ergibt so den für die Bestimmung der maximalen Ladeleistung verwendeten Ausgangswert *Efficiency*.

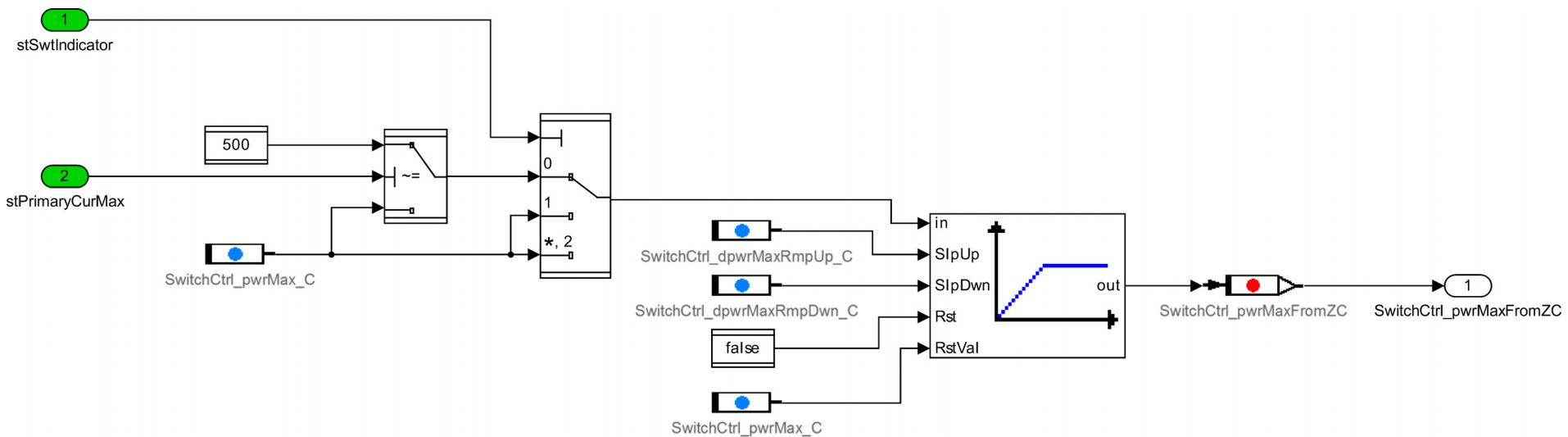
Zu Meßzwecken wird zusätzlich noch die Systemeffizienz [PwrCtrl_effSystemRaw](#) als Verhältnis der auf der Oru Seite gemessenen Ladesleistung *pwrMeasORU* zu der aus dem Netz aufgenommenen Leistung [PwrCtrl_pwrGrid](#) berechnet.

6.2.4 Block PowerLoss_Compensator



Hier wird die Verlustleistung `pwrLoss` berechnet. ???

6.2.5 Block PowerDerating



Hier wird die maximale Ladeleistung im falle eines Kapazitiven schaltens ([SwitchCtrl_pwrMaxFromZC](#)) der B6 Endstufe durchgeführt Siehe “Block ChkZeroCrossState” auf Seite 24.

Im Fall des kapazitiven Schaltens ($stSwtIndicator = 0$) wird der Eingang der nachfolgenden Rampe konstant auf 500W gesetzt. Für die beiden anderen Fälle auf den Wert [SwitchCtrl_pwrMax_C](#) (Normal 3680W). Der Ausgangswert [SwitchCtrl_pwrMaxFromZC](#) wird immer in Richtung des Eingangswerts mit dem konstanten Gradienten [SwitchCtrl_dpwrMaxRmpUp_C](#) in positive und [SwitchCtrl_dpwrMaxRmpDwn_C](#) in negative Richtung gefahren.

6.2.6 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

6.2.6.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

6.2.6.2 Ausgangsgrößen

keine	
-------	--

6.2.6.3 Interfaces

6.2.6.3.1 Interface

Data Element	Data Type	Auflösung	Einheit	Bedeutung

6.2.7 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

PwrCtrl_effMagnFilt	Estimated efficiency of Magnetic Coupling_ - Filtered Value
PwrCtrl_effMagnRaw	Estimated efficiency of Magnetic Coupling - RAW value
PwrCtrl_effMagnSat	Estimated efficiency of Magnetic Coupling - Saturated value
PwrCtrl_effSystemRaw	Estimated system Efficiency
PwrCtrl_fraqReqB6	B6 operating frequency
PwrCtrl_iMeas	Measured ORU DC Current
PwrCtrl_iMeasORU	Measured ORU DC Current
PwrCtrl_iReqORU	Required ORU DC Current
PwrCtrl_pwrGrid	Measured Grid Power
PwrCtrl_pwrLoss	Power Loss Estimated value
PwrCtrl_pwrMeasORU	ORU Measured Power
PwrCtrl_pwrOutputMaxEstim	ORU estimated Maximum output Power
PwrCtrl_pwrReqB6	Inverter Required Power
PwrCtrl_pwrReqORU	ORU requested Power
PwrCtrl_pwrReqVariation	Variation of the Required Power
PwrCtrl_uMeas	Measured ORU OutputDC Voltage
PwrCtrl_uMeasORU	Measured ORU OutputDC Voltage
PwrCtrl_uReqB6	Required Voltage at Inverter Input

6.2.8 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

PwrCtrl_effMagnFTC_C	Filter time constant for magnetic efficiency
PwrCtrl_effMagnMax_C	Maximum cosidered Efficiency in the calculation of the maximum ORU output power
PwrCtrl_effMagnMin_C	Minimum cosidered Efficiency in the calculation of the maximum ORU output power
PwrCtrl_frqInverter_Cur	
PwrCtrl_pwrLossKi_C	Integral Gain of the Loss Estimator
PwrCtrl_pwrLossMax_C	Loss estimator maximum power
PwrCtrl_pwrLossMin_C	Loss estimator minimum power
PwrCtrl_uBooster_Cur	

6.2.9 Fehlerpfade

6.2.9.1 TBD

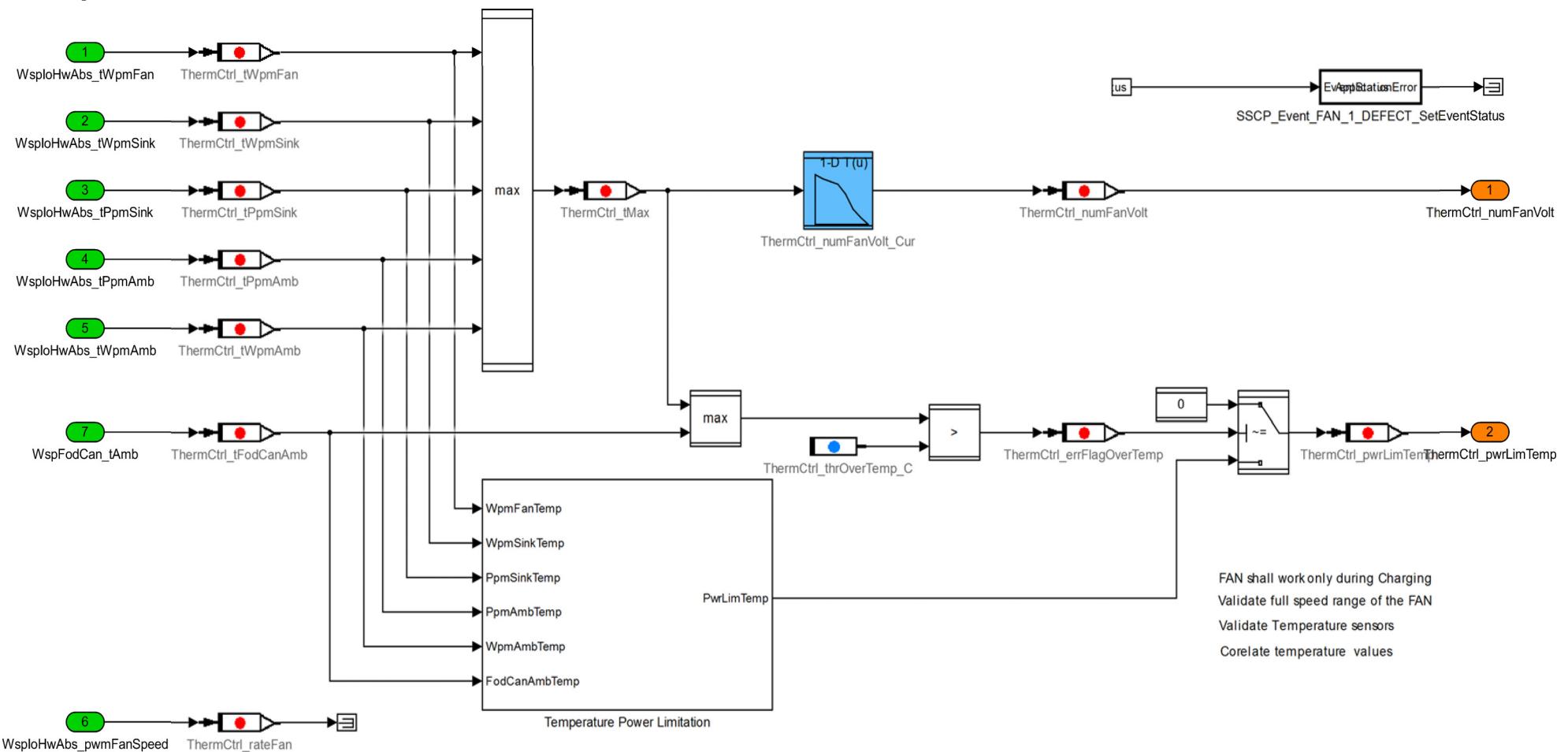
tbd

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

6.2.10 Initialisierung

tbd

6.3 Komponente ThermCtrl



In der Komponente *ThermCtrl* wird die maximale Ladeleistung *ThermCtrl_pwrLimTemp* (*PwrMaxTemp*) in Abhängigkeit der Temperaturen der einzelnen Systemkomponenten des PAD's berechnet (Derating).

Darüber hinaus wird ebenfalls in Abhängigkeit dieser Temperaturen die Lüftergeschwindigkeit bestimmt.

Zur Berechnung der Lüftergeschwindigkeit wird zunächst das Maximum *ThermCtrl_tMax* der Komponententemperaturen *ThermCtrl_tWpmFan*, *ThermCtrl_tWpmSink*, *ThermCtrl_tPpmSink*, *ThermCtrl_tPpmAmb* und *ThermCtrl_tWpmAmb* bestimmt. In Abhängigkeit dieser Maximaltemperatur wird dann mittels der Kennlinie *ThermCtrl_numFanVolt_Cur* direkt der numerische Wert *ThermCtrl_numFanVolt* berechnet, welcher das PWM Verhältnis zur Lüfteransteuerung ist.

Softwaredokumentation

primove
true e-mobility

Zur Bestimmung der temperaturabhängigen Leistungslimitierung *ThermCtrl_pwrLimTemp* wird nun zunächst überprüft, ob die zuvor bestimmte Maximaltemperatur *ThermCtrl_tMax* oder die Temperatur *ThermCtrl_tFodCanAmb* der FOD Platine größer sind als *ThermCtrl_thrOverTemp_C*. In diesem Fall wird die maximale Ladeleistung *ThermCtrl_pwrLimTemp* konstant auf 0 gesetzt.

Ist dies nicht der Fall wird die maximale Ladeleistung auf den im Block *Temperature Power Limitation* bestimmten Wert *PwrLimTemp* gesetzt.

Softwaredokumentation

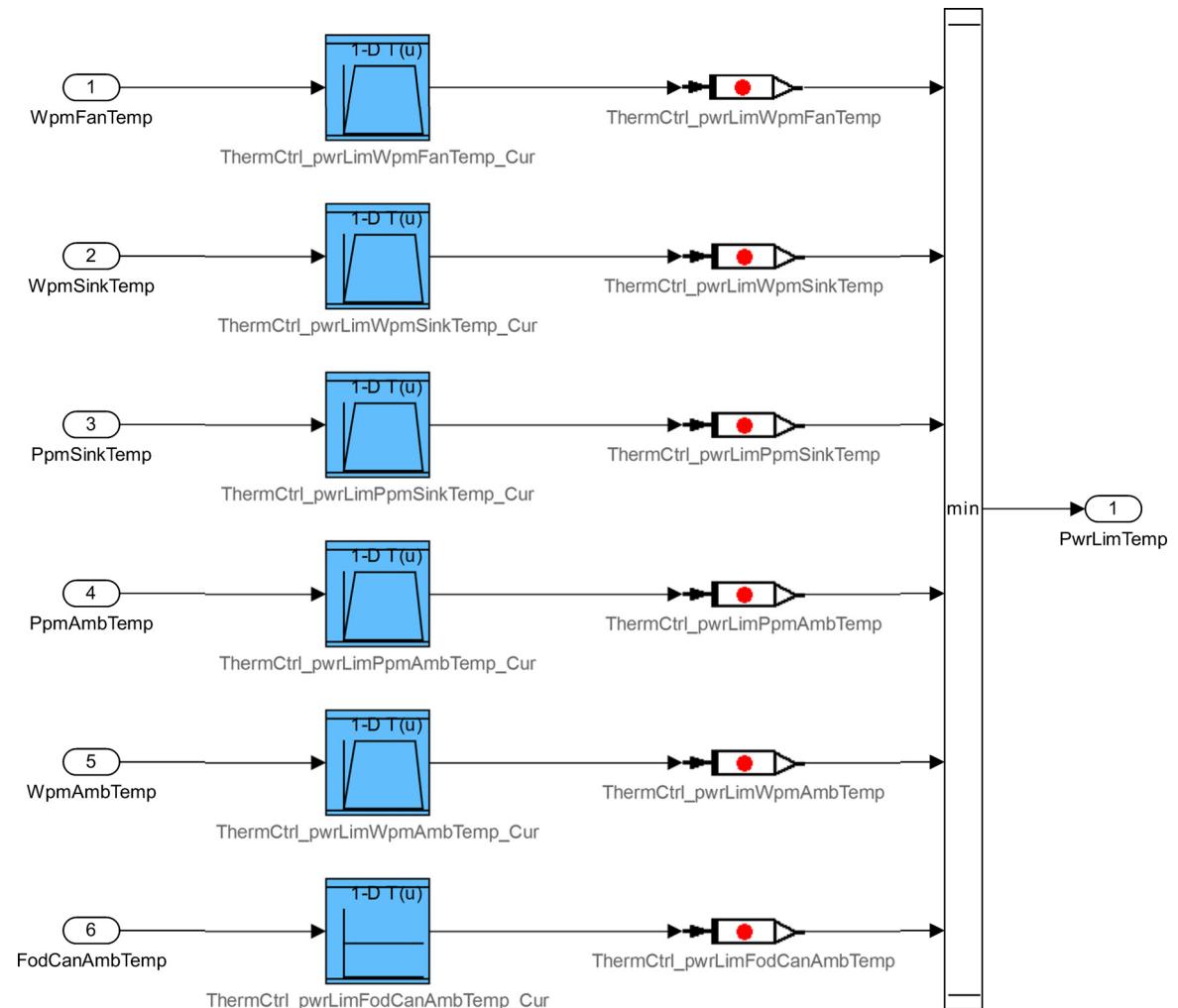
6.3.1 Block Temperature Power Limitation

Hier wird die limitierte Ladeleistung PwrLimTemp bestimmt. Hierzu wird zunächst für jede gemessene Systemtemperatur mittels einer Kennlinie die zugehörige Leistungslimitierung berechnet.

Dies sind im folgenden die Limitierungen:

- *ThermCtrl_pwrLimWpmFanTemp* aus der Temperatur *WpmFanTemp* über die Kennlinie *ThermCtrl_pwrLimWpmFanTemp_Cur*
- *ThermCtrl_pwrLimWpmSinkTemp* aus der Temperatur *WpmSinkTemp* über die Kennlinie *ThermCtrl_pwrLimWpmSinkTemp_Cur*
- *ThermCtrl_pwrLimPpmSinkTemp* aus der Temperatur *PpmSinkTemp* über die Kennlinie *ThermCtrl_pwrLimPpmSinkTemp_Cur*
- *ThermCtrl_pwrLimPpmAmbTemp* aus der Temperatur *PpmAmbTemp* über die Kennlinie *ThermCtrl_pwrLimPpmAmbTemp_Cur*
- *ThermCtrl_pwrLimWpmAmbTemp* aus der Temperatur *WpmAmbTemp* über die Kennlinie *ThermCtrl_pwrLimWpmAmbTemp_Cur*
- *ThermCtrl_pwrLimFodCanAmbTemp* aus der Temperatur *FodCanAmbTemp* über die Kennlinie *ThermCtrl_pwrLimFodCanAmbTemp_Cur*

Die limitierte Ladeleistung *PwrLimTemp* bezüglich der Temperatur ist dann das Minimum aller oben genannten Leistungslimits.



6.3.2 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

6.3.2.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

6.3.2.2 Ausgangsgrößen

keine	
-------	--

6.3.2.3 Interfaces

6.3.2.3.1 Interface

Data Element	Data Type	Auflösung	Einheit	Bedeutung

6.3.3 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

<i>ThermCtrl_errFlagOverTemp</i>	Overtemperature Flag detected ya Power Transfer component
<i>ThermCtrl_numFanVolt</i>	PWM Duty Cycle applied to the FAN - [32767 0] coresponds to [0V 24V]
<i>ThermCtrl_pwrLimFodCanAmbTemp</i>	Maximum Power calculated based on Temperature of FOD Board
<i>ThermCtrl_pwrLimPpmAmbTemp</i>	Maximum Power calculated based on Temperature of Ppm Ambient
<i>ThermCtrl_pwrLimPpmSinkTemp</i>	Maximum Power calculated based on Temperature of Ppm Sink
<i>ThermCtrl_pwrLimTemp</i>	Maximum Power calculated based on Temperature sensors
<i>ThermCtrl_pwrLimWpmAmbTemp</i>	Maximum Power calculated based on Temperature of Wpm Ambient
<i>ThermCtrl_pwrLimWpmFanTemp</i>	Maximum Power calculated based on Temperature of Wpm Fan Input
<i>ThermCtrl_pwrLimWpmSinkTemp</i>	Maximum Power calculated based on Temperature of Wpm Sink
<i>ThermCtrl_rateFan</i>	FAN measured Rotational Speed
<i>ThermCtrl_tFodCanAmb</i>	Temperature FOD Board
<i>ThermCtrl_tMax</i>	Maximum Value between all measured temperatures
<i>ThermCtrl_tPpmAmb</i>	Temperature Ppm Ambient
<i>ThermCtrl_tPpmSink</i>	Temperature Ppm Sink
<i>ThermCtrl_tWpmAmb</i>	Temperature Wpm Ambient
<i>ThermCtrl_tWpmFan</i>	Temperature Wpm Board
<i>ThermCtrl_tWpmSink</i>	Temperature Wpm Sink

6.3.4 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>ThermCtrl_numFanVolt_Cur</i>	PWM Ansteuerwert Lüfter
<i>ThermCtrl_pwrLimFodCanAmbTemp_Cur</i>	Power Limitation by HWIO Ambient Temperature
<i>ThermCtrl_pwrLimPpmAmbTemp_Cur</i>	Power Limitation by WPC3 Temperature
<i>ThermCtrl_pwrLimPpmSinkTemp_Cur</i>	Power Limitation by WPC2 Temperature
<i>ThermCtrl_pwrLimWpmAmbTemp_Cur</i>	Power Limitation by MOD Ambient Temperature
<i>ThermCtrl_pwrLimWpmFanTemp_Cur</i>	Power Limitation by Boosttemperatur
<i>ThermCtrl_pwrLimWpmSinkTemp_Cur</i>	Power Limitation by WPC1 Temperature
<i>ThermCtrl_thrOverTemp_C</i>	Overtemperature Threshold

6.3.5 Fehlerpfade

6.3.5.1 TBD

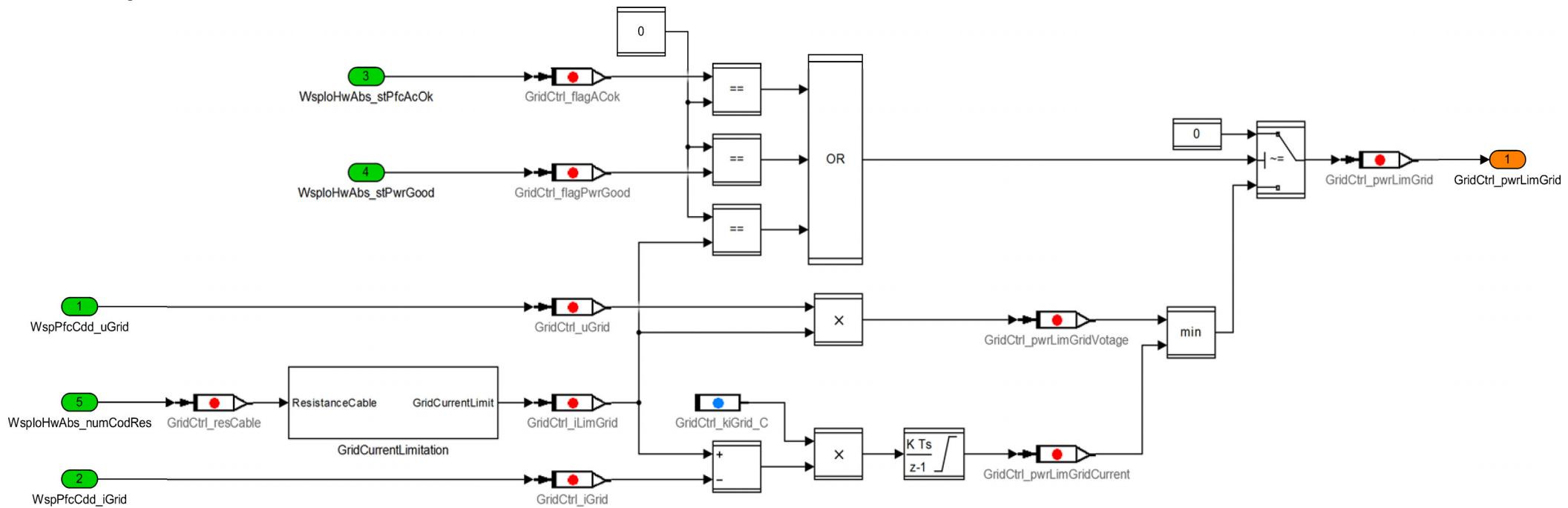
tbd

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

6.3.6 Initialisierung

tbd

6.4 Komponente GridCtrl

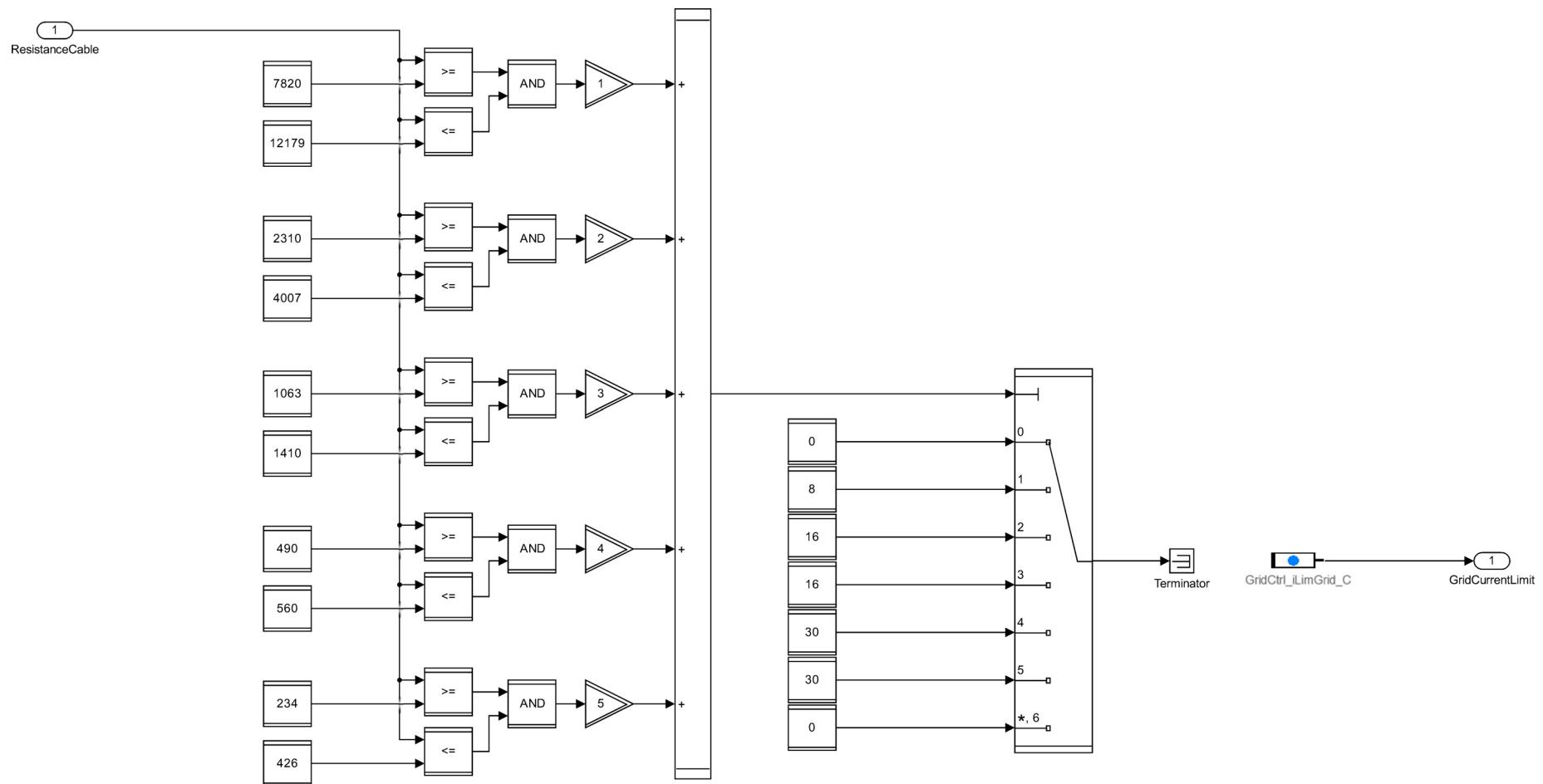


Hier wird die maximale Ladeleistung *GridCtrl_pwrLimGrid* bezüglich der Netz-Versorgung bestimmt.

Für den Fall, daß die vom Pfc - Controller kommenden Stati *GridCtrl_flagACok* oder *GridCtrl_flagPwrGood* den Wert 0 haben, oder die im Block *GridCurrentLimitation* bestimmte Stromlimitierung *GridCtrl_iLimGrid* den Wert 0 hat, wird die maximale Ladeleistung *GridCtrl_pwrLimGrid* konstant auf 0 gesetzt.

Für den anderen Fall wird zunächst die Leistungslimitierung *GridCtrl_pwrLimGridVotage* bezüglich der Netz - Stromlimitierung als Produkt aus der Netzspannung *GridCtrl_uGrid* und dem Netz - Stromlimitierung *GridCtrl_iLimGrid* berechnet. Dann wird die Abweichung zwischen der Stromlimitierung *GridCtrl_iLimGrid* und dem gemessenen Netzstrom *GridCtrl_iGrid* mit dem Gewichtungsfaktor *GridCtrl_kiGrid_C* aufintegriert. Der Integrator ist hierbei nach unten auf 1000W und nach oben auf 3680W begrenzt. die Leistungsbegrenzung *GridCtrl_pwrLimGridCurrent* wird also bei einer Überschreitung des Stromlimits auf 1000W heruntergerampt und bei einer Unterschreitung auf 3680W hochgerampt. Das Minimum der beiden Limits ist dann das Ladeleisungslimit *GridCtrl_pwrLimGrid* bezüglich des Stromnetzes.

6.4.1 Block GridCurrentLimitation



tbd.

6.4.2 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

6.4.2.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

6.4.2.2 Ausgangsgrößen

keine	
-------	--

6.4.2.3 Interfaces

6.4.2.3.1 Interface

Data Element	Data Type	Auflösung	Einheit	Bedeutung

6.4.3 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

<i>GridCtrl_flagACok</i>	Flag AC ok from PFC cip
<i>GridCtrl_flagPwrGood</i>	Flag Power Good from PFC cip
<i>GridCtrl_iGrid</i>	Grid Current
<i>GridCtrl_iLimGrid</i>	Grid Current Limitation value- shall be calculated based on the cable resistance
<i>GridCtrl_pwrLimGrid</i>	Maximum Power calculated based on available grid voltage
<i>GridCtrl_pwrLimGridCurrent</i>	Maximum Power calculated based on available grid voltage
<i>GridCtrl_pwrLimGridVoltage</i>	Maximum Power calculated based on available grid voltage
<i>GridCtrl_resCable</i>	Cable resistance
<i>GridCtrl_uGrid</i>	Grid Voltage

6.4.4 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>GridCtrl_iLimGrid_C</i>	Grid Current Limitation value- shall be calculated based on the cable resistance
<i>GridCtrl_kiGrid_C</i>	Integral Gain for Input Grid Current

6.4.5 Fehlerpfade

6.4.5.1 TBD

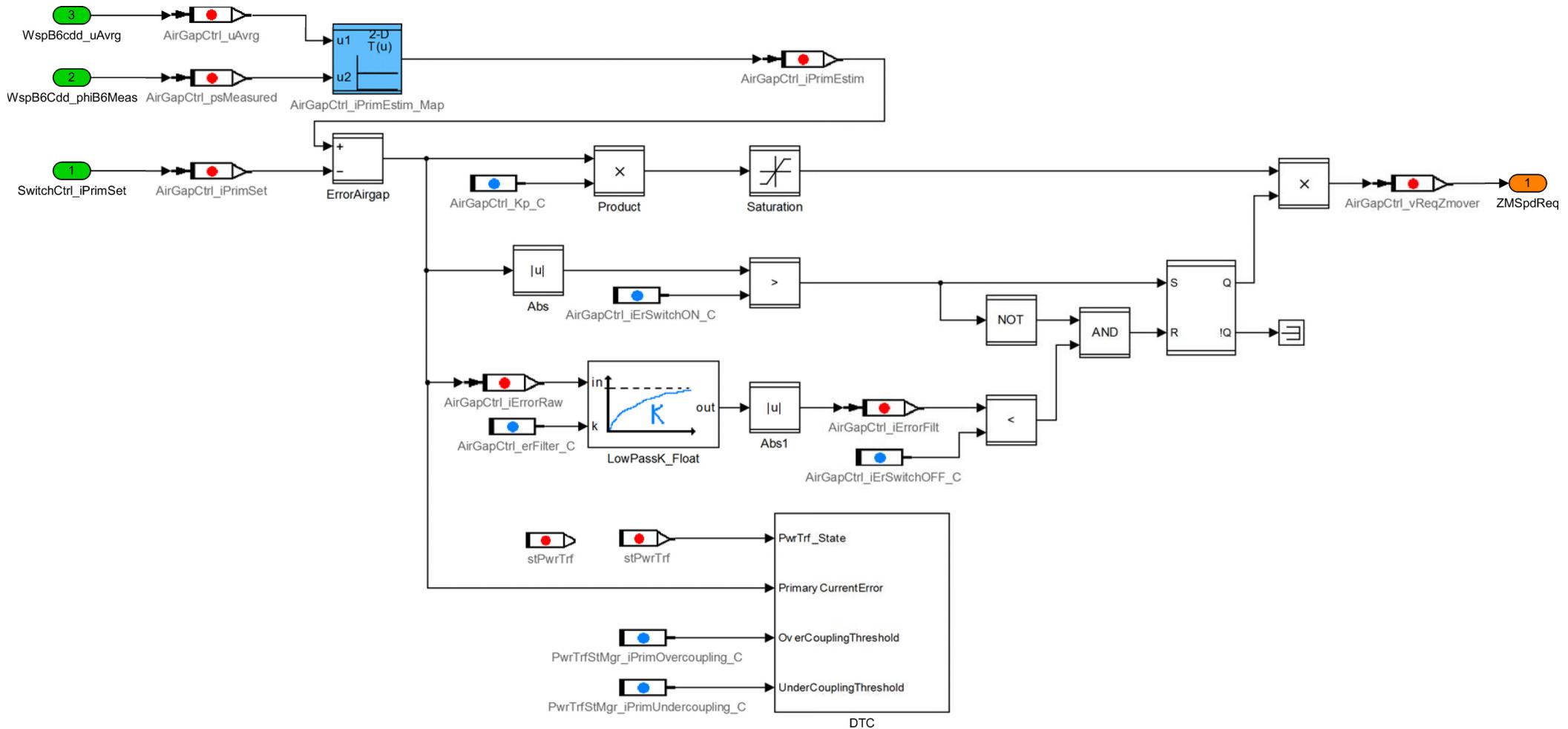
tbd

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

6.4.6 Initialisierung

tbd

6.5 Komponente AirGapCtrl



Hier findet die Feinjustierung des Soll - Primärstroms [SwitchCtrl_iPrimSet](#) mittels Variation des Luftspalts statt. Da der aktuelle Primärstrom [AirGapCtrl_iPrimEstim](#) vom Steuergerät nicht direkt gemessen werden kann, wird er in Abhängigkeit von der gemessenen Boostspannung [B6cdd_uAvrg](#) und der aktuellen Phasenverschiebung [AirGapCtrl_psMeasured](#) über das extern ausgemessene Kennfeld [AirGapCtrl_iPrimEstim_Map](#) bestimmt.

Aus der Regelabweichung [AirGapCtrl_iErrorRaw](#) = [AirGapCtrl_iPrimEstim](#) - [SwitchCtrl_iPrimSet](#) wird dann mittels des Proportionalitätsfaktors [AirGapCtrl_Kp_C](#) die Verstellgeschwindigkeit [AirGapCtrl_vReqZmover](#) des ZMovers berechnet.

Softwaredokumentation

primove
true e-mobility

Das Verfahren des ZMovers wird jedoch nur dann freigegeben, wenn der Betrag der Regelabweichung *AirGapCtrl_iErrorRaw* den Wert *AirGapCtrl_iErSwitchON_C* überschreitet. Die Freigabe wird zurückgenommen, wenn der Betrag der Regelabweichung diesen Wert nicht mehr überschreitet und der Betrag des tiefpassgefilterten Wertes *AirGapCtrl_iErrorFilt* der Regelabweichung kleiner ist als *AirGapCtrl_iErSwitchOFF_C*.

6.5.1 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

6.5.1.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

6.5.1.2 Ausgangsgrößen

keine	
-------	--

6.5.1.3 Interfaces

6.5.1.3.1 Interface

Data Element	Data Type	Auflösung	Einheit	Bedeutung

6.5.2 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

AirGapCtrl_psErrorFilt	Phase Shift Error Filtered value
AirGapCtrl_psErrorRaw	Phase Shift Error RAW value
AirGapCtrl_psMeasured	Phase Shift Measured value
AirGapCtrl_psRequest	Phase Shift Request value
AirGapCtrl_vReqZmover	Required ZMover Speed

6.5.3 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

AirGapCtrl_erFilter_C	Gain of the Coupling Adjustment Controller
AirGapCtrl_Kp_C	Gain of the Coupling Adjustment Controller
AirGapCtrl_maxNegSpeed_C	Maximum Negative Speed for Airgap Control
AirGapCtrl_maxPosSpeed_C	Maximum Positive Speed for Airgap Control
AirGapCtrl_psErSwitchOFF_C	Threshold where Airgap control is switched OFF
AirGapCtrl_psErSwitchON_C	Threshold where Airgap control is switched ON

6.5.4 Fehlerpfade

6.5.4.1 TBD

tbd

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

6.5.5 Initialisierung

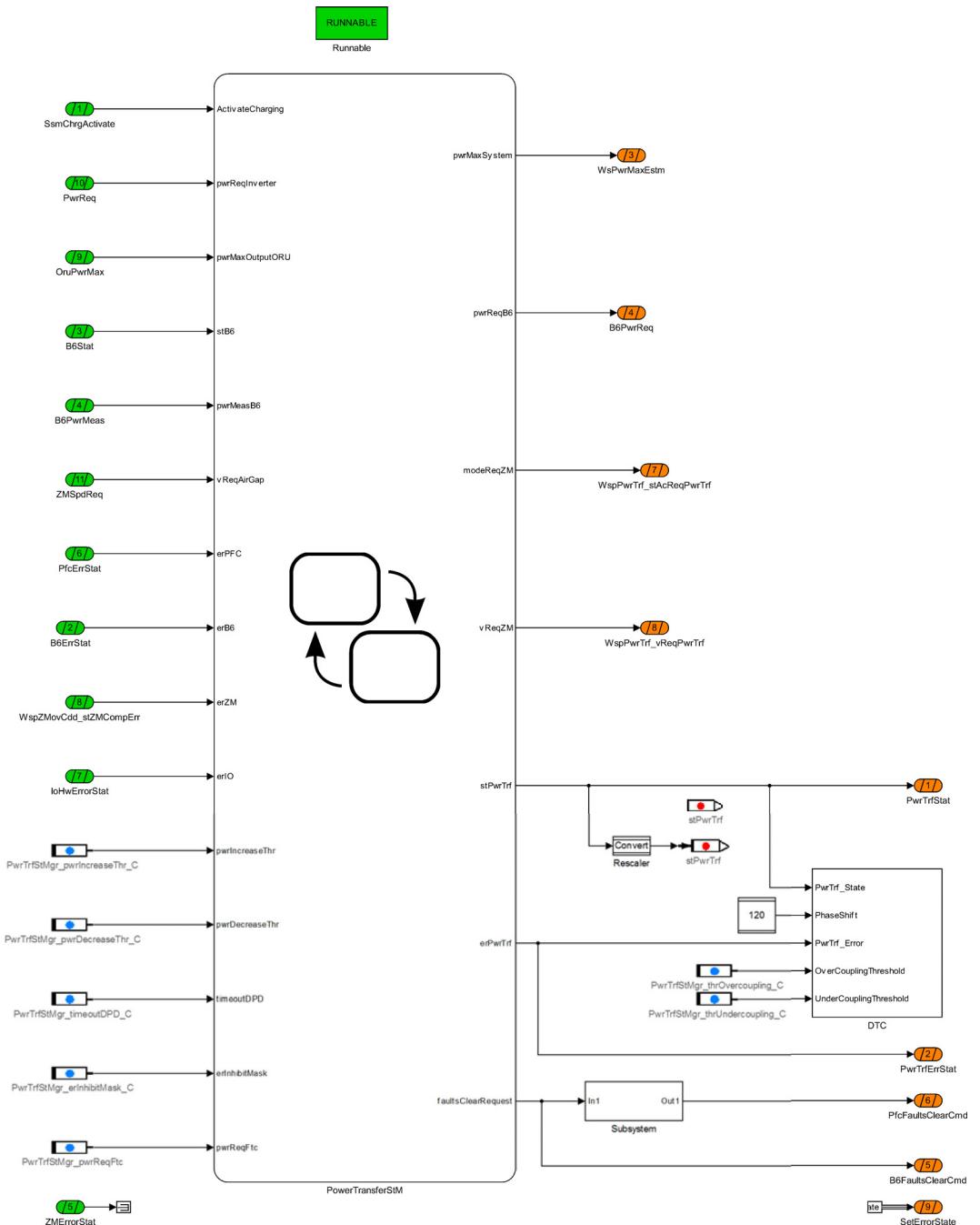
tbd

Softwaredokumentation

6.6 Komponente OpCtrl

In der Komponente OpCtrl werden die in den entsprechenden Modulen zuvor bestimmten Größen für die Ladeleistung und den ZMover überwacht und falls nötig modifiziert und dann für den Komponententreiber der B6Brücke und des ZMovers auf der RTE ausgegeben. Die Fehlerüberwachung findet ebenfalls hier statt.

Die Funktion ist im wesentlichen als Stateflow Diagramm *PowerTransferStM* implementiert. Lediglich die Zuweisung der DTC's entsprechend der bitcodierten Fehler- und Betriebsstati *stPwrTrf* (*PwrTrfStMgr_stPwrTrf*) und *erPwrTrf* (*PwrTrfStMgr_erPwrTrf*) ist im Block *DTC* realisiert.



6.6.1 States von OpCtrl

Die mögliche States *stPwrTrf* (*PwrTrfStMgr_stPwrTrf*) sowie die Bitcodierten Errorstates *erPwrTrf* (*PwrTrfStMgr_erPwrTrf*) werden im nachfolgenden aufgelistet.

6.6.1.1 State *PwrTrfStMgr_stPwrTrf*

Der Ausgangsstate *PwrTrfStMgr_stPwrTrf* (*stPwrTrf*) kann folgenden Werte annehmen:

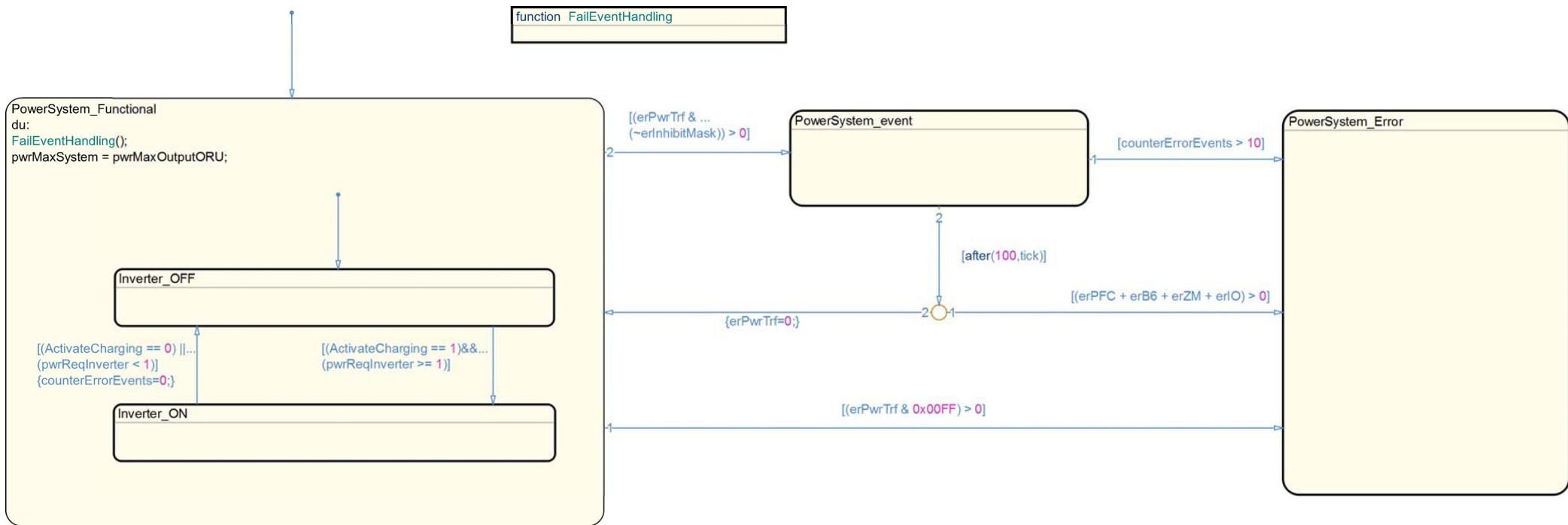
Wert	Bedeutung
1	STANDBY -> Inverter aus, Laden inaktiv
11	DOM_PHASE_DETECT -> Dominant Phase Detection
15	POWER_CONTROLLER -> Laden aktiv, PowerControl aktiviert nach Dominant Phase Detection
254	PRE_ERROR -> Vorläufiger Fehler, wird versucht zu heilen
255	ERROR -> Endgültiger Fehler, Reset der Wayside notwendig

6.6.1.2 ErrorState *PwrTrfStMgr_erPwrTrf*

Der Fehlerstatus *PwrTrfStMgr_erPwrTrf* (*erPwrTrf*) ist Bitcodiert- Folgende Fehler sind hierbei definiert:

Bit	Bedeutung
0	PFC Error (not activated)
1	B6 Error
2	ZMover Error
3	IO Error
5	Dominant Phase Detection failed (timeout)
8	PFC Error
9	B6 Error: Over- UnderVoltage or Over- UnderCurrent or Underpower
10	ZMover Error: Current to high, Currentoffset to high or Lowerd with active force
11	IO Error
13	B6 Charging power to low

6.6.2 Statediagramm PowerTransferStM



Dies ist der Haupt State des Stateflow Diagramms *PowerTransferStM*. Es stellt die im nachfolgenden beschriebenen States zur Verfügung, sowie die Funktion *FailEventHandling*. Dieser wird im State *PowerSystem_Functional* zyklisch aufgerufen.

6.6.2.1 State *PowerSystem_Functional*

Dieser State wird zunächst nach dem Neustart des Systems aktiviert. Er ruft, solange aktiv, zyklisch die Funktion *FailEventHandling* auf und weist dem Ausgangswert *pwrMaxSystem* (maximale Ladeleistung) zunächst die in der Komponente *PwrCtrl* bestimmte maximale Ladeleistung *pwrMaxOutputORU* zu.

Bei der Aktivierung des States *PowerSystem_Functional* wird der SubState *Inverter_OFF* aktiviert.

State Transition -> *PowerSystem_event*

Der State *PowerSystem_Functional* verzweigt zum State *PowerSystem_event*, wenn im Fehlerstatus *erPwrTrf* (aus Funktion *FailEventHandling*) mindestens ein nicht durch den Parameter *erInhibitMask* ([PwrTrfStMgr_erInhibitMask_C](#)) ausmaskiertes Fehlerbit gesetzt ist.

State Transition -> *PowerSystem_Error*

Der State *PowerSystem_Functional* verzweigt zum State *PowerSystem_Error*, wenn im Fehlerstatus *erPwrTrf* mindesten eines der unteren 8 Fehlerbits gesetzt ist.

6.6.2.2 SubState Inverter_OFF

Beim aktivieren dieses States werden folgende Aktionen durchgeführt:

- Der Powertransfer Status *stPwrTrf* wird auf STANDBY (1) gesetzt
- Die Soll Ladeleistungsanforderung *pwrReqB6* für die B6 Brücke wird auf 0W gesetzt
- Der ZMover wird freigegeben -> *modeReqZM* = 0
- Die Schwelle *pwrTooLowThr* für die Fehlererkennung ‚Ladeleistung zu niedrig‘ wird mit -500W initialisiert.

```
Inverter_OFF
en:
stPwrTrf = STANDBY;

/* Stop the Inverter */
pwrReqB6 = 0;

/* Release control of the ZMover */
modeReqZM = 0;

pwrTooLowThr = -500;
```

State Transition -> Inverter_ON

Der State *Inverter_OFF* wechselt zum State *Inverter_ON*, wenn eine Ladeanforderung vom System Statemanager besteht (*ActivateCharging* = 1) und die angeforderte Ladeleistung *pwrReqInverter* größer als 0W ist.

6.6.2.3 SubState Inverter_ON

Der Substate *Inverter_ON*, welcher bei einer Ladeanforderung aktiviert wird, enthält folgende zwei SubStates:

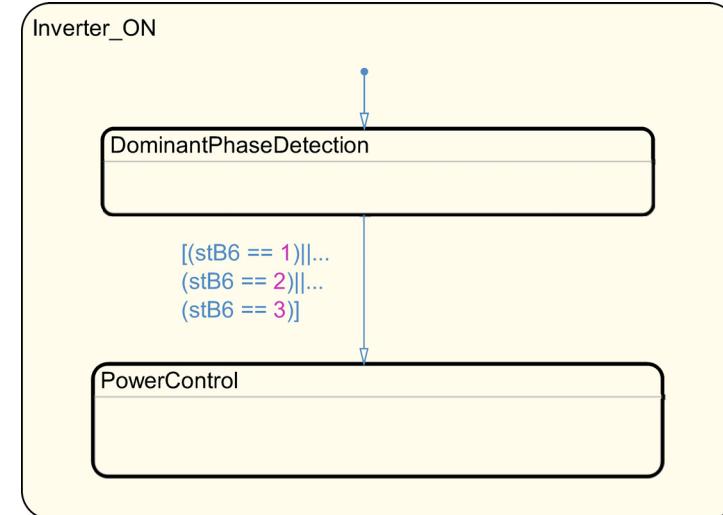
DominantPhaseDetection - Dies ist der Initiale State des SubStates *Inverter_ON*. Er wird ausgeführt, bis eine Dominante Phase vom B6 - Brückenmodul erkannt wurde.

PowerControl - Dieser State wird nach der DominatPhasedetection ausgeführt und regelt den Ladevorgang.

State Transition -> Inverter_OFF

Der State *Inverter_ON* wechselt zurück zum State *Inverter_OFF*, wenn die Ladeanforderung von SystemStatemanager zurückgenommen wird, d.h. *ActivateCharging* = 0 oder die Angeforderte Ladeleistung *pwrReqInverter* 0W ist.

Beim Ausführen dieser Transition wird der Fehlerzähler *counterErrorEvents* auf 0 gesetzt.



6.6.2.4 SubState DominantPhaseDetection

Der State enthält zusätzlich die Substates *ZMover_fix* und *ZMover_lift*. Beim aktivieren des States *DominantPhaseDetection* wird zunächst der SubState *ZMover_fix* aktiviert.

Beim aktivieren des States *DominantPhaseDetection* wird darüber hinaus der Anforderungsstatus *stPwrTrf* für den B6 Brückentreiben auf *DOM_PHASE_DETECT* (11) gesetzt. Die Anforderung *modeReqZM* an den ZMover wird auf 1 (*SpeedControl*) gesetzt. Die Soll - Ladeleistungsanforderung *pwrReqB6* für den B6 Brückentreiben wird auf 500W gesetzt.

Der Timeout - Counter *counterDPD* für die Dominant Phase Detection wird mit dem Applikationswert *timeoutDPD* (*PwrTrfStMgr_timeoutDPD_C*, Def. 2500) initialisiert.

Solange der State *DominantPhaseDetection* aktiv ist, wird bei jedem Aufruf der Timeout Counter *counterDPD* dekrementiert.

State Transition -> PowerControl

Der State *DominantPhaseDetection* wechselt zum State *PowerControl*, wenn der Status *stB6* des B6 Brückentreibers eine dominante Phase gefunden hat, das heisst, dass eine der folgenden Bedingungen zutrifft:

- *stB6* = 1 -> Phase U als dominant erkannt
- *stB6* = 2 -> Phase V als dominant erkannt
- *stB6* = 3 -> Phase W als dominant erkannt

6.6.2.5 SubState ZMover_fix

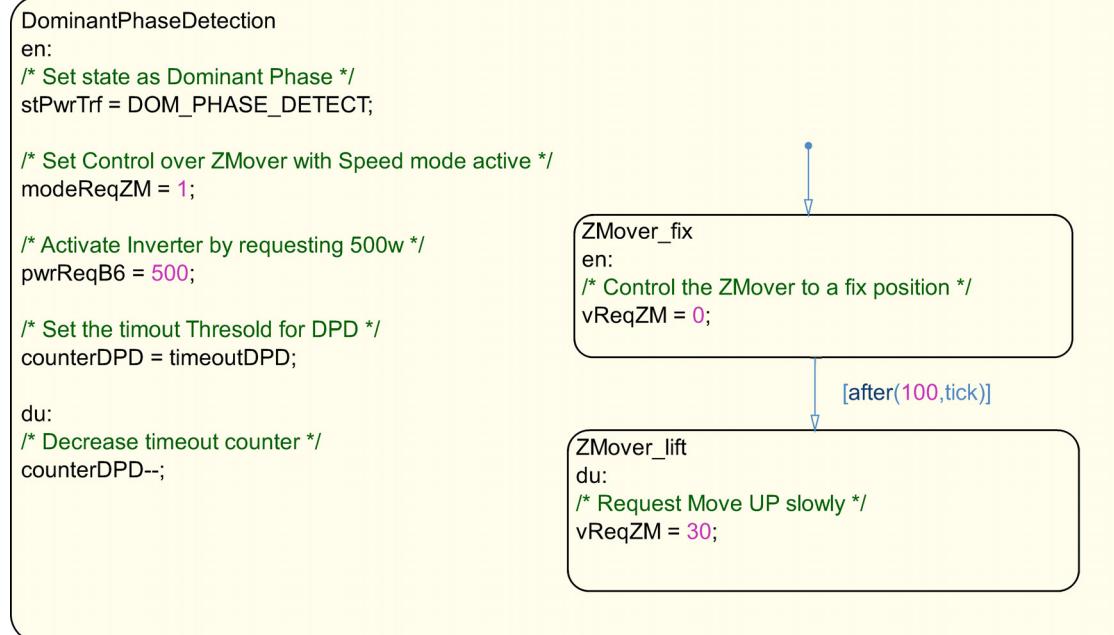
Dies ist der initiale SubState des States *DominantPhaseDetection*. Beim aktivieren dieses States wird die Sollgeschwindigkeit *vReqZM* für den ZMover auf 0 gesetzt.

State Transition -> ZMover_lift

Der State *ZMover_fix* wechselt 1 sek. (100 Ticks) nach seiner Aktivierung zum State *ZMover_lift*.

6.6.2.6 SubState ZMover_lift

Der State *ZMover_lift* setzt die Sollgeschwindigkeit *vReqZM* konstant auf den (geringen) Wert 30.

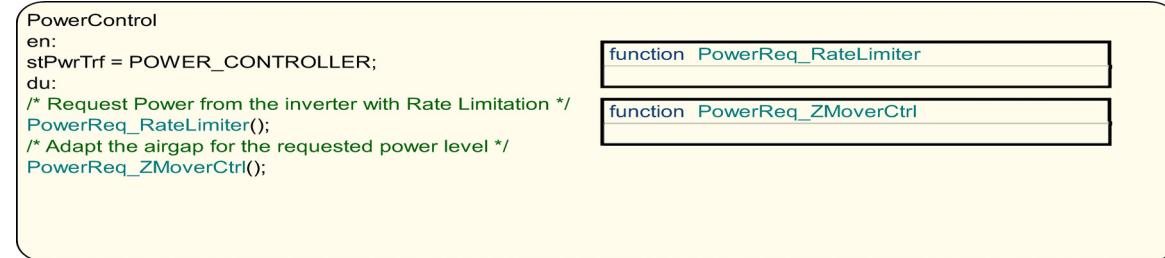


6.6.2.7 SubState PowerControl

Der SubState *PowerControl* regelt den induktiven Ladevorgang. Hierzu beinhaltet er die Funktionen *PowerReq_RateLimiter* und *PowerReq_ZMoverCtrl*.

Beim aktivieren dieses States wird der Anforderungsstatus *stPwrTrf* für den B6 Brückentreiber auf *POWER_CONTROLLER* (15) gesetzt.

Solange der State *PowerControl* aktiv ist wird zyklisch zunächst die Funktion *PowerReq_RateLimiter* und dann *PowerReq_ZMoverCtrl* aufgerufen.



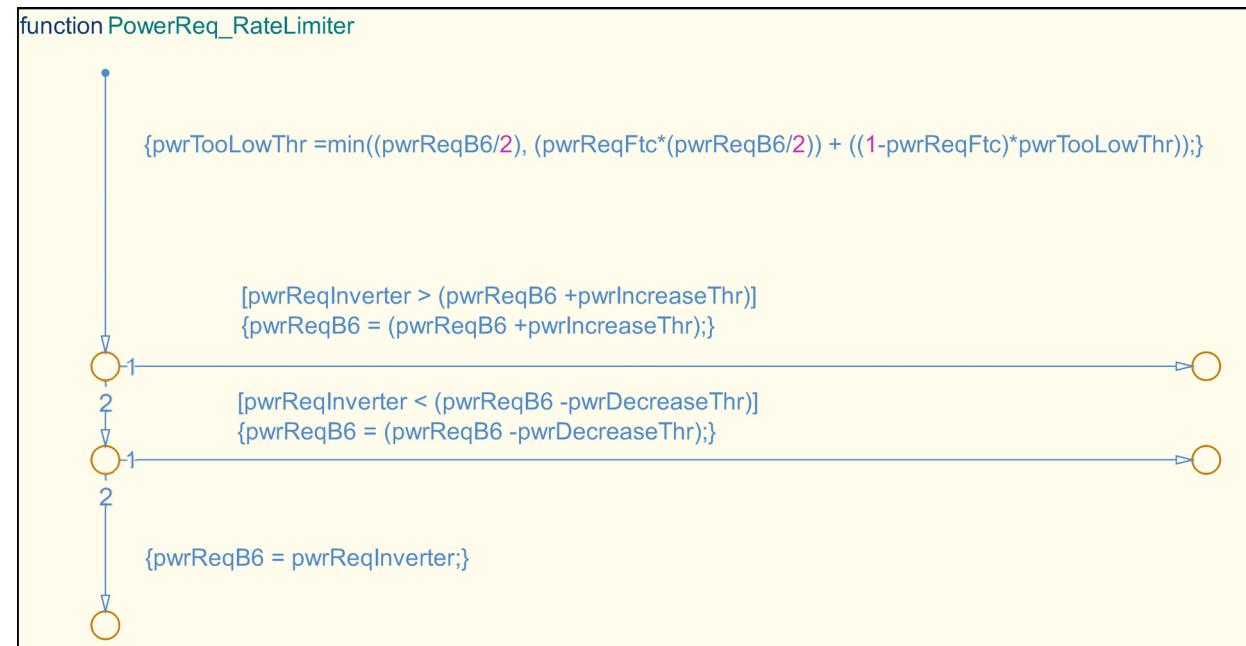
function *PowerReq_RateLimiter*

function *PowerReq_ZMoverCtrl*

6.6.2.8 Funktion PowerReq_RateLimiter

In dieser Funktion wird zunächst die Schwelle *pwrTooLowThr* für die Diagnose berechnet. Sie ergibt sich aus dem Minimum der Hälfte der an die B6 Brücke angeforderten Ladeleistung *pwrReqB6* und deren Tiefpassgefiltertem Wert.

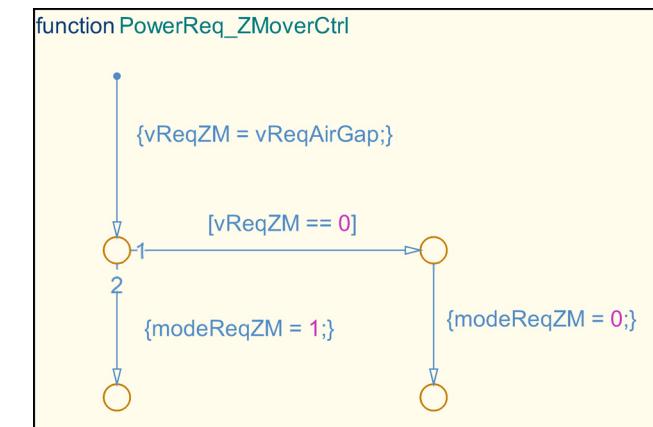
Als nächstes wird die an den B6 Brückentreiber übermittelte Soll - Ladeleistung *pwrReqB6* mit der Stepweite *pwrIncreaseThr* (*PwrTrfStMgr_pwrIncreaseThr_C*) nach oben bzw. *pwrDecreaseThr* (*PwrTrfStMgr_pwrDecreaseThr_C*) nach unten bis zur vom System geforderten Ladeleistung *pwrReqInverter* gerampft, d.h. Gradientenlimitiert.



6.6.2.9 Funktion PowerReq_ZMoverCtrl

Diese Funktion schaltet die Modusanforderung *modeReqZM* an den ZMover entsprechend der Geschwindigkeitsanforderung *vReqAirGap* vom *AirGapCtrl* Modul. Ist die ZMover Geschwindigkeitsanforderung von *AirGapCtrl* Modul 0, wird die ZMover Anforderung auf 0 (= keine Anforderung) gesetzt, ist sie 1, wird die ZMover Anforderung auf 1 (= *SpeedMode*) gesetzt.

Die Geschwindigkeitsanforderung *vReqAirGap* von *AirGapCtrl* Modul wird dann mittels der Sollgeschwindigkeit *vReqZM* an das ZMover Modul übermittelt.



6.6.2.10 State PowerSystem_event

Der State *PowerSystem_event* wird für 1 sek. aktiviert wenn im State *PowerSystem_Functional* ein Fehler auftritt welcher nicht ausmaskiert wurde.

Beim aktivieren dieses States werden folgende Aktionen ausgeführt:

- Der Fehlerzähler *counterErrorEvents* wird inkrementiert.
- Der Status *stPwrTrf* wird auf *PRE_ERROR* (254 = vorläufig defekt) gesetzt
- Die maximale Ladeleistung *pwrMaxSystem* sowie die Soll - Ladeleistung *pwrReqB6* werden auf 0W gesetzt
- Der Anforderungsstatus *faultsClearRequest* zum Löschen der Fehler wird (auf 1) gesetzt

```

PowerSystem_event
en:
counterErrorEvents++;
stPwrTrf = PRE_ERROR;
pwrMaxSystem = 0;
modeReqZM = 0;
pwrReqB6 = 0;
faultsClearRequest = 1;
ex:
faultsClearRequest = 0;
  
```

Beim verlassen dieses States wird der Anforderungsstatus *faultsClearRequest* zum Löschen der Fehler (auf 0) zurückgesetzt.

State Transition -> PowerSystem_Error

Der State *PowerSystem_event* wechselt zum State *PowerSystem_Error*, wenn der Fehlerzähler *counterErrorEvents* den Wert 10 überschreitet. Dies ist der Fall, wenn bei aktiviertem Laden ein Powertransferfehler (*erPwrTrf*) 10 mal erneut aufgetreten ist.

Er wechselt ebenfalls nach *PowerSystem_Error*, wenn 1 sek. nach Aktivierung trotz Aufforderung zum Löschen der Fehler immer noch ein Fehler des PFC (*erPFC*), der B6 Brücke (*erB6*), des ZMovers (*erZM*) oder der HWIO (*erI/O*) vorliegt.

State Transition -> PowerSystem_Functional

Der State *PowerSystem_event* wechselt zurück zum State *PowerSystem_Functional*, wenn 1 sek nach seiner Aktivierung (und implizit der Fehlerlöschanfrage) kein neuer Fehler vom PFC oder dem B6 - Brückentreiber gesetzt wurde. Einmal gesetzte ZMover - oder HWIO fehler werden von der Fehlerlöschanfrage nicht berücksichtigt.

Beim Wechsel zum *StatePowerSystem_Functional* wird der Status *erPwrTrf* gelöscht.

6.6.2.11 State PowerSystem_Error

Beim aktivieren des States *PowerSystem_Error* wird der Status *stPwrTrf* auf ERROR (255) gesetzt.

Die maximale Ladeleistung *pwrMaxSystem* sowie die Soll - Ladeleistung *pwrReqB6* werden auf 0W gesetzt.

Die Anforderung an den ZMover wird zurückgenommen (*modeReqZM* = 0).

```
PowerSystem_Error
en:
stPwrTrf = ERROR;
pwrReqB6 = 0;
modeReqZM = 0;
pwrMaxSystem = 0;
```

6.6.3 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

6.6.3.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

6.6.3.2 Ausgangsgrößen

keine	
-------	--

6.6.3.3 Interfaces

6.6.3.3.1 Interface

Data Element	Data Type	Auflösung	Einheit	Bedeutung

6.6.4 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

ZMOpCtrl_cntr	Current counter value
ZMOpCtrl_erComponent	Error field for Zmover component
ZMOpCtrl_stReqDiag	Diagnostic requested access
ZMOpCtrl_stReqPwrTrf	Power Transfer requested control
ZMOpCtrl_stReqStMgr	State Manager requested State
ZMOpCtrl_vReqDiag	Motor speed request from State Manager
ZMOpCtrl_vReqPwrTrf	Motor speed request from State Manager
ZMOpCtrl_vReqStMgr	Motor speed request from State Manager

6.6.5 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

6.6.6 Fehlerpfade

6.6.6.1 TBD

tbd

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

6.6.7 Initialisierung

tbd

7 Modul WspStMgr

Autor: Michael Münch

Software-Struktur-Position: 30_Implementation/10_WSP/30_STMGR

7.1 Allgemein

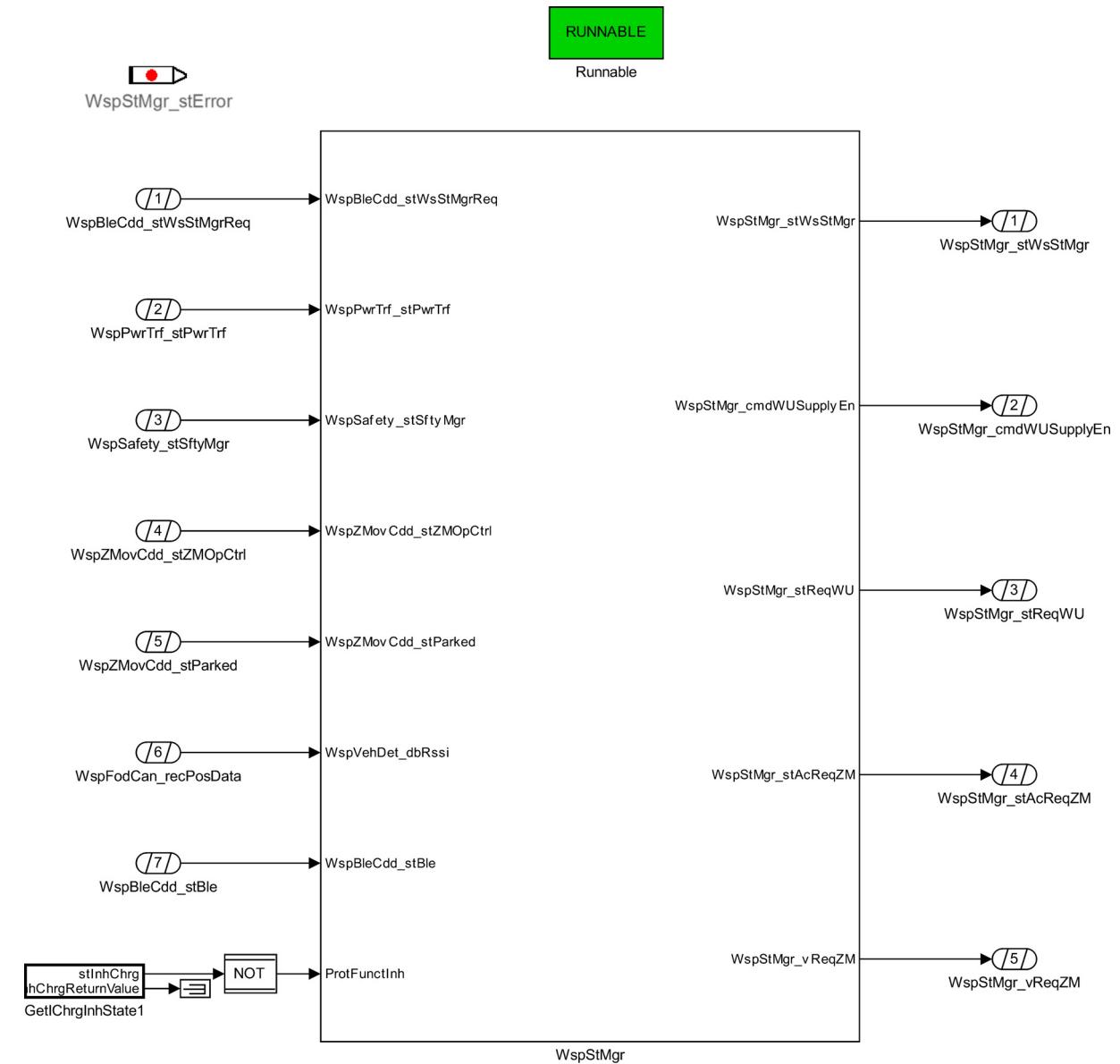
Das Modul *WspSfMgr* implementiert den Wayside System State - Manager. der Wayside System Statemanager steuert in Abhängigkeit der Vorgaben vom Oru Chargingmanager sowie vom aktuellen State des Wayside Safetymanagers den Z Mover sowie die Anforderungen an das Wayside Ladesystem.

Das Modul *WspSfMgr* enthält folgende Komponenten:

- *WspSfMgr* - Der Wayside Sysstem State Manager

7.2 Komponente WspStMgr_10ms

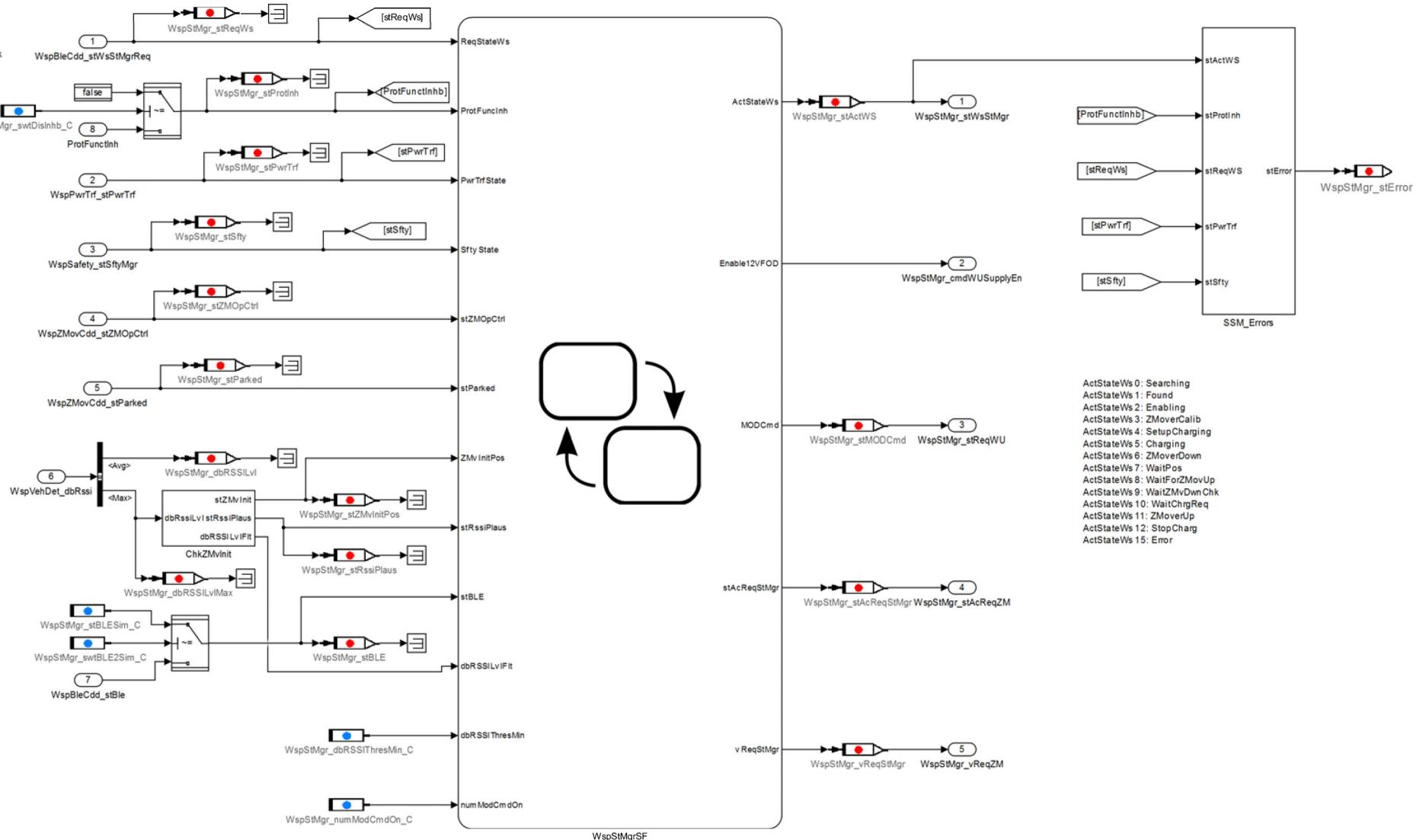
Das Subsystem **WspStMgr_10ms** stellt im Wesentlichen die Autosar IO - Schnittstellen zur Verfügung (siehe Schnittstellen). Diese werden an das Subsystem **WspStMgr** weitergegeben bzw. von diesem ausgelesen.



Softwaredokumentation

7.2.1 Block WspStMgr

```
ReqStateWs 0 -> 2Searching
ReqStateWs 1 -> 2Found
ReqStateWs 2 -> 2Enabling
ReqStateWs 3 -> 2WaitFodPre
ReqStateWs 4 -> 2WaitFodPre
ReqStateWs 5 -> 2WaitPos
ReqStateWs 6 -> 2ZMoverDown
ReqStateWs 7 -> 2WaitForZMovDwn
```



Hier ist der Wayside State Manager implementiert. Der überwiegende Teil des Statemanagers ist im Stateflow Block *WspStMgrSF* realisiert.

Im Block *ChkZMvInit* wird das Positionssignal der Oru ausgewertet, um den Z Mover nach Ladeanforderung bis zu einer definierten Höhe hochzufahren.

Im Block *SSM_Errors* werden alle Fehler des Wayside State Managers gesammelt und Bitcodiert in der Variablen *WspStMgr_stError* zur Verfügung gestellt.

Der aktuelle State des Wayside System Satemanagers wird in der Messgröße [*WspStMgr_stActWS*](#) zur Verfügung gestellt.

7.2.1.1 Systemstates *WspStMgr_stActWS*

Im folgenden sind die States [*WspStMgr_stActWS*](#) des Wayside System Statemanagers aufgelistet:

0	Standby / Searching
1	Found
2	Enabling
3	ZMoverCalib
4	SetupCharging
5	Charging
6	ZMoverDown
7	WaitPos
8	WaitForZMoverUp
9	WaitZMoverDwnChk
10	WaitChrgReq
11	ZMoverUp
12	StopCharging
15	Error

Softwaredokumentation

7.2.1.2 Fehlerstates

WspStMgr_stError

Hier werden die Fehlerbits der Statusvariablen [WspStMgr_stError](#) geändert.

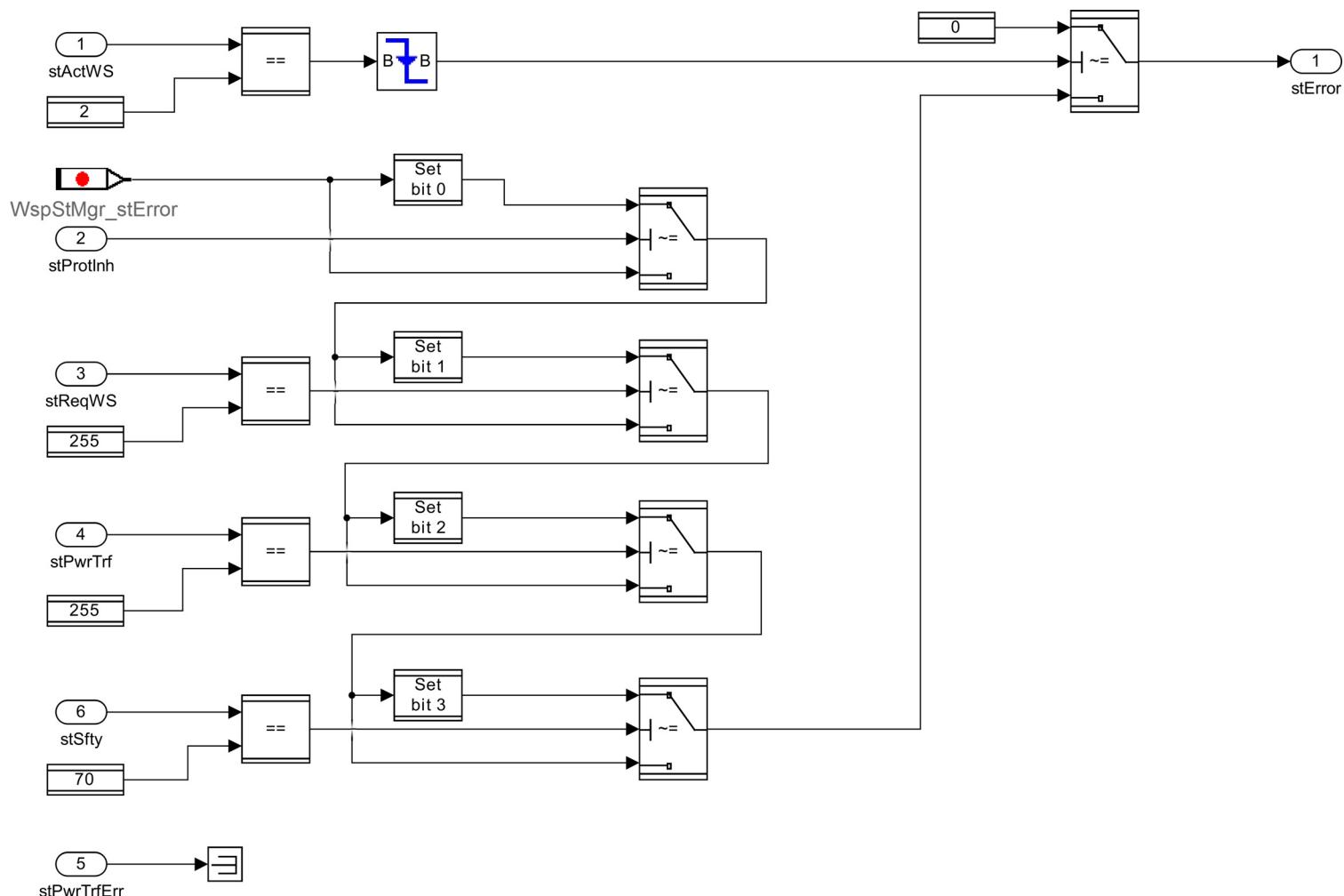
Alle Bits werden gelöscht, sobald der Enabling State (2) verlassen wird.

Bit 0 wird gesetzt, wenn der globale Inhibit-Charging Status gesetzt ist (`stProtInh = true`).

Bit 1 wird gesetzt, wenn der Anforderungsstatus `stReqWS` von der Oru den ungültigen Wert 255 hat.

Bit 2 wird gesetzt, wenn der Status `stPwrTrf` vom Power Transfermanager den Wert 255 = „nicht heilbarer Fehler“ hat.

Bit 3 wird gesetzt, wenn sich der Wayside Safety Manager im Status „WaitErroNotify“ befindet, d.h. `stSfty = 70`.



Softwaredokumentation

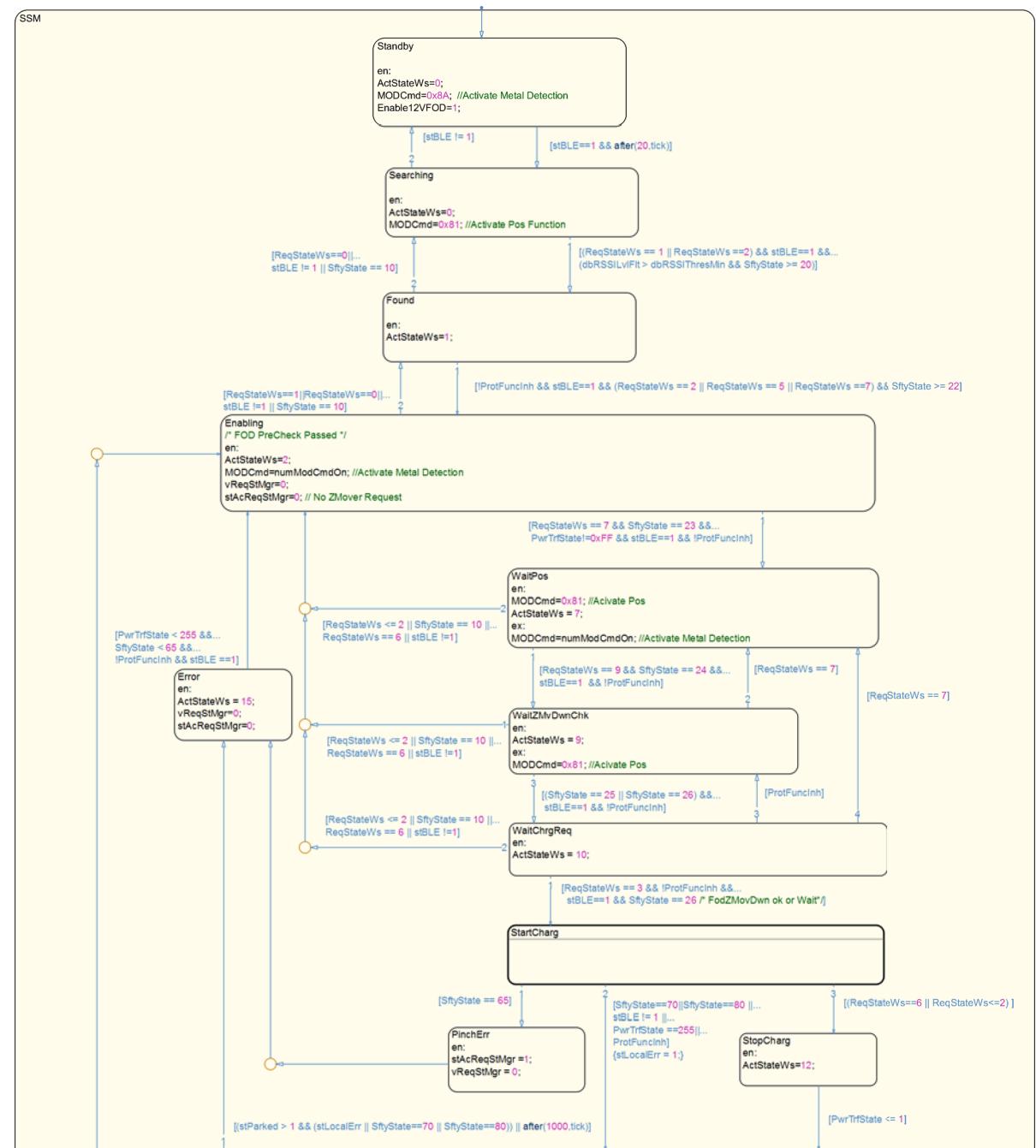
Die Fehlerbits der Variablen *WspStMgr_stError* werden in folgender Tabelle beschrieben:

Bit	Fehler
0	Inhibit Charging ist gesetzt
1	Anforderungsstatus <i>WspStMgr_stReqWs</i> der Oru hat Init Wert (255)
2	Status <i>WspStMgr_stPwrTrf</i> des Wayside Power Transfer Managers meldet Fehler (255)
3	Der Wayside Safety Manager befindet sich im Fehlerstatus (<i>WspStMgr_stSfty</i> = 70)

Softwaredokumentation

primove
true e-mobility

7.2.2 Stateflow Block WspStMgrSF



Softwaredokumentation

primove
true e-mobility

Dies ist der eigentliche Wayside System Statemanager. Der initiale State nach Start des Steuergerätes ist der State *Standby*.

7.2.2.1 State Standby

Beim aktivieren dieses States (nach Reset der Wayside) wird der Status *ActStateWs* ([WspStMgr_stActWS](#)) auf 0 gesetzt. Die Spannungsversorgung der FOD Platine wird freigegeben, d.h.

Enable12VFOD ([WspStMgr_stEna12VFOD](#)) wird auf 1 gesetzt und der FOD Modus wird aktiviert, indem der Sollstatus *MODCmd* ([WspStMgr_stMODCmd](#)) auf 0x8A gesetzt wird. Dieser Zustand wird von der Diagnose benötigt um den Nullvektor Abgleich durchzuführen.

State Transition -> Searching

Der State Standby wechselt zum State Searching, wenn die BLE Verbindung hergestellt ist, d.h. *stBLE* = 1 ([WspStMgr_stBLE](#)); jedoch frühestens nach 200ms nach aktivierung des States.

7.2.2.2 State Searching

Beim aktivieren dieses States wird der Status *ActStateWs* ([WspStMgr_stActWS](#)) auf 0 gesetzt. Der POS Modus wird aktiviert, indem der Sollstatus *MODCmd* ([WspStMgr_stMODCmd](#)) auf 0x81 gesetzt wird.

State Transition -> Found

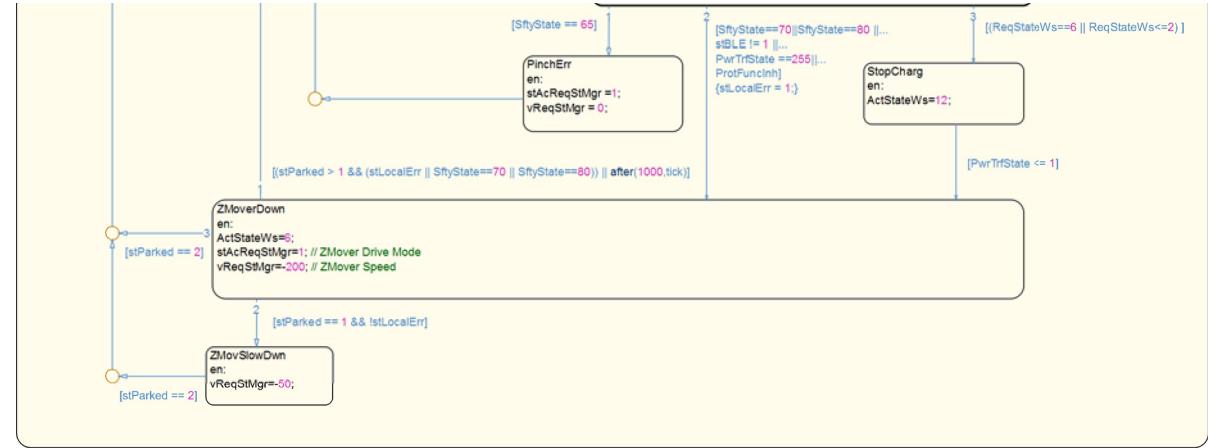
Der State Searching wechselt zum State Found, wenn die Oru sicher erkannt wurde. Dies ist der Fall, wenn alle folgende Bedingungen zutreffen:

- Die Weiterschaltungsanforderung *ReqStateWs* ([WspStMgr_stReqWs](#)) des Oru Charging Managers ist 1 oder 2
- Die BLE Verbinsung ist etabliert (*stBLE* = 1)
- Es wird ein ausreichendes POS Signal von der Oru empfangen *numRSSILvlFlt* > *numRSSIThresMin*
- Der Wayside Safety Manager befindet sich im State 20 (Idle)

State Transition -> Standby

Der State Searchning wechsel zurück zum State Standby, wenn die BLE Verbindung verloren geht (*stBLE* != 1)

7.2.2.3 State Found



Beim aktivieren des State Found wird die Statusvariable *ActStateWs* ([WspStMgr_stActWS](#)) auf 1 gesetzt

State Transition -> Enabling

Der State *Found* wechselt zum State *Enabling*, wenn alle folgenden Bedingungen erfüllt sind:

- Der Status des Wayside Safety Managers ist mindesten 22 (*FodPreChk*)
- Es liegt keine Ladeverhinderungsanforderung des Error Handlers vor, d.h. *ProtFuncInh* ([WspStMgr_stProtInh](#)) = *false*
- Die BLE Verbindung ist noch immer aktiv (*stBLE* = 1)
- Die Weiterschaltanforderung *ReqStateWs* des Oru Charging Mangers ist 2, 5 oder 7

State Transition -> Searching

Der State *Found* wechselt zurück zu State *Searching*, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Der Anforderungsstatus *ReqStateWs* des Oru Charging Manager ist 0
- die BLE Verbindung ist verloren gegangen (*stBLE* != 1)
- Der Wayside Safety Manger befindet sich im State Standby (*SftyState* = 10)

7.2.2.4 State Enabling

Beim aktivieren des States Enabling wird die Statusvariable *ActStateWs* auf 2 gesetzt. Der FOD Anforderungsstatus *MODCmd* wird auf den Applikationsparameter *numModCmdOn* (Standardmäßig 0x8A -> Activate FOD) gesetzt. Der Anforderungsstatus *stAcReqStMgr* ([WspStMgr_stAcReqStMgr](#)) für den Z Mover wird auf 0 (keine Anforderung) gesetzt.

State Transition -> WaitPos

Der State *Enabling* wechselt zum State *WaitPos*, wenn alle folgenden Bedingungen erfüllt sind:

- Der Anforderungsstatus *ReqStateWs* von Oru Charging Manager ist 7
- Der aktuelle Status *SftyState* des Wayside Safety Managers ist 23 (*FodChkZMoverDwn*)
- Es liegt kein Fehler des *PwrTrf* Moduls vor, d.h. *PwrTrfState* ([WspStMgr_stPwrTrf](#)) ist nicht 255 (0xFF)
- Die BLE Verbindung ist noch aktiv (*stBLE* = 1)

- Es liegt keine Ladeverhinderungsanforderung des Error Handler vor (*ProtFuncInh = false*)

State Transition -> Found

Der State *Enabling* wechselt zurück zum State *Found*, wenn mindesten eine der folgenden Bedingungen vorliegt:

- Der Anforderungsstatus *ReqStateWs* des Oru Charging Managers ist 0 oder 1
- Die BLE Verbindung ist nicht mehr aktiv (*stBLE != 1*)
- Der Safety Statemanager der Wayside befindet sich im Standby, d.h. *SftyState* = 10

7.2.2.5 State WaitPos

Beim aktivieren dieses States wird der POS Modus aktiviert, indem der FOD Anforerungsstatus *MODCmd* auf 0x81 gesetzt wird. Die System Status Variabole *ActStateWs* wird auf 7 gesetzt. Beim verlassen dieses States wird der FOD Anforderungsstate *MODCmd* auf den Applizierbaren Wert *numModCmdOn* ([WspStMgr_numModCmdOn_C](#)) gesetzt. Dieser hat normalerweise den Wert 0x8A (MOD Mode).

State Transition -> WaitZMvDwnChk

Der State *WaitPos* wechselt zum State *WaitZMvDwnChk*, wenn alle folgenden Bedingungen erfüllt sind:

- Der von der Oru geforderte State *ReqStateWs* ist 9 (*WaitZMvDwnChk*)
- Der aktuelle State *SftyState* des Wayside Safety Managers ist 24 (*WaitPos*)
- Die BLE Verbindung ist noch hergestellt (*stBLE = 1*)
- Es ist bisher keine Ladeverhinderungsanforderung erfolgt (*ProtFuncInh = false*)

State Transition -> Enabling

Der State *WaitPos* wechselt zurück zum State *Enabling*, wenn mindesten einer der folgenden Bedingungen erfüllt ist:

- Der Anforderungsstatus *ReqStateWs* vom Oru Chargingmanger ist kleiner oder gleich 2 (*Enabling*) oder 6 (*ZMoverDown*)
- Der Wayside Safety Manager befindet sich im Standby State (*SftyState* = 10)
- Es besteht keine BLE Verbindung mehr (*stBLE != 1*)

7.2.2.6 State WaitZMvDwnChk

Beim aktivieren des State *WaitZMvDwnChk* wird der Status *ActStateWs* auf 9 gesetzt. Beim verlassen dieses States wird der POS Modus aktiviert, indem der diesbezügliche Anforderungsstatus *MODCmd* auf 0x81 gesetzt wird.

State Transition -> WaitChrgReq

Der State *WaitZMvDwnChk* wechselt zum State *WaitChrgReq*, wenn alle folgenden Bedingungen erfüllt sind:

- Der Safety Status *SftyState* der Wayside ist 25 (*WaitReqChrg*) oder 26 (*FodChkZMoverUp*)
- Die BLE Verbindung ist noch hergestellt (*stBLE* = 1)
- Es ist bisher keine Ladeverhinderungsanforderung erfolgt (*ProtFuncInh* = *false*)

State Transition -> WaitPos

Der State *WaitZMvDwnChk* wechselt zurück zum State *WaitPos*, wenn der Anforderungsstatus *ReqStateWs* der Oru 7 ist.

State Transition -> Enabling

Der State *WaitPos* wechselt zurück zum State *Enabling*, wenn mindesten einer der folgenden Bedingungen erfüllt ist:

- Der Anforderungsstatus *ReqStateWs* vom Oru Chargingmanger ist kleiner oder gleich 2 (*Enabling*) oder 6 (*ZMoverDown*)
- Der Wayside Safety Manager befindet sich im Standby State (*SftyState* = 10)
- Es besteht keine BLE Verbindung mehr (*stBLE* != 1)

7.2.2.7 State WaitChrgReq

Beim aktivieren des State *WaitChrgReq* wird der Status *ActStateWs* auf 10 gesetzt.

State Transition -> StartCharg

Der State *WaitChrgReq* wechselt zum State *StartCharg* wenn alle folgenden Bedingungen erfüllt sind:

- Der Anforderungsstatus *ReqStateWs* der Oru ist 3
- Es ist bisher keine Ladeverhinderungsanforderung erfolgt (*ProtFuncInh* = *false*)

Softwaredokumentation

- Die BLE Verbindung ist noch hergestellt (*stBLE* = 1)
- Der Safety Status *SftyState* der Wayside ist 26 (*FodChkZMoverUp*)

7.2.2.8 State StartCharg

Der State *StartCharg* behandelt den Ladevorgang inclusive das hochfahren des ZMovers. Die darin enthaltenen SubStates werden im folgenden beschrieben.

State Transition -> PinchErr

Der State *StartCharg* wechselt zum State *PinchErr*, wenn der Status *SftyState* des Wayside Safety Managers 65 (*PinchErr*) ist.

State Transition -> StopCharg

Der State *StartCharg* wechselt zum State *StopCharg*, wenn der Anforderungsstatus *ReqStateWs* vom Oru Charging Manager gleich 6 (*ZMoverDown*) oder kleiner / gleich 2 (*Enabling*) ist.

State Transition -> ZMoverDown

Der State *StartChrg* wechselt zum State *ZMoverDown*, wenn mindesten eine der folgenden Bedingungen erfüllt ist:

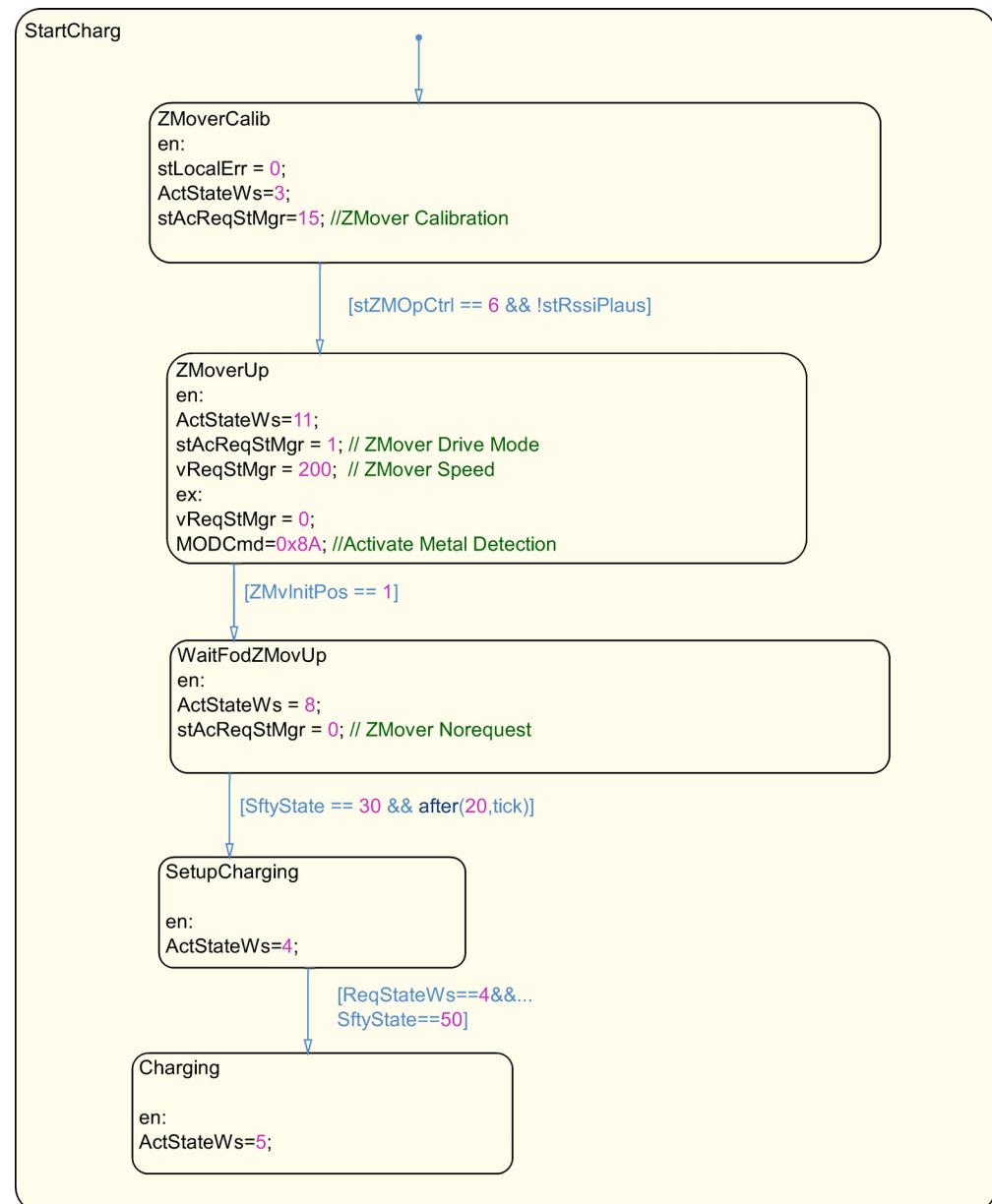
- Es besteht keine BLE Verbindung mehr (*stBLE* != 1)
- Der Status *SftyState* des Wayside Safety Managers ist 70 (*Error*) oder 80 (*Shutdown*)
- Der Status *PwrTrfState* des Powertransfermanagers ist 255 (*Error*)
- Der Ladeverhinderungsstatus *ProtFuncInh* ist gesetzt

Beim Ausführen dieser State Transition wird der lokale Fehlerstatus *stLocalErr* auf 1 gesetzt.

7.2.2.9 SubState ZMoverCalib

Der Substate *ZMoverCalib* ist der Initiale State nach der Aktivierung des States *StartCharg*. Beim aktivieren des SubStates *ZMoverCalib* wird der Anforderungsstatus *stAcReqStMgr* an den ZMover auf 15 (*Calibration*) gesetzt. Der der lokale Fehlerstatus *stLocalErr* wird auf 0 gesetzt und die Statusvariable *ActStateWs* des Wayside State Managers wird auf 3 gesetzt.

State Transition -> ZMoverUp



Der State *ZMoverCalib* wechselt zum State *ZMoverUp*, wenn der Zustand des ZMovers gleich 6 (*Standby*) ist und kein Plausibilitätsfehler des POS Signals vorliegt, d.h. *stRSSIPlaus* (*WspStMgr_stRssiPlaus*) = *false*.

7.2.2.10 SubState *ZMoverUp*

Beim aktivieren des SubStates *ZMoverUp* wird . Die Statusvariable *ActStateWs* des Wayside State Managers wird auf 11 gesetzt. Der Anforderungsstatus *stAcReqStMgr* (*WspStMgr_stAcReqStMgr*) für den ZMover wird auf 1 (DriveMode) gesetzt. Die Sollgeschwindigkeit *vReqStMgr* (*WspStMgr_vReqStMgr*) für den ZMover wird auf 200 gesetzt.

Beim verlassen diese Substates wird die ZMover Sollgeschwindigkeit *vReqStMgr* auf 0 gesetzt und der FOD Modus angefordert (*MODCmd* wird auf 0x8A gesetzt).

State Transition -> *WaitFodZMovUp*

Der State *ZMoverUp* wechselt zum State *ZMoverCalib*, wenn der Status *ZMvInitPos* (*WspStMgr_stZMvInitPos*) den Wert 1 hat, also der POS Signalpegel einen definierten Wert erreicht hat.

7.2.2.11 SubState *WaitFodZMovUp*

Beim aktivieren des SubStates *WaitFodZMovUp* wird die Statusvariable *ActStateWs* auf 8 gesetzt. Der Anforderungsstatus *stAcReqStMgr* an den ZMover wird auf 0 (*No-Request*) gesetzt .

State Transition -> *SetupCharging*

Der State *WaitFodZMovUp* wechselt zum State *SetupCharging*, wenn der Status *SftyState* des Wayside Safetymanagers den Wert 30 (*VehInChargePosition*) hat; frühestens jedoch 200ms nach Aktivierung des States.

7.2.2.12 SubState *SetupCharging*

Bei Aktivierung dieses State wird die Statusvariable *ActStateWs* auf 4 gesetzt. Dieser State dient der Synchronisierung der anderen Statemanger während des FOD ZMover up und des Vehicle Detection Checks

State Transition -> *Charging*

Der State *SetupCharging* wechselt zum State *Charging* wenn der Anforderungsstatus *ReqStateWs* vom Oru ChargingManager gleich 4 (*ActivateCharging*) ist und der Status *SftyState* des Wayside Safety Managers 50 (*Charging*) ist.

7.2.2.13 SubState *Charging*

Beim aktivieren des States *Charging* wird der Status *ActStateWs* des Wayside System Manager auf 5 (*Charging*) gesetzt.

State Transition -> keine

Der State *Charging* kann nur implizit durch eine StateTransition des übergeordneten States *StartCharg* wieder verlassen werden.

7.2.2.14 State PinchErr

Beim aktivieren des States *PinchErr* wird das Beenden des Ladevorgangs angefordert, indem der Anforderungsstatus *ActivateCharging* ([WspStMgr_stActvCharging](#)) auf 0 gesetzt wird. Der ZMover wird in den DriveMode geschaltet (*stAcReqStMgr* = 1) und die Sollgeschwindigkeit des ZMovers auf 0 gesetzt (*vReqStMgr* = 0).

State Transition -> Error

Der State *PinchErr* wechselt ohne weitere Bedingung nach einmaliger Ausführung zum State *Error*.

7.2.2.15 State StopCharg

Beim aktivieren des States *StopCharg* wurde das Beenden des Ladevorgangs angefordert. Beim aktivieren des States *StopCharg* wird der Status *ActStateWs* des Wayside System Manager auf 12 (*StopCharging*) gesetzt.

State Transition -> ZMoverDown

Der State *WaitPwrTrf* wechselt zum State *ZMoverDown*, wenn der Status *PwrTrf* des Power Transfermanagers 0 oder 1 ist.

7.2.2.16 State ZMoverDown

Beim aktivieren des States *ZMoverDown* wird die Statusvariable *ActStateWs* des Wayside System Statemanagers auf 6 gesetzt. Das Beenden des Ladevorgangs wird angefordert, indem der entsprechende Anforderungsstatus *ActivateCharging* auf 0 gesetzt wird. Der ZMover wird in den DriveMode geschaltet (*stAcReqStMgr* = 1) und die Sollgeschwindigkeit des ZMovers auf -200 gesetzt (*vReqStMgr* = -200) um den ZMover runter zu fahren.

State Transition -> ZMovSlowDwn

Der State *ZMoverDwn* wechselt zum State *ZMovSlowDwn*, wenn der Z-Mover Status *stParked* ([WspStMgr_stParked](#)) den Wert 1 hat, d.h. ein Bodenkontaktsensor von zweien ausgelöst ist.

State Transition -> Error

Der State *ZMoverDown* wechselt zum State *Error*, wenn beide Bodenkontaktsensor ausgelöst sind (*stParked* > 1) und mindestens eine der folgenden Fehlerbedingungen vorliegt:

- der Ladevorgang wurde durch einen Fehler beendet (*stLoaciErr* = 1)
- Der Wayside Safetymanager befindet sich im Status Error (*SftyState* = 70)
- Der Wayside Safetymanager befindet sich im Status ShutDown (*SftyState* = 80)

Der State *ZMoverDwn* wechselt auch dann zum State *Error*, wenn er bis spätesten 10sek. nach seiner Aktivierung nicht anderweitig verlassen wurde.

State Transition -> Enabling

Der State *ZMoverDown* wechselt zum State *Enabling*, wenn beide Bodenkontaktsensoren ausgelöst sind (*stParked* = 2).

7.2.2.17 State *ZMovSlowDwn*

Beim aktivieren dieses States wird die Sollgeschwindigkeit *vReqStMgr* des Z-Movers auf -50 herabgesetzt, damit der Z-Mover beim normalen herunterfahren nicht zu hart gegen die Bodenplatte schlägt.

State Transition -> Enabling

Der State *ZMovSlowDwn* wechselt zum State *Enabling*, wenn beide Bodenkontaktsensoren ausgelöst sind (*stParked* = 2).

7.2.2.18 State *Error*

Beim aktivieren des States *Error* wird die Statusvariable *ActStateWs* des Wayside System Satemanagers auf 15 gesetzt. Der Sollmodus *stAcReqStMgr* des Z-Movers wird auf 0 (keine Anforderung) und die Sollgeschwindigkeit *vReqStMgr* auf 0 gesetzt.

State Transition -> Enabling

Der State *Error* wechselt zum State *Enabling*, wenn alle folgenden Bedingungen erfüllt sind:

- Der Status *PwrTrfState* des Power Transfermanagers ist nicht 255 (*Error*)
- Der Wayside Safetymanager befindet sich in keinem der Error States (*SftyState* < 65)
- Es besteht keine Ladeverhinderungsanforderung (*ProtFuncInh* = *false*)
- Es besteht eine BLE Verbindung (*stBLE* = 1)

7.2.3 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Requiered Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

7.2.3.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
RP_ReqStateWs	RPort	SCI_SSM_ReqStateWs_if	Angeforderter State von Oru Charging Manager
GetChrgInhState	OperationCall	ifInhibitCharging	Abfrage Ladeverhinderungsanforderung vom Error Handler
RP_PwrTrfState	RPort	SCI_SSM_PwrTrfState_if	Status des Wayside Power Transfer Managers
RP_PwrTrf_ErrorState_u32_T	RPort	SCI_SSM_PwrTrf_ErrorStat e_u32_T_if	Fehlerstatus des Wayside Power Transfer Managers
RP_SftyMgrError_u32_T	RPort	SCI_SSM_SftyMgrError_u32 _T_if	Fehlerstatus des Wayside Safety Manager
RP_SftyMgrState_u8_T	RPort	SCI_SSM_SftyMgrState_u8_ T_if	Status des Wasyside Safety Managers
WspZMovCdd_stZMOpCtrl	RPort	StatusByte	Staus des Z-Mover Managers
WspZMovCdd_stParked	RPort	StatusByte	Status der Bodenplattensensoren des Z-Movers
WspZMovCdd_stZMCompErr	RPort	StatusDWord	Fehlerstatus des Z-Movers
RssiLvl	RPort	RssiLvl	Signalfeldstärke des POS Signals
RP_BLE_State	RPort	ifBleState	Verbindungsstatus des BLE

7.2.3.2 Ausgangsgrößen

Name	Typ	Interface / Argument	Beschreibung
PP_ActStateWs	PPort	SCI_SSM_ActStateWs_if	Aktueller Status des Wayside System Statemanagers
PP_ActivateCharging	PPort	SCI_SSM_Activate_charging_b_T_if	Anforderungsstatus Laden
PP_Z_MoverUp	PPort	SCI_SSM_Z_MoverUp_if	Status Z-Mover ist hochgefahren
PP_Z_MoverDown	PPort	SCI_SSM_Z_MoverDown_if	Status Z-Mover ist runtergefahren
PP_Enable_12V_FOD_b_T	PPort	SCI_SSM_Enable_12V_FOD_b_T_if	Sollstatus Spannungsversorgung FOD Platine
PP_MODCmd	PPort	SCI_SSM_MODCmd_if	Sollstatus FOD -> POS oder FOD
WspStMgr_stAcReqStMgr	PPort	StatusByte	Anforderungsstatus Z-Mover Betriebsmodus
WspStMgr_vReqStMrg	PPort	ifZMoverMotSpeed	Anforderung Z-Mover Sollgeschwindigkeit

7.2.3.3 Interfaces

7.2.3.3.1 Interface SCI_SSM_ReqStateWs_if

Data Element	Data Type	Auflösung	Einheit	Bedeutung
SCI_SSM_ReqStateWsValue	uint8	1	-	Status byte

7.2.3.3.2 Interface ifInhibitCharging

Data Element	Data Type	Auflösung	Einheit	Bedeutung
isChrgInhibited	bool	1	-	Ladeverhinderungsstatus

7.2.3.3.3 Interface SCI_SSM_PwrTrfState_if

Data Element	Data Type	Auflösung	Einheit	Bedeutung
SCI_SSM_PwrTrfStateValue	uint8	1	-	Statusbyte

7.2.3.3.4 Interface SCI_SSM_PwrTrf_ErrorState_u32_T_if

Data Element	Data Type	Auflösung	Einheit	Bedeutung
SCI_SSM_PwrTrf_ErrorState_u32_TValue	uint32	1	-	Status DWord

7.2.3.3.5 Interface StatusByte

Data Element	Data Type	Auflösung	Einheit	Bedeutung
StatusByte	uint8	1	-	Status Byte

7.2.3.3.6 Interface RssiLvl

Data Element	Data Type	Auflösung	Einheit	Bedeutung
Value	sint16	1	-	Wert Empfangsfeldstärke

7.2.3.3.7 Interface ifZMoverMotSpeed

Data Element	Data Type	Auflösung	Einheit	Bedeutung
ZMoverMotSpeedVal	sint16	1	-	Z-Mover Geschwindigkeit

7.2.4 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>WspStMgr_numReqZMovPos</i>	Angeforderte Z-Mover Position
<i>WspStMgr_numReqZMovRate</i>	Angeforderte Z-Mover Geschwindigkeit
<i>WspStMgr_stAcReqStMgr</i>	Geforderte ZMover Status: 0-No request; 1 - Drive Request; 15 - Calibration request possible only if actual state is Wait-ForCalib(3)
<i>WspStMgr_stActWS</i>	Aktueller Status Wayside Statemachine
<i>WspStMgr_stBLE</i>	aktuell vom Port eingelesener BLE Status in PAD Statemanager
<i>WspStMgr_stEna12VFOD</i>	Anforderungsstatus 12V Versorgung FOD einschalten
<i>WspStMgr_stError</i>	Fehlerstati SystemStateMachine WSP Bit Codiert
<i>WspStMgr_stMODCmd</i>	Anforderungsstatus Metallobjekt Erkennung von WS Statemachine
<i>WspStMgr_stParked</i>	Status Z Mover Parking position: 0 - both parking sensor detection are OFF; 1 - one parking sensor is ON; 2 - system completely Parked - both sensors ON
<i>WspStMgr_stProtInh</i>	Status Protection 1 = Charging inhibited
<i>WspStMgr_stPwrTrf</i>	Aktueller Status WS PowerTransfer (B6-Brücke)
<i>WspStMgr_stReqWs</i>	Anforderungsstatus an WS von Oru Charging Manager
<i>WspStMgr_stRssiPlaus</i>	Plausibilität des Rssi Signals (0 = Ok, 1 = nicht Plausibel)
<i>WspStMgr_stSfty</i>	Aktueller Status WS Safety State Manager
<i>WspStMgr_stZMOpCtrl</i>	Actual Z-Mover State: 1- WaitForVoltage; 2 - Parking; 3 - WaitForCalib; 4 - Current_Test; 5 - Hall_Test; 6 - StandBy; 7 - StMgr_Control; 8 - PwrTrf_Control; 21 - Diag_Control; 255 - Deffect
<i>WspStMgr_stZMvInitPos</i>	Status Z.Mover init Pos erreicht (falls tue)
<i>WspStMgr_vReqStMgr</i>	Angefordete Fahrgeschwindigkeit Z-Mover in 'motor ticks'/10ms
<i>WspStMgr_dbRSSILvl</i>	Zahl Signalfeldstärke PosSignal RSSI
<i>WspStMgr_dbRSSILvlFilt</i>	Zahl Signalfeldstärke PosSignal RSSI gefiltert
<i>WspStMgr_dbRSSILvlMax</i>	Zahl Signalfeldstärke PosSignal RSSI Maximalwert

7.2.5 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

WspStMgr_facLpRSSI_C	Dämpfungsfaktor Lowpass RSSI Signalstärke
WspStMgr_numModCmdOn_C	Wert um MOD einzuschalten (normal 138 = 8Ah)
WspStMgr_dbRSSIThresMin_C	Minimaler RSSI FarField Wert zur Freigabe der Wsp System State Machine
WspStMgr_dbZMvInitThres_C	Schwelle RSSI Signalstärke zur InitPos Erkennung
WspStMgr_stBLESim_C	BLE Status im Simulationsmodus
WspStMgr_swtBLE2Sim_C	Schaltet BLE Status auf Simulationsmodus (wenn true)
WspStMgr_tiDebDefRssiPlaus_C	Defektentprellzeit Rssi Plausibilisierung
WspStMgr_tiDebOkRssiPlaus_C	Heilentprellzeit Rssi Plausibilisierung
WspStMgr_swtDisInhb_C	Disables Inhibit Charging (if true)

7.2.6 Fehlerpfade

7.2.6.1 keine

keine

Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

7.2.7 Initialisierung

tbd.

8 Modul WspLed (PAD Led Steuerung)

Autor: Bogdan Marcu

Software-Struktur-Position: 30_Implementation/10_WSP/35_LED

8.1 Allgemein

Das Modul *WspLed* implementiert die Ansteuerung inclusive Farbgebung der LED des Pad's.

Das Modul WspLed enthält folgende Komponente:

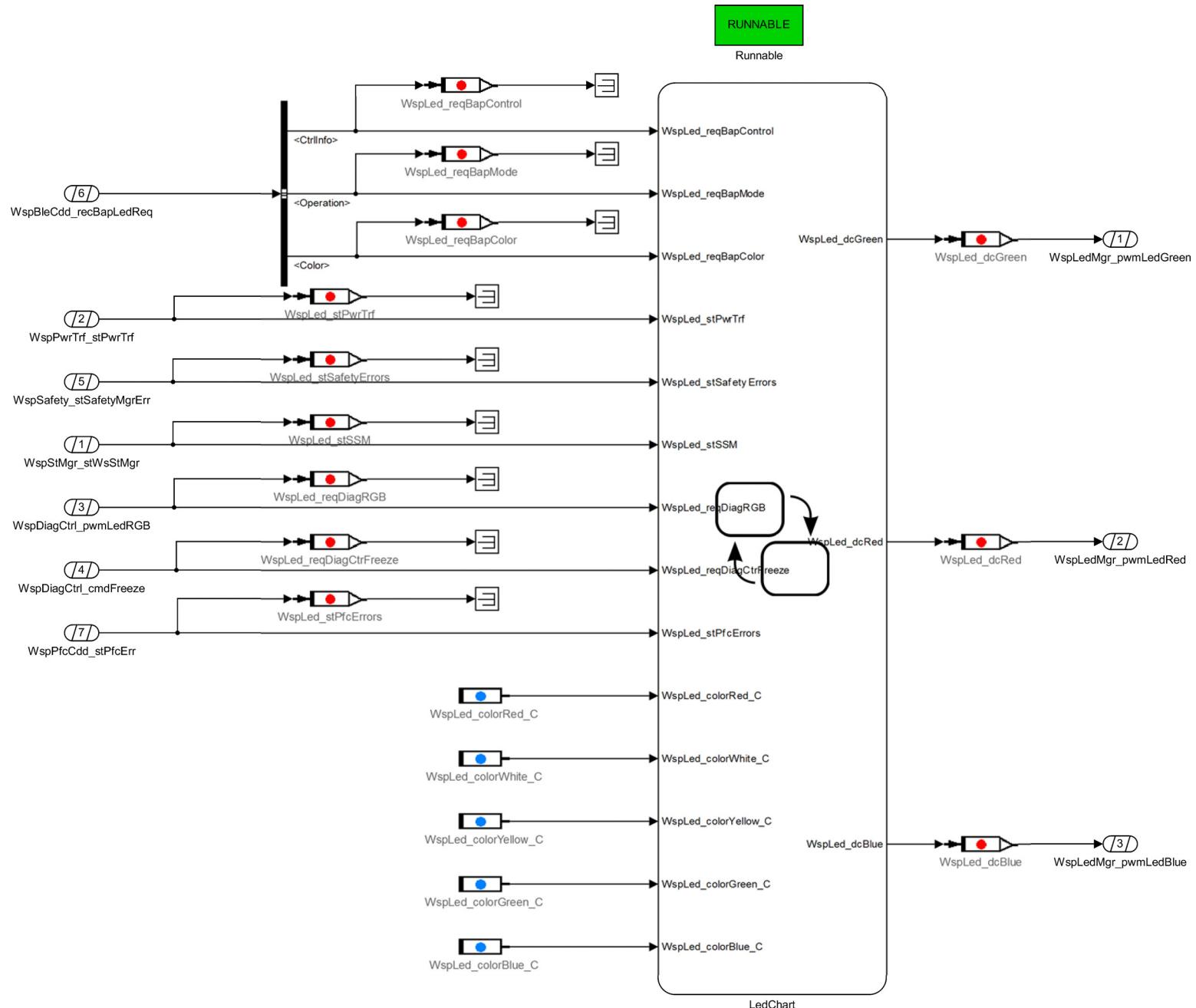
- *LedMgr_MainFunction* - StateManager zur Ansteuerung der Grundfarben der PAD - LED.

Softwaredokumentation

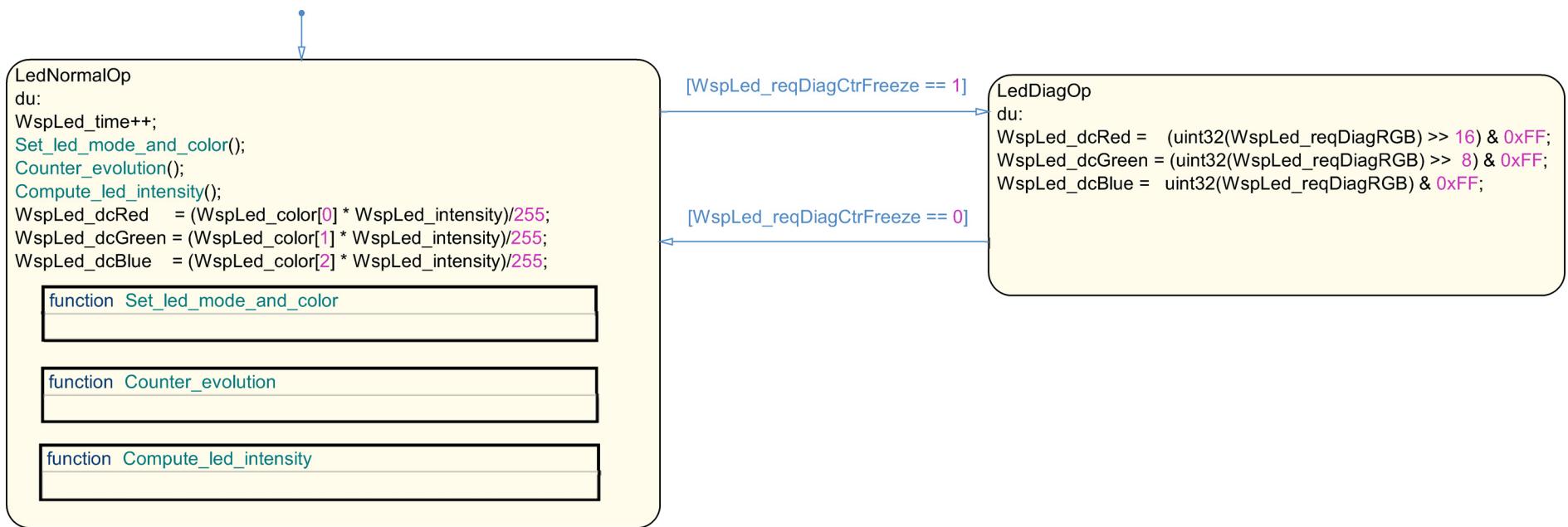
8.2 Komponente LedMgr_MainFunction

Im Modul LedMgr_MainFunction ist die Ansteuerlogik für die 3 Grundfarben der PAD LED im Stateflow Diagramm LedChart implementiert.

Dort wird in Abhängigkeit von den jeweiligen Systemzuständen des: Wayside System und Safety Managers, des Powertransfer Managers, des Zustandes des PFC, der BAP Anforderung sowie der Anforderung des Diagnosesystems die Auszugebende LED Farbe bestimmt.



8.2.1 Stateflow Chart LedChart



Hier wird die Farbe und gegebenenfalls die Blinkfrequenz der PAD LED in Abhängigkeit vom Systemzustand bestimmt. Die Funktion glieder sich in zwei States. Der Initiale State ist *LedNormalOp* für die normale Ausführung der LED Funktion. Falls das Diagnosessttem die Steuerung der LED anfordert (*WspLed_reqDiagCtrlFreeze* = 1) wird zum State *LedDiagOp* verzweigt.

8.2.2 State LedNormalOp

Dies ist der initiale State. Er wird nach dem Systemstart aktiviert. Bei jedem Taskdurchlauf wird die Variable *WspLed_time* inkrementiert. Sie dient als Zeitbasis für das Intitative Blinken der LED nach Neustart des PAD's.

Im folgenden werden dann zyklisch nacheinander die Funktionen *Set_led_mode_and_color()*, *Counter_evolution()*, und *Compute_led_intensity()*, welche ebenfalls in diesem State definiert sind, ausgeführt.

Das Ergebnis der Funktion *Set_led_mode_and_color* ist dann jeweils der Rotanteil in *WspLed_color[0]*, der Grünanteil in *WspLed_color[1]* sowie Der Blauanteil in *WspLed_color[2]* der darzustellenden RGB LED Farbe. Die einzelnen Farbanteile liegen jeweils im Wertebereich [0, 255];

Das Ergebnis der Funktion *Compute_led_intensity* ist die aktuelle Helligkeit *WspLed_intensity* der darzustellenden LED Farbe. Der Wert liegt ebenfalls im Bereich zwischen 0 = LED aus und 255 = maximale Helligkeit.

Die RGB mit der Helligkeit multiplizierten Farbanteile *WspLed_dcRed*, *WspLed_dcGreen* und *WspLed_dcBlue* werden dann, normiert auf den Wertebereich [0, 255] an den LED Treiber übermittelt.

```
LedNormalOp
du:
WspLed_time++;
Set_led_mode_and_color();
Counter_evolution();
Compute_led_intensity();
WspLed_dcRed = (WspLed_color[0] * WspLed_intensity)/255;
WspLed_dcGreen = (WspLed_color[1] * WspLed_intensity)/255;
WspLed_dcBlue = (WspLed_color[2] * WspLed_intensity)/255;
```

```
function Set_led_mode_and_color
```

```
function Counter_evolution
```

```
function Compute_led_intensity
```

Softwaredokumentation

8.2.2.1 Funktion Set_led_mode_and_color

In der Funktion `Set_led_mode_and_color` wird die aktuell darzustellende LED Farbe in Abhängigkeit vom Systemzustand bestimmt. Es wird auch der Status des Ansteuermodus (konstant an/aus oder blinken) hier bestimmt.

Die LED Farbe und er Ansteuermodus wird im folgenden mit absteigender Priorität festgelegt:

Die höchste Priorität hat hierbei ein Pfc Fehler.

Prio1 Pfc Fehler

Wird ausgeführt, wenn im Pfc Fehlerstatus `WspLed_stPfcErrors` Bit 14 gesetzt ist.

LED Ansteuerungsmodus Pfc Fehler	
Farbe	Rot
Modus	Blinken
Frequenz	1 sek.

Prio2 Bap Control



Softwaredokumentation

Wird ausgeführt, wenn das BAP die LED Ansteuerung Anfordert, d.h. wenn der diesbezügliche Anforderungsstatus `WspLed_reqBapControl` auf 1 gesetzt ist. In diesem Fall fordert das BAP sowohl den Ansteuermodus mittels `WspLed_reqBapMode` als auch die Farbe mittels `WspLed_reqBapColor` an.

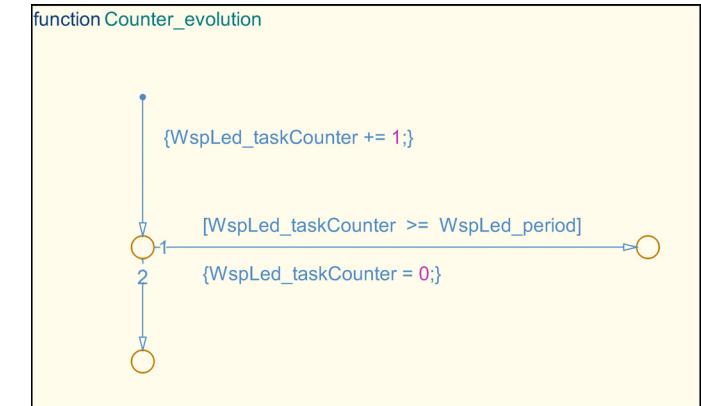
LED Ansteuerungsmodus Bap Control	
Farbe	1=Grün, 2=Gelb, 3=Ror, 4=Weiß, 5= Blau
Modus	1=Konstant, 2=Blinken,3=Pulsieren, 4=Blitzen
Frequenz	1=n.a., 2=1sek, 3=2sek., 3=2sek, 4=2sek.

Prio3 PowerTransfer oder Safety Fehler

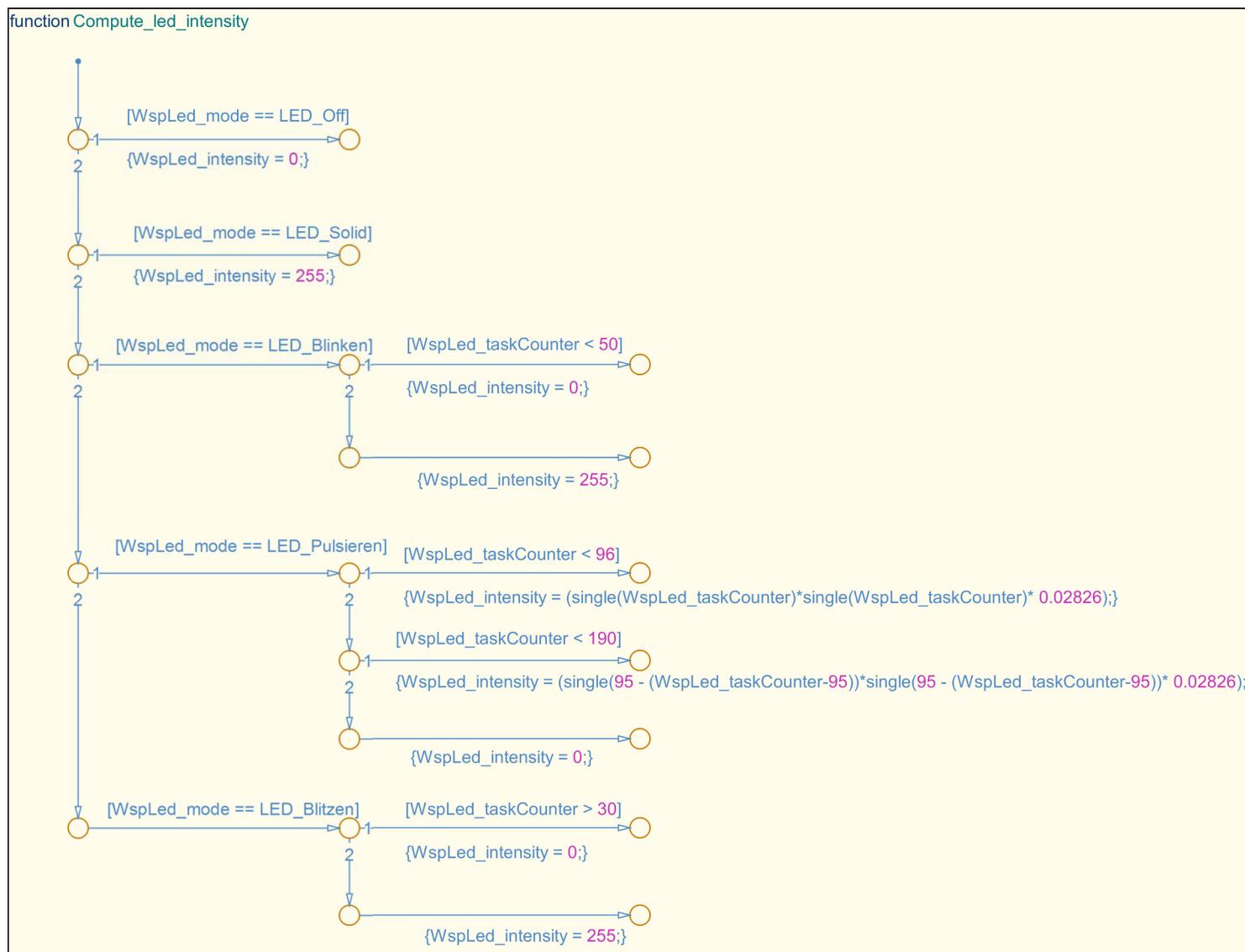


8.2.2.2 Funktion Counter_evolution

Diese Hilfsfunktion dient dazu den Taskcounter `WspLed_taskCounter`, welcher die Blink- bzw. Blitzperiodenzeit im entsprechenden LED Modus verwendet wird hochzuzählen. Die Funktion setzt den Counter zurück auf 0, wenn er die in der Funktion `Set_led_mode_and_color` festgelegte Zeit `WspLed_period` erreicht hat



8.2.2.3 Funktion Compute_led_intensity



In der Funktion *Compute_led_intensity* wird aktuelle die Gesamthelligkeit in Abhängigkeit vom LED Modus *WspLed_intensity* bestimmt. Hiermit wird das Blinken, Pulsieren und Blitzen realisiert.

8.2.3 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portinterfaces und für Operation Calls der Name der Operation Arguments.

8.2.3.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung

8.2.3.2 Ausgangsgrößen

Name	Typ	Interface / Argument	Beschreibung

8.2.3.3 Interfaces

8.2.3.3.1 Interface StatusByte

Data Element	Data Type	Auflösung	Einheit	Bedeutung
StatusByte	uint8	1	-	Status byte

8.2.4 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>WspLed_color</i>	Displayed LED color
<i>WspLed_dcBlue</i>	Duty Cycle Blue LED
<i>WspLed_dcGreen</i>	Duty Cycle Green LED
<i>WspLed_dcRed</i>	Duty Cycle Red LED
<i>WspLed_intensity</i>	Displayed Intensity
<i>WspLed_mode</i>	Displayed Mode
<i>WspLed_period</i>	Operation Period
<i>WspLed_reqBapColor</i>	BAP requested color
<i>WspLed_reqBapControl</i>	BAP control request over LED
<i>WspLed_reqBapMode</i>	BAPDisplayed Mode
<i>WspLed_reqDiagCtrFreeze</i>	Diagnostic control request over LED
<i>WspLed_reqDiagRGB</i>	DIAG color request
<i>WspLed_stPfcErrors</i>	Error field provided by PFC
<i>WspLed_stPwrTrf</i>	Power Transfer State
<i>WspLed_stSafetyErrors</i>	Error field provided by Safety
<i>WspLed_stSSM</i>	PAD system state manager
<i>WspLed_taskCounter</i>	LED Task Counter
<i>WspLed_time</i>	Led execution time

8.2.5 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>WspLed_colorBlue_C</i>	Displayed LED color Blue
<i>WspLed_colorGreen_C</i>	Displayed LED color Green
<i>WspLed_colorRed_C</i>	Displayed LED color Red
<i>WspLed_colorWhite_C</i>	Displayed LED color White
<i>WspLed_colorYellow_C</i>	Displayed LED color Yellow

8.2.6 Fehlerpfade

8.2.6.1 tbd

Fehler

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

8.2.7 Initialisierung

tbd.

9 Modul ZMover

Autor: Bogdan Marcu

Software-Struktur-Position: 30_Implementation/10_WSP/50_ZMOVCDD

9.1 Allgemein

Das Modul *ZMover* implementiert die Z-Mover Funktionalität der Wayside.

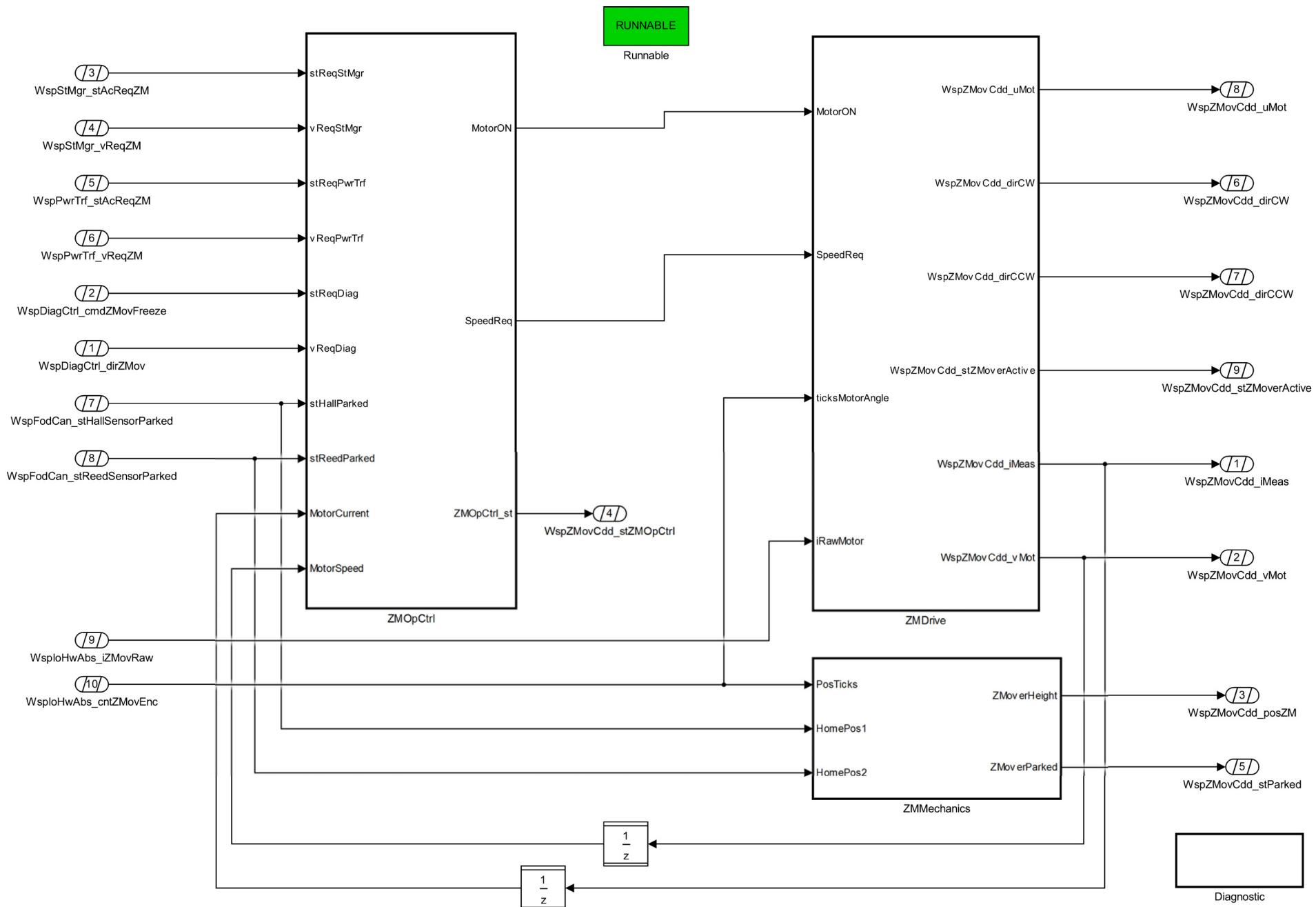
Das Modul *ZMover* enthält folgende Komponenten:

- *ZMControl* - ZMover Komponenten Treiber

9.2 Komponente ZMControl

Softwaredokumentation

9.2.1 Übersicht



Softwaredokumentation

Dies ist die oberste Ebene des Moduls **ZMover**. Hier werden im wesentlichen die Autosar IO Schnittstellen zur Verfügung gestellt. Sie werden dann an die einzelnen Submodule weiter gegeben, bzw. von ihnen ausgelesen.

Die ZMover Funktion wird im 10ms Raster ausgeführt.

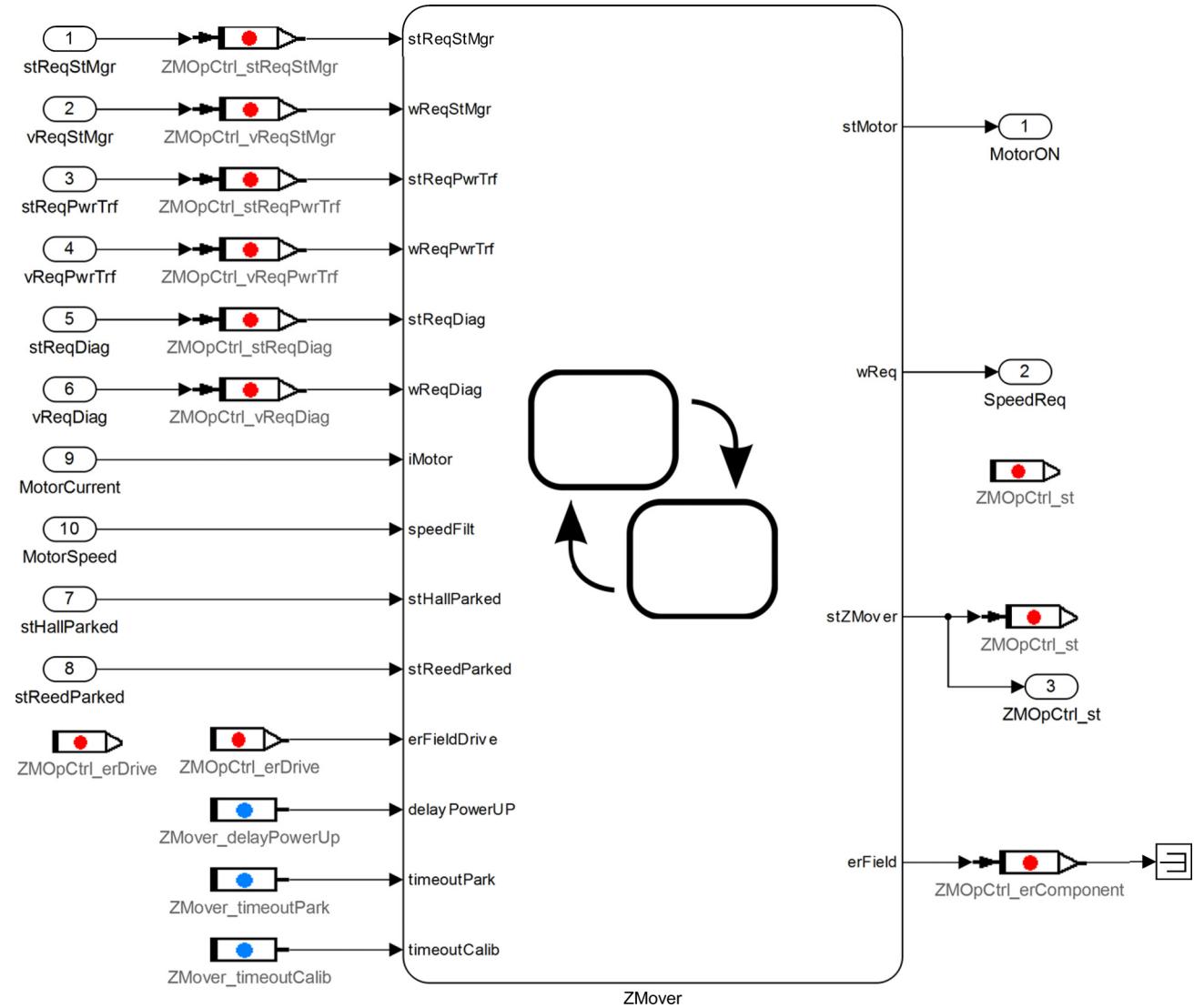
Hinweis: Die ZMover Geschwindigkeiten wie z.B. *WspStMgr_vReqZM*, *vMotor*, *wReqStMgr* (= *ZMOpCtrl_vReqStMgr*) mit negativem Vorzeichen bedeuten ZMover nach unten, mit positivem nach oben. Ebenso verhält es sich mit den ZMover Strom- und Spannungswerten.

Es sind folgende Submodule implementiert:

- | | |
|--------------------|---|
| <i>ZMOpCtrl</i> | - Bereitstellung der Sollwertvorgaben für die Ansteuerung des ZMover Motors |
| <i>ZMDrive</i> | - Ansteuerung des ZMover Motors |
| <i>ZMMechanics</i> | - Bereitstellung der ZMover Position |
| <i>Diagnostic</i> | - Diagnose ZMover |

9.2.2 Subkomponente ZMOpCtrl

Die Subkomponente **ZMOpCtrl** (ZMover Operation Control) bestimmt den aktuellen Bewegungszustand des ZMovers in Abhängigkeit der eingehenden Anforderungen. Die Komponente ist als Stateflow Chart **ZMover** implementiert.



9.2.2.1 ZMover State ZMOpCtrl_st

Die ZMover Statemachine kann hierbei folgende States, representiert in der Statevariable `ZMOpCtrl_st`, annehmen:

1	WaitForVoltage
2	Parking
3	WaitForCalib
4	Calibration - Current_Test
5	Calibration - CalibDone
6	StandBy
7	StMgr_Control
8	PwrTrf_Control
21	Diag_Control

9.2.2.2 Stateflow ZMover

Das Stateflowchart `ZMover` enthält die beiden Haupt States `Operational` und `Diag_Control`. Beim Start des Steuergerätes wird zunächst der State `Operational` aktiviert.

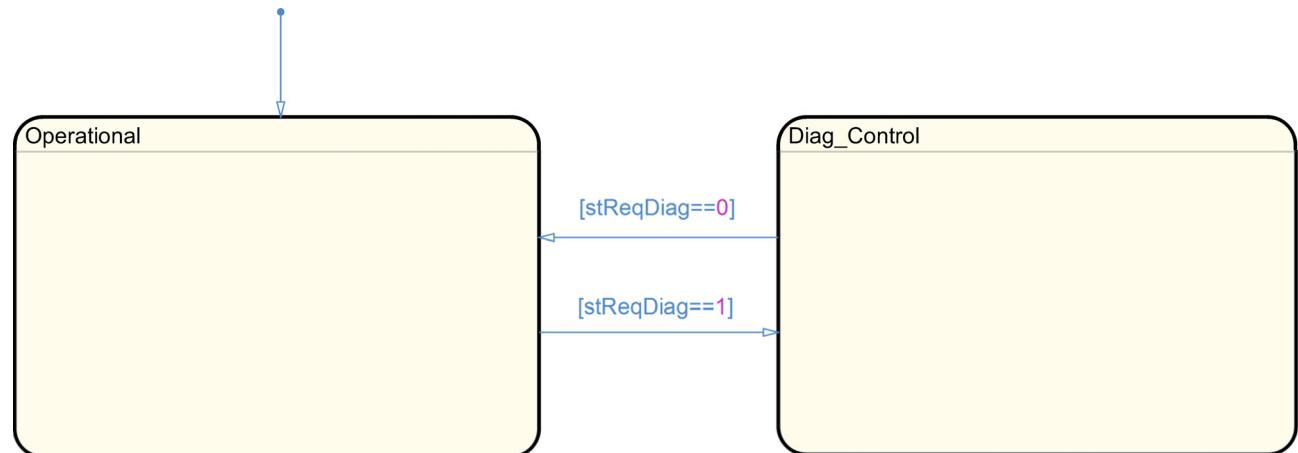
9.2.2.3 State Operational

Dies ist der initiale State der Statediagramms `ZMover`.

Im State `Operational` sind die Substates zur Realisierung der ZMover Ablaufkontrolle implementiert (s.u.).

State Transition -> Diag_Control

Der State `Operational` wechselt zum State `Diag_Control`, wenn der vom Diagnosesystem kommende Anforderungsstatus `stReqDiag` (`ZMOpCtrl_stReqDiag` = Eingansport `WspDiagCtrl_stAcReqZM`) den Wert 1 hat.



9.2.2.4 State Diag_Control

Im State *Diag_Control* ist der Substate implementiert, welcher die Anforderungen des Diagnosesystems umsetzt.

State Transition -> Operational

Der State *Diag_Control* wechselt zum State *Operational*, wenn keine Anforderung vom Diagnosesystem mehr vorliegt d.h., wenn *stReqDiag* den Wert 0 hat.

9.2.2.5 SubState Operational

Im SubState Operational ist die eigentliche StateMachine der ZMover Operation Control implementiert.

9.2.2.6 State WaitForVoltage

Dies ist der Initiale State des Substates *Operational*, welcher also nach Aktivierung der Substates *Operational* zunächst ausgeführt wird. Er dient neben der Initialisierung der States auch dazu, nach dem Start des Steuergerätes eine kurze Zeit zu warten bis sich die Ansteuerendstufe des ZMover Motors eingeschwungen hat.

Beim aktivieren dieses State wird der Status *stZMover* (*ZMOpCtrl_st*) auf 1 gesetzt, der Status *stMotor* (=Soll Aktivierungsstatus des ZMover Motors) auf 0 (Aus).

Der Wartetimer *cntrCurrentValue* zum halten dieses States wird auf *delayPowerUP* (*ZMover_delayPowerUp*, typ. 0,5sek.) gesetzt. Dieser Timer wird bei jedem Satedurchlauf (10ms) dekrementiert.

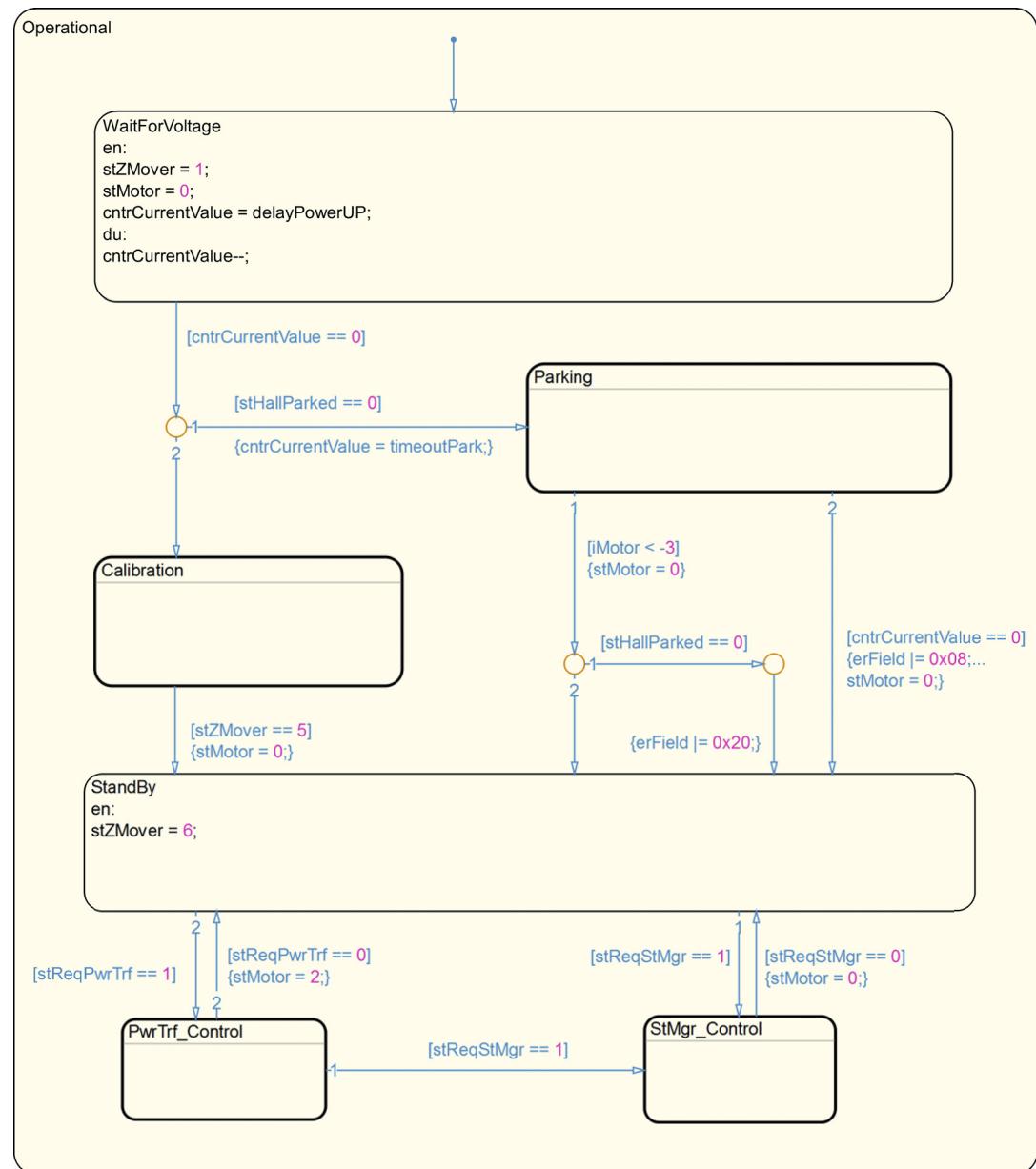
State Transition -> Parking

Der State *WaitForVoltage* wechselt zum State *Parking*, wenn der Wartezähler *cntrCurrentValue* auf 0 abgelaufen ist und der vom FOD CAN kommende Parking Status *stHallParked* 0 ist d.h. nicht den Hall Sensor der Bodenplatte ausgelöst hat.

Der globale Wartezähler *cntrCurrentValue* wird bei Ausführung dieser State Transition auf *timeoutPark* (*ZMover_timeoutPark*, typ. 15sek.) gesetzt.

State Transition -> WaitForCalib (in Calibration)

Der State *WaitForVoltage* wechselt zum State *WaitForCalib*, wenn wenn der Wartezähler *cntrCurrentValue* auf 0 abgelaufen ist und der FOD CAN kommende Parking Status *stHallParked* 1 ist, d.h. der Hall Sensor der Bodenplatte ausgelöst hat.



9.2.2.7 SubState Parking

Beim aktivieren des States Parking wird der ZMover Status *stZMover* (*ZMOpCtrl_st*) auf 2 gesetzt. Der (Soll) Aktivierungsstatus *stMotor* des ZMover - Motors wird auf 1 (An) gesetzt.

Bei jedem Durchlauf dieses States (10ms) wird die Countervariable *cntrCurrentValue* dekrementiert und die Sollgeschwindigkeit *wReq* des ZMover auf den Rückgabewert der Funktion *CalcSpeedReq* gesetzt.

State Transition -> StandBy

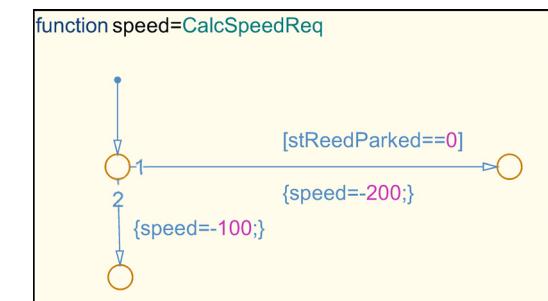
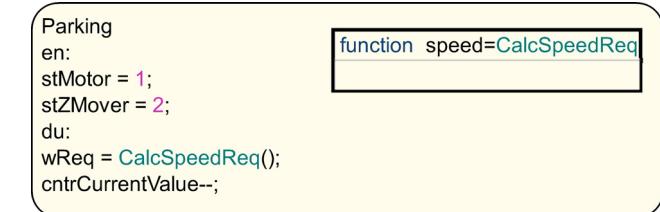
Der State Parking wechselt zum State Calibration, wenn eine der beiden folgenden Bedingungen zutrifft:

- Der Strom *iMotor* des ZMover Motors ist kleiner als -3A. Falls in diesem Fall der Status *stHallParked* nicht gesetzt ist, wird zusätzlich Bit 5 des Fehlerstatus *erField* gesetzt.
- Der Timeout Counter *cntrCurrentValue* ist abgelaufen. In diesem Fall wird Bit 3 des Fehlerstatus *erField* gesetzt.

In jedem Fall wird beim Ausführen einer der obigen Transitions der Soll Status *stMotor* des ZMover Motors auf 0 (Aus) gesetzt.

9.2.2.8 SF Subfunction CalcSpeedReq

In dieser Hilfsfunktion wird die ZMovergeschwindigkeit beim Ausführen des Parking States ermittelt. Für den Fall, daß der Status *stReedParked* vom zweiten Bodenplatten - Kontaktsensors den Wert 0 hat, also nicht dieser nicht betätigt ist, wird die Zurückgegebene Sollgeschwindigkeit *speed* auf -200 gesetzt, anderenfalls auf -100.



9.2.2.9 SubState Calibration

Der State *Calibration* dient zur erstmaligen Vorbereitung des ZMovers nach einem Neustart. Hier wird die grundsätzliche Funktionalität des ZMovers geprüft und dieser nach unten gefahren. Der State *Calibration* muß nach einem Neustart einmal durchlaufen werden, bevor der ZMover von anderen Modulen genutzt werden kann. Beim aktivieren des States *Calibration* wird zunächst der Substate *WaitForCalib* aktiviert.

9.2.2.10 SubState WaitForCalib

Beim aktivieren dieses States wird der ZMover Status *stZMover* (*ZMOpCtrl_st*) auf 3 gesetzt und der (Soll) Aktivierungsstatus *stMotor* des ZMover Motors auf 0 (*Aus*).

Die Sollgeschwindigkeit *wReq* des ZMver wird ebenfalls auf 0 gesetzt.

State Transition -> Current_Test

Der State *WaitForCalib* verzweigt zum State *Calibration*, wenn der Anforderungsstatus *stReqStMgr* (*ZMOpCtrl_stReqStMgr* = Eingangsport *WspStMgr_stAcReqZM*) den Wert 15 (*Calibration*) hat.

Beim ausführen dieser State Transition wird die globale Counter-variabole *cntrCurrentValue* auf *timeoutCalib* (typ. 500) gesetzt.

9.2.2.11 SubState Current_Test

Beim aktivieren dieses States wird der ZMover Status *stZMover* (*ZMOpCtrl_st*) auf 4 gesetzt und der (Soll) Aktivierungsstatus *stMotor* des ZMover Motors auf 1 (*An*). Die Sollgeschwindigkeit *wReq* wird auf -300 gesetzt.

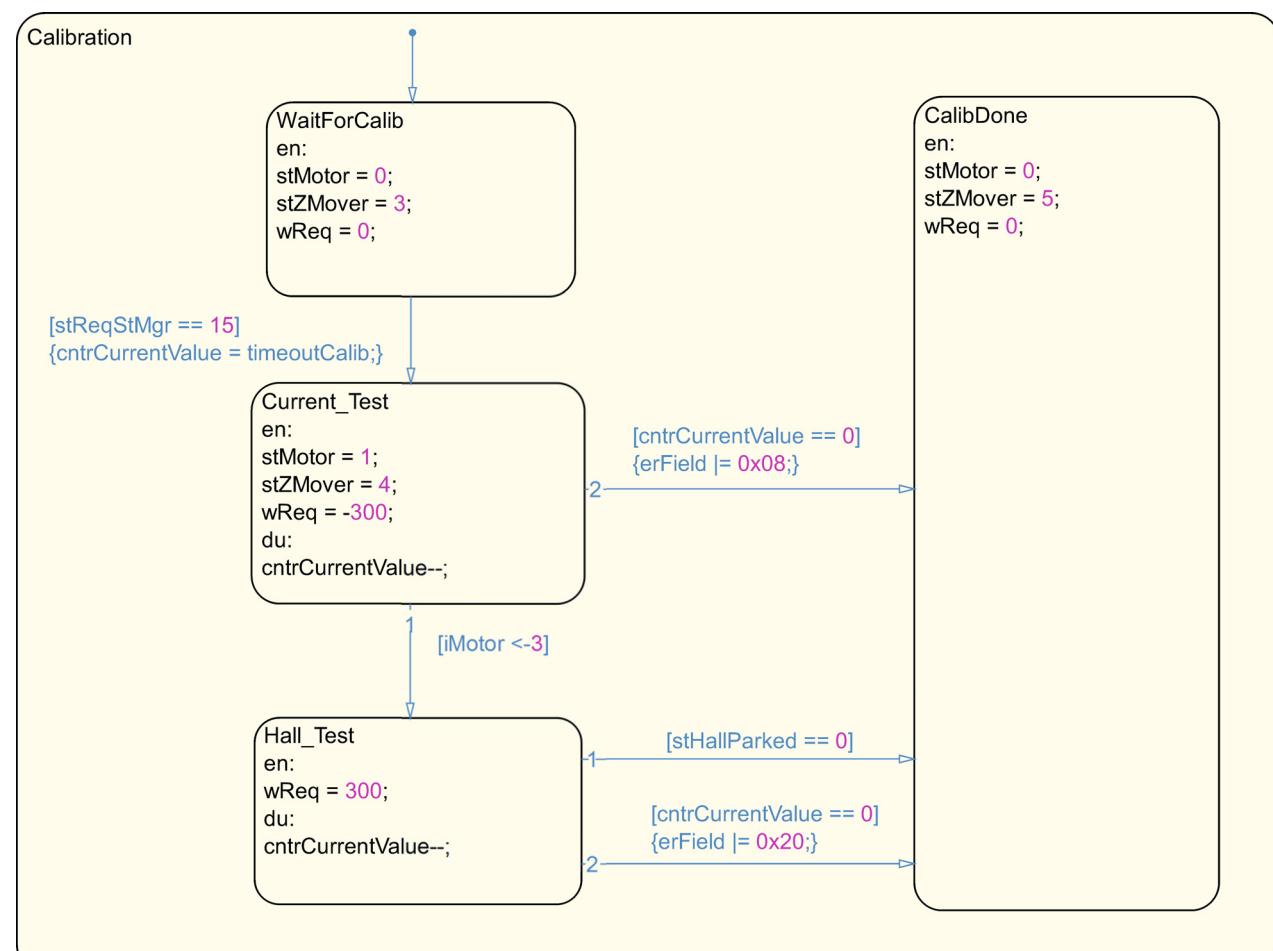
Bei jedem Statedurchlauf wird der Timeout Counter *cntrCurrentValue* dekrementiert.

State Transition -> Hall_Test

Der State *Current_Test* wechselt zum State *Hall_Test*, wenn der ZMoverstrom *iMotor* -3A unterschreitet.

State Transition -> CalibDone

Der State *Current_Test* wechselt zum State *CalibDone*, wenn der Timeout Counter *cntrCurrentValue* abgelaufen ist. In diesem Fall wird noch Bit 3 des Fehlerstatus *errField* gesetzt.



9.2.2.12 SubState Hall_Test

Beim aktivieren des States *Hall_Test* wird der ZMover wieder nach oben gefahren, indem dessen Sollgeschwindigkeit *wReq* auf 300 gesetzt wird. Bei jedem Durchlauf dieses States wird der Timeout Counter *cntrCurrentValue* weiter inkrementiert.

State Transition -> CalibDone

Der State *Current_Test* wechselt zum State *CalibDone*, wenn der Status *stHallParked* des Hall Sensors nicht (mehr) gesetzt ist.

Der State *Current_Test* wechselt ebenfalls zum State *CalibDone*, wenn der Timeout Counter *cntrCurrentValue* abgelaufen ist. In diesem Fall wird noch Bit 5 des Fehlerstatus *errField* gesetzt.

9.2.2.13 SubState CalibDone

Beim aktivieren des States *CalibDone* wird Aktivierungsstatus *stMotor* des ZMover Motors auf 0 (*Aus*) und der Status *stZMover* ([ZMOpCtrl_st](#)) auf 5 gesetzt .

Die Sollgeschwindigkeit *wReq* des ZMver wird auf 0 gesetzt.

State Transition Calibration -> StandBy

Der State *Calibration* wechselt zum State *Standby*, wenn der Status *stZMover* den Wert 5 (*CalibDone*) hat.

9.2.2.14 State StandBy

Beim aktivieren des States *StandBy* wird der ZMover Status *stZMover* ([ZMOpCtrl_st](#)) auf 6 gesetzt.

State Transition -> StMgr_Control

Der State *StandBy* wechselt zum State *StMgr_Control*, wenn der Anforderungsstate *stReqStMgr* des Wayside System State Managers den Wert 1 hat (Prio 1).

State Transition -> PwrTrf_Control

Der State *StandBy* wechselt zum State *PwrTrf_Control*, wenn der Anforderungsstate *stReqPwrTrf* des Wayside Power Transfer Managers den Wert 1 hat (Prio 2).

9.2.2.15 SubState PwrTrf_Control

Beim aktivieren des State *PwrTrf_Control* wird der ZMover Status *stZMover* ([ZMOpCtrl_st](#)) auf 8 gesetzt. Der (Soll) Aktivierungsstatus *stMotor* des ZMover Motors wird auf 1 (*An*) gesetzt.

Bei jedem Durchlauf diese States wird die Soll- ZMovergeschwindigkeit *wReq* mit der aktuell vom Power Transfermodul angeforderten ZMover Geschwindigkeit *wReqPwrTrf* ([ZMOpCtrl_vReqPwrTrf](#)) aktualisiert.

PwrTrf_Control
en:
stZMover = 8;
stMotor = 1;
du:
wReq = wReqPwrTrf;

State Transition -> StandBy

Der State *PwrTrf_Control* wechselt zurück zum State *StandBy*, wenn der Anforderungsstatus *stReqPwrTrf* vom PowerTransfer Modul zurückgenommen wird.

State Transition -> StMgr_Control

Der State *PwrTrf_Control* wechselt zum State *StMgr_Control*, wenn der Anforderungsstatus *stReqStMgr* vom Wayside System Statemangere gesetzt ist, da dieser die höchste Priorität hat.

9.2.2.16 SubState StMgr_Control

Beim aktivieren des State *StMgr_Control* wird der ZMover Status *stZMover* (*ZMOpCtrl_st*) auf 7 gesetzt. Der (Soll) Aktivierungsstatus *stMotor* des ZMover Motors wird auf 1 (An) gesetzt.

Bei jedem Durchlauf diese States wird die Soll- ZMovergeschwindigkeit *wReq* mit der aktuell vom Wayside System Statemanager angeforderten ZMover Geschwindigkeit *wReqStMgr* (*ZMOpCtrl_vReqStMgr*) aktualisiert.

```
StMgr_Control
en:
stZMover = 7;
stMotor = 1;
du:
wReq = wReqStMgr;
```

State Transition -> StandBy

Der State *StMgr_Control* wechselt zurück zum State *StandBy*, wenn der Anforderungsstatus *stReqStMgr* vom Wayside System Statemanager zurückgenommen wird.

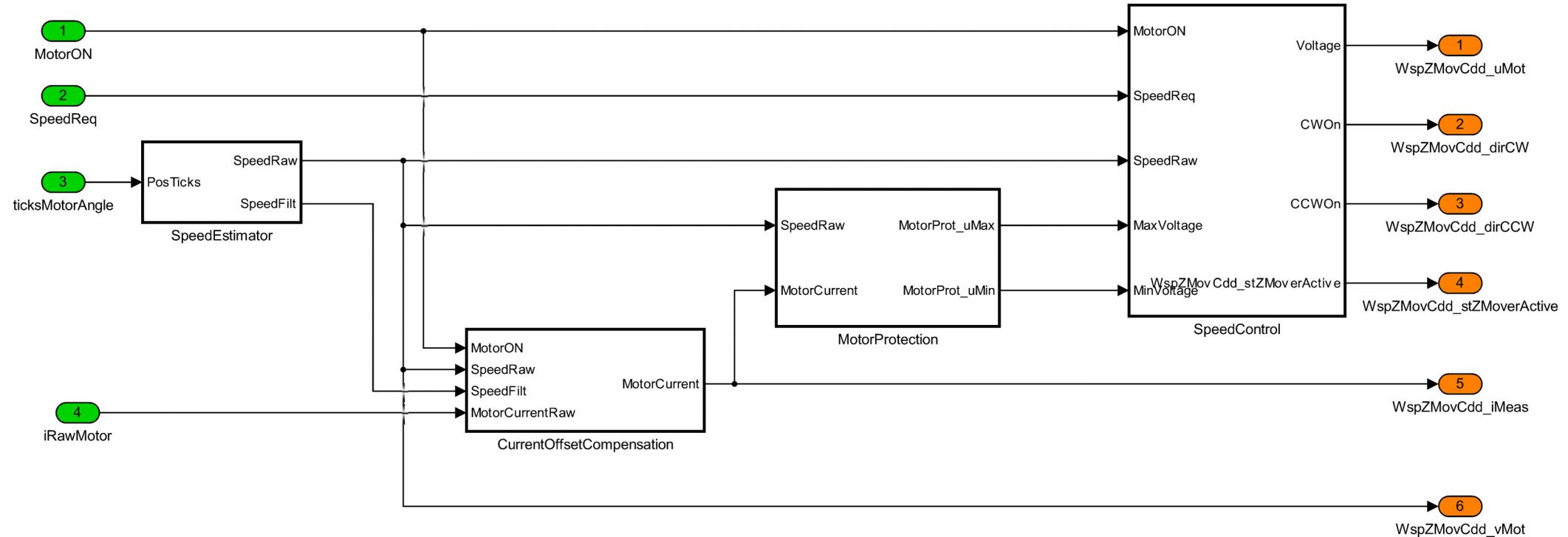
9.2.2.17 State Diag_Control

```
Diag_Control
en:
stMotor = 1;
stZMover = 21;
du:
wReq = 300*(wReqDiag-1);
```

Beim aktivieren des State *Diag_Control* wird der ZMover Status *stZMover* (*ZMOpCtrl_st*) auf 21 gesetzt. Der (Soll) Aktivierungsstatus *stMotor* des ZMover Motors wird auf 1 (An) gesetzt.

Bei jedem Durchlauf diese States wird die Soll- ZMovergeschwindigkeit *wReq* mit der aktuell vom Diagnosesystem angeforderten ZMover Geschwindigkeit *wReqDiag* (*ZMOpCtrl_vReqDiag*) aktualisiert, wobei die angeforderte Diagnosegeschwindigkeit in die Einheit des ZMover Moduls umgerechnet werden muss.

9.2.3 Subkomponente ZMDrive



In der Komponente ZMDrive wird die angerforderte ZMover- Motorgeschwindigkeit über die Ansteuerspannung *WspZMovCdd_uMot* (*Voltage*) des ZMovers geregelt.

Hierzu wird zunächst im Block *SpeedEstimator* mittels des Zählerwerts *ticksMotorAngle* des Quadratursensors die aktuelle ZMovergeschwindigkeit ermittelt.

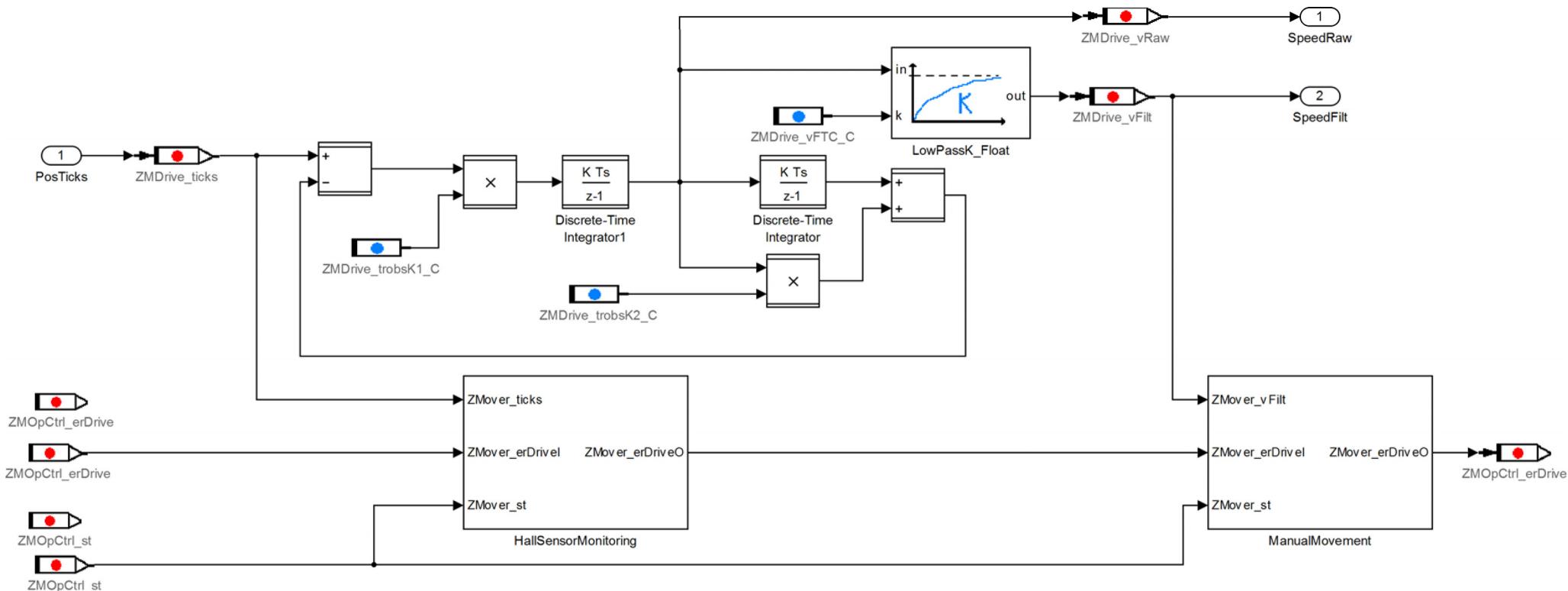
Im Block *CurrentOffsetCompensation* wird der gemessene ZMover Strom gefiltert.

Im Block *MotorProtection* wird zum Schutz des ZMover die Ansteuerspannung in Abhängigkeit von der Motortemperatur und Last begrenzt.

Im Block *SpeedControl* wird die Geschwindigkeitsregelung des ZMovers durchgeführt.

Softwaredokumentation

9.2.3.1 Block SpeedEstimator

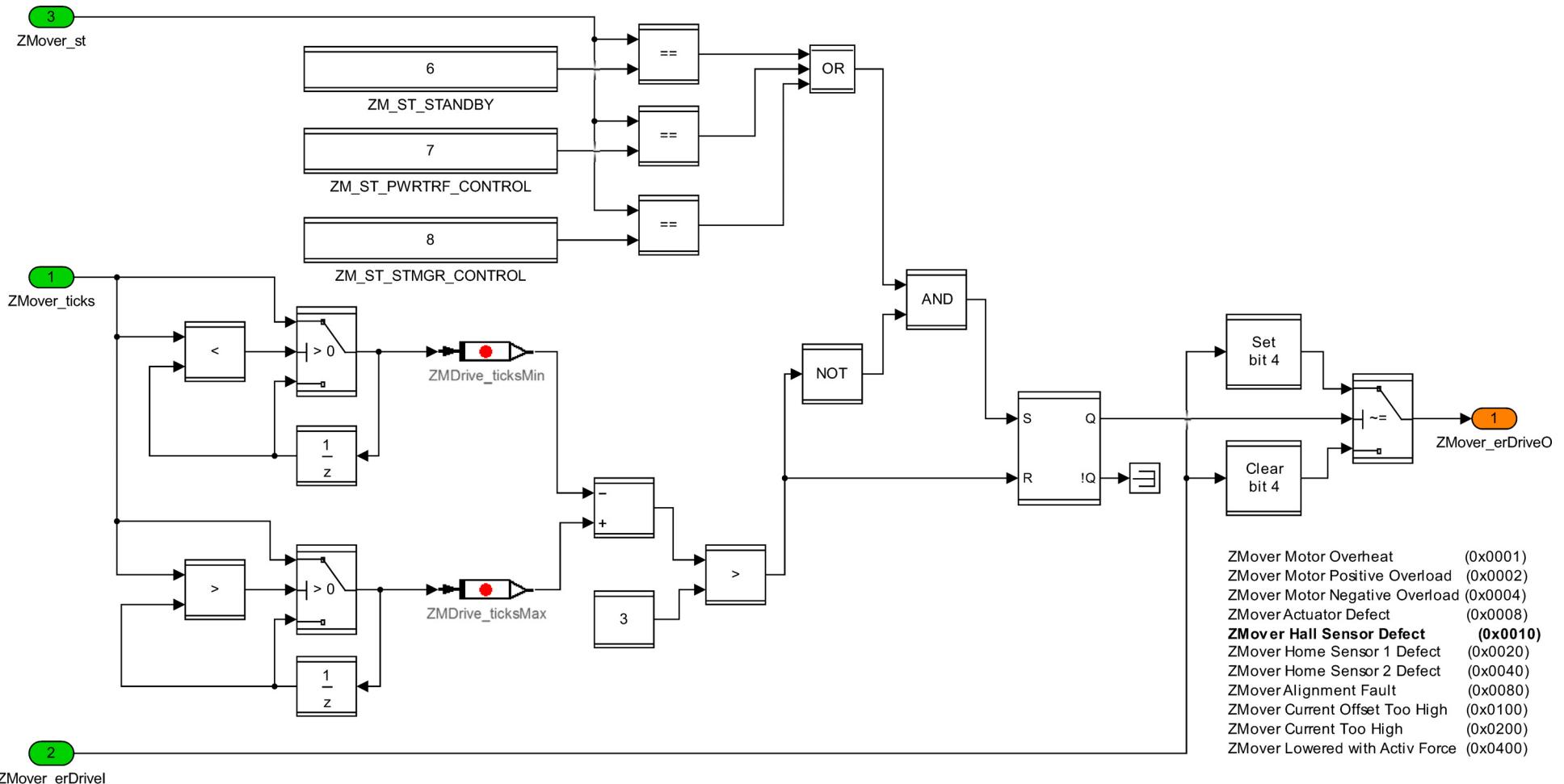


Hier wird zunächst aus der gefilterten zeitlichen Differenz des Zählerstandes [ZMDrive_ticks](#) des Quadraturdekoders des ZMover Motors die Winkelgeschwindigkeit [ZMDrive_vRaw](#) sowie die gefilterte Winkelgeschwindigkeit [ZMDrive_vFilt](#) ermittelt.

Im Block [HallSensorMonitoring](#) wird dann der Hall Sensor für die Positionen und Winkelgeschwindigkeitserfassung des ZMovers plausibilisiert.

Im Block [ManualMovement](#) wird überprüft, ob der ZMover unter Einwirkung einer äußeren Kraft bewegt wird.

9.2.3.2 Block HallSensorMonitoring



Im Block *HallSensorMonitoring* wird der Hallsensor des ZMovers plausibilisiert. Hierbei wird davon ausgegangen, daß der ZMover während der Kalibrierung mindestens eine minimale Bewegung durchführt, welche mittels des Hallsensors erfasst wird und so der Zählerstand *ZMover_ticks* der ZMover Winkelpositionen entsprechend verändert.

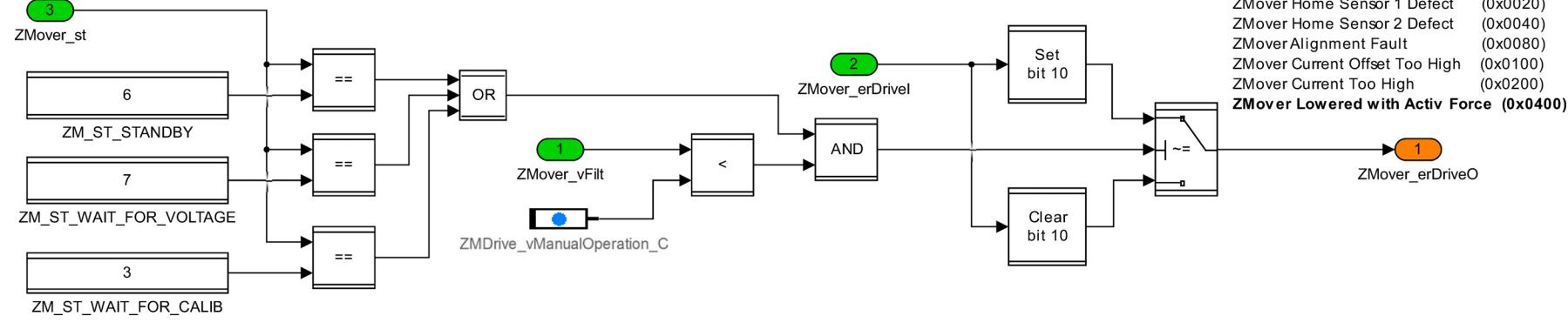
Hierzu wird der aktuelle Zählerstand *ZMover_ticks* auf laufend Veränderung überprüft. Bei einer negativen Veränderung wird der danach aktuelle Stand in der Variablen *ZMDrive_ticksMin* vermerkt. Bei einer positiven Veränderung in der Variablen *ZMDrive_ticksMax*.

Ist nun der aktuelle Status *ZMover_st* des ZMovers größer oder gleich 6 (Standby), 7 (PwrTrfCtrl) oder 8 (StMgrCtrl) , wurde also die Kalibrierung durchgeführt, liegt dann ein Hallsensorfehler vor, wenn die Veränderung der Hallsensor Ticks (*ZMDrive_ticksMin* - *ZMDrive_ticksMax*) nicht größer ist als 3. In diesem Fall wird das Bit 4 des Fehlerstatus *ZMOpCtrl_erDrive* gesetzt, anderenfalls gelöscht.

Softwaredokumentation

9.2.3.3 Block ManualMovement

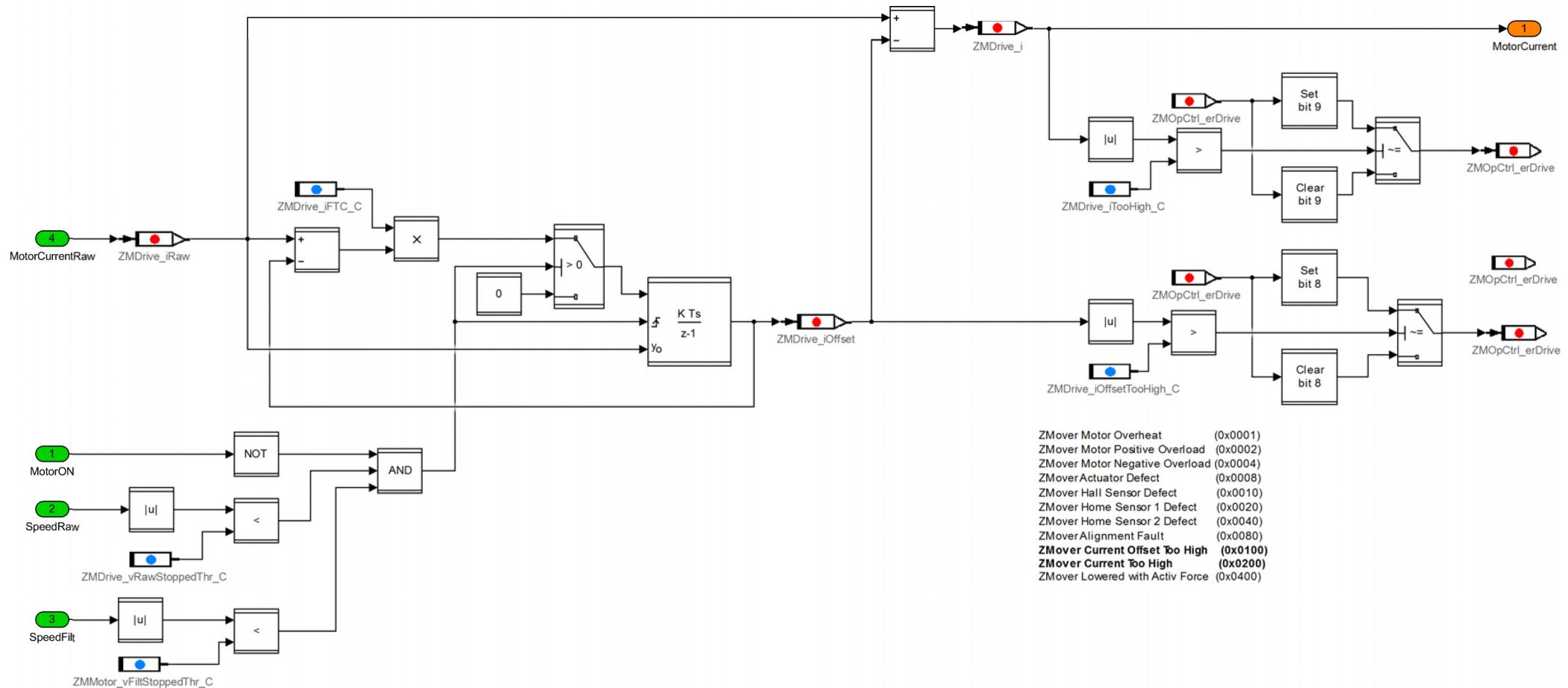
Hier wird geprüft, ob der ZMover durch äußere Krafteinwirkung nach unten gedrückt wird. Die Überprüfung wird nur im ZMover State `ZMover_st = 6` (Standby), `7` (`PwrTrfCtrl`) oder `3` (`WaitForCalib`) durchgeführt, in welchem der ZMover seitens der Ansteuerung bewegungslos sein müsste.



Ist hier die Winkelgeschwindigkeit `ZMover_vFilt` dann kleiner als `ZMDrive_vManualOperation_C`, kann dies nur durch äußere Krafteinwirkung hervorgerufen worden sein. In diesem Fall wird das Bit 10 des Fehlerstatus `ZMOpCtrl_erDrive` gesetzt.

ZMover Motor Overheat	(0x0001)
ZMover Motor Positive Overload	(0x0002)
ZMover Motor Negative Overload	(0x0004)
ZMover Actuator Defect	(0x0008)
ZMover Hall Sensor Defect	(0x0010)
ZMover Home Sensor 1 Defect	(0x0020)
ZMover Home Sensor 2 Defect	(0x0040)
ZMover Alignment Fault	(0x0080)
ZMover Current Offset Too High	(0x0100)
ZMover Current Too High	(0x0200)
ZMover Lowered with Activ Force	(0x0400)

9.2.3.4 Block CurrentOffsetCompensation



Hier wird der Offset der ZMover Strommessung bestimmt, welcher durch Toleranzen der Meßschaltung hervorgerufen wird.

Für den Fall, daß der Motor nicht angesteuer wird, d.h. der Status MotorON = false und die Winkelgeschwindigkeit SpeedRaw bzw. SpeedFilt ist kleiner als [ZMDrive_vRawStoppedThr_C](#) bzw. [ZMMotor_vFiltStoppedThr_C](#), wird der aktuell gemessene ZMover Strom [ZMDrive_iRaw](#) mit dem Faktor [ZMDrive_iFTC_C](#) multipliziert und von dem vorhergehenden Ausgangswert des nachfolgenden Integrators subtrahiert. Der Integrator dient hierbei als Akkumulator. Die gesamte Schaltung weist hierbei also eine PT1 Eigenschaft bezüglich der Stromabweichung von 0A auf. Für den Fall das der Motor angesteuert wird, wird der letzte Stand des Integrators (Akkumulators) eingefroren, indem der Eingang auf 0 gesetzt wird. Der Ausgang des Integrators ist der so erhaltene Stromoffset [ZMDrive_iOffset](#).

Dieser Offset wird dann vom gemessenen Strom-Rohwert [ZMDrive_iRaw](#) subtrahiert und ergibt somit den Offsetkompensierten Z-Moverstrom [ZMDrive_i](#).

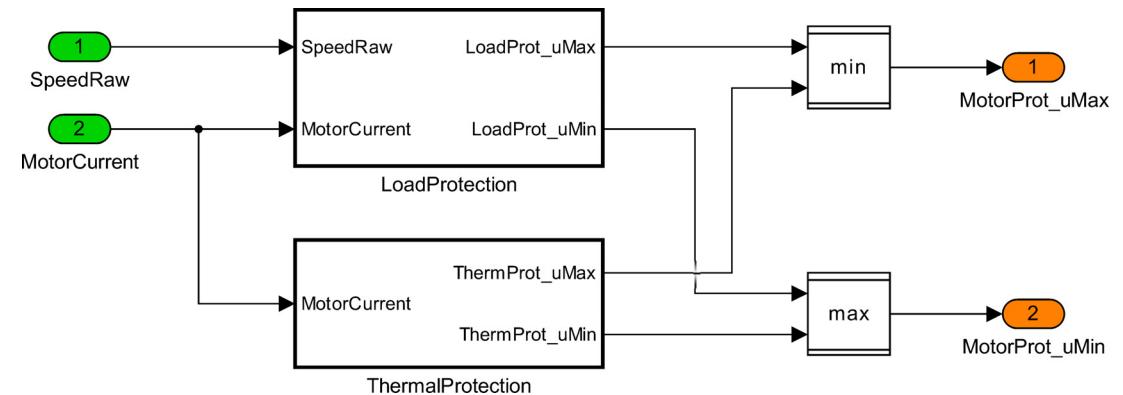
Der so erhaltene Z-Moverstrom [ZMDrive_i](#) sowie der Stromoffset [ZMDrive_iOffset](#) wird nun noch auf die Einhaltung zulässiger Grenzen überprüft.

Übersteigt der Betrag des ZMover Stroms die Schwelle [ZMDrive_iTooHigh_C](#), wird das Bit 9 des Fehlerstatus [ZMOpCtrl_erDrive](#) gesetzt. Übersteigt der Betrag des Stromoffsets die Schwelle [ZMDrive_iOffsetTooHigh_C](#), wird Bit 8 des Fehlerstatus gesetzt.

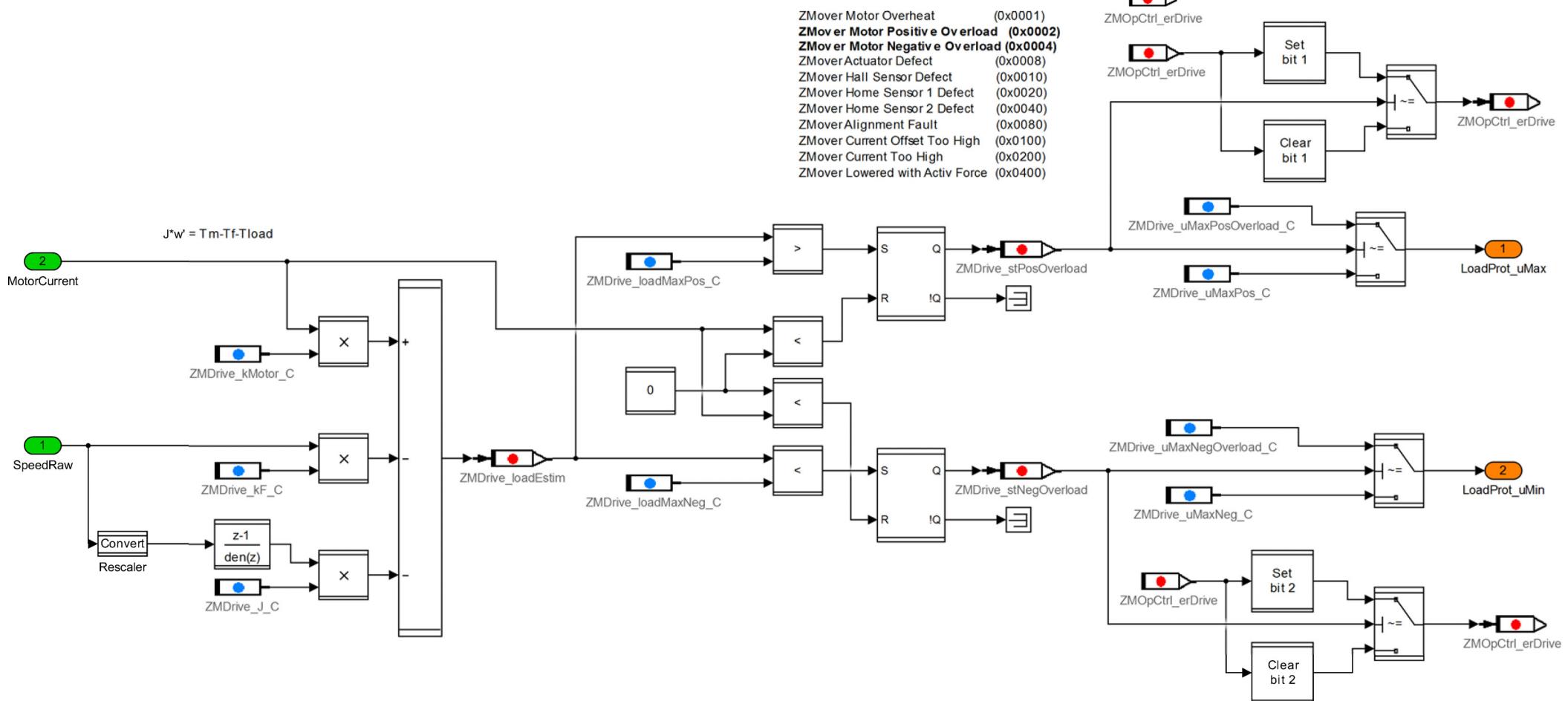
9.2.3.5 Block MotorProtection

Hier wird zum Schutze des Z-Movermotors die Ansteuerspannung des Zmovers in Abhängigkeit von der Motorlast sowie der erwarteten Motortemperatur begrenzt.

Im Block *LoadProtection* werden die Maximal- sowie Minimalspannungsgrenzen *LoadProt_uMax* bzw. *LoadProt_uMin* bezüglich der aktuellen Motorlast bestimmt. Im Block *ThermalProtection* die Min/Max-Grenzen *ThremProt_uMax* sowie *ThremProt_uMin* bezüglich der zu erwartenden Motortemperatur. Der kleinere Werte der beiden Maximalspannungen ergibt dann die obere zulässige Ansteuerspannung *MotorProt_uMax* und der größere der Minimalspannungen die untere zulässige Ansteuerspannung *MototProt_uMin* des Z-Mover Motors.



9.2.3.6 Block LoadProtection



Hier wird der Überlastschutz des ZMovers bezüglich des Drehmoments durchgeführt. Hierzu wird zunächst das Primäre Drehmoment aus dem Ansteuerstrom *MotorCurrent* des ZMovers als Produkt aus diesem mit der Motorkonstanten *ZMDrive_kMotor_C* berechnet. Hiervon abgezogen wird das Reibmoment welches sich als Produkt der Motorgeschwindigkeit *SpeedRaw* mit der Konstanten *ZMDrive_kF_C* ergibt sowie das Trägheitsmoment, weches sich als Produkt aus der Beschleunigung des Motors (aus Ableitung der Winkelgeschwindigkeit *SpeedRaw*) mit dem Massenträgheitsmoment *ZMDrive_J_C* errechnet.

Das so erhaltene Effektivmoment *ZMDrive_loadEstim* wird dann für die positive Drehrichtung mit dem Schwellwert *ZMDrive_loadMaxPos_C* bzw. bei negativer Drehrichtung mit *ZMDrive_loadMaxNeg_C* verglichen.

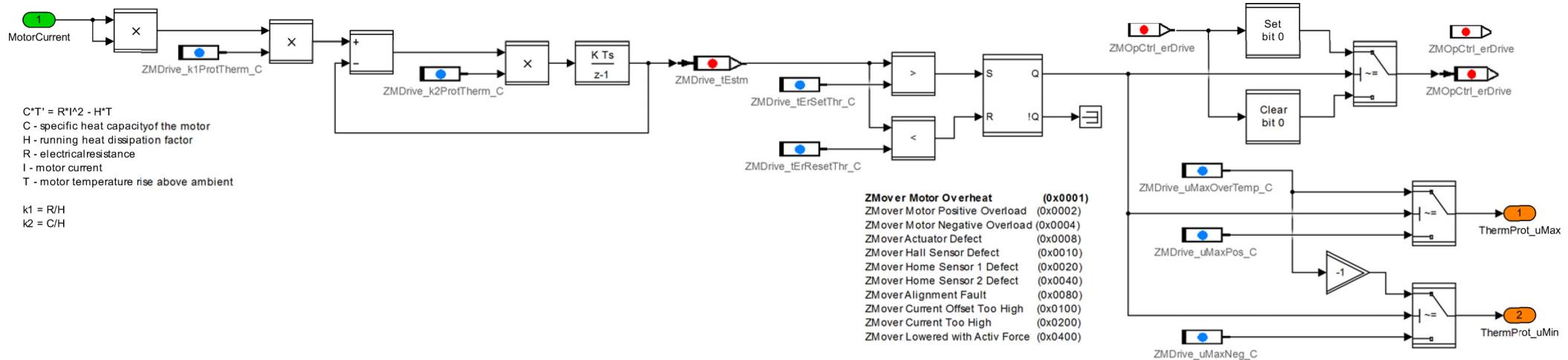
Bei Überschreitung der entsprechenden Schwellen in positiver Drehrichtung wird der Fehlerstatus *ZMDrive_stPosOverload* gesetzt; entsprechend in negativer Drehrichtung *ZMDrive_stNegOverload*. Diese Fehlerstati werden nur bei Drehrichtungswechsel wieder zurückgesetzt.

Softwaredokumentation

primove
true e-mobility

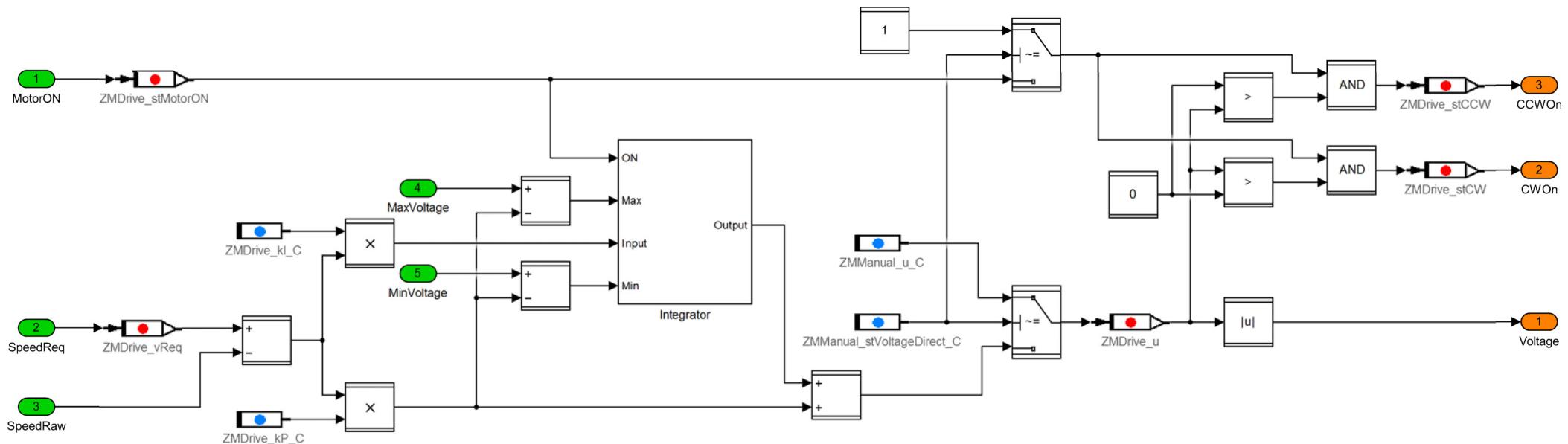
Ist der Fehlerstatus [ZMDrive_stPosOverload](#) gesetzt, wird die maximale Ansteuerspannung [LoadProt_uMax](#) auf [ZMDrive_uMaxPosOverload_C](#) begrenzt und Bit 1 des globalen Fehlerstatus [ZMOpCtrl_erDrive](#) gesetzt. Ist der Fehlerstatus [ZMDrive_stNegOverload](#) gesetzt, wird die minimale Ansteuerspannung [LoadProt_uMin](#) auf [ZMDrive_uMaxNegOverload_C](#) begrenzt und Bit 2 des globalen Fehlerstatus [ZMOpCtrl_erDrive](#) gesetzt.

9.2.3.7 Block ThermalProtection



tbd

9.2.3.8 Block SpeedControl



Im Block *SpeedControl* wird die Geschwindigkeitsregelung des Z-Mover Motors durchgeführt. Eine Ansteuerung der Ausgangsendstufen findet hier nur statt, wenn der Status [ZMDrive_stMotorON](#) den Wert *true* hat.

Die Drehrichtung des Z-Mover Motors wird hierbei durch die Status [ZMDrive_stCW](#) (Clockweise = im Uhrzeigersinn) bzw. [ZMDrive_stCCW](#) (CounterClockwise = im Gegen-uhzeigersinn) bestimmt. Diese Stati bestimmen die Polung der Ausgangsendstufe des ZMovers. Sind beide Stati = *false* (0) ist die Ausgangsendstufe inaktiv.

Für den Fall, das der manuelle Ansteuermodus aktiv ist ([ZMManual_stVoltageDirect_C](#) = *true*), wird der Sollstatus [ZMDrive_stMotorON](#) ignoriert und die Drehrichtungsstati entsprechend des Sollspannung [ZMDrive_u](#), welche im manuellen Modus dem Parameter [ZMManual_u_C](#) entspricht, gesetzt.

Im normalen Regelbetrieb, d.h. [ZMDrive_stMotorON](#) = *true* und [ZMManual_stVoltageDirect_C](#) = *false* ist die Regelung mittels PI - Regler implementiert. Der P-Anteil ergibt sich aus der Regelabweichung [ZMDrive_vReq](#) - SpeedRaw (Soll minus Ist-Geschwindigkeit) multipliziert mit dem Faktor [ZMDrive_kP_C](#). Der I-Anteil wird im Block Integrator berechnet. Als Eingangsgröße dient hier die Regelabweichung multipliziert mit dem (Gain) Faktor [ZMDrive_kI_C](#).

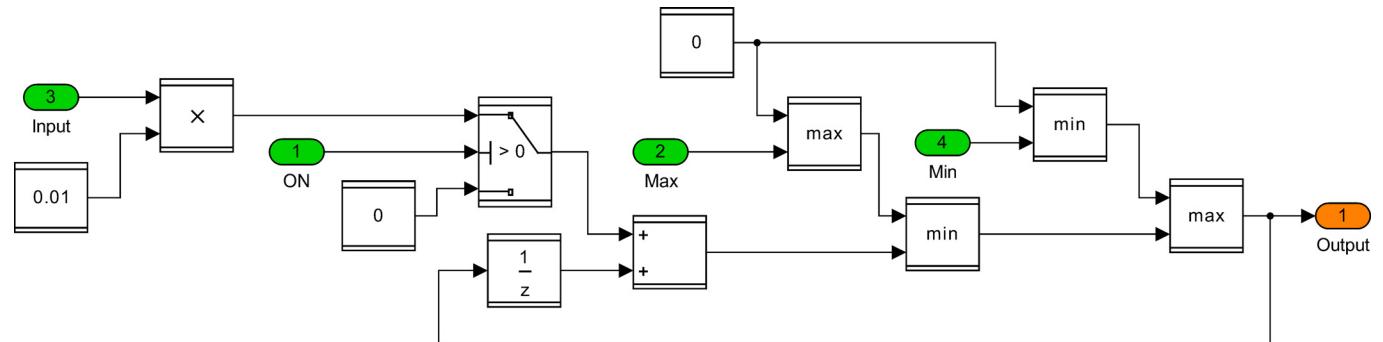
Die resultierende Ansteuerspannung für den ZMover ist dann die Summe aus P- ind I-Anteil. Aus dem Vorzeichen dieser Spannung werden dann die Drehrichtungsstatis bestimmt. Für positive Spannungswerte wird die Drehrichtung im Uhrzeigersinn gesetzt, d.h. [ZMDrive_stCW](#) = *true* und [ZMDrive_stCCW](#) = *false*, für negative Spannungswerte im Gegenuhrzeigersinn, d.h. [ZMDrive_stCW](#) = *false* und [ZMDrive_stCCW](#) = *true*.

Der Ausgabewert *Voltage* der Absteuerspannung ist dann der Betrag vom [ZMDrive_u](#).

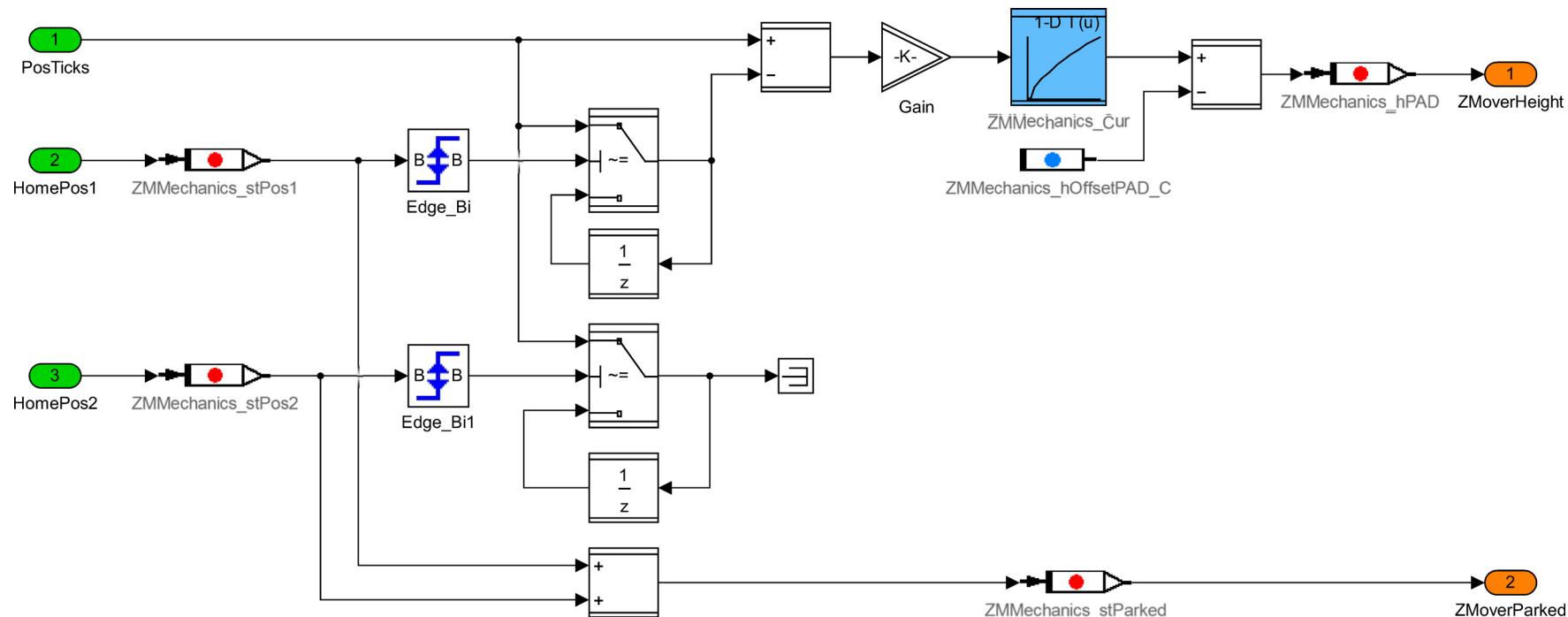
9.2.3.9 Block Integrator

Hier ist der Integrator für den I-Anteil implementiert. Der Eingangswert *Input* wird mit dem Zeitschritt 0,01sek skaliert.

Für den Fall, daß der Eingangsstatus *ON* = *false* ist, wird der aktuelle Integratorstand eingefroren, ist *ON* = *true*, wird bei jedem Zeitschritt der skalierte Eingangswert zum Akkumulator $1/Z$ addiert. Die Summe wird nach oben hin durch den Eingangswert *Max* und nach unten hin durch den Eingangswert *Min* begrenzt.



9.2.4 Subkomponente ZMMechanics



Im Block **ZMMechanics** wird die Höhe des ZMovers über der Bodenplatte bestimmt.

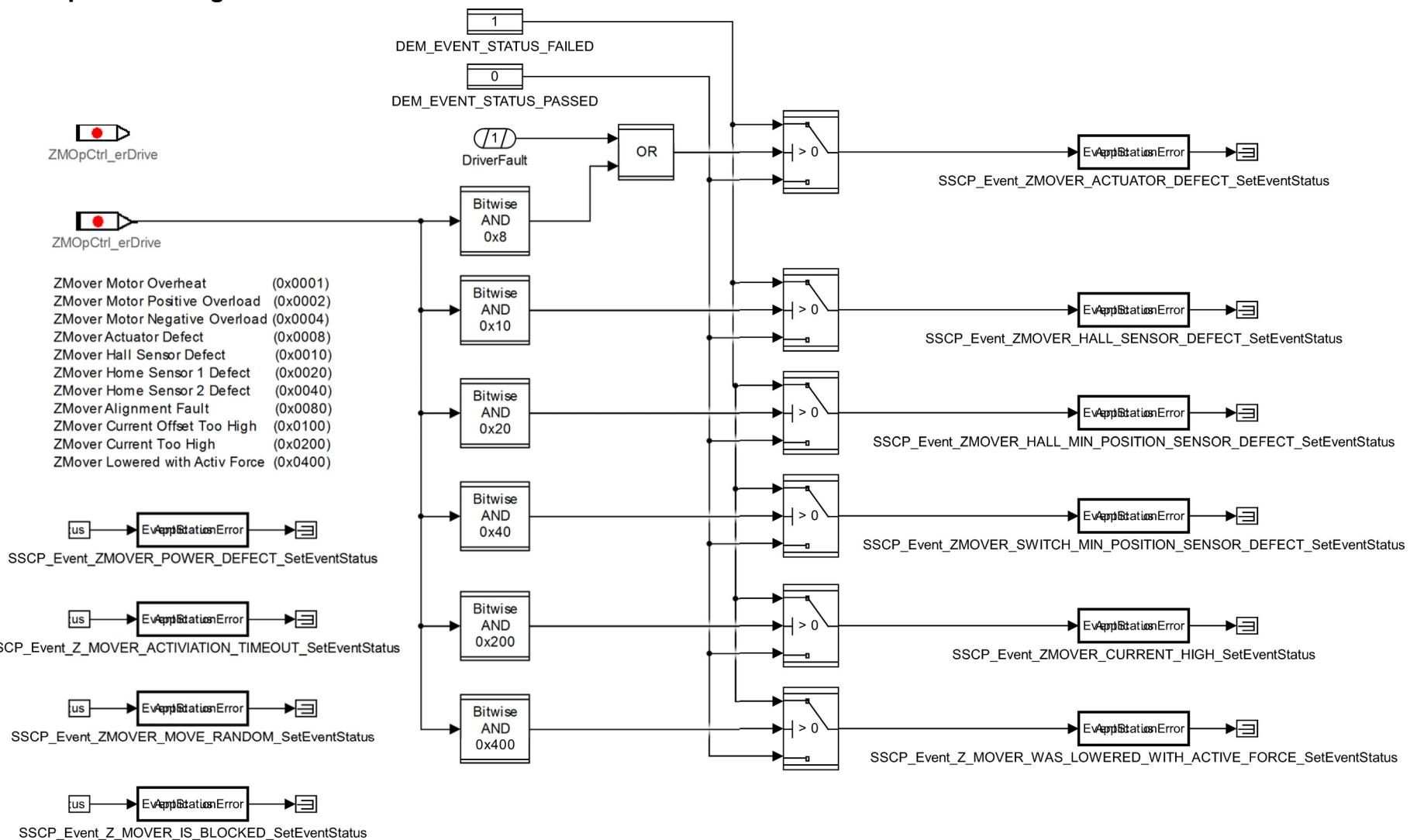
Zunächst wird der Zustand der beiden Bodenplatten Kontaktssensoren **ZMMechanics_stPos1** und **ZMMechanics_stPos2** ausgewertet. Die beiden Kontaktstati werden jeweils summiert und ergeben so den Park Status **ZMMechanics_stParked**. Detektiert keiner der beiden Sensoren den ZMover Bodenkontakt hat demzufolge der Status **ZMMechanics_stParked** den Wert 0 und meldet beide Sensoren Kontakt den Wert 2.

Bei jedem Statuswechsel von Bodenkontaktelement 1 wird die aktuelle Motorwinkelposition **PosTicks** in einem Akkumulator ($1/Z$) gesichert. Die so gesicherte Winkelposition wird immer von der aktuell gemessenen Winkelposition **PosTicks** subtrahiert, so das bei Bodenkontakt ein auf 0 normierter Winkelwert vorliegt.

Der normierte Winkelwert wird dann mittels der Kennlinie **ZMMechanics_Cur** in eine Höhe **ZMMechanics_hPad** des Z-Movers über der Bodenplatte umgerechnet. Diese Höhe kann bei Bedarf noch mittels der Konstanten **ZMMechanics_hOffsetPAD_C** justiert werden.

Softwaredokumentation

9.2.5 Subkomponente Diagnostic



tbd

9.2.6 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Requiered Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

9.2.6.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
WspConf_posZM	RPort	ZMPos	
WspDiagCtrl_cmdZMovFreeze	RPort	DiagCtrlFreeze	Anforderungsstatus ZMover Steuerung von Diagnose
WspDiagCtrl_dirZMov	RPort	DiagCtrlZMovDir	Anforderung ZMover Geschwindigkeit von Diagnose
WspFodCan_stHallSensorParked	RPort	BoolVal	Kontakt-Status Bodenplatten Hall Sensor
WspFodCan_stReedSensorParked	RPort	BoolVal	Kontakt-Status Bodenplatten Reed Sensor
WspoHwAbs_cntZMovEnc	RPort	Cntr	Motorposition in Ticks
WspoHwAbs_iZMovRaw	RPort	Current_mA	gemesserner ZMover Strom aus HW Abstraction
WspoHwAbs_stZMovFault	RPort	StatusByte	Fehler Codes ZMover
WspPwrTrf_stAcReqZM	RPort	StatusByte	Anforderungsstatus ZMover Steuerung von Wsp PowerTransfer
WspPwrTrf_vReqZM	RPort	ZMoverMotSpeedVal	Anforderung ZMover Geschwindigkeit von Wsp PowerTransfer
WspStMgr_stAcReqZM	RPort	StatusByte	Anforderungsstatus ZMover Steuerung von Wsp State Manager
WspStMgr_vReqZM	RPort	ZMoverMotSpeedVal	Anforderung ZMover Geschwindigkeit von Wsp State Manager

9.2.6.2 Ausgangsgrößen

Name	Typ	Interface / Argument	Beschreibung
WspZMovCdd_iMeas	PPort		
WspZMovCdd_vMot	PPort		
WspConf_posZM	PPort		
WspZMovCdd_stZMOpCtrl	PPort		
WspZMovCdd_stParked	PPort		
WspZMovCdd_stZMCompErr	PPort		
RP_ZMoverError	PPort		
WspZMovCdd_dirCW	PPort		
WspZMovCdd_dirCCW	PPort		
WspZMovCdd_uMot	PPort		

9.2.6.3 Interfaces

9.2.6.3.1 Interface StatusByte

Data Element	Data Type	Auflösung	Einheit	Bedeutung
StatusByte	uint8	1	-	Status byte

9.2.6.3.2 Interface ifZMoverMotSpeed

Data Element	Data Type	Auflösung	Einheit	Bedeutung
ZMoverMotSpeedVal	sinr16	1	-	ZMover Geschwindigkeit

9.2.7 Messgrößen

ZMDrive_i	ZMover motor current with offset compensation
ZMDrive_iOffset	Motor Current estimated Offset
ZMDrive_iRaw	Motor Current Raw Data
ZMDrive_loadEstim	Load estimation
ZMDrive_stCCW	CounterClockWise digital IO
ZMDrive_stCW	ClockWise digital IO
ZMDrive_stMotorON	Request Motor Active
ZMDrive_stNegOverload	Negative Overload active
ZMDrive_stPosOverload	Positive Overload active
ZMDrive_tEstm	Temperature estimation ZMover motor
ZMDrive_ticks	Current no of ticks value
ZMDrive_ticksMax	Maximum Ticks during current execution cycle
ZMDrive_ticksMin	Minimum Ticks during current execution cycle
ZMDrive_u	Voltage applied to the Motor
ZMDrive_vFilt	Motor speed filtered in Ticks per 10 milliseconds
ZMDrive_vManualOperation_C	Speed Threshold for detectiong manual operation
ZMDrive_vRaw	Motor speed Raw in Ticks per 10 milliseconds
ZMDrive_vReq	Motor speed Request
ZMMechanics_hPAD	Height of the PAD in mm
ZMMechanics_stParked	Praking state of the ZMover (sum of Home position sensors)
ZMMechanics_stPos1	State of the Home Position sensor 1
ZMMechanics_stPos2	State of the Home Position sensor 2
ZMOpCtrl_cntr	Current counter value
ZMOpCtrl_erComponent	Error field for Zmover component

Softwaredokumentation

ZMOpCtrl_stReqDiag	Diagnostic requested access
ZMOpCtrl_stReqPwrTrf	Power Transfer requested control
ZMOpCtrl_stReqStMgr	State Manager requested State
ZMOpCtrl_vReqDiag	Motor speed request from Diagnostic
ZMOpCtrl_vReqPwrTrf	Motor speed request from Power Transfer
ZMOpCtrl_vReqStMgr	Motor speed request from State Manager
ZMover_stCalib	Z Mover Calibration State
ZMover_uSet	ZMover Set Voltage

9.2.8 Applikationswerte

ZMDrive_iFTC_C	Filter Time constant used in Current Offset estimation
ZMDrive_iOffsetTooHigh_C	Sensor offset too high
ZMDrive_iTooHigh_C	Motor Overcurrent Threshold
ZMDrive_J_C	Moment of inertia of the rotor [kg· m ²]
ZMDrive_k1ProtTherm_C	Temperature estimation Gain 1 - Winding resistance devided by Heat disipation threshold
ZMDrive_k2ProtTherm_C	Temperature estimation Gain 2 - Heat Capacity devided by Heat disipation
ZMDrive_kAntiW_C	Gain of the Antiwindup Path
ZMDrive_kF_C	Damping (friction) of the mechanical system [Nms]
ZMDrive_kI_C	Speed Controller Integral Gain
ZMDrive_kMotor_C	Motor Torque Constant
ZMDrive_kP_C	Speed Controller Proportional Gain
ZMDrive_loadMaxNeg_C	Overload Negative threshold
ZMDrive_loadMaxPos_C	Overload Positive threshold
ZMDrive_tErResetThr_C	Threshold for resetting the Overtemperature Error
ZMDrive_tErSetThr_C	Threshold for setting the Overtemperature Error
ZMDrive_trobsK1_C	Tracking Observer gain 1
ZMDrive_trobsK2_C	Tracking Observer gain 2
ZMDrive_uMaxNeg_C	Maximum negative Voltage for the Motor
ZMDrive_uMaxNegOverload_C	Maximum Voltage when Up Overload is detected
ZMDrive_uMaxOverTemp_C	Maximum Voltage when Down Overload is detected
ZMDrive_uMaxPos_C	Maximum Positive Voltage for the Motor
ZMDrive_uMaxPosOverload_C	Maximum Voltage when Down Overload is detected
ZMDrive_vRawStoppedThr_C	Raw speed threshold where the motor can be considered as stopped
ZMManual_stON_C	Manual State control over the ZMover

Softwaredokumentation

ZMManual_stReqDiag_C	Diagnostic Freeze Request
ZMManual_stReqPwrTrf_C	Test State request PwrTrf
ZMManual_stReqStMng_C	Test State request StMgr
ZMManual_stVoltageDirect_C	Manual Voltage Control activation Flag
ZMManual_u_C	Voltage applied to the Motor
ZMManual_vReqDiag_C	Speed request Diagnostic
ZMManual_vReqPwrTrf_C	Speed request PwrTrf
ZMManual_vReqStMng_C	Speed request StMgr
ZMMechanics_Cur	
ZMMechanics_hOffsetPAD_C	Offset of the PAD Height
ZMMotor_vFiltStoppedThr_C	Filtered speed threshold where the motor can be considered as stopped

9.2.9 Fehlerpfade

9.2.9.1 tbd

Fehler

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

9.2.10 Initialisierung

tbd

10 Modul WsploHwAbs (Wsp Hardwareabstraction)

Autor: Michael Münch

Software-Struktur-Position: 30_Implementation/10_WSP/60_HWIOABS

10.1 Allgemein

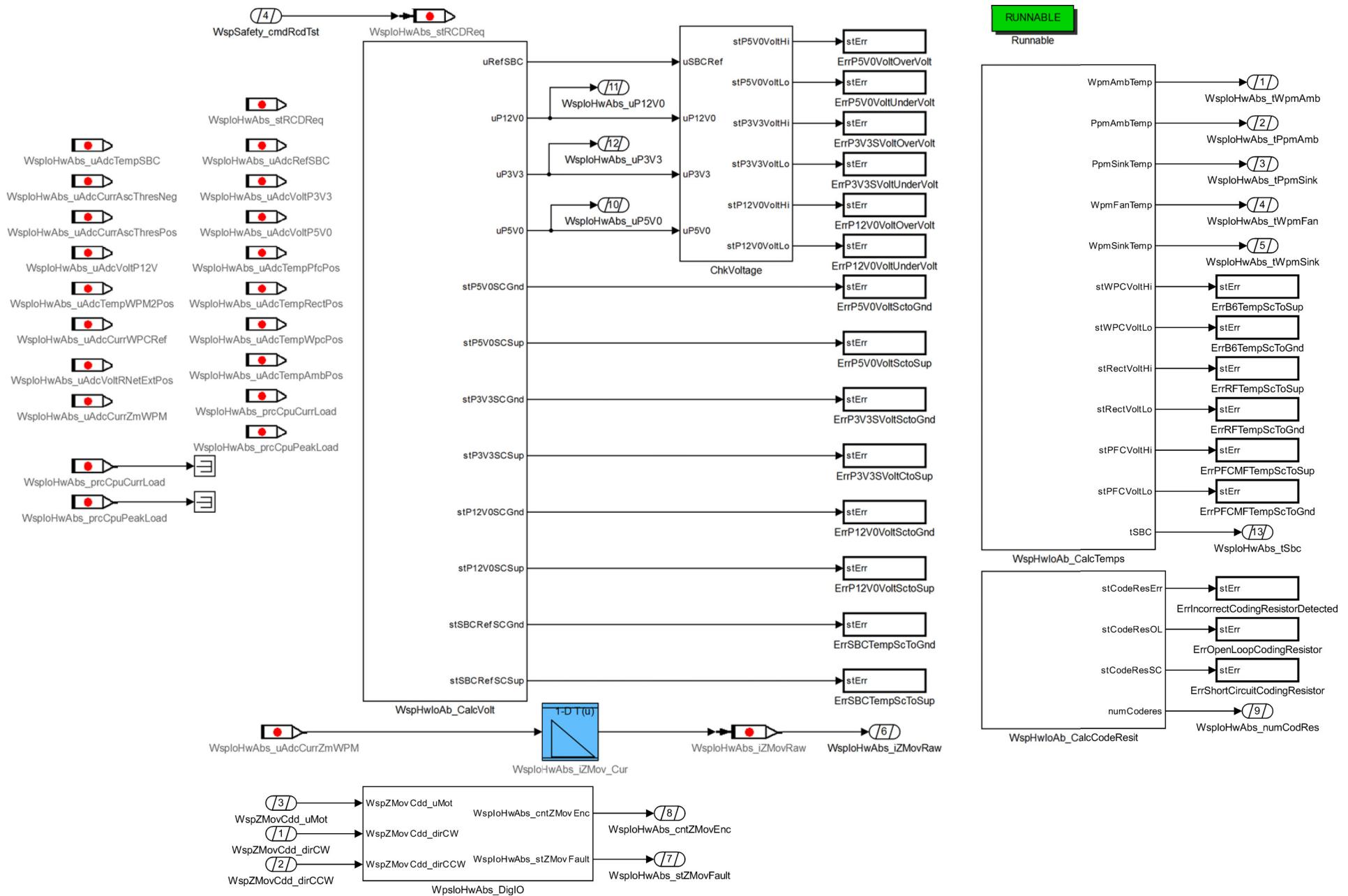
Im Modul *WspoHwAbs* werden die eingelesenen ADC Spannungswerte überwacht und ggf. in physikalische Größen umgewandelt.

Das Modul *WspoHwAbs* enthält folgende Komponente:

- *WspoHwAb_Calc* Überwachung und Umrechnung der ADC Rohwerte.

Softwaredokumentation

10.2 Komponente WsploHwAb_Calc



Softwaredokumentation

Hier werden die ADC Rohwerte der Wayside eingelesen, überwacht und in physikalische Werte umgerechnet.

Zur Unterdrückung der HW Fehlererkennung während des RCD Selbsttests wird der diesbezügliche Eingangsport *WspSafety_CmdRcdTst* zunächst auf die Variable *WsploHwAbs_stRCDReq* eingelesen. Ist dieser Status (auf *true*) gesetzt, läuft der RCD Selbsttest.

Im Block *WspHwloAb_CalcVolt* werden die Versorgungs- und Referenzspannungen des Phanter sowie der Peripherie überwacht. Im Block *ChkVoltage* werden die Versorgungsspannungen des Phanter plausibilisiert.

Im Block *WspHwloAb_CalcTemps* werden die ADC Rohwerte der Temperatursensoren überwacht und in physikalische Werte umgerechnet.

Im Block *WspHwloAb_calcCodeResist* wird der Codierwiderstand überwacht und einem Code zugeordnet.

Im Block *WsploHwAbs_DigIO* werden die digitalen Eingänge gelesen und die digitalen Ausgänge sowie PWM - Ausgänge geschrieben.

Unten wird noch der ADC Spannungswert *WsploHwAbs_uAdcCurrZmWPM* des Z-Mover Stroms mittels der Kennlinie *WsploHwAbs_iZMov_Cur* in den physikalischen Stromwert *WsploHwAbs_iZMovRaw* umgerechnet und auf dem Port *WsploHwAbs_iZMovRaw* ausgegeben.

Softwaredokumentation

10.2.1 Block WspHwIoAb_CalcVolt

Hier werden die Versorgungs- und Referenzspannungen des Prozessors sowie der Peripherie auf Kurzschluss überwacht.

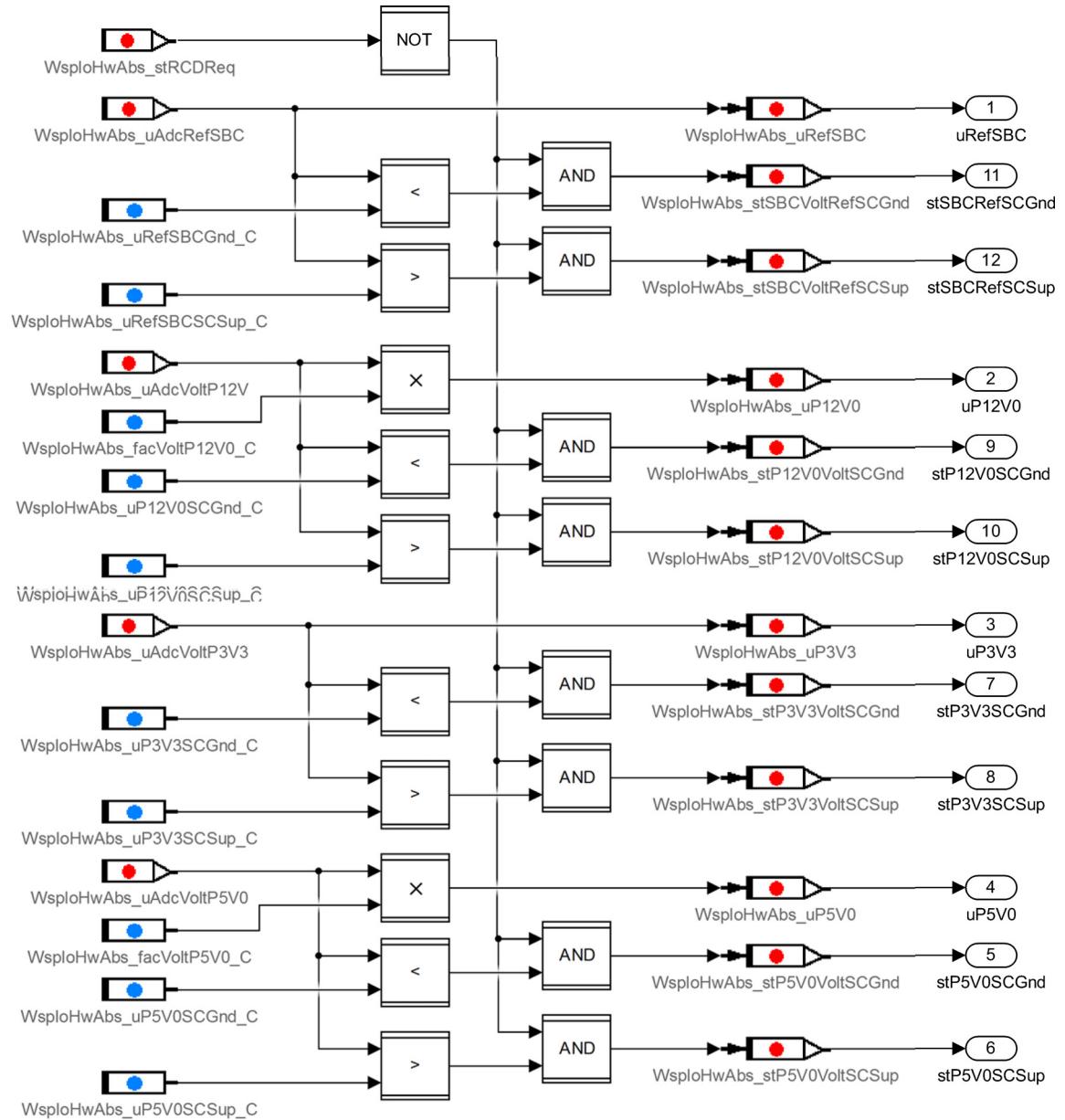
Die Fehlerstati werden nur dann gesetzt, wenn der RCD Selbsttest nicht aktiv ist, also `WsploHwAbs_stRCDReq` den Wert `false` hat.

Der ADC Wert *Wspl0HwAbs_uAdcVoltMuxSBC* der Ausgangsspannung des SBC Multiplexers wird mit der Spannungsschwelle *Wspl0HwAbs_uSBCGnd_C* verglichen. Unterschreitet der ADC Wert die Schwelle, wird der Fehlerstatus *Wspl0HwAbs_stSBCVoltSCGnd* (*Siehe "SBCTempScToGnd" auf Seite 163*) für Kurzschluss nach Masse gesetzt, sonst zurückgesetzt.

Überschreitet der ADC Wert die Schwelle *WsploHwAbs_uSBCSCSup_C*, wird der Fehlerstatus *WsploHwAbs_stSBCVoltSCSup* für eine Kurzschluss zur Versorgung gesetzt. Der physikalische Wert *WsploHwAbs_uSBC* entspricht hier dem ADC Rohwert.

Entsprechend wird für die anderen ADC Spannungen verfahren. Der ADC Rohwert *Wspl0HwAbs_uAdcVoltP12V* der 12V Versorgung muss mittels des Skalierungsfaktors *Wspl0HwAbs_facVoltP12V0_C* in den physikalischen Wert *Wspl0HwAbs_uP12V0* umgerechnet werden.

Ebenso der ADC Rohwert `WsploHwAbs_uAdcVoltP5V0`. Dieser wird mittels des Faktors `WsploHwAbs_facVoltP5V0_C` in den physikalischen Wert `WsploHwAbs_uP5V0` umgerechnet.



10.2.2 Block ChkVoltage

Im Block *ChkVoltage* werden definierte Spannungen auf die Einhaltung zulässiger Werte überprüft.

Die Fehlerstati werden nur dann gesetzt, wenn der RCD Selbsttest nicht aktiv ist, also *WsploHwAbs_stRCDReq* den Wert *false* hat.

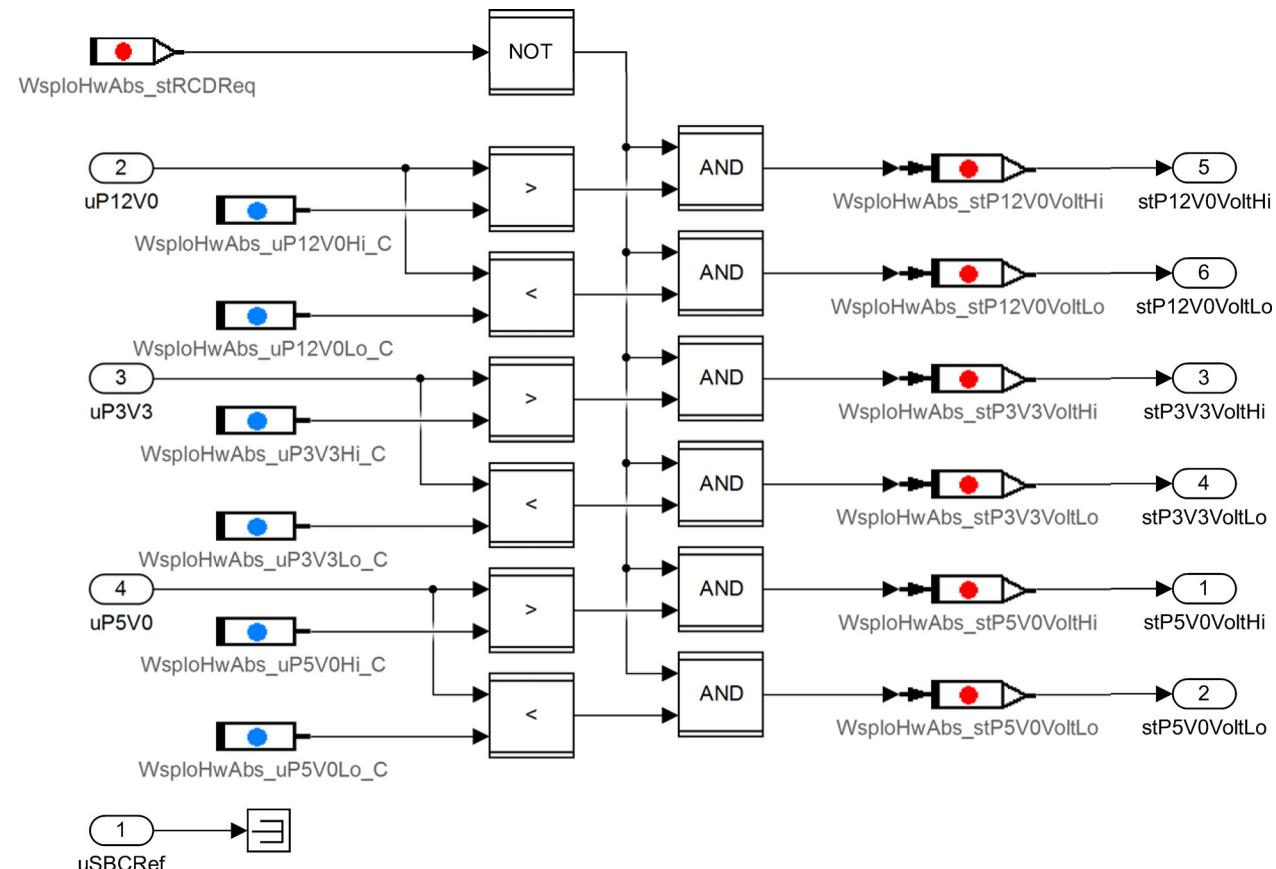
Liegt die 12V Versorgungsspannung *uP12V0* über dem Grenzwert *WsploHwAbs_uP12V0Hi_C*, wird der Fehlerstatus *WsploHwAbs_stP12V0VoltHi* (auf true) gesetzt (bei nicht zu treffender Bedingungen werden die Fehlerstati zurück, d.h. auf false gesetzt).

Liegt *uP12V0* unterhalb von *WsploHwAbs_uP12V0Lo_C*, wird der Fehlerstatus *WsploHwAbs_stP12V0VoltLo* gesetzt.

Entsprechend wird der Fehlerstatus *WsploHwAbs_stP3V3VoltHi* gesetzt, wenn die 3.3V Versorgungsspannung *uP3V3* über der Schwelle *WsploHwAbs_uP3V3Hi_C* liegt, der Fehlerstatus *WsploHwAbs_stP3V3VoltLo*, wenn diese Spannung unter *WsploHwAbs_uP3V3Lo_C* liegt.

Für die 5V Versorgungsspannung *uP5V0* wird der Fehlerstatus *WsploHwAbs_stP5V0VoltHi* gesetzt, wenn diese über *WsploHwAbs_uP5V0Hi_C* liegt oder der Fehlerstatus *WsploHwAbs_stP5V0VoltLo*, wenn die Spannung unter *WsploHwAbs_uP5V0Lo_C* liegt.

Die Fehlerstatis werden anschliessend im übergeordneten Block den entsprechenden Fehlerpfaden übergeben.



10.2.3 Block WspHwloAb_CalcTemps

Hier werden die ADC Rohwerte der Temperatursensoren in physikalische Werte umgerechnet und der Signal Range Check der ADC Werte, sofern spezifiziert, durchgeführt.

Die Fehlerstati werden nur dann gesetzt, wenn der RCD Selbsttest nicht aktiv ist, also [WsploHwAbs_stRCDReq](#) den Wert *false* hat.

Zunächst wird die ADC Rohspannung

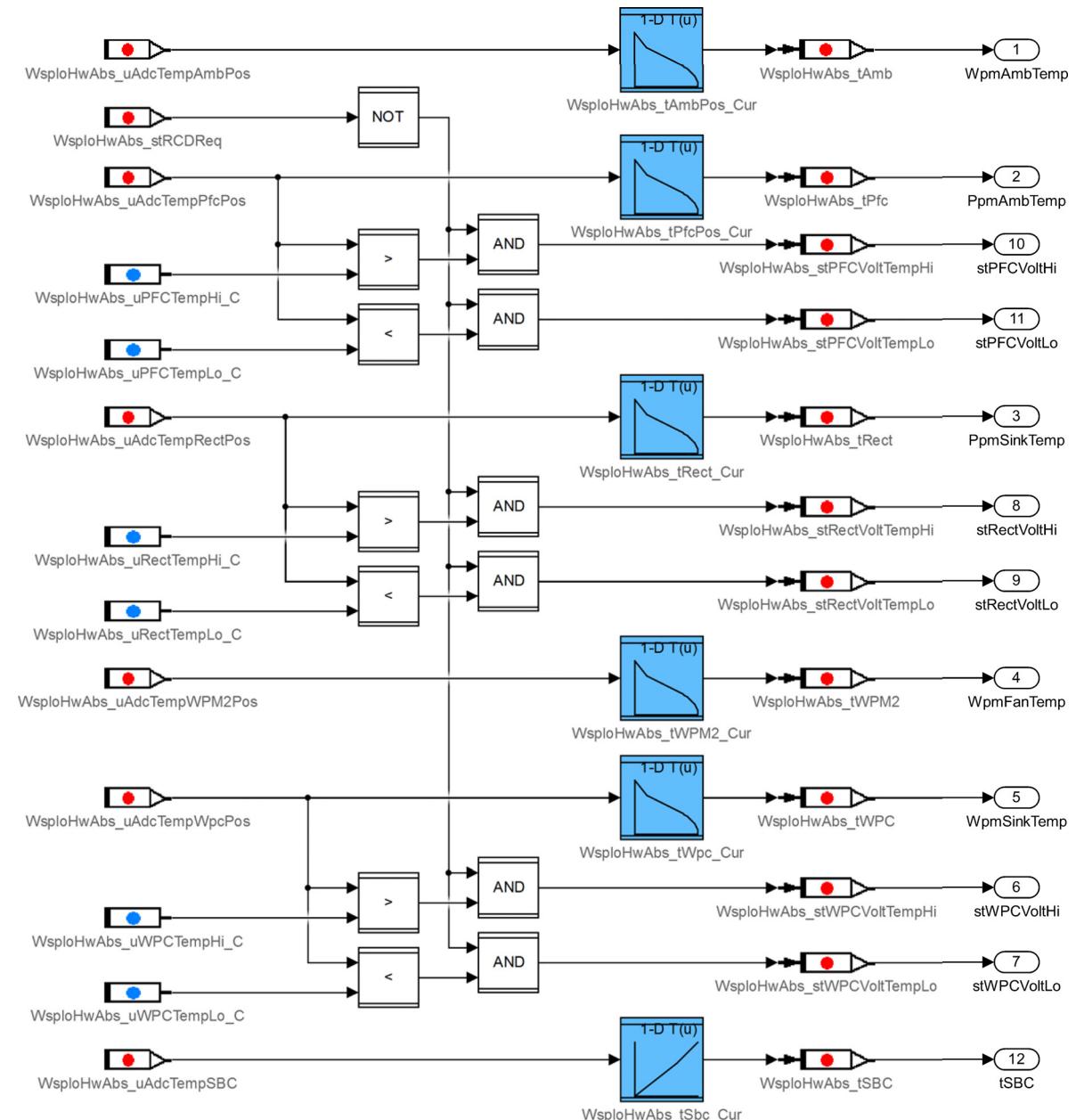
[WsploHwAbs_uAdcTempAmbPos](#) des Umgebungstemperatursensors mittels der Kennlinie [WsploHwAbs_tAmbPos_Cur](#) in den Temperaturwert [WsploHwAbs_tAmb](#) umgerechnet.

Als nächstes wird ein Signal Range Check der Temperatursensorspannung des Pfc Sensors durchgeführt. Hierzu wird die ADC Spannung [WsploHwAbs_uAdcTempPfcPos](#) mit der oberen Schwelle [WsploHwAbs_uPFCTempHi_C](#) verglichen. Liegt die Spannung darüber, wird der Fehlerstatus [WsploHwAbs_stPFCVoltTempHi](#) gesetzt. Liegt die Spannung unter [WsploHwAbs_uPFCTempLo_C](#), wird der Fehlerstatus [WsploHwAbs_stPFCVoltTempLo](#) gesetzt.

Anschließend wird der ADC Rohwert mittels der Kennlinie [WsploHwAbs_tPfcPos_Cur](#) in den Temperaturwert [WsploHwAbs_tPfc](#) umgerechnet.

Als nächstes wird der Signal Range Check der Temperatursensorspannung [WsploHwAbs_uAdcTempRectPos](#) des Gleichrichterteils durchgeführt. Liegt die ADC Spannung über [WsploHwAbs_uRectTempHi_C](#), wird der Fehlerstatus [WsploHwAbs_stRectVoltTempHi](#) gesetzt. Liegt sie unter [WsploHwAbs_uRectTempLo_C](#) der Fehlerstatus [WsploHwAbs_stRectVoltTempLo](#).

Der ADC Spannungswert wird dann mittels der Kennlinie [WsploHwAbs_tRect_Cur](#) in den Temperaturwert [WsploHwAbs_tRect](#) umgerechnet.



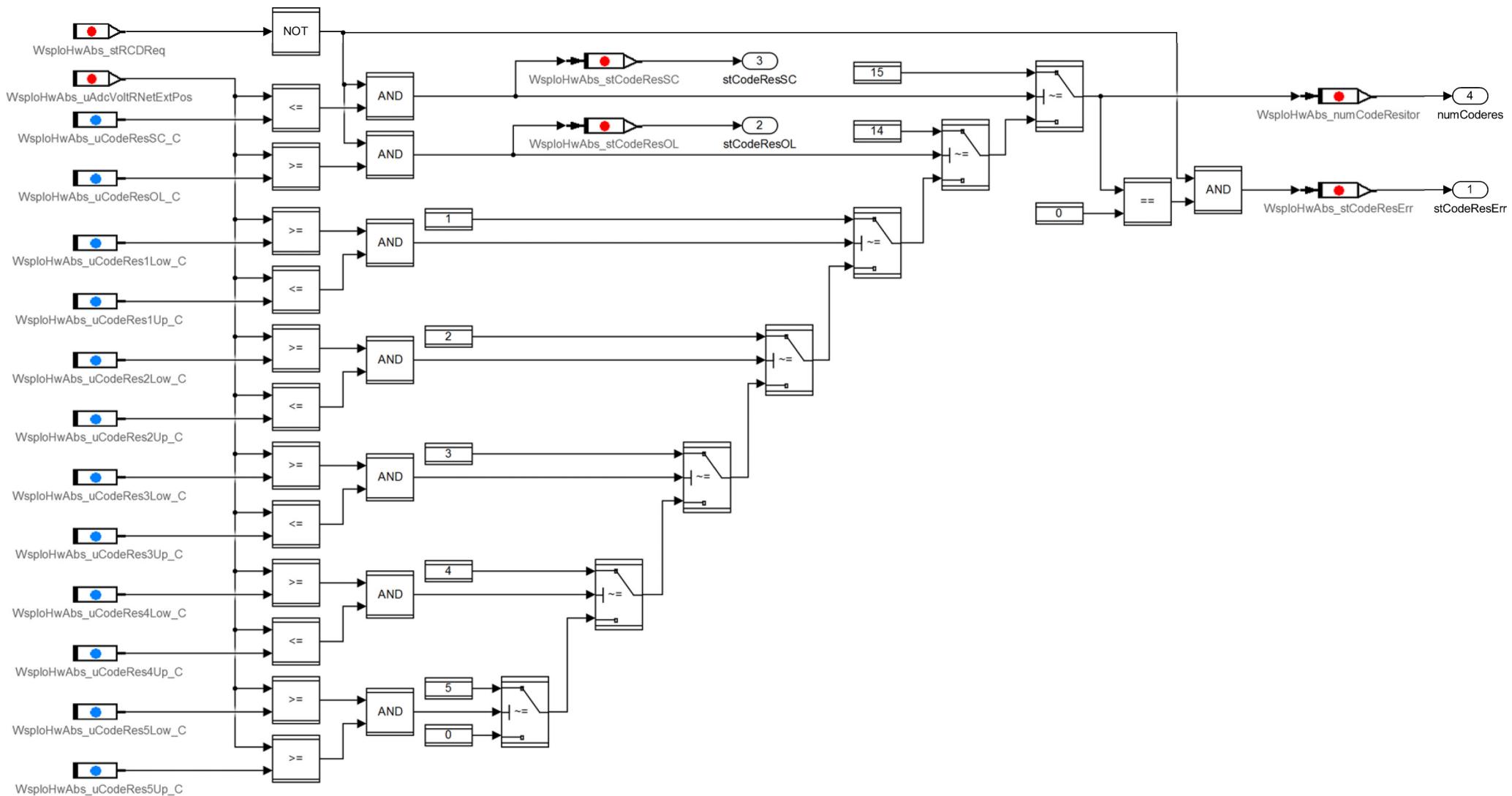
Softwaredokumentation

primove
true e-mobility

Der ADC Rohwert *WsploHwAbs_uAdcTempWPM2Pos* wird über die Kennlinie *WsploHwAbs_tWPM2_Cur* in den Temperaturwert *WsploHwAbs_tWPM2* umgerechnet. Schließlich wird noch der Signal Range Check für die Temperatursensorspannung *WsploHwAbs_uAdcTempWpcPos* durchgeführt. Liegt sie über *WsploHwAbs_uWPCTempHi_C*, wird der Fehlerstatus *WsploHwAbs_stWPCVoltTempHi* gesetzt. Liegt sie unter *WsploHwAbs_uWPCTempLo_C*, wird der Fehlerstatus *WsploHwAbs_stWPCVoltTempLo* gesetzt.

Die ADC Spannung wird anschließend noch mittels der Kennlinie *WsploHwAbs_tWpc_Cur* in den Temperaturwert *WsploHwAbs_tWPC* umgerechnet. Die Fehlerstati werden dann im Übergeordneten Block den entsprechenden Fehlerpfaden übergeben.

10.2.4 Block WspHwloAb_CalcCodeResist



Im Block *WspHwloAb_CalcCodeResit* wird der Kodierwiderstand des PAD - Netzsteckers überprüft sowie ein Status für eine erkannte Kodierung zur Verfügung gestellt. Hierzu wird die Spannung *WsploHwAbs_uAdcVoltRNetExtPos* über dem Kodierwiderstand mit den folgenden Spannungsleveln verglichen: Liegt die Spannung *WsploHwAbs_uAdcVoltRNetExtPos* unterhalb der Schwelle *WsploHwAbs_uCodeResSC_C*, wird der Fehlerstatus *WsploHwAbs_stCodeResSC* gesetzt (Kurzschluss).

Die Fehlerstati werden nur dann gesetzt, wenn der RCD Selbsttest nicht aktiv ist, also *WsploHwAbs_stRCDReq* den Wert *false* hat.

Softwaredokumentation

Der Statuscode *WsploHwAbs_numCodeResistor* wird in diesem Fall auf 15 gesetzt. Liegt die Spannung über *WsploHwAbs_uCodeResOL_C*, wird der Fehlerstatus *WsploHwAbs_stCodeResOL* gesetzt (Open Load). Der Statuscode wird dann auf 14 gesetzt.

Anschließend wird geprüft, ob die Spannung innerhalb eines der Intervalle [*WsploHwAbs_uCodeRes1Low_C*, *WsploHwAbs_uCodeRes1Up_C*] ... [*WsploHwAbs_uCodeRes5Low_C*, *WsploHwAbs_uCodeRes5Up_C*] liegt. Wird kein passendes Spannungsintervall gefunden, wird der Statuscode auf 0 und der Fehlerstatus *WsploHwAbs_stCodeResErr* auf true gesetzt.

Die Spannungsgrenzen für die 5 Bereiche sind derzeit wie folgt Appliziert:

Statuscode <i>WsploHwAbs_numCodeResistor</i>	Spannung <i>WsploHwAbs_uAdcVoltRNetExtPos [V]</i>
0 -> Ungültiger Kodierwiderstand	Ausserhalb definierter Bereiche (incl. Fehlerbereiche)
1 -> not used	0.5 - 0.8 -> 330 Ohm
2 -> not used	0.9 - 1.1 -> 560 Ohm
3 -> Country Variant 1,2,3,4,9 / 16A / 220V - 230V	1.6 - 1.9 -> 1K2 Ohm
4 -> Country Variant 5,10 / 16A / 240V	2.5 - 3.0 -> 2K7 Ohm
5 -> Country Variant 7 / 8A / 220V	3.7 - 4.5 -> 10K0 Ohm
14 -> Open Load	> 4.6
15 -> Shortcut	< 0.3

10.2.5 Block WsiloHwAbs_DigIO

Im Block WsiloHwAbs_DigIO werden die digitalen Eingänge gelesen und die digitalen Ausgänge sowie PWM - Ausgänge geschrieben.

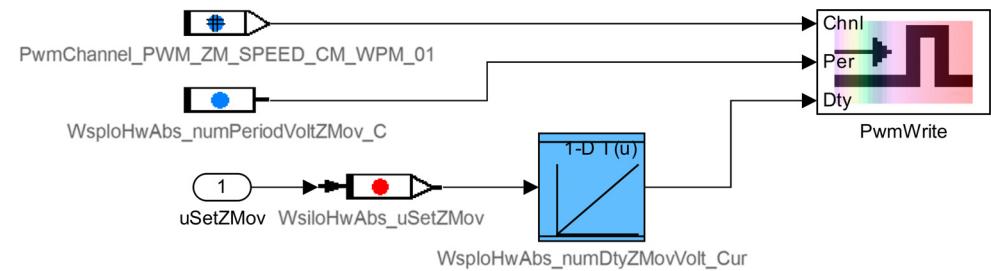
Der CustomCode Block GetMotorTicks liest direkt vom Timer1 Register den aktuellen Impulscounter des Quadraturdekoders des Z Mover Motors aus.

Im Block Pwm werden die Pwm Signale ausgegeben. Die Ausgangsstati (Fehlerstati) der einzelnen Blöcke werden hier den entsprechenden Fehlerpfaden zugeordnet.

10.2.5.1 Block Pwm

Im Block Pwm wird die vom ZMover Modul kommende Soll - Ansteuerspannung *WspZMovCdd_uMot* mittels der Kennlinie *WsiloHwAbs_numDtyZMovVolt_Cur* in den numerischen Wert konvertiert, welche der für die Spannungszeugung entsprechenden PWM Ratio entspricht.

Mit diesem Ratiowert und die PWM Periodenzeit *WsiloHwAbs_numPeriodVoltZMov_C* wird dann die PwmWrite Funktion für den PWM Channel *PwmChannel_PWM_ZM_SPEED_CM_WPM_01* aufgerufen.

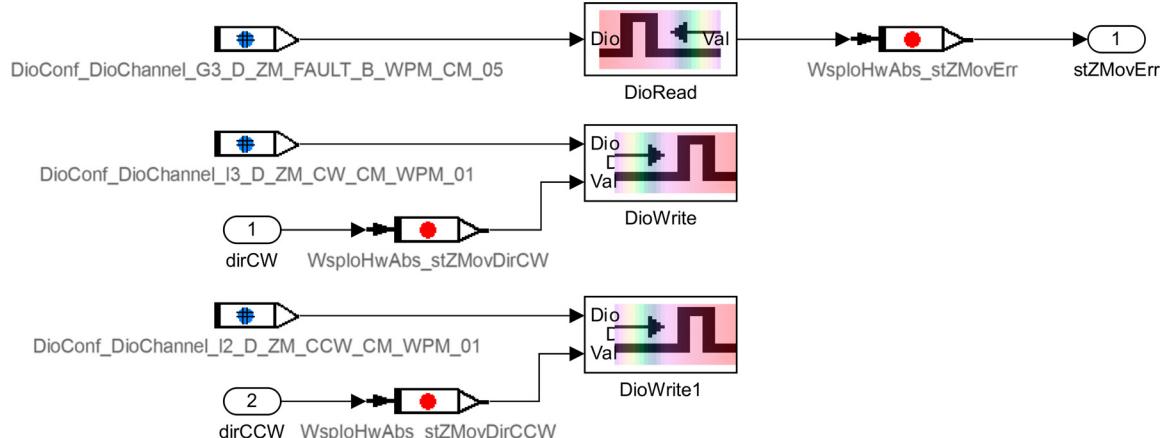


10.2.5.2 Block Dio

Im Block Dio werden digitale Eingänge gelesen und digitale Ausgänge geschrieben.

Der Fehlerstatus *WsiloHwAbs_stZMovErr* des Z Mover Ansteuermoduls (HW) wird vom Dio Input Channel *DioConf_DioChannel_G3_D_ZM_FAULT_B_WPM_CM_05* eingelesen. Ist dieser true, hat die Hardware einen Fehler festgestellt.

Zur Festlegung der Drehrichtung des Z Mover Motors werden die beiden komplementären Dio Pins *DioConf_DioChannel_I3_D_ZM_CW_CM_WPM_01* sowie *DioConf_DioChannel_I2_D_ZM_CCW_CM_WPM_01* entsprechend der vom Z Mover Modul kommenden Sollwerte *WspZMovCdd_dirCW* und *WspZMovCdd_dirCCW* geschrieben.



10.2.6 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Requiered Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

10.2.6.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
WspZMovCdd_dirCW	RPort	ifBool	Status Z-Movermotor Drehrichtung im Uhrzeigersinn
WspZMovCdd_dirCCW	RPort	ifBool	Status Z-Movermotor Drehrichtung im Gegenuhrzeigersinn
WspZMovCdd_uMot	RPort	ifVoltage	Sollspannung Z-Movermotor

10.2.6.2 Ausgangsgrößen

Name	Typ	Interface / Argument	Beschreibung
WpmAmbTemp	PPort	ifTemperatureC	Umgebungstemperatur PAD Innenraum bei B6 Brücke
PpmAmbTemp	PPort	ifTemperatureC	Umgebungstemperatur PAD Innenraum bei PFC
PpmSinkTemp	PPort	ifTemperatureC	Temperatur PFC Kühlkörper
WpmFanTemp	PPort	ifTemperatureC	Temperatur am Lüfter
WpmSinkTemp	PPort	ifTemperatureC	Temperatur B6 Kühlkörper
WsploHwAbs_iZMovRaw	PPort	CurrentSmallRange_mA	Rohwert Z-Mover Ist Strom
WsploHwAbs_stZMovFault	PPort	StatusByte	Fehlerstatus Z-Mover von HW-Pin
WsploHwAbs_cntZMovEnc	PPort	Cntr16Bit	Counter Wert ZMover Motor Quadraturdekoder

10.2.6.3 Interfaces

Softwaredokumentation

10.2.6.3.1 Interface ifBool

Data Element	Data Type	Auflösung	Einheit	Bedeutung
Bool/Val	bool	1	-	Status true/false

10.2.6.3.2 Interface ifVoltage

Data Element	Data Type	Auflösung	Einheit	Bedeutung
Value	uint16	0.01	V	Spannungswert

10.2.6.3.3 Interface ifTemperatureC

Data Element	Data Type	Auflösung	Einheit	Bedeutung
TempCValue	sint16	0.1	°C	Temperaturwert

10.2.6.3.4 Interface CurrentSmallRange_mA

Data Element	Data Type	Auflösung	Einheit	Bedeutung
Current_mA	sint16	0.001	A	Stromwert

10.2.6.3.5 Interface StatusByte

Data Element	Data Type	Auflösung	Einheit	Bedeutung
StatusByte	uint8	1	-	Status Byte

10.2.6.3.6 Interface Cntr16Bit

Data Element	Data Type	Auflösung	Einheit	Bedeutung
Cntr	sint16	1	-	Zählerwert

10.2.7 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

<i>WspoHwAbs_iZMovRaw</i>	Rohwert ZMover-Strom
<i>WspoHwAbs_numCodeResistor</i>	Nummer des erkannten Kodierwiderstandes (1 - 5 -> gültige Widerstände, 0 -> Kein gültiger Widerstandswert erkannt)
<i>WspoHwAbs_prcCpuCurrLoad</i>	Aktueller Cpu Load Pad
<i>WspoHwAbs_prcCpuPeakLoad</i>	Maximaler Cpu Load Pad
<i>WspoHwAbs_stCodeResErr</i>	Status Fehler Kodierwiderstand. true = Fehler, false = kein Fehler
<i>WspoHwAbs_stCodeResOL</i>	Status Fehler Kodierwiderstand OpenLoop
<i>WspoHwAbs_stCodeResSC</i>	Status Fehler Kodierwiderstand Shortcut
<i>WspoHwAbs_stP12V0VoltHi</i>	State Internal Voltage Measurement P12V0 Voltage to low
<i>WspoHwAbs_stP12V0VoltLo</i>	State Internal Voltage Measurement P12V0 Voltage to high
<i>WspoHwAbs_stP12V0VoltSCGnd</i>	State Internal Voltage Measurement P12V0 Voltage ShortCut to Ground
<i>WspoHwAbs_stP12V0VoltSCSup</i>	State Internal Voltage Measurement P12V0 Voltage ShortCut to Supply
<i>WspoHwAbs_stP3V3VoltHi</i>	State Internal Voltage Measurement P3V3 Voltage to low
<i>WspoHwAbs_stP3V3VoltLo</i>	State Internal Voltage Measurement P3V3 Voltage to high
<i>WspoHwAbs_stP3V3VoltSCGnd</i>	State Internal Voltage Measurement P5V0 Voltage ShortCut to Ground
<i>WspoHwAbs_stP3V3VoltSCSup</i>	State Internal Voltage Measurement P5V0 Voltage ShortCut to Supply
<i>WspoHwAbs_stP5V0VoltHi</i>	State Internal Voltage Measurement P5V0 Voltage to low
<i>WspoHwAbs_stP5V0VoltLo</i>	State Internal Voltage Measurement P5V0 Voltage to high
<i>WspoHwAbs_stP5V0VoltSCGnd</i>	State Internal Voltage Measurement P5V0 Voltage ShortCut to Ground
<i>WspoHwAbs_stP5V0VoltSCSup</i>	State Internal Voltage Measurement P5V0 Voltage ShortCut to Supply
<i>WspoHwAbs_stPFCVoltTempHi</i>	Status ADC Spannung PFC Temperatursensor zu hoch (wenn true)
<i>WspoHwAbs_stPFCVoltTempLo</i>	Status ADC Spannung PFC Temperatursensor zu niedrig (wenn true)
<i>WspoHwAbs_stRectVoltTempHi</i>	Status ADC Spannung Rectifier Temperatursensor zu hoch (wenn true)
<i>WspoHwAbs_stRectVoltTempLo</i>	Status ADC Spannung Rectifier Temperatursensor zu niedrig (wenn true)

Softwaredokumentation

<i>WsploHwAbs_stSBCVoltSCGnd</i>	State SBC Voltage Measurement ShortCut to Ground
<i>WsploHwAbs_stSBCVoltSCSup</i>	State SBC Voltage Measurement ShortCut to Supply
<i>WsploHwAbs_stWPCHoltTempHi</i>	Status ADC Spannung WPC POS Temperatursensor zu hoch (wenn true)
<i>WsploHwAbs_stWPCHoltTempLo</i>	Status ADC Spannung WPC POS Temperatursensor zu niedrig (wenn true)
<i>WsploHwAbs_tAmb</i>	PAD Umgebungstemperatur
<i>WsploHwAbs_tPfc</i>	PAD Pfc FET Temperatur
<i>WsploHwAbs_tRect</i>	PAD Gleichrichtertemperatur
<i>WsploHwAbs_tWPC</i>	PAD WPC FET Temperatur
<i>WsploHwAbs_tWPM2</i>	PAD WPM2 FET Temperatur
<i>WsploHwAbs_uAdcCurrAscThresNeg</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> I_ASC_THRESHOLD_NEGATIVE_AdcVal
<i>WsploHwAbs_uAdcCurrAscThresPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> I_ASC_THRESHOLD_POSITIVE_AdcVal
<i>WsploHwAbs_uAdcCurrWPCRef</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> I_MEAS_WPC_REF_CM_01_AdcVal
<i>WsploHwAbs_uAdcCurrZmWPM</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> I_MEAS_ZM_WPM_CM_05_AdcVal
<i>WsploHwAbs_uAdcTempAmbPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> T_MEAS_AMB_POS_WPM_CM_01_AdcVal
<i>WsploHwAbs_uAdcTempPfcPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> T_MEAS_PFC_POS_PPM_CM_01_AdcVal
<i>WsploHwAbs_uAdcTempRectPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> T_MEAS_RECT_POS_PPM_CM_01_AdcVal
<i>WsploHwAbs_uAdcTempWpcPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> T_MEAS_WPC_POS_WPM_CM_01_AdcVal
<i>WsploHwAbs_uAdcTempWPM2Pos</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> T_MEAS_WPM2_POS_WPM_CM_01_AdcVal
<i>WsploHwAbs_uAdcVoltMuxSBC</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> U_MEAS_MUX_SBC CU_01_AdcVal
<i>WsploHwAbs_uAdcVoltP12V</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> U_MEAS_P12V0 CU_AdcVal
<i>WsploHwAbs_uAdcVoltP3V3</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> U_MEAS_P3V3_S CU_AdcVal
<i>WsploHwAbs_uAdcVoltP5V0</i>	Adc Eingangsspannung 12Bit von Kanal Unit1-> U_MEAS_P5V0 CU_AdcVal
<i>WsploHwAbs_uAdcVoltRNetExtPos</i>	Adc Eingangsspannung 12Bit von Kanal Unit0-> U_MEAS_R_NETWORK_EXTERNAL_POS_05_AdcVal

Softwaredokumentation

<i>WsploHwAbs_uP12V0</i>	Internal Voltage Measurement P12V0
<i>WsploHwAbs_uP3V3</i>	Internal Voltage Measurement P3V3
<i>WsploHwAbs_uP5V0</i>	Internal Voltage Measurement P5V0
<i>WsploHwAbs_uSBC</i>	Spare Voltage Measurement configurable by
<i>WsploHwAbs_uSetZMov</i>	Ansteuerspannung ZMover
<i>WsploHwAbs_numMotorTicks</i>	ZMover Motorposition in Ticks
<i>WsploHwAbs_stZMovDirCCW</i>	State of ZMover Direction CounterClockWise
<i>WsploHwAbs_stZMovDirCW</i>	State of ZMover Direction ClockWise
<i>WsploHwAbs_stZMovErr</i>	State of ZMover Error read from DIO

10.2.8 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>WsploHwAbs_facVoltP12V0_C</i>	Umrechnungsfaktor ADC Rohsspannung in P12V0 Spannung
<i>WsploHwAbs_facVoltP5V0_C</i>	Umrechnungsfaktor ADC Rohsspannung in P5V0 Spannung
<i>WsploHwAbs_iZMov_Cur</i>	Umrechnung ADC Spannung in Z Mover Strom
<i>WsploHwAbs_uCodeRes1Low_C</i>	Untere Spannungsschwelle Kodier-Widerstand 1 (330 Ohm)
<i>WsploHwAbs_uCodeRes1Up_C</i>	Obere Spannungsschwelle Kodier-Widerstand 1 (330 Ohm)
<i>WsploHwAbs_uCodeRes2Low_C</i>	Untere Spannungsschwelle Kodier-Widerstand 2 (560 Ohm)
<i>WsploHwAbs_uCodeRes2Up_C</i>	Obere Spannungsschwelle Kodier-Widerstand 2 (560 Ohm)
<i>WsploHwAbs_uCodeRes3Low_C</i>	Untere Spannungsschwelle Kodier-Widerstand 3 (1K2 Ohm)
<i>WsploHwAbs_uCodeRes3Up_C</i>	Obere Spannungsschwelle Kodier-Widerstand 3 (1K2 Ohm)
<i>WsploHwAbs_uCodeRes4Low_C</i>	Untere Spannungsschwelle Kodier-Widerstand 4 (2K7 Ohm)
<i>WsploHwAbs_uCodeRes4Up_C</i>	Obere Spannungsschwelle Kodier-Widerstand 4 (2K7 Ohm)
<i>WsploHwAbs_uCodeRes5Low_C</i>	Untere Spannungsschwelle Kodier-Widerstand 5 (10K Ohm)
<i>WsploHwAbs_uCodeRes5Up_C</i>	Obere Spannungsschwelle Kodier-Widerstand 5 (10K Ohm)
<i>WsploHwAbs_uCodeResOL_C</i>	Spannungsschwelle Kodier-Widerstand Open Loop
<i>WsploHwAbs_uCodeResSC_C</i>	Spannungsschwelle Kodier-Widerstand Shortcut
<i>WsploHwAbs_uP12V0Hi_C</i>	Schwelle interne Versorgungsspannung P12V0 zu hoch
<i>WsploHwAbs_uP12V0Lo_C</i>	Schwelle interne Versorgungsspannung P12V0 zu niedrig
<i>WsploHwAbs_uP12V0SCGnd_C</i>	Schwelle interne Versorgungsspannung P12V0 ShortCut to Ground
<i>WsploHwAbs_uP12V0SCSup_C</i>	Schwelle interne Versorgungsspannung P12V0 ShortCut to Supply
<i>WsploHwAbs_uP3V3Hi_C</i>	Schwelle interne Versorgungsspannung P3V3 zu hoch
<i>WsploHwAbs_uP3V3Lo_C</i>	Schwelle interne Versorgungsspannung P3V3 zu niedrig
<i>WsploHwAbs_uP3V3SCGnd_C</i>	Schwelle interne Versorgungsspannung P3V3 ShorCut to Ground
<i>WsploHwAbs_uP3V3SCSup_C</i>	Schwelle interne Versorgungsspannung P3V3 ShorCut to Supply

<i>WsploHwAbs_uP5V0Hi_C</i>	Schwelle interne Versorgungsspannung P5V0 zu hoch
<i>WsploHwAbs_uP5V0Lo_C</i>	Schwelle interne Versorgungsspannung P5V0 zu niedrig
<i>WsploHwAbs_uP5V0SCGnd_C</i>	Schwelle interne Versorgungsspannung P5V0 ShorCut to Ground
<i>WsploHwAbs_uP5V0SCSup_C</i>	Schwelle interne Versorgungsspannung P5V0 ShorCut to Supply
<i>WsploHwAbs_uPFCTempHi_C</i>	Schwelle ADC Spannung PFC Temperatursensor zu hoch
<i>WsploHwAbs_uPFCTempLo_C</i>	Schwelle ADC Spannung PFC Temperatursensor zu niedrig
<i>WsploHwAbs_uRectTempHi_C</i>	Schwelle ADC Spannung Rectifier Temperatursensor zu hoch
<i>WsploHwAbs_uRectTempLo_C</i>	Schwelle ADC Spannung Rectifier Temperatursensor zu niedrig
<i>WsploHwAbs_uSBCGnd_C</i>	Schwelle Spannung SBC ShorCut to Ground
<i>WsploHwAbs_uSBCSCSup_C</i>	Schwelle Spannung SBC ShorCut to Supply
<i>WsploHwAbs_uWPCTempHi_C</i>	ADC Spannung WPC POS Temeraturesensor zu hoch
<i>WsploHwAbs_uWPCTempLo_C</i>	Schwelle ADC Spannung WPC POS Temperatursensor zu niedrig
<i>WsploHwAbs_numPeriodVoltZMov_C</i>	Frequenzvorgabe PWM Ausgang für ZMoverspannung (numerisch)
<i>WsploHwAbs_tAmbPos_Cur</i>	TemperaturUmrechnungskurve Ambienttemperatur
<i>WsploHwAbs_tPfcPos_Cur</i>	TemperaturUmrechnungskurve PfcTemperatur
<i>WsploHwAbs_tRect_Cur</i>	Umrechnung Gleichrichtertemperatur
<i>WsploHwAbs_tWPM2_Cur</i>	Umrechnung WPM2 Temperatur
<i>WsploHwAbs_tWpc_Cur</i>	Umrechnung Wpc Temperatur
<i>WsploHwAbs_numDtyZMovVolt_Cur</i>	Umrechnung Sollspannung ZMover in numerischen Wert zum setzen des entsprechenden DutyCycle mit Tresos Funktion

10.2.9 Fehlerpfade

10.2.9.1 P5V0VoltOverVolt

Fehler Spannung P5V0 zu hoch

P5V0VoltOverVolt

Softwaredokumentation

Fehlerindex	EHI_P5V0VOLTOVERVOLT 151
Fehler Status	ErrHndl_stP5V0VoltOverVolt
Defektbedingung	P5V0 Voltage Sensor: Overvoltage
Heilbedingung	P5V0 Voltage Sensor Fehlermeldung: Keine Überspannung
Fehlerklasse	ErrHndl_Class_P5V0VoltOverVolt_C
Defekt Entprellzeit	ErrHndl_tiDebDef_P5V0VoltOverVolt_C
Heil Entprellzeit	ErrHndl_tiDebOk_P5V0VoltOverVolt_C
DTC	0x151015

10.2.9.2 P5V0VoltUnderVolt

Fehler Spannung P5V0 zu niedrig

P5V0VoltUnderVolt	
Fehlerindex	EHI_P5V0VOLTUNDERVOLT 152
Fehler Status	ErrHndl_stP5V0VoltUnderVolt
Defektbedingung	P5V0 Voltage Sensor: Undervoltage
Heilbedingung	P5V0 Voltage Sensor Fehlermeldung: Keine Unterspannung
Fehlerklasse	ErrHndl_Class_P5V0VoltUnderVolt_C
Defekt Entprellzeit	ErrHndl_tiDebDef_P5V0VoltUnderVolt_C
Heil Entprellzeit	ErrHndl_tiDebOk_P5V0VoltUnderVolt_C
DTC	0x151016

10.2.9.3 P3V3SVoltOverVolt

Fehler Spannung P3V3 zu hoch

P3V3SVoltOverVolt	
Fehlerindex	EHI_P3V3SVOLTOVERVOLT 153

Softwaredokumentation

Fehler Status	<i>ErrHndl_stP3V3SVoltOverVolt</i>
Defektbedingung	P3V3_S Temperature Sensor_1: Overvoltage
Heilbedingung	P3V3_S Temperature Sensor_1: Keine Überspannung / No Overvoltage
Fehlerklasse	<i>ErrHndl_Class_P3V3SVoltOverVolt_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P3V3SVoltOverVolt_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P3V3SVoltOverVolt_C</i>
DTC	0x15101A

10.2.9.4 P3V3SVoltUnderVolt

Fehler Spannung P3V3 zu niedrig

P3V3SVoltUnderVolt	
Fehlerindex	<i>EHI_P3V3SVOLTUNDERVOLT</i> 154
Fehler Status	<i>ErrHndl_stP3V3SVoltUnderVolt</i>
Defektbedingung	P3V3_S Temperature Sensor_1: Undervoltage
Heilbedingung	P3V3_S Temperature Sensor_1: Keine Unterspannung / No Undervoltage
Fehlerklasse	<i>ErrHndl_Class_P3V3SVoltUnderVolt_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P3V3SVoltUnderVolt_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P3V3SVoltUnderVolt_C</i>
DTC	0x15101B

10.2.9.5 P12V0VoltOverVolt

Fehler Spannung P12V0 zu hoch

P12V0VoltOverVolt	
Fehlerindex	<i>EHI_P12V0VOLTOVERVOLT</i> 143
Fehler Status	<i>ErrHndl_stP12V0VoltOverVolt</i>

Defektbedingung	P12V0 Voltage Sensor: Overvoltage
Heilbedingung	P12V0 Voltage Sensor Fehlermeldung: Keine Überspannung
Fehlerklasse	<i>ErrHndl_Class_P12V0VoltOverVolt_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P12V0VoltOverVolt_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P12V0VoltOverVolt_C</i>
DTC	0x151005

10.2.9.6 P12V0VoltUnderVolt

Fehler Spannung P12V0 zu niedrig

P12V0VoltUnderVolt	
Fehlerindex	<i>EHI_P12V0VOLTUNDERVOLT</i> 144
Fehler Status	<i>ErrHndl_stP12V0VoltUnderVolt</i>
Defektbedingung	P12V0 Voltage Sensor: Undervoltage
Heilbedingung	P12V0 Voltage Sensor Fehlermeldung: Keine Unterspannung
Fehlerklasse	<i>ErrHndl_Class_P12V0VoltUnderVolt_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P12V0VoltUnderVolt_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P12V0VoltUnderVolt_C</i>
DTC	0x151006

10.2.9.7 P5V0VoltSctoGnd

Fehler Spannung P5V0 Kurzschluss nach Masse

P5V0VoltSctoGnd	
Fehlerindex	<i>EHI_P5V0VOLTSCTOGND</i> 148
Fehler Status	<i>ErrHndl_stP5V0VoltSctoGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltP5V0 < WsploHwAbs_uP5V0SCGnd_C</i>

Heilbedingung	<i>WsploHwAbs_uAdcVoltP5V0 >= WsploHwAbs_uP5V0SCGnd_C</i>
Fehlerklasse	<i>ErrHndl_Class_P5V0VoltSctoGnd_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P5V0VoltSctoGnd_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P5V0VoltSctoGnd_C</i>
DTC	0x151012

10.2.9.8 P5V0VoltSctoSup

Fehler Spannung P5V0 Kurzschluss zur Versorgung

P5V0VoltSctoSup	
Fehlerindex	<i>EHI_P5V0VOLTSCTOSUP</i> 149
Fehler Status	<i>ErrHndl_stP5V0VoltSctoSup</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltP5V0 > WsploHwAbs_uP5V0SCSup_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltP5V0 <= WsploHwAbs_uP5V0SCSup_C</i>
Fehlerklasse	<i>ErrHndl_Class_P5V0VoltSctoSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P5V0VoltSctoSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P5V0VoltSctoSup_C</i>
DTC	0x151013

10.2.9.9 P3V3SVoltSctoGnd

Fehler Spannung P3V3 Kurzschluss nach Masse

P3V3SVoltSctoGnd	
Fehlerindex	<i>EHI_P3V3SVOLTSCTOGND</i> 155
Fehler Status	<i>ErrHndl_stP3V3SVoltSctoGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltP3V3 < WsploHwAbs_uP3V3SCGnd_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltP3V3 >= WsploHwAbs_uP3V3SCGnd_C</i>

Softwaredokumentation

Fehlerklasse	<i>ErrHndl_Class_P3V3SVoltSctoGnd_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P3V3SVoltSctoGnd_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P3V3SVoltSctoGnd_C</i>
DTC	0x151027

10.2.9.10 P3V3SVoltCtoSup

Fehler Spannung P3V3 Kurzschluss zur Versorgung

P3V3SVoltCtoSup	
Fehlerindex	<i>EHI_P3V3SVOLTCTOSUP</i> 156
Fehler Status	<i>ErrHndl_stP3V3SVoltCtoSup</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltP3V3 > WsploHwAbs_uP3V3SCSup_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltP3V3 <= WsploHwAbs_uP3V3SCSup_C</i>
Fehlerklasse	<i>ErrHndl_Class_P3V3SVoltCtoSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_P3V3SVoltCtoSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_P3V3SVoltCtoSup_C</i>
DTC	0x151028

10.2.9.11 P12V0VoltSctoGnd

Fehler Spannung P12V0 Kurzschluss nach Masse

P12V0VoltSctoGnd	
Fehlerindex	<i>EHI_P12V0VOLTSCTOGND</i> 140
Fehler Status	<i>ErrHndl_stP12V0VoltSctoGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltP12V < WsploHwAbs_uP12V0SCGnd_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltP12V >= WsploHwAbs_uP12V0SCGnd_C</i>
Fehlerklasse	<i>ErrHndl_Class_P12V0VoltSctoGnd_C</i>

Defekt Entprellzeit	ErrHndl_tiDebDef_P12V0VoltSctoGnd_C
Heil Entprellzeit	ErrHndl_tiDebOk_P12V0VoltSctoGnd_C
DTC	0x151002

10.2.9.12 P12V0VoltSctoSup

Fehler Spannung P12V0 Kurzschluss zur Versorgung

P12V0VoltSctoSup	
Fehlerindex	EHI_P12V0VOLTSCTOSUP 141
Fehler Status	ErrHndl_stP12V0VoltSctoSup
Defektbedingung	WsploHwAbs_uAdcVoltP12V > WsploHwAbs_uP12V0SCSup_C
Heilbedingung	WsploHwAbs_uAdcVoltP12V <= WsploHwAbs_uP12V0SCSup_C
Fehlerklasse	ErrHndl_Class_P12V0VoltSctoSup_C
Defekt Entprellzeit	ErrHndl_tiDebDef_P12V0VoltSctoSup_C
Heil Entprellzeit	ErrHndl_tiDebOk_P12V0VoltSctoSup_C
DTC	0x151003

10.2.9.13 SBCTempScToGnd

Fehler Ausgangs-Spannung SBC Kurzschluss nach Masse

SBCTempScToGnd	
Fehlerindex	EHI_SBCTEMPSCTOGND 108
Fehler Status	ErrHndl_stSBCTempScToGnd
Defektbedingung	SBC Temperature Sensor_1 Shortcut to GND
Heilbedingung	SBC Temperature Sensor_1 Kein Shortcut to GND / No SC to GND
Fehlerklasse	ErrHndl_Class_SBCTempScToGnd_C
Defekt Entprellzeit	ErrHndl_tiDebDef_SBCTempScToGnd_C

Heil Entprellzeit	<i>ErrHndl_tIDebOk_SBCTempScToGnd_C</i>
DTC	0x150001

10.2.9.14 SBCTempScToSup

Fehler Ausgangs-Spannung SBC Kurzschluss zur Versorgung

SBCTempScToSup	
Fehlerindex	<i>EHI_SBCTEMPSCTOSUP_109</i>
Fehler Status	<i>ErrHndl_stSBCTempScToSup</i>
Defektbedingung	SBC Temperature Sensor_1 Shortcut to SUPPLY
Heilbedingung	SBC Temperature Sensor_1 Kein Shortcut to SUPPLY / No SC to SUP
Fehlerklasse	<i>ErrHndl_Class_SBCTempScToSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tIDebDef_SBCTempScToSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tIDebOk_SBCTempScToSup_C</i>
DTC	0x150002

10.2.9.15 B6TempScToSup

Fehler B6 Temperatursensor Kurzschluss zur Versorgung

B6TempScToSup	
Fehlerindex	<i>EHI_B6TEMPSCTOSUP_73</i>
Fehler Status	<i>ErrHndl_stB6TempScToSup</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempWpcPos > WsploHwAbs_uWPCTempHi_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempWpcPos <= WsploHwAbs_uWPCTempHi_C</i>
Fehlerklasse	<i>ErrHndl_Class_B6TempScToSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tIDebDef_B6TempScToSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tIDebOk_B6TempScToSup_C</i>

DTC	0x120009
-----	----------

10.2.9.16 B6TempScToGnd

Fehler B6 Temperatursensor Kurzschluss nach Masse

B6TempScToGnd	
Fehlerindex	<i>EHI_B6TEMPSCTOGND_72</i>
Fehler Status	<i>ErrHndl_stB6TempScToGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempWpcPos < WsploHwAbs_uWPCTempLo_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempWpcPos >= WsploHwAbs_uWPCTempLo_C</i>
Fehlerklasse	<i>ErrHndl_Class_B6TempScToGnd_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_B6TempScToGnd_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_B6TempScToGnd_C</i>
DTC	0x120008

10.2.9.17 RFTempScToSup

Fehler Gleichrichter Temperatursensor Kurzschluss zur Versorgung

RFTempScToSup	
Fehlerindex	<i>EHI_RFTEMPSCTOSUP_75</i>
Fehler Status	<i>ErrHndl_stRFTempScToSup</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempRectPos > WsploHwAbs_uRectTempHi_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempRectPos <= WsploHwAbs_uRectTempHi_C</i>
Fehlerklasse	<i>ErrHndl_Class_RFTempScToSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_RFTempScToSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_RFTempScToSup_C</i>
DTC	0x12000E

10.2.9.18 RFTempScToGnd

Fehler Gleichrichter Temperatursensor Kurzschluss nach Masse

RFTempScToGnd	
Fehlerindex	<i>EHI_RFTEMPSCTOGND</i> 74
Fehler Status	<i>ErrHndl_stRFTempScToGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempRectPos < WsploHwAbs_uRectTempLo_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempRectPos >= WsploHwAbs_uRectTempLo_C</i>
Fehlerklasse	<i>ErrHndl_Class_RFTempScToGnd_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_RFTempScToGnd_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_RFTempScToGnd_C</i>
DTC	0x12000D

10.2.9.19 PFCMFTempScToSup

Fehler PFC Temperatursensor Kurzschluss zur Versorgung

PFCMFTempScToSup	
Fehlerindex	<i>EHI_PFCMFTEMPSCTOSUP</i> 77
Fehler Status	<i>ErrHndl_stPFCMFTempScToSup</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempPfcPos > WsploHwAbs_uPFCTempHi_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempPfcPos <= WsploHwAbs_uPFCTempHi_C</i>
Fehlerklasse	<i>ErrHndl_Class_PFCMFTempScToSup_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_PFCMFTempScToSup_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_PFCMFTempScToSup_C</i>
DTC	0x120013

10.2.9.20 PFCMFTempScToGnd

Softwaredokumentation

Fehler PFC Temperatursensor Kurzschluss nach Masse

PFCMFTempScToGnd	
Fehlerindex	<i>EHI_PFCMFTEMPSCTOGND</i> 76
Fehler Status	<i>ErrHndl_stPFCMFTempScToGnd</i>
Defektbedingung	<i>WsploHwAbs_uAdcTempPfcPos < WsploHwAbs_uPFCTempLo_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcTempPfcPos >= WsploHwAbs_uPFCTempLo_C</i>
Fehlerklasse	<i>ErrHndl_Class_PFCMFTempScToGnd_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_PFCMFTempScToGnd_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_PFCMFTempScToGnd_C</i>
DTC	0x120012

10.2.9.21 IncorrectCodingResistorDetected

Fehler ungültiger Kodierwiderstand

IncorrectCodingResistorDetected	
Fehlerindex	<i>EHI_INCORRECTCODINGRESISTORDETECTED</i> 162
Fehler Status	<i>ErrHndl_stIncorrectCodingResistorDetected</i>
Defektbedingung	<i>WsploHwAbs_stCodeResErr == 1</i>
Heilbedingung	<i>WsploHwAbs_stCodeResErr == 0</i>
Fehlerklasse	<i>ErrHndl_Class_IncorrectCodingResistorDetected_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_IncorrectCodingResistorDetected_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_IncorrectCodingResistorDetected_C</i>
DTC	0x153022

10.2.9.22 OpenLoopCodingResistor

Softwaredokumentation

Fehler Kodierwiderstand nicht verbunden

OpenLoopCodingResistor	
Fehlerindex	<i>EHI_OPENLOOPCODINGRESISTOR</i> 164
Fehler Status	<i>ErrHndl_stOpenLoopCodingResistor</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltRNetExtPos >= WsploHwAbs_uCodeResOL_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltRNetExtPos < WsploHwAbs_uCodeResOL_C</i>
Fehlerklasse	<i>ErrHndl_Class_OpenLoopCodingResistor_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_OpenLoopCodingResistor_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_OpenLoopCodingResistor_C</i>
DTC	0x153024

10.2.9.23 ShortCircuitCodingResistor

Fehler Kodierwiderstand Kurzschluss nach Masse

ShortCircuitCodingResistor	
Fehlerindex	<i>EHI_SHORTCIRCUITCODINGRESISTOR</i> 163
Fehler Status	<i>ErrHndl_stShortCircuitCodingResistor</i>
Defektbedingung	<i>WsploHwAbs_uAdcVoltRNetExtPos <= WsploHwAbs_uCodeResSC_C</i>
Heilbedingung	<i>WsploHwAbs_uAdcVoltRNetExtPos > WsploHwAbs_uCodeResSC_C</i>
Fehlerklasse	<i>ErrHndl_Class_ShortCircuitCodingResistor_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_ShortCircuitCodingResistor_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_ShortCircuitCodingResistor_C</i>
DTC	0x153023

10.2.10 Initialisierung

tbd

11 Modul WspProtect

Autor: Michael Münch

Software-Struktur-Position: 30_Implementation/10_WSP/80_PROTEC

11.1 Allgemein

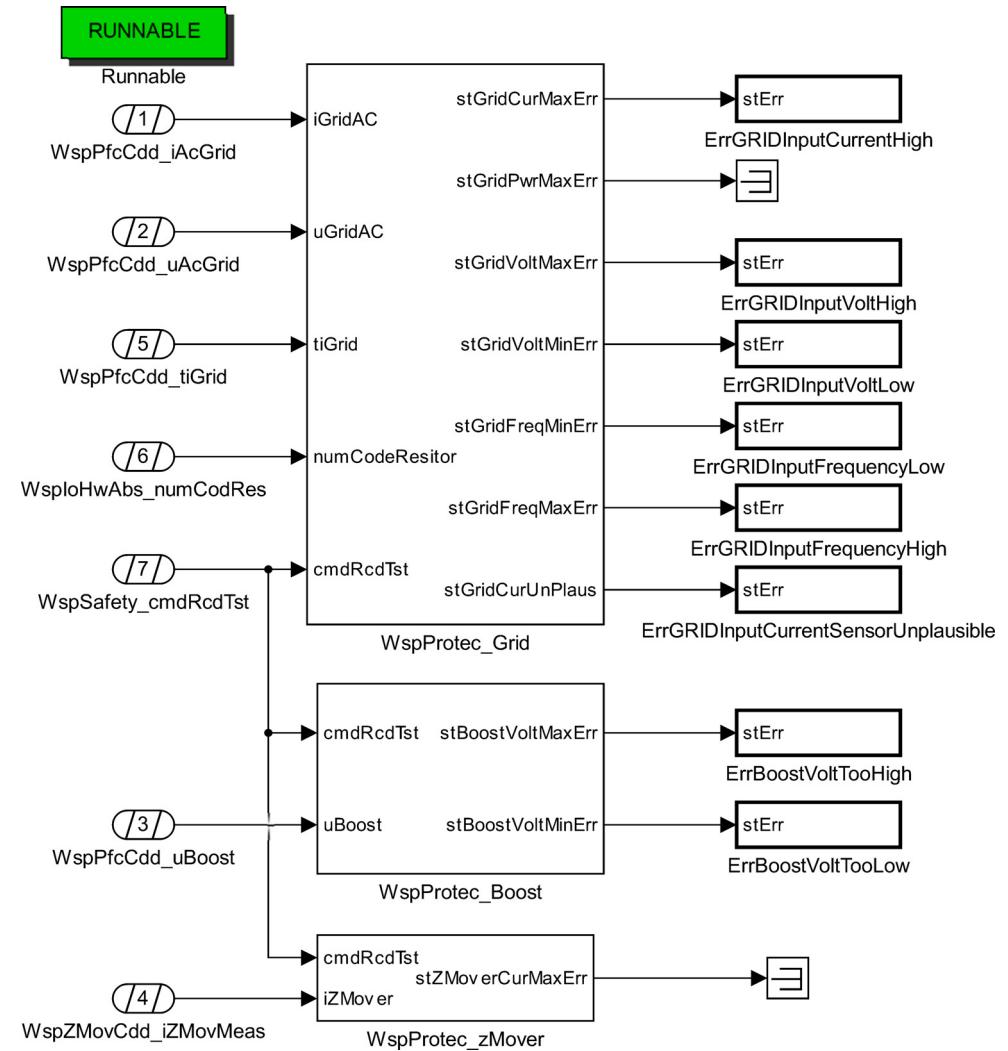
Das Modul *WspProtect* dient dem Bauteilschutz des PAD. Hier werden die in Frage kommenden Spannungen, Ströme und Temperaturen auf die Einhaltung definierter Grenzen überwacht.

Das Modul *WspProtect* enthält folgende Komponenten:

- *WspProtec_10ms* Protection im 10ms Raster zur Überwachung von Strömen und Spannungen
- *WspProtec_100ms* Protection im 100ms Raster zur Überwachung von Temperaturen

11.2 Komponente WspProtec_10ms

Dies ist die oberste Ebene der Komponente *WspProtec_10ms*. Hier werden die benötigten Autosarports eingelesen und an die entsprechenden Submodule (Blocks) weitergegeben. Die von den Modulen zurückgelieferten Fehlerstati werden den entsprechenden Fehlerpfaden zugeordnet. Im Block *WspProtec_Grid* wird der vom Stromnetz gelieferte Strom, die Netzspannung und die daraus resultierende Leistungsaufnahme überwacht. Im Block *WspProtec_Boost* wird die Boost- (PFC-) Spannung zur Speisung der B6 Brücke überwacht. Im Block *WspProtec_zMover* wird der Strom des Z - Movers überwacht.



11.2.1 Block WspProtec_Grid

Im Block *WspProtec_Grid* wird Eingangsseitig die Netzspannung sowie der Eingangsstrom und die daraus resultierende Eingangsleistung überwacht.

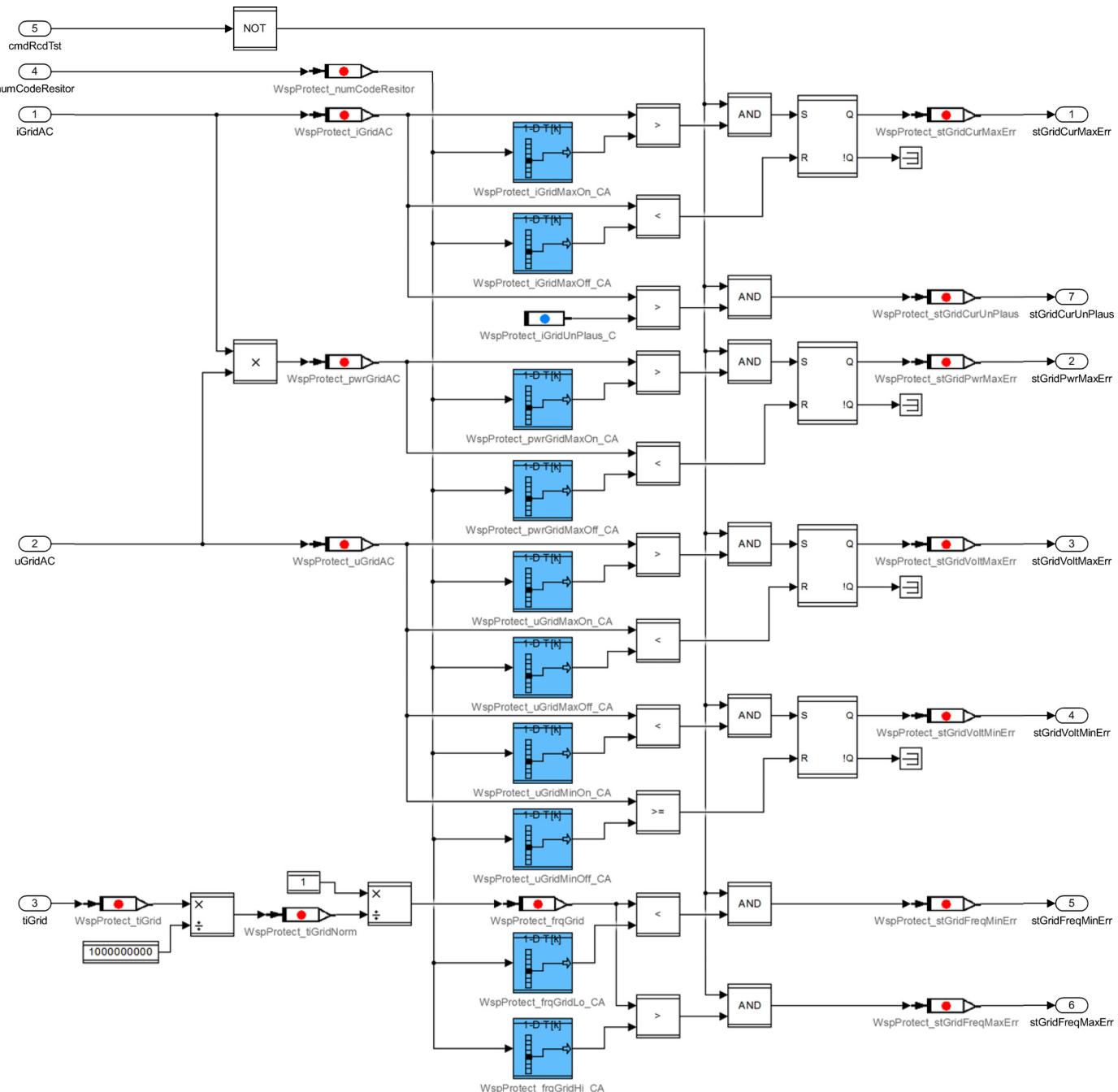
Die im nachfolgenden Benutzten Schwellen sind größtenteils als Arrays implementiert, wobei der jeweils selektierte Wert vom Index *WspProtect_numCodeResistor* des Codierungswiderstandes Abhängt.

Fehler werden jedoch nur gesetzt, wenn kein RCD Test aktiv ist (*cmdRcdTst = false*).

Übersteigt der Eingangsstrom *WspProtect_iGridAC* (*iGridAC*) die obere Schwelle *WspProtect_iGridMaxOn_CA*, wird der Fehlerstatus *WspProtect_stGridCurMaxErr* gesetzt. Unterschreitet der Eingangstrom die Schwelle *WspProtect_iGridMaxOff_CA*, wird der Fehlerstatus zurückgesetzt.

Es wird dann der Eingangsstrom *WspProtect_iGridAC* auf Plausibilität untersucht, indem dieser mit der Schwelle *WspProtect_iGridUnPlaus_C* verglichen wird. Beim überschreiten dieses Schwellwertes wird der Fehlerstatus *WspProtect_stGridCurUnPlaus* gesetzt.

Als nächstes wird die Eingangsleistung *WspProtect_pwrGridAC* überprüft. Diese berechnet sich als das Produkt aus Eingangsstrom *WspProtect_iGridAC* und Eingangsspannung *WspProtect_uGridAC*. Übersteigt die Eingangsleistung den Wert *WspProtect_pwrGridMaxOn_CA*, wird der Fehlerstatus *WspProtect_stGridPwrMaxErr* gesetzt. Unterschreitet sie die Schwelle *WspProtect_pwrGridMaxOff_CA*, wird dieser Fehlerstatus zurückgesetzt.



Softwaredokumentation

primove
true e-mobility

Dann wird die Eingangsspannung (Netzspannung) `WspProtect_uGridAC` betrachtet. Bei Überschreitung der Schwelle `WspProtect_uGridMaxOn_CA` wird der Fehlerstatus `WspProtect_stGridVoltMaxErr` gesetzt, bei Unterschreitung von `WspProtect_uGridMaxOff_CA` zurückgesetzt.

Unterschreitet die Netzspannung die Schwelle `WspProtect_uGridMinOn_CA`, wird der Fehlerstatus `WspProtect_stGridVoltMinErr` gesetzt, überschreite die Spannung den Wert `WspProtect_uGridMinOff_CA`, wird er wieder zurückgesetzt

Zuletzt wird noch die Netzfrequenz überprüft. Hierzu wird zunächst die vom PFC Modul gemessene Periodenzeit `tiGrid` (`WspProtect_tiGrid`) der Netzfrequenz von der Einheit ns auf sek. umgerechnet. Von der so erhaltenen normalisierten Periodenzeit `WspProtect_tiGridNorm` wird dann der Kehrwert genommen und ergibt somit die gemessene Netzfrequenz `WspProtect_frqGrid` in [Hz].

Unterschreitet nun die Netzfrequenz `WspProtect_frqGrid` die Schwelle `WspProtect_frqGridLo_CA`, wird der Fehlerstatus `WspProtect_stGridFreqMinErr` gesetzt. Überschreitet die Netzfrequenz die Schwelle `WspProtect_frqGridHi_CA`, wird der Fehlerstatus `WspProtect_stGridFreqMaxErr` gesetzt.

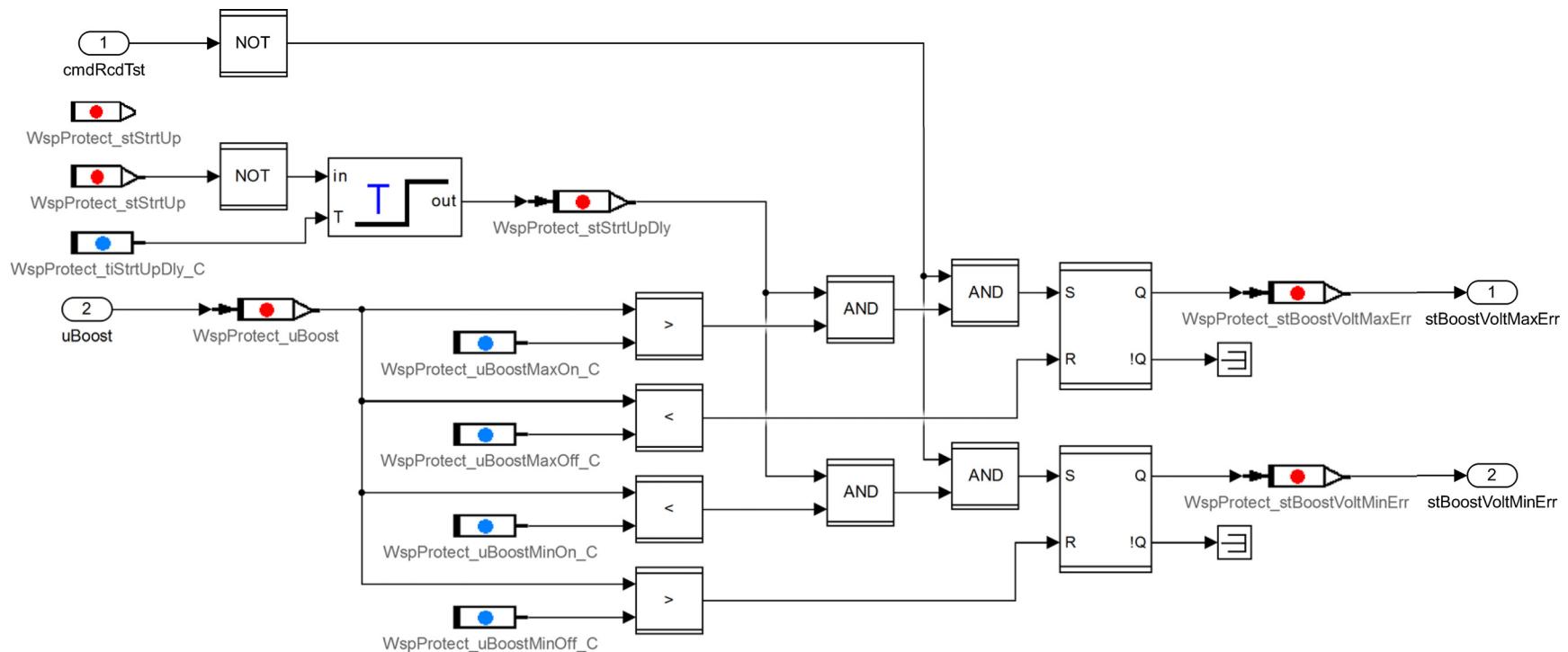
11.2.2 Block WspProtec_Boost

Hier wird die Boost-
(PFC Ausgangs-) Span-
nung

WspProtect_uBoost
(*uBoost*) überwacht.

Fehler werden jedoch
nur gesetzt, wenn kein
RCD Test aktiv ist
(*cmdRcdTst* = *false*).

Da nach einem Neustart
des PAD's der PFC et-
was Zeit braucht um die
Boostspannung hochzu-
fahren, wird die
Fehlererkennung erst
nach einer Delayzeit
WspProtect_tiStrtUpDly
_C freigegeben indem
nach dieser Zeit der
Freigabestatus *WspProtect_stStrtUpDly* gesetzt wird.



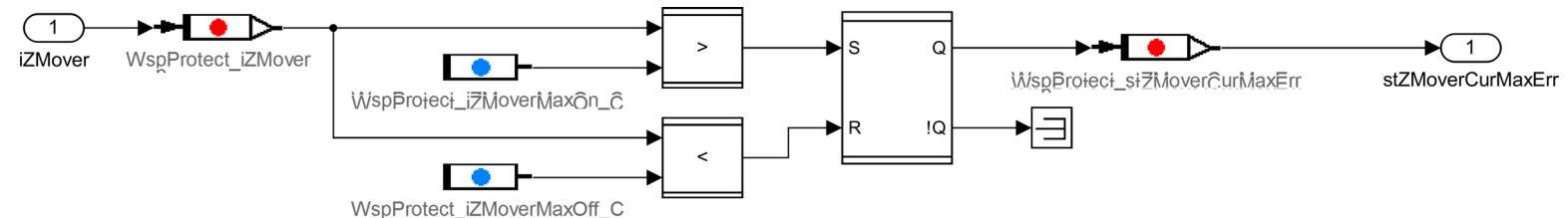
Überschreitet *uBoost* den Wert *WspProtect_uBoostMaxOn_C*, wird der Fehlerstatus *WspProtect_stBoostVoltMaxErr* gesetzt. Unterschreitet die Boostspannung den Wert *WspProtect_uBoostMaxOff_C*, wird er wieder zurückgesetzt.

Unterschreitet die Boostspannung den Wert *WspProtect_uBoostMinOn_C*, wird der Fehlerstatus *WspProtect_stBoostVoltMinErr* gesetzt und bei Überschreitung von *WspProtect_uBoostMinOff_C* wieder zurückgesetzt.

11.2.3 Block WspProtec_zMover

Im Block Wspprotect_zMover wird der Z-Mover Strom auf Übetstrom Überwacht.

Fehler werden jedoch nur gesetzt, wenn kein RCD Test aktiv ist (*cmdRcdTst = false*).



Überschreitet der Z-Mover Strom *WspProtect_iZMover* die Schwelle *WspProtect_iZMoverMaxOn_C*, wird der Fehlerstatus *WspProtect_stZMoverCurMaxErr* gesetzt. Unter- schreitet der Strom den Wert *WspProtect_iZMoverMaxOff_C*, wird der Fehlerstatus zurückgesetzt.

11.2.4 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Requiered Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

11.2.4.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
<i>rpAcCurrGrid</i>	RPort	SCI_Protec_AcCurrGrid_s32_T_if	Eingangsstrom Netztseitig
<i>rpAcVoltGrid</i>	RPort	SCI_Protec_AcVoltGrid_s32_TValue	Eingangsspannung Netztseitig
<i>rpBoostVolt</i>	RPort	SCI_Protec_BoostVolt_s32_T_if	PFC Spannung
<i>rpZMoverCurr</i>	RPort	ifZMoverCurrent	Z-Mover Strom

11.2.4.2 Ausgangsgrößen

keine	
-------	--

11.2.4.3 Interfaces

11.2.4.3.1 Interface SCI_Protec_AcCurrGrid_s32_T_if

Data Element	Data Type	Auflösung	Einheit	Bedeutung
<i>SCI_Protec_AcCurrGrid_s32_TValue</i>	sint32	0.001	V	Spannung hohe Auflösung

11.2.5 Messgrößen

Für Details bezüglich Datentypen und Auflösung der Messgrößen siehe WspAppConst.xlsx -> Tab: Measurements

<i>WspProtect_iGridAC</i>	AC Gitterstrom Wayside
<i>WspProtect_iZMover</i>	Ansteuerstrom Z-Mover
<i>WspProtect_pwrGridAC</i>	AC Gittersleistung Wayside
<i>WspProtect_stBoostVoltMaxErr</i>	Fehlerstatus Boost-Spannung zu hoch
<i>WspProtect_stBoostVoltMinErr</i>	Fehlerstatus Boost-Spannung zu niedrig
<i>WspProtect_stGridCurMaxErr</i>	Fehlerstatus Strom am Gitter zu hoch
<i>WspProtect_stGridPwrMaxErr</i>	Fehlerstatus Leistung am Gitter zu hoch
<i>WspProtect_stGridVoltMaxErr</i>	Fehlerstatus Spannung am Gitter zu hoch
<i>WspProtect_stGridVoltMinErr</i>	Fehlerstatus Spannung am Gitter zu niedrig
<i>WspProtect_stZMoverCurMaxErr</i>	Fehlerstatus z-Mover Strom zu hoch
<i>WspProtect_uBoost</i>	Boosterspannung Wayside
<i>WspProtect_uGridAC</i>	AC Gitterspannung Wayside
<i>WspProtect_numCodeResistor</i>	Codenummer Codierungswiderstand aus WspHwloAbs
<i>WspProtect_stGridFreqMaxErr</i>	Fehlerstatus Netzfrequenz zu hoch
<i>WspProtect_stGridFreqMinErr</i>	Fehlerstatus Netzfrequenz zu niedrig
<i>WspProtect_stStrtUp</i>	Startup Init Wert für StartupDelay
<i>WspProtect_stStrtUpDly</i>	Startup Status für StartupDelay (true = Startup ready)
<i>WspProtect_tiGrid</i>	Periodendauer Netz Eingangsspannung
<i>WspProtect_tiGridNorm</i>	Periodendauer Netz Eingangsspannung in Sekunden

11.2.6 Applikationswerte

Für Details bezüglich Datentypen und Auflösung der Applikationswerte siehe WspAppConst.xlsx -> Tab: ApplConst Skalar und ApplConst Curve

<i>WspProtect_iZMoverMaxOff_C</i>	Unteres Limit Maximal zulässiger Strom an z-Mover, Fehler setzen zurücksetzen
<i>WspProtect_iZMoverMaxOn_C</i>	Oberes Limit Maximal zulässiger Strom an z-Mover, Fehler setzen
<i>WspProtect_tiStrtUpDly_C</i>	Startup Verzögerungszeit für Diagnose
<i>WspProtect_uBoostMaxOff_C</i>	Unteres Limit Maximal zulässige Boostspannung, Fehler setzen zurücksetzen
<i>WspProtect_uBoostMaxOn_C</i>	Oberes Limit Maximal zulässige Boostspannung, Fehler setzen
<i>WspProtect_uBoostMinOff_C</i>	Oberes Limit Minimal zulässige Boostspannung, Fehler setzen zurücksetzen
<i>WspProtect_uBoostMinOn_C</i>	Unteres Limit Minimal zulässige Boostspannung, Fehler setzen
<i>WspProtect_iGridMaxOff_CA</i>	Unteres Limit Maximal zulässiger Gitterstrom, Fehler setzen zurücksetzen
<i>WspProtect_iGridMaxOn_CA</i>	Oberes Limit Maximal zulässiger Gitterstrom, Fehler setzen
<i>WspProtect_pwrGridMaxOff_CA</i>	Unteres Limit Maximal zulässige Gitterleistung, Fehler setzen zurücksetzen
<i>WspProtect_pwrGridMaxOn_CA</i>	Oberes Limit Maximal zulässige Gitterleistung, Fehler setzen
<i>WspProtect_uGridMaxOff_CA</i>	Unteres Limit Maximal zulässige Gitterspannung, Fehler setzen zurücksetzen
<i>WspProtect_uGridMaxOn_CA</i>	Oberes Limit Maximal zulässige Gitterspannung, Fehler setzen
<i>WspProtect_uGridMinOff_CA</i>	Oberes Limit Minimal zulässige Gitterspannung, Fehler setzen zurücksetzen
<i>WspProtect_uGridMinOn_CA</i>	Unteres Limit Minimal zulässige Gitterspannung, Fehler setzen
<i>WspProtect_fqGridHi_CA</i>	Schwelle Netzfrequenz zu hoch
<i>WspProtect_fqGridLo_CA</i>	Schwelle Netzfrequenz zu niedrig

11.2.7 Fehlerpfade

11.2.7.1 GRIDInputCurrentHigh

Fehler Eingangsstrom netzseitig zu hoch

GRIDInputCurrentHigh	
Fehlerindex	<i>EHI_GRIDINPUTCURRENTHIGH</i> 93
Fehler Status	<i>ErrHndl_stGRIDInputCurrentHigh</i>
Defektbedingung	<i>WspProtect_iGridAC > WspProtect_iGridMaxOn_C</i>
Heilbedingung	<i>WspProtect_iGridAC < WspProtect_iGridMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_GRIDInputCurrentHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_GRIDInputCurrentHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_GRIDInputCurrentHigh_C</i>
DTC	0x12200F

11.2.7.2 WPCInputPowerHigh

Fehler Eingangsleistung netzseitig zu hoch

WPCInputPowerHigh	
Fehlerindex	<i>EHI_WPCINPUTPOWERHIGH</i> 105
Fehler Status	<i>ErrHndl_stWPCInputPowerHigh</i>
Defektbedingung	<i>WspProtect_pwrGridAC > WspProtect_pwrGridMaxOn_C</i>
Heilbedingung	<i>WspProtect_pwrGridAC < WspProtect_pwrGridMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_WPCInputPowerHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_WPCInputPowerHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_WPCInputPowerHigh_C</i>
DTC	0x128000

11.2.7.3 GRIDInputVoltHigh

Fehler Eingangsspannung netzseitig zu hoch

GRIDInputVoltHigh	
Fehlerindex	<i>EHI_GRIDINPUTVOLTHIGH</i> 84
Fehler Status	<i>ErrHndl_stGRIDInputVoltHigh</i>
Defektbedingung	<i>WspProtect_uGridAC > WspProtect_uGridMaxOn_C</i>
Heilbedingung	<i>WspProtect_uGridAC < WspProtect_uGridMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_GRIDInputVoltHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_GRIDInputVoltHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_GRIDInputVoltHigh_C</i>
DTC	0x121004

11.2.7.4 GRIDInputVoltLow

Fehler Eingangsspannung netzseitig zu niedrig

GRIDInputVoltLow	
Fehlerindex	<i>EHI_GRIDINPUTVOLTLOW</i> 85
Fehler Status	<i>ErrHndl_stGRIDInputVoltLow</i>
Defektbedingung	<i>WspProtect_uGridAC < WspProtect_uGridMinOn_C</i>
Heilbedingung	<i>WspProtect_uGridAC >= WspProtect_uGridMinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_GRIDInputVoltLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_GRIDInputVoltLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_GRIDInputVoltLow_C</i>
DTC	0x121006

11.2.7.5 BoostVoltTooHigh

Softwaredokumentation

Fehler Boostspannung PFC zu hoch

BoostVoltTooHigh	
Fehlerindex	<i>EHI_BOOSTVOLTTOOHIGH</i> 83
Fehler Status	<i>ErrHndl_stBoostVoltTooHigh</i>
Defektbedingung	<i>WspProtect_uBoost > WspProtect_uBoostMaxOn_C</i>
Heilbedingung	<i>WspProtect_uBoost < WspProtect_uBoostMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_BoostVoltTooHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_BoostVoltTooHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_BoostVoltTooHigh_C</i>
DTC	0x121001

11.2.7.6 BoostVoltTooLow

Fehler Boostspannung PFC zu niedrig

BoostVoltTooLow	
Fehlerindex	<i>EHI_BOOSTVOLTTOOLOW</i> 82
Fehler Status	<i>ErrHndl_stBoostVoltTooLow</i>
Defektbedingung	<i>WspProtect_uBoost < WspProtect_uBoostMinOn_C</i>
Heilbedingung	<i>WspProtect_uBoost > WspProtect_uBoostMinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_BoostVoltTooLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_BoostVoltTooLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_BoostVoltTooLow_C</i>
DTC	0x121000

11.2.8 Initialisierung

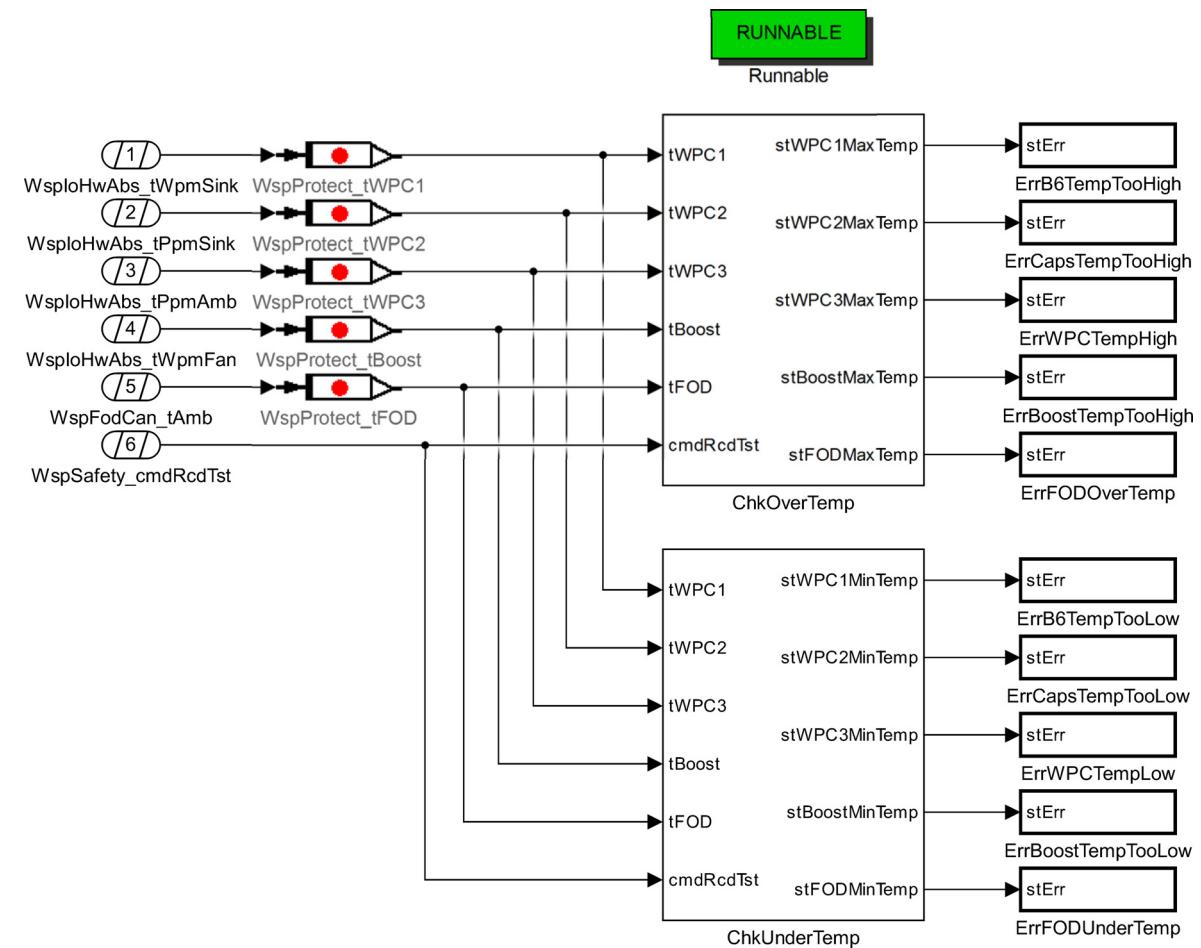
tbd

11.3 Komponente WspProtec_100ms

In der Komponente *WspProtect_100ms* werden die Temperaturen der einzelnen Bauteikomponenten auf Über- und Untertemperatur überwacht. Dies stellt die Oberste Ebene dar. Hier werden die Autosarports der benötigten Messgrößen eingelesen und an den entsprechenden Block übergeben.

Die von den Block gelieferten Fehlerstati werden den zugehörigen Fehlerpfaden übergeben.

Im Block *ChkOverTemp* werden die HW Komponenten auf eine für den Betrieb zu hohe und im Block *ChkUnderTemp* auf eine zu niedrige Temperatur überwacht.



11.3.1 Block ChkOverTemp

Hier werden die zur Verfügung stehenden Temperaturwerte auf eine für die entsprechenden Bauteile schädliche Übertemperatur überwacht.

Fehler werden jedoch nur gesetzt, wenn kein RCD Test aktiv ist (*cmdRcdTst = false*).

Liegt die Temperatur *tWPC1* (*WspProtect_tWPC1*) des Kühlkörpers an der B6 Brücke oberhalb der Schwelle

WspProtect_tWPC1MaxOn_C, wird der Fehlerstatus

WspProtect_stWPC1MaxTempErr gesetzt. Liegt diese Temperatur unterhalb des Wertes *WspProtect_tWPC1MaxOff_C*, wird der Fehlerstatus zurückgesetzt.

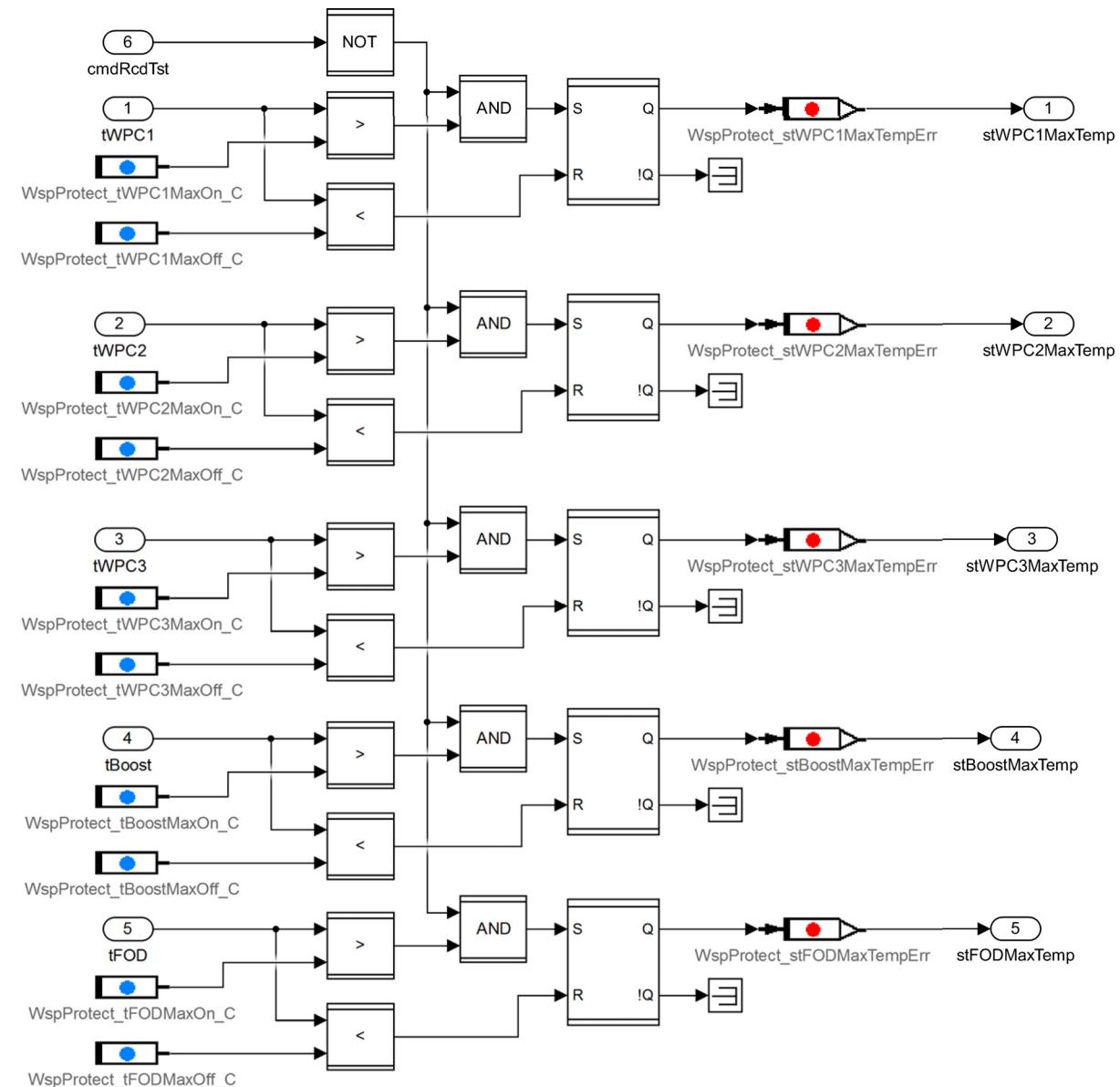
Entsprechend wird für die Kühlkörpertemperatur des Gleichrichters / PFC verfahren. Ist hier die Temperatur *tWPC2* (*WspProtect_tWPC2*) größer als *WspProtect_tWPC2MaxOn_C*, wird der Fehlerstatus

WspProtect_stWPC2MaxTempErr gesetzt. Liegt diese Temperatur unter *WspProtect_tWPC2MaxOff_C* wird der Fehlerstatus zurückgesetzt.

Für die Umgebungstemperatur *tWPC3* (*WspProtect_tWPC3*) des Gleichrichters wird geprüft, ob diese über *WspProtect_tWPC3MaxOn_C* liegt. In diesem Fall wird der Fehlerstatus *WspProtect_stWPC3MaxTempErr* gesetzt. Liegt die Temperatur unter *WspProtect_tWPC3MaxOff_C*, wird der Fehlerstatus zurückgesetzt.

Ebenso wird für die Lüftertemperatur *tBoost* (*WspProtect_tBoost*) geprüft, ob diese über *WspProtect_tBoostMaxOn_C* liegt. Dann wird der Fehlerstatus *WspProtect_stBoostMaxTempErr* gesetzt. Liegt sie unter *WspProtect_tBoostMaxOff_C*. wird der Fehlerstatus zurückgesetzt.

Schließlich wird noch der Temperatur *tFOD* (*WspProtect_tFOD*) der FOD Platine auf Übertemperatur überprüft. Übersteigt diese die Schwelle *WspProtect_tFODMaxOn_C*, wird der Fehlerstatus *WspProtect_stFODMaxTempErr* gesetzt. Liegt diese Temperatur unterhalb von *WspProtect_tFODMaxOff_C*, wird der Fehlerstatus zurückgesetzt.



11.3.2 Block ChkUnderTemp

Im Block *ChkUnderTemp* werden die zur Verfügung stehenden Temperaturwerte auf eine für den Betrieb notwendige Mindesttemperatur überwacht.

Fehler werden jedoch nur gesetzt, wenn kein RCD Test aktiv ist (*cmdRcdTst* = *false*).

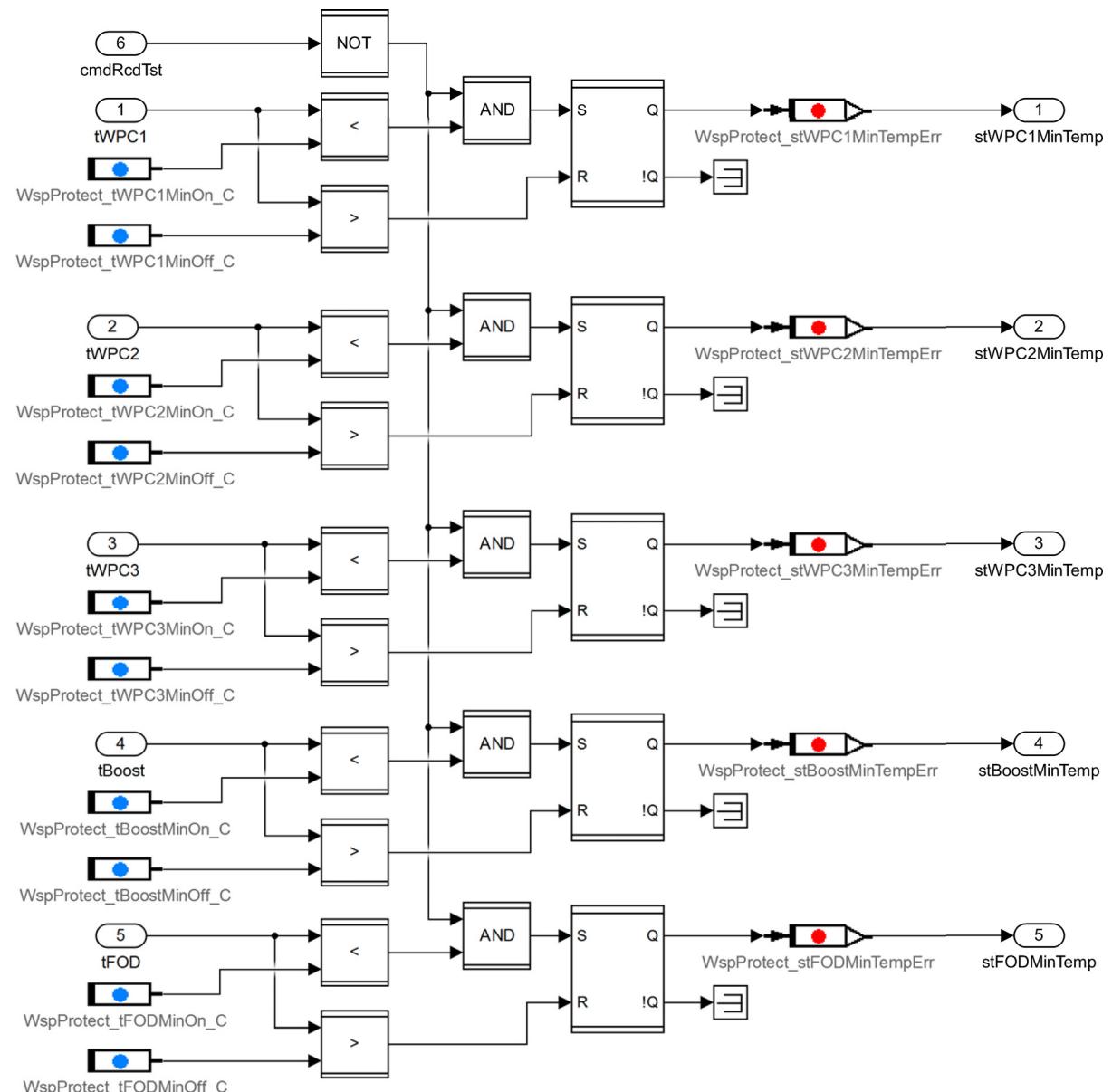
Liegt die Temperatur *tWPC1* (*WspProtect_tWPC1*) des Kühlkörpers an der B6 Brücke unthalb der Schwelle *WspProtect_tWPC1MinOn_C*, wird der Fehlerstatus *WspProtect_stWPC1MinTempErr* gesetzt. Liegt diese Temperatur oberhalb des Wertes *WspProtect_tWPC1MinOff_C*, wird der Fehlerstatus zurückgesetzt.

Entsprechend wird für die Kühlkörpertemperatur des Gleichrichters / PFC verfahren. Ist hier die Temperatur *tPWC2* (*WspProtect_tPWC2*) kleiner als *WspProtect_tWPC2MinOn_C*, wird der Fehlerstatus *WspProtect_stWPC2MinTempErr* gesetzt. Liegt diese Temperatur über *WspProtect_tWPC2MinOff_C* wird der Fehlerstatus zurückgesetzt.

Für die Umgebungstemperatur *tPWC3* (*WspProtect_tPWC3*) des Gleichrichters wird geprüft, ob diese unter *WspProtect_tWPC3MinOn_C* liegt. In diesem Fall wird der Fehlerstatus *WspProtect_stWPC3MinTempErr* gesetzt. Liegt die Temperatur über *WspProtect_tWPC3MinOff_C*, wird der Fehlerstatus zurückgesetzt.

Ebenso wird für die Lüftertemperatur *tBoost* (*WspProtect_tBoost*) geprüft, ob diese unter *WspProtect_tBoostMinOn_C* liegt. Dann wird der Fehlerstatus *WspProtect_stBoostMinTempErr* gesetzt. Liegt sie über *WspProtect_tBoostMinOff_C*, wird der Fehlerstatus zurückgesetzt.

Schließlich wird noch die Temperatur *tFOD* (*WspProtect_tFOD*) der FOD Platine auf Untertemperatur überprüft. Unterschreitet diese die Schwelle *WspProtect_tFODMinOn_C*, wird der Fehlerstatus *WspProtect_stFODMinTempErr* gesetzt. Liegt diese Temperatur oberhalb von *WspProtect_tFODMinOff_C*, wird der Fehlerstatus zurückgesetzt.



11.3.3 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Requiered Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

11.3.3.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
WpmSinkTemp	RPort	TempSftyC	Temperatur Kühlkörper B6 Brücke
PpmSinkTemp	RPort	TempSftyC	Temperatur Kühlkörper Gleichrichter / PFC
PpmAmbTemp	RPort	TempSftyC	Umgebungstemperatur Gleichrichter / PFC
WpmFanTemp	RPort	TempSftyC	Temperatur Lüfter
FodCanAmbTemp	RPort	TempSftyC	Temperatur FOD Platine

11.3.3.2 Ausgangsgrößen

keine	
-------	--

11.3.3.3 Interfaces

11.3.3.3.1 Interface TempSftyC

Data Element	Data Type	Auflösung	Einheit	Bedeutung
TempCVal_s16	sint16	0.1	°C	Temperaturwert

11.3.4 Messgrößen

WspProtect_frqGrid	
<i>WspProtect_stBoostMaxTempErr</i>	Fehlerstatus Temperatur an Booster zu hoch
<i>WspProtect_stBoostMinTempErr</i>	Fehlerstatus Temperatur an Booster zu niedrig
<i>WspProtect_stFODMaxTempErr</i>	Fehlerstatus Temperatur am FOD zu hoch
<i>WspProtect_stFODMinTempErr</i>	Fehlerstatus Temperatur am FOD zu niedrig
<i>WspProtect_stWPC1MaxTempErr</i>	Fehlerstatus Temperatur an WPC1 zu hoch
<i>WspProtect_stWPC1MinTempErr</i>	Fehlerstatus Temperatur an WPC1 (B6 Brücke) zu niedrig
<i>WspProtect_stWPC2MaxTempErr</i>	Fehlerstatus Temperatur an WPC2 zu hoch
<i>WspProtect_stWPC2MinTempErr</i>	Fehlerstatus Temperatur an WPC2 zu niedrig
<i>WspProtect_stWPC3MaxTempErr</i>	Fehlerstatus Temperatur an WPC3 zu hoch
<i>WspProtect_stWPC3MinTempErr</i>	Fehlerstatus Temperatur an WPC3 zu niedrig
<i>WspProtect_tBoost</i>	Eingangswert Temperatur an Booster
<i>WspProtect_tFOD</i>	Eingangswert Temperatur an FOD Platine vom FOD CAN
<i>WspProtect_tWPC1</i>	Eingangswert Temperatur an WPC1 (B6 Brücke)
<i>WspProtect_tWPC2</i>	Eingangswert Temperatur an WPC2
<i>WspProtect_tWPC3</i>	Eingangswert Temperatur an WPC3

11.3.5 Applikationswerte

<code>WspProtect_tBoostMaxOff_C</code>	Oberes Limit Maximal zulässige Temperatur an Booster Fehler zurücksetzen
<code>WspProtect_tBoostMaxOn_C</code>	Oberes Limit Maximal zulässige Temperatur an Booster, Fehler setzen
<code>WspProtect_tBoostMinOff_C</code>	Unteres Limit Minimal zulässige Temperatur an Booster, Fehler zurücksetzen
<code>WspProtect_tBoostMinOn_C</code>	Unteres Limit Minimal zulässige Temperatur an Booster, Fehler setzen
<code>WspProtect_tFODMaxOff_C</code>	Oberes Limit Maximal zulässige Temperatur an FOD Fehler zurücksetzen
<code>WspProtect_tFODMaxOn_C</code>	Oberes Limit Maximal zulässige Temperatur an FOD Fehler setzen
<code>WspProtect_tFODMinOff_C</code>	Unteres Limit Minimal zulässige Temperatur an FOD Fehler zurücksetzen
<code>WspProtect_tFODMinOn_C</code>	Unteres Limit Minimal zulässige Temperatur an FOD Fehler setzen
<code>WspProtect_tWPC1MaxOff_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC1, Fehler zurücksetzen
<code>WspProtect_tWPC1MaxOn_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC1, Fehler setzen
<code>WspProtect_tWPC2MaxOff_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC2 Fehler zurücksetzen
<code>WspProtect_tWPC2MaxOn_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC2, Fehler setzen
<code>WspProtect_tWPC3MaxOff_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC3 Fehler zurücksetzen
<code>WspProtect_tWPC3MaxOn_C</code>	Oberes Limit Maximal zulässige Temperatur an WPC3, Fehler setzen
<code>WspProtect_tWPC1MinOff_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC1, Fehler zurücksetzen
<code>WspProtect_tWPC1MinOn_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC1, Fehler setzen
<code>WspProtect_tWPC2MinOff_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC2, Fehler zurücksetzen
<code>WspProtect_tWPC2MinOn_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC2, Fehler setzen
<code>WspProtect_tWPC3MinOff_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC3, Fehler zurücksetzen
<code>WspProtect_tWPC3MinOn_C</code>	Unteres Limit Minimal zulässige Temperatur an WPC3, Fehler setzen

11.3.6 Fehlerpfade

11.3.6.1 B6TempTooHigh

Fehler Kühlkörpertemperatur B6 Brücke zu hoch

B6TempTooHigh	
Fehlerindex	<i>EHI_B6TEMPTOOHIGH</i> 69
Fehler Status	<i>ErrHndl_stB6TempTooHigh</i>
Defektbedingung	<i>WspProtect_tWPC1 > WspProtect_tWPC1MaxOn_C</i>
Heilbedingung	<i>WspProtect_tWPC1 < WspProtect_tWPC1MaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_B6TempTooHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_B6TempTooHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_B6TempTooHigh_C</i>
DTC	0x120005

11.3.6.2 CapsTempTooHigh

Fehler Kühlkörpertemperatur Gleichrichter zu hoch

CapsTempTooHigh	
Fehlerindex	<i>EHI_CAPSTEMPTOOHIGH</i> 71
Fehler Status	<i>ErrHndl_stCapsTempTooHigh</i>
Defektbedingung	<i>WspProtect_tWPC2 > WspProtect_tWPC2MaxOn_C</i>
Heilbedingung	<i>WspProtect_tWPC2 < WspProtect_tWPC2MaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_CapsTempTooHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_CapsTempTooHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_CapsTempTooHigh_C</i>
DTC	0x120007

11.3.6.3 WPCTempHigh

Fehler Umgebungstemperatur Gleichrichter zu hoch

WPCTempHigh	
Fehlerindex	<i>EHI_WPCTEMPHIGH</i> 80
Fehler Status	<i>ErrHndl_stWPCTempHigh</i>
Defektbedingung	<i>WspProtect_tWPC3 > WspProtect_tWPC3MaxOn_C</i>
Heilbedingung	<i>WspProtect_tWPC3 < WspProtect_tWPC3MaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_WPCTempHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_WPCTempHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_WPCTempHigh_C</i>
DTC	0x12001C

11.3.6.4 BoostTempTooHigh

Fehler Lüftertemperatur zu hoch

BoostTempTooHigh	
Fehlerindex	<i>EHI_BOOSTTEMPTOOHIGH</i> 67
Fehler Status	<i>ErrHndl_stBoostTempTooHigh</i>
Defektbedingung	<i>WspProtect_tBoost > WspProtect_tBoostMaxOn_C</i>
Heilbedingung	<i>WspProtect_tBoost < WspProtect_tBoostMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_BoostTempTooHigh_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_BoostTempTooHigh_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_BoostTempTooHigh_C</i>
DTC	0x120001

11.3.6.5 FODOverTemp

Fehler Temperatur FOD Platine zu hoch

FODOverTemp	
Fehlerindex	<i>EHI_FODOVERTEMP_6</i>
Fehler Status	<i>ErrHndl_stFODOverTemp</i>
Defektbedingung	<i>WspProtect_tFOD > WspProtect_tFODMaxOn_C</i>
Heilbedingung	<i>WspProtect_tFOD < WspProtect_tFODMaxOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_FODOverTemp_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_FODOverTemp_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_FODOverTemp_C</i>
DTC	0x100005

11.3.6.6 B6TempTooLow

Fehler Kühlkörpertemperatur B6 Brücke zu niedrig

B6TempTooLow	
Fehlerindex	<i>EHI_B6TEMPTOOLOW_68</i>
Fehler Status	<i>ErrHndl_stB6TempTooLow</i>
Defektbedingung	<i>WspProtect_tWPC1 < WspProtect_tWPC1MinOn_C</i>
Heilbedingung	<i>WspProtect_tWPC1 > WspProtect_tWPC1MinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_B6TempTooLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_B6TempTooLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_B6TempTooLow_C</i>
DTC	0x120004

11.3.6.7 CapsTempTooLow

Softwaredokumentation

Fehler Kühlkörpertemperatur Gleichrichter zu niedrig

CapsTempTooLow	
Fehlerindex	<i>EHI_CAPSTEMPTOOLOW 70</i>
Fehler Status	<i>ErrHndl_stCapsTempTooLow</i>
Defektbedingung	<i>WspProtect_tWPC2 < WspProtect_tWPC2MinOn_C</i>
Heilbedingung	<i>WspProtect_tWPC2 > WspProtect_tWPC2MinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_CapsTempTooLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_CapsTempTooLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_CapsTempTooLow_C</i>
DTC	0x120006

11.3.6.8 WPCTempLow

Fehler Umgebungstemperatur Gleichrichter zu niedrig

WPCTempLow	
Fehlerindex	<i>EHI_WPCTEMPLOW 81</i>
Fehler Status	<i>ErrHndl_stWPCTempLow</i>
Defektbedingung	<i>WspProtect_tWPC3 < WspProtect_tWPC3MinOn_C</i>
Heilbedingung	<i>WspProtect_tWPC3 > WspProtect_tWPC3MinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_WPCTempLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_WPCTempLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_WPCTempLow_C</i>
DTC	0x12001D

11.3.6.9 BoostTempTooLow

Softwaredokumentation

Fehler Lüftertemperatur zu niedrig

BoostTempTooLow	
Fehlerindex	<i>EHI_BOOSTTEMPTOOLOW</i> 66
Fehler Status	<i>ErrHndl_stBoostTempTooLow</i>
Defektbedingung	<i>WspProtect_tBoost < WspProtect_tBoostMinOn_C</i>
Heilbedingung	<i>WspProtect_tBoost > WspProtect_tBoostMinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_BoostTempTooLow_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_BoostTempTooLow_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_BoostTempTooLow_C</i>
DTC	0x120000

11.3.6.10 FODUnderTemp

Fehler Temperatur FOD Platine zu niedrig

FODUnderTemp	
Fehlerindex	<i>EHI_FODUNDERTEMP</i> 7
Fehler Status	<i>ErrHndl_stFODUnderTemp</i>
Defektbedingung	<i>WspProtect_tFOD < WspProtect_tFODMinOn_C</i>
Heilbedingung	<i>WspProtect_tFOD < WspProtect_tFODMinOff_C</i>
Fehlerklasse	<i>ErrHndl_Class_FODUnderTemp_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_FODUnderTemp_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_FODUnderTemp_C</i>
DTC	0x100006

11.3.7 Initialisierung

tbd

12 Modul WspSfty (Wayside Safety)

Autor: Michael Münch

Software-Struktur-Position: 30_Implementation/10_WSP/90_SAFETY

12.1 Allgemein

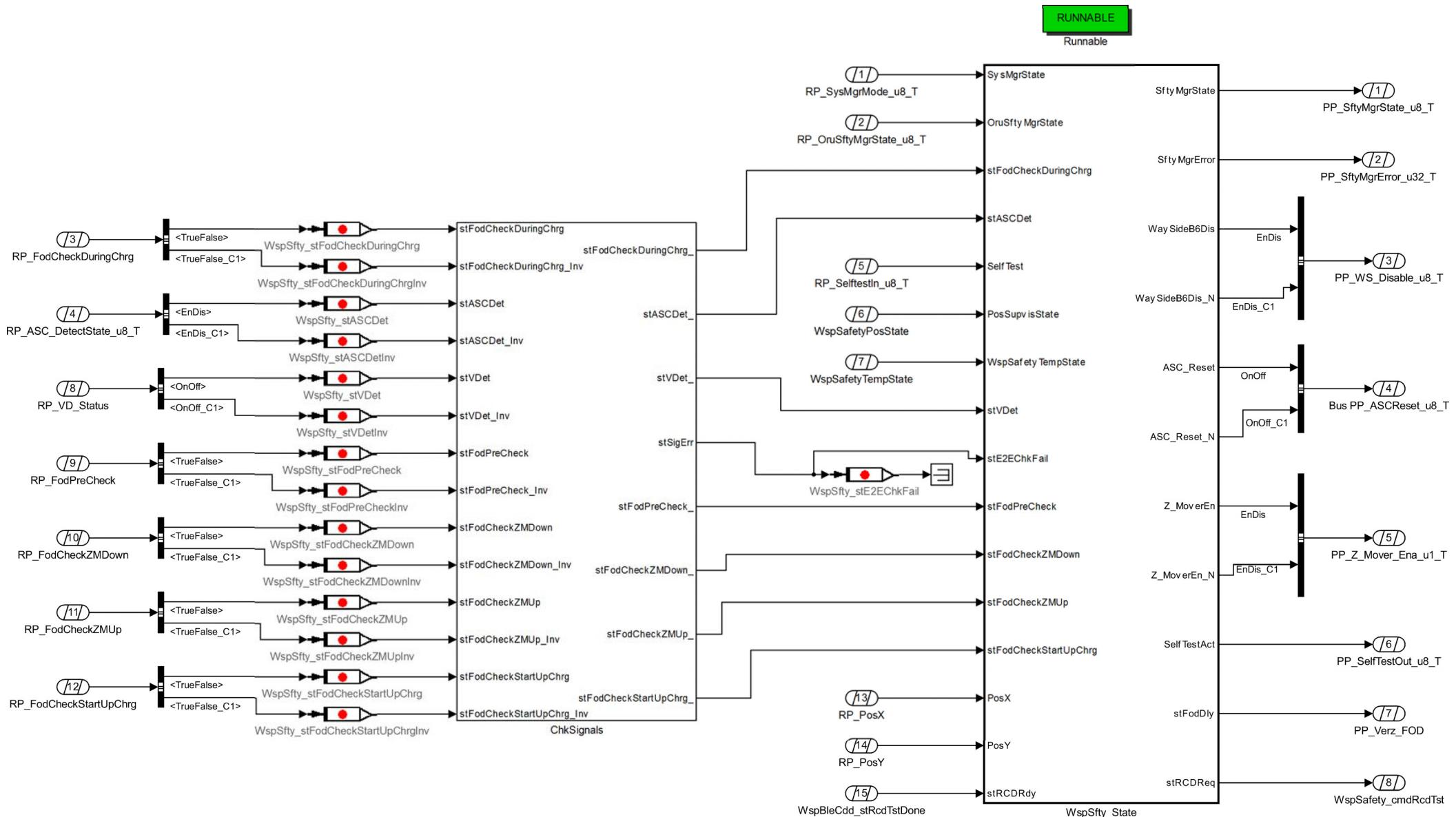
Das Modul *WspSfty* implementiert den Wayside Safety Manager. Im Wayside Safety Manager werden die für ASIL 2 notwendigen Überwachungen durchgeführt. Der Wayside Safety Statemanger stellt, in Abhängigkeit der Überwachungsergebnisse, dann den Safety Status für den Wayside System Statemanager und den Oru Safetymanager zur Verfügung.

Das Modul *WspSfty* enthält folgende Komponenten:

- *WspSfty_State* - Der Wayside Safety State Manager
- *WspSfty_Pinch* - Die Pinch Protektion (Einklemmschutz)
- *WspSfty_Calc* - Die Überwachung der Pad Gehäusetemperatur und des Oru Positionssenders

Softwaredokumentation

12.2 Komponente WspSfty_StateTL (Wayside Safety Statemanager)



Das Subsystem **WspSfty_StateTL** stellt im wesentlichen die Autosar IO - Schnittstellen zur Verfügung (siehe Schnittstellen). Diese werden an das Subsystem **WspSfty_State** weitergegeben bzw. aus diesem ausgelesen. Darüberhinaus wird hier im Block **ChkSignals** die Siignalqualifikations der hierfür definierten Autosar Eingangssignale durchgeführt.

12.2.1 Block ChkSignals

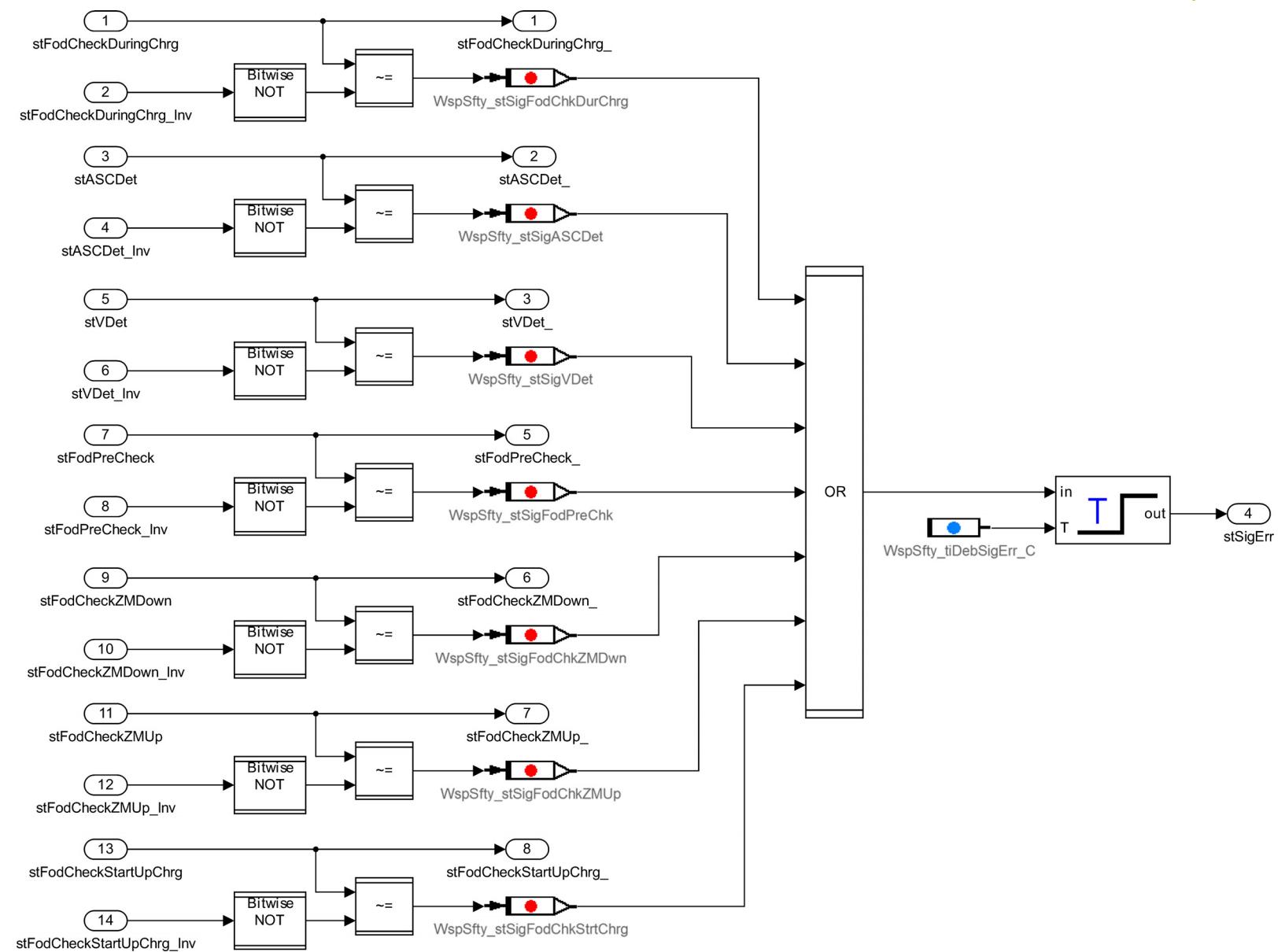
Hier wird die Signalqualifikation für Definierte Autosar- Eingangssignale durchgeführt.

Die entsprechenden Autosar Eingangssports enthalten hierbei immer den eigentlichen Signalwert als auch den entsprechenden Bitinvertierten Wert, z.B. *stFodCheckDuringChrg* und *stFodCheckDuringChrgInv*.

Es wird nun zum Überprüfen der korrekten Signalübermittlung über die Autosar RTE der jeweils der Bitinvertierte Wert wieder Bitinvertiert und mit dem Originalwert verglichen.

Sind beide Werte gleich, wird der Signal-Fehlerstatus (z.B. *WspSfty_stSigFodChkDurChrg*) auf false (= Ok) gesetzt, bei unterschiedlichen Wertem auf true (= Signalfehler).

Tritt nun in mindestens einem dieser Signale ein solcher Fehler für länger als die Debouncezeit *WspSfty_tiDebSigErr_C* auf, wird der Blockausgang *stSigErr* auf true gesetzt.



Softwaredokumentation

12.2.2 WspSfty_State

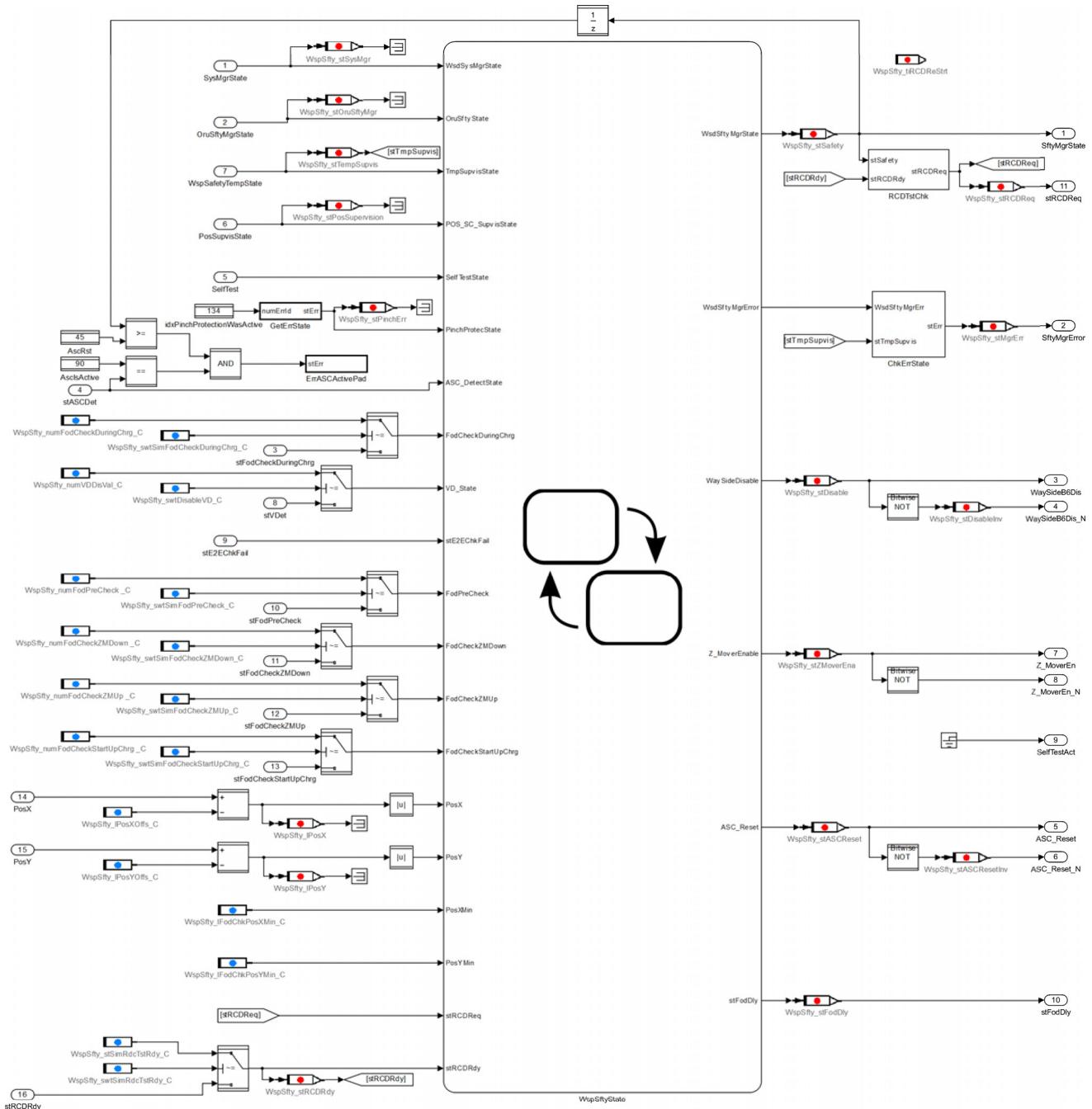
In diesem Block ist die Wsp Safety State Machine im Stafeflow Block *WspSftyState* und eine letztendliche Fehlerbereitstellung im Block *ChkErrState* implementiert. Bitcodierung Siehe "Block ChkErrState" auf Seite 213.

Im Block *RCDTstChk* befindet sich die Aktivierungslogik für den RDC Selbsttest.

Darüber hinaus wird hier noch der Fehlerspeichereintrag *ASCAactivePad* gesetzt, falls der von der HWIo kommende Status *stASCDet* den Wert 90 (AscActive) hat. jedoch nur, wenn der aktuelle Status *WspSfty_stSafety* des SafetyState Managers mindestens den Status 45 (AscReset_Impuls) aufweist. Dies ist notwendig, da im initialen Fall, also bevor die ASC Logik geresetet wurde die HW einen ausgelösten ASC meldet.

Der eigentliche State der Wayside Safety Statemachine wird über die Messvariable *WspSfty_stSafety*, welche dem Ausgangsport *SftyMgrState* zugewiesen wird, repräsentiert.

Folgende Stati sind hierbei definiert:



Softwaredokumentation

State Index	WspSfty_stSafety (Wayside Safety State)
0	Startup
10	Standby
20	Idle
21	FodChkMinPos
22	FodPreChk
23	FodChkZMoverDwn
24	WaitPos
25	WaitReqChrg
26	FodChkZMoverUp
27	MetalObjectSearch
30	VehInChargePosition
35	Z_MoverElevate
40	SetupCharging
45	ASC_Resetimpuls
48	Sfty_WCP_Enable
50	Charging
60	EndOfCharging
65	Pinch Detected
70	WaitSftyErrorNotif
80	Shutdown

Der Status [WspSfty_stPinchErr](#) der Pinchprotection entspricht dem entprellten Fehlerstatus des Fehlerpfads [PinchProtectionWasActive](#) (Id 134).

Die Stati der einzelnen FOD Checks können mit den Applikationsschaltern [WspSfty_swtSimFodPreCheck_C](#), [WspSfty_swtSimFodCheckZMDown_C](#), [WspSfty_swtSimFodCheckZMUp_C](#), [WspSfty_swtSimFodCheckStartUpChrg_C](#) und [WspSfty_swtSimFodCheckDuringChrg_C](#) und den applizierbaren Werten [WspSfty_numFodPreCheck_C](#) ... [WspSfty_numFodCheckDuringChrg_C](#) umgeschaltet werden, indem die Schalter jeweils auf true gesetzt werden.

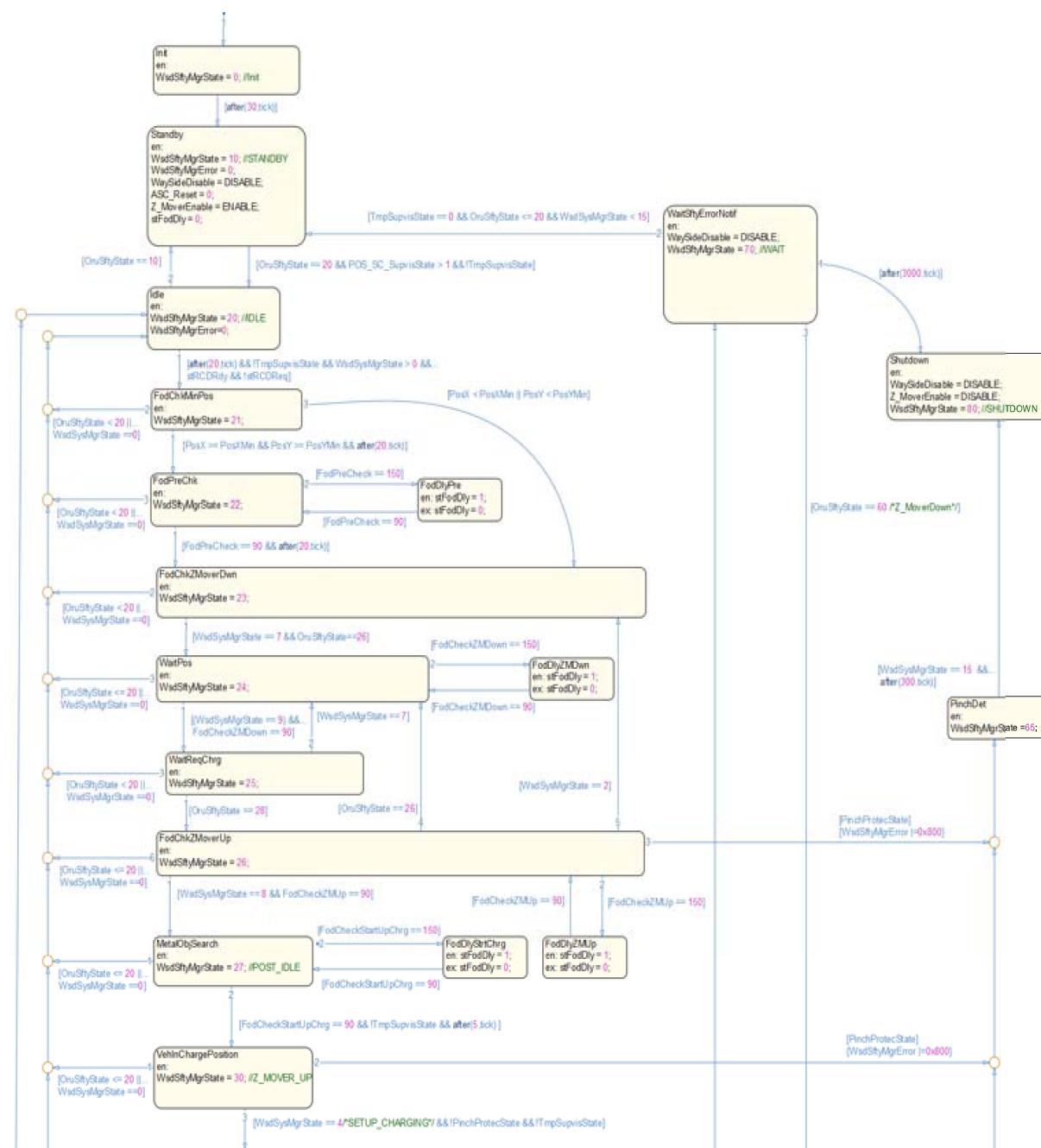
Softwaredokumentation

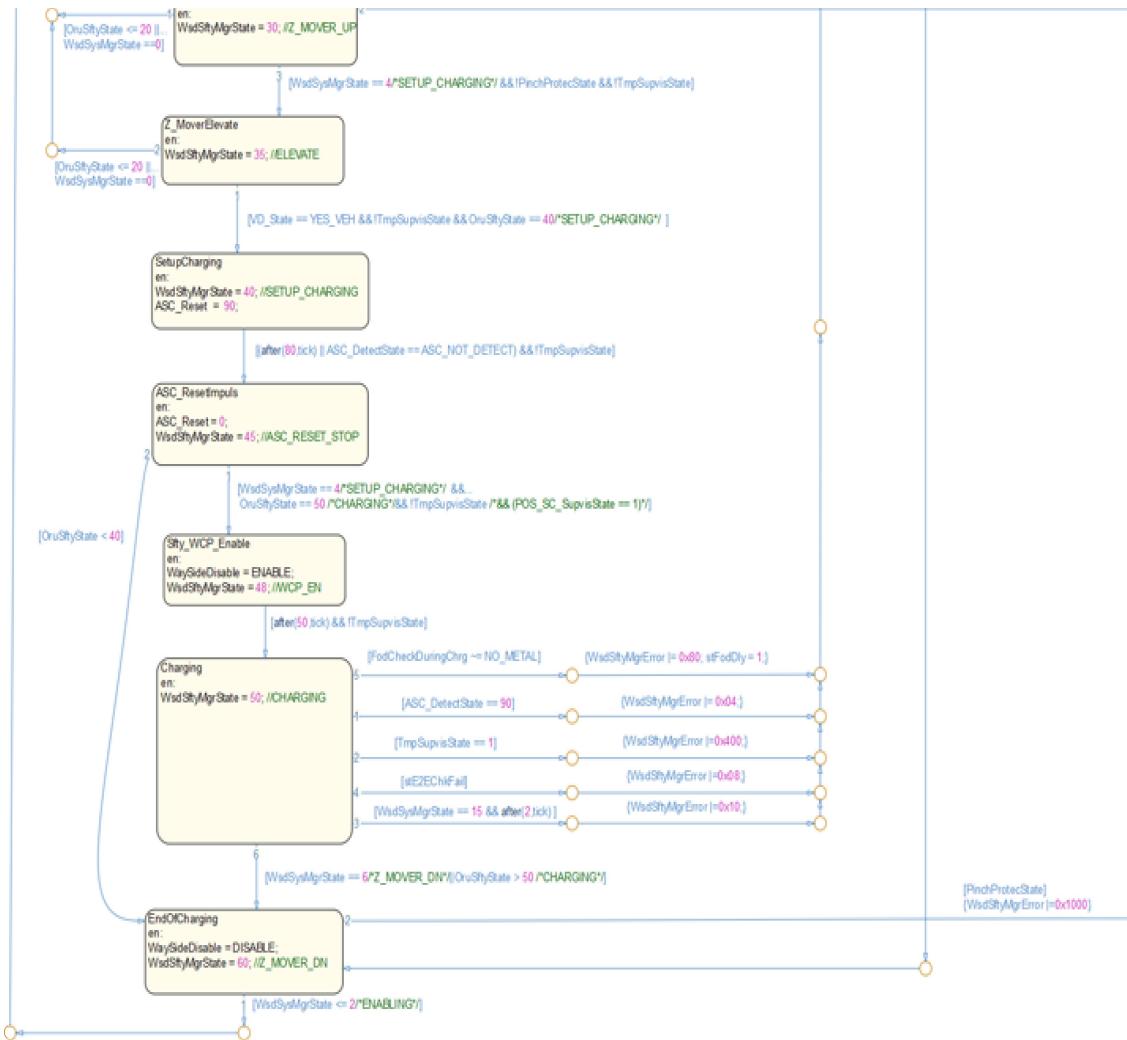
Ein numFOD Wert von 90 (0x5A) entspricht hierbei kein Metall erkannt und 150 (0x96) Metall erkannt.

Ebenso lässt sich die Vehicle Detection mittels des Schalters *WspSfty_swtDisableVD_C* auf den Applikationswert *WspSfty_numVDDisVal_C* umschalten. Hier entspricht ein Wert von 90 (0x5A) kein Vehicle erkannt und 150 (0x96) Vehicle erkannt.

Softwaredokumentation

StateMachine (Stateflow)





Dies ist die Statechart des Wayside Safety State Managers. Es folgt nun die Beschreibung der einzelnen States:

12.2.2.1 State Standby

Dies ist der Initialstate, welcher nach dem (Neu-) Start des Pad aktiviert wird. Beim aktivieren dieses States wird der aktuelle Safety Status *WspSftyMgrState* (*WspSfty_stSafety*) auf 10 (Standby) gesetzt. Der Fehlerstatus *WsdSftyMgrError* wird gelöscht.

Der Sollstatus *WaySideDisable* (*WspSfty_stDisable*) für die B6 - Brücke wird auf 0x5A (DISABLE) gesetzt. Der Sollstatus *ADC_Reset* (*WspSfty_stASCRest*) wird mit 0 (INACTIV) initialisiert.

Der Sollstatus *Z_MoverEnable* ([WspSfty_stZMoverEna](#)) der Z-Mover Endstufe wird auf 0x96 (ENABLE) gesetzt. Dies ist notwendig, da nach einem (Neu)Start des Steuergerätes der Z-Mover von der Z-Mover Komponente in Ausgangsposition gefahren, kalibriert und überprüft wird.

Der Status *stFodDly* ([WspSfty_stFodDly](#)) für die Verzögerung FOD wird mit 0 initialisiert.

State Transition -> Idle

Aus dem State *Standby* wird in den State *Idle* verzweigt, wenn der Status *OruSftyState* ([WspSfty_stOruSftyMgr](#)) des Oru Safety Managers den Wert 20 (IDLE) hat und die Aktivität der Oru Positionsantenne erkannt wurde, d.h. *POS_SC_SupvisState* ([WspSfty_stPosSupervision](#)) ist größer als 1 und kein Fehler der Safety Temperaturüberwachung vorliegt, d.h. *TmpSupvisState* ([WspSfty_stTempSupvis](#)) ist *false*.

12.2.2.2 State Idle

Beim aktivieren dieses States wird der aktuelle Safety Status *WspSftyMgrState* auf 20 (IDLE) gesetzt und der Fehlerstatus *WsdSftyMgrErr* des Safetymanagers gelöscht.

State Transition -> Standby

Aus dem State *Idle* wird zurück in den State *Standby* verzweigt, wenn der Status *OruSftyState* des Oru Saftemanagers 10 (INIT) ist.

State Transition -> FodChkMinPos

Aus dem State *Idle* wird in den State *FodChkMinPos* verzweigt, wenn alle folgenden Bedingungen erfüllt sind:

- Der Status *WsdSysMgrState* ([WspSfty_stSysMgr](#)) des Wayside System Statemanagers ist größer als 0 (*Searching*)
- Es sind mindestens 200ms nach Aktivierung des *Idle* States abgelaufen (zur Pos Erkennung)
- Kein Fehler der Safety Temperaturüberwachung vorliegt, also *TmpSupvisState* den Wert *false* hat
- Die aktivierung des RCD Tests ist nicht angefordert (*stRCDReq = false*)
- Der Status des RCD Selftests meldet ‚fertig‘ (*stRCDRdy = true*)

12.2.2.3 State FodChkMinPos

Beim aktivieren diese States wird der aktuelle Safety Status *WspSftyMgrState* auf 21 gesetzt.

State Transition -> FodPreChk

Aus dem State *FodChkMinPos* wird in den State *FodPreChk* verzweigt, wenn alle folgenden Bedingungen erfüllt sind:

- Die der x - Abstand *PosX* ([WspSfty_lPosX](#)) zum Fahrzeug ist größer oder gleich *PosXMin* ([WspSfty_lFodChkPosXMin_C](#))

- Die der y - Abstand *PosY* ([WspSfty_IPosY](#)) zum Fahrzeug ist größer oder gleich *PosYMin* ([WspSfty_IFodChkPosYMin_C](#))
- Nach Aktivierung dieses States mindestens 200ms vergangen sind.

Zum State *FodPreChk* wird also nur verzweigt, wenn das Fahrzeug noch eine genügend großen Abstand zum Pad hat.

State Transition -> Idle

Aus dem State *FodChkMinPos* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

State Transition -> FodChkZMoverDwn

Aus dem State *FodChkMinPos* wird in den State *FodChkZMoverDwn* verzweigt, wenn das Fahrzeug für den FodPre Check schon zu dicht am Pad ist, also mindestens eine der folgenden Bedingungen erfüllt ist:

- Die der x - Abstand *PosX* ([WspSfty_IPosX](#)) zum Fahrzeug ist kleiner *PosXMin* ([WspSfty_IFodChkPosXMin_C](#))
- Die der y - Abstand *PosY* ([WspSfty_IPosY](#)) zum Fahrzeug ist kleiner *PosYMin* ([WspSfty_IFodChkPosYMin_C](#))

12.2.2.4 State FodPreChk

Beim aktivieren diese States wird der aktuelle Safety Status *WspSftyMgrState* auf 22 gesetzt.

State Transition -> FodChkZMoverDwn

Der State *FodPreChk* verzweigt zum State *FodChkZMoverDwn*, wenn der FOD Eingangsstatus *FodPreChk* ([WspSfty_stFodPreCheck](#)) den Wert 90 (No Metal detected) hat; jedoch frühestens 200ms nach Aktivierung des States.

State Transition -> Idle

Aus dem State *FodPreChk* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

State Transition -> FodDlyPre

Der State *FodPreChk* verzweigt zum State *FodDlyPre*, wenn ein Metallobjekt erkannt wurde, d.h. *FodPreChk* ([WspSfty_stFodPreCheck](#)) den Wert 150 hat.

12.2.2.5 State FodDlyPre

Bei aktivieren dieses FOD Fehlerstates wird der Status *stFodDly* ([WspSfty_stFodDly](#)) auf 1 gesetzt (Verzögerung FOD aktiv). Beim verlassen dieses States wird der Status *stFodDly* zurück auf 0 gesetzt (Verzögerung FOD inaktiv).

State Transition -> FodPreChk

Der State *FodDlyPre* verzweigt zurück zum State *FodPreChk*, wenn der Status *FodPreChk* der Metallobjekterkennung den Wert 90 hat (kein Metall erkannt).

12.2.2.6 State FodChkZMoverDwn

Beim aktivieren diese States wird der aktuelle Safety Status *WspSftyMgrState* auf 23 gesetzt.

State Transition -> WaitPos

Der State *FodChkZMoverDwn* verzweigt zum State *WaitPos*, wenn der Status *WsdSysMgrState* des Wayside Systemmanagers den Status 7 (WaitPos) hat, also auf den POS Modus umgeschaltet hat und der Status *OruSftyState* des Oru Safety Managers den Status 26 hat, die Oru also auf die Positionierung des Fahrzeugs wartet.

State Transition -> Idle

Aus dem State *FodChkZMoverDwn* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.7 State WaitPos

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 24 gesetzt.

State Transition -> WaitReqChrg

Der State *WaitPos* verzweigt in den State *WaitReqChrg*, wenn der Status *WsdSysMgrState* des Wayside Systemmanagers den Status 9 (WaitZMvDwnChk) hat, also in den FOD Modus zurückgeschaltet hat und der Status *FodChkZMDwn* (*WspSfty_stFodCheckZMDwn*) der Metallobjekterkennung den Wert 90 hat (kein Metall erkannt).

State Transition -> FodDlyZMDwn

Der State *WaitPos* verzweigt zum State *FodDlyZMDwn*, wenn ein Metallobjekt erkannt wurde, d.h. *FodChkZMDwn* (*WspSfty_stFodCheckZMDwn*) den Wert 150 hat.

State Transition -> Idle

Aus dem State *WaitPos* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.8 State WaitReqChrg

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 25 gesetzt.

State Transition -> FodChkZMoverUp

Der State *WaitReqChrg* verzweigt zum State *FodChkZMoverUp*, wenn der Status *OruSftyState* des Oru Safety State Managers den Wert 28 hat und damit anzeigt, dass die Ladeanforderung vom Fahrzeug akzeptiert wurde und die Anforderung zum Hochfahren des Z Movers erfüllt wurde.

State Transition -> WaitPos

Der State *WaitReqChrg* verzweigt zurück zum State *WaitPos*, wenn der Status *WsdSysMgrState* des Wayside System Statemanagers auf 7 (WaitPos) zurückgegangen ist.

State Transition -> Idle

Aus dem State *WaitReqChrg* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.9 State FodChkZMoverUp

Beim aktivieren diese States wird der aktuelle Safety Status *WspSftyMgrState* auf 26 gesetzt.

State Transition -> MetalObjSearch

Der State *FodChkZMoverUp* verzweigt zum State *MetalObjSerach*, wenn der Status *WsdSysMgrState* des Wayside State Managers den Wert 8 hat, also die Initialpositionierung des Z Movers abgeschlossen ist und der Status *FodChkZMUp* (*WspSfty_stFodCheckZMUp*) der Metallobjekterkennung den Wert 90 hat (kein Metall erkannt).

State Transition -> FodDlyZMUp

Der State *FodChkZMoverUp* verzweigt zum State *FodDlyZMUp*, wenn ein Metallobjekt erkannt wurde, d.h. *FodChkZMUp* (*WspSfty_stFodCheckZMUp*) den Wert 150 hat.

State Transition -> PinchDet

Der State *FodChkZMoverUp* verzweigt zum State *PinchDet*, wenn der Status *PinchProtectState* (*WspSfty_stPinchErr*) gesetzt (=1) ist und damit anzeigt, dass der Einklemmschutz ausgelöst wurde. Durch diese Verzweigung wird Bit 11 des Fehlerstatus *WsdSftyMgrError* gesetzt.

State Transition -> WaitPos

Der State *FodChkZMoverUp* verzweigt zurück zum State *WaitPos*, wenn der Status *OruSftyState* des Oru Safety Statemangers zurückgegangen ist auf 26 (WaitPos).

State Transition -> FodChkZMoverDwn

Der State *FodChkZMoverUp* verzweigt zurück zum State *FodChkZMoverDwn*, wenn der Status *WsdSysMgrState* des Wayside System Statemanagers zurückgegangen ist auf 2 (Enabling); z.B. durch Ladeabbruch.

State Transition -> Idle

Aus dem State *FodChkZMoverUp* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.10 State MetalObjSearch

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 27 gesetzt.

State Transition -> VehInChargePosition

Der State *MetalObjSearch* verzweigt zum State *VehInChargePosition*, wenn der State *FodChekStartUpChrg* ([WspSfty_stFodCheckStartUpChrg](#)) den Wert 90 hat, also vom FOD kein Metall erkannt wurde und der Status *TmpSupvisState* der Safety Temperaturüberwachung nicht gesetzt ist, jedoch frühestens nach Ablauf von 50ms nach Aktivierung des States.

State Transition -> FodDlyStrtChrg

Der State *MetalObjSearch* verzweigt zum State *FodDlyStrtChrg*, wenn ein Metallobjekt erkannt wurde, d.h. *FodCheckStartupChrg* ([WspSfty_stFodCheckStartUpChrg](#)) den Wert 150 hat.

State Transition -> Idle

Aus dem State *MetalObjSearch* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.11 State VehInChargePosition

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 30 gesetzt.

State Transition -> Z_MoverElevate

Der State *MetalObjSearch* verzweigt zum State *Z_MoverElevate*, wenn der Status *WsdSysMgrState* des Wayside System Satemanagers den Wert 4 (Setup Charging) hat und der Einklemmschutz keinen Fehler anzeigt (*PinchProtecState = false*) und die Safety Temperaturüberwachung keine Fehler anzeigt (*TmpSupvisState = false*).

State Transition -> PinchDet

Der State *FodChkZMoverUp* verzweigt zum State *PinchDet*, wenn der Status *PinchProtectState* ([WspSfty_stPinchErr](#)) gesetzt (=1) ist und damit anzeigt, dass der Einklemmschutz ausgelöst wurde. Durch diese Verzweigung wird Bit 11 des Fehlerstatus *WsdSftyMgrError* gesetzt.

State Transition -> Idle

Aus dem State *VehInChargePosition* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.12 State Z_MoverElevate

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 35 gesetzt.

State Transition -> SetupCharging

Der State *Z_MoverElevate* verzweigt zum State *SetupCharging*, wenn der Status *VD_State* ([WspSfty_stVDDet](#)) der Vehicle Detection den Wert 150 (Vehicle detected) hat und der Oru Safety Statemanager im Status *OruSftyState = 40* (*SetupCharging*) ist und kein Safety Temperaturüberwachungsfehler vorliegt (*TmpSupvisState = false*).

State Transition -> Idle

Aus dem State *ZMoverElevate* wird zurück zum State *Idle* verzweigt, wenn der Status *OruSftyState* des Oru Safetymanagers kleiner ist als 20 (IDLE) oder der Status *WsdSysMgrState* des Wayside Systemmangers 0 (Searching) ist.

12.2.2.13 State SetupCharging

Beim aktivieren diese States wird der aktuelle Safety Status *WspSftyMgrState* auf 40 gesetzt. Die ASC Erkennung wird durch eine Flanke am entsprechend DIO Pin zurückgesetzt d.h. der Sollstatus *ASC_Reset* (*WspSfty_stASCRest*) wird hier auf 90 und im nächsten State wieder auf 0 gesetzt.

State Transition -> ASC_ResetImpuls

Der State *SetupCharging* verzweigt zum State *ASC_ResetImpuls*, wenn der Status *ASC_DetectState* (*WspSfty_stASCDet*) den Wert 150 (ASC_NOT_DETECT) hat oder nach Ablauf von 800 ms, jedoch nur, wenn kein Temperaturüberwachungsfehler angezeigt wird.

12.2.2.14 State ASC_ResetImpuls

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 45 gesetzt. Zum beenden des Resetimpulses wird der Sollstatus *ASC_Reset* auf 0 zurückgesetzt.

State Transition -> Sfty_WCP_Enable

Der State *ASC_ResetImpuls* verzweigt zum State *Sfty_WCP_Enable*, wenn der Status *WsdSysMgrState* des Wayside System Statemanagers 4 (SETUP_CHARGING) ist und der Status *OruSftyState* des Oru Safety Manager 50 (CHARGING) ist und auch kein Temperaturüberwachungsfehler vorliegt, d.h. *TmpSupvisState* nicht gesetzt ist.

State Transition -> EndOfCharging

Der State *ASC_Resetimpuls* verzweigt zum State *EndOfCharging*, wenn der Status *OruSftyState* des Oru Safety Statemanagers kleiner ist als 40 (PreCharging).

12.2.2.15 State Sfty_WCP_Enable

Beim aktivieren dieser States wird der aktuelle Safety Status *WspSftyMgrState* auf 48 gesetzt. Der Sollstatus *WaysideDisable* (*WspSfty_stDisable*) für die B6 Brücke wird auf 150 (ENABLE) gesetzt und diese damit freigegeben.

State Transition -> Charging

Der State *Sfty_WCP_Enabe* verzweigt zum State *Charging*, wenn nach Aktivierung dieses States 500ms vergangen sind und kein Temperaturüberwachungsfehler vorliegt.

12.2.2.16 State Charging

Beim aktivieren des States *Charging* wird der aktuelle Safety Status *WspSftyMgrState* auf 50 gesetzt.

State Transition -> EnfOfCharging

Der State *Charging* verzweigt zum State *EndOfCharging*, wenn der Status *WsdSysMgrState* des Wayside System Statemanagers gleich 6 (Z_Mover_Down) ist oder *OruSftyState* des Oru Safety Statemanagers größer ist als 50 (Charging).

State Transition -> WaitSftyErrorNotif

Der State *Charging* verzweigt zum State *WaitSftyErrorNotif*, wenn eine der folgenden Fehlerbedingungen gesetzt ist:

- Metallobjekt erkannt (*FodCheckDuringCharging* = 150). Durch diese Bedingung wird im Fehlerstatus *WsdSftyMgrErr* das Bit 7 gesetzt und der Status *stFodDly* wird auf 1 gesetzt.
- Temperaturüberwachungsfehler (*TmpSupvisState* = 1). Durch diese Bedingung wird im Fehlerstatus *WsdSftyMgrErr* das Bit 2 gesetzt.
- ASC wurde ausgelöst (*ASC_DetectState* = 90). Durch diese Bedingung wird im Fehlerstatus *WsdSftyMgrErr* das Bit 10 gesetzt.
- End to End Fehler (*stE2EChkFail* = true). Durch diese Bedingung wird im Fehlerstatus *WsdSftyMgrErr* das Bit 3 gesetzt.
- Fehler System Statemanager (*WsdSysMgrState* = 15). Durch diese Bedingung wird im Fehlerstatus *WsdSftyMgrErr* das Bit 4 gesetzt.

12.2.2.17 State EndOfCharging

Beim aktivieren des States *EndOfCharging* wird der aktuelle Safety Status *WspSftyMgrState* auf 60 gesetzt. Die B6 Brücke wird abgeschaltet (*WaySideDisable* = 90 -> DISABLE).

State Transition -> Idle

Der State *EndOfCharging* wechselt zurück zum State *Idle*, wenn der Status **WsdSysMgrState** des Wayside System Statmanagers mindesten bis zum State 2 (ENABLING) zurückgegangen ist.

State Transition -> PinchDet

Der State *EndOfCharging* wechselt zum State *PinchDet*, wenn der Status *PinchProtectState* des Einklemmschutzes gesetzt ist. In diesem Fall wird Bit 12 des Fehlerstatus *WsdSftyMgrErr* gesetzt.

12.2.2.18 State PinchDet

Beim aktivieren des States *PinchDet* wird der aktuelle Safety Status *WspSftyMgrState* auf 65 gesetzt.

State Transition -> Shutdown

Der State *PinchDet* wechselt zum State *Shutdown*, wenn der Status *WsdSysMgrState* des Wayside System Statmanagers den Status 15 (ERROR) hat und seit Aktivierung des States *PinchDet* mindesten 3s vergangen sind.

12.2.2.19 State WaitSftyErrorNotif

Beim aktivieren des States *WaitErrorNotif* wird der aktuelle Safety Status *WspSftyMgrState* auf 70 gesetzt. Die B6 Brücke wird deaktiviert, d.h. *WaySideDisable* wird auf 90 (0x5A) gesetzt.

State Transition -> Standby

Der State *WaitErrorNotif* wechselt zum State *Standby*, wenn kein Temperaturüberwachungsfehler vorliegt (*TmpSupvisState = false*) und der Oru Safety Manager im State IDLE (*OruSftyState = 20*) ist und der Wayside System State Manager nicht im State ERROR (*WsdSysMgrState = 15*) ist.

State Transition -> Shutdown

Der State *WaitErrorNotif* wechselt zum State *Shutdown*, wenn spätestens 30s nach aktivierung dieses States nicht zum State *Standby* verzweigt wurde.

12.2.20 State Shutdown

Beim aktivieren des States *Shutdown* wird der aktuelle Safety Status *WspSftyMgrState* auf 80 gesetzt. Die B6 Brücke wird deaktiviert, d.h. *WaySideDisable* wird auf 90 (0x5A) gesetzt. Ebenso wird der Z-Mover deaktiviert, indem der Sollstatus *Z_MoverEnable* des Z-Mover HW Treibers auf 90 (DISABLE) gesetzt wird.

State Transition -> keine

Der State *Shutdown* kann nur durch Reset des Steuergerätes wieder verlassen werden.

12.2.3 Block RCDTstChk

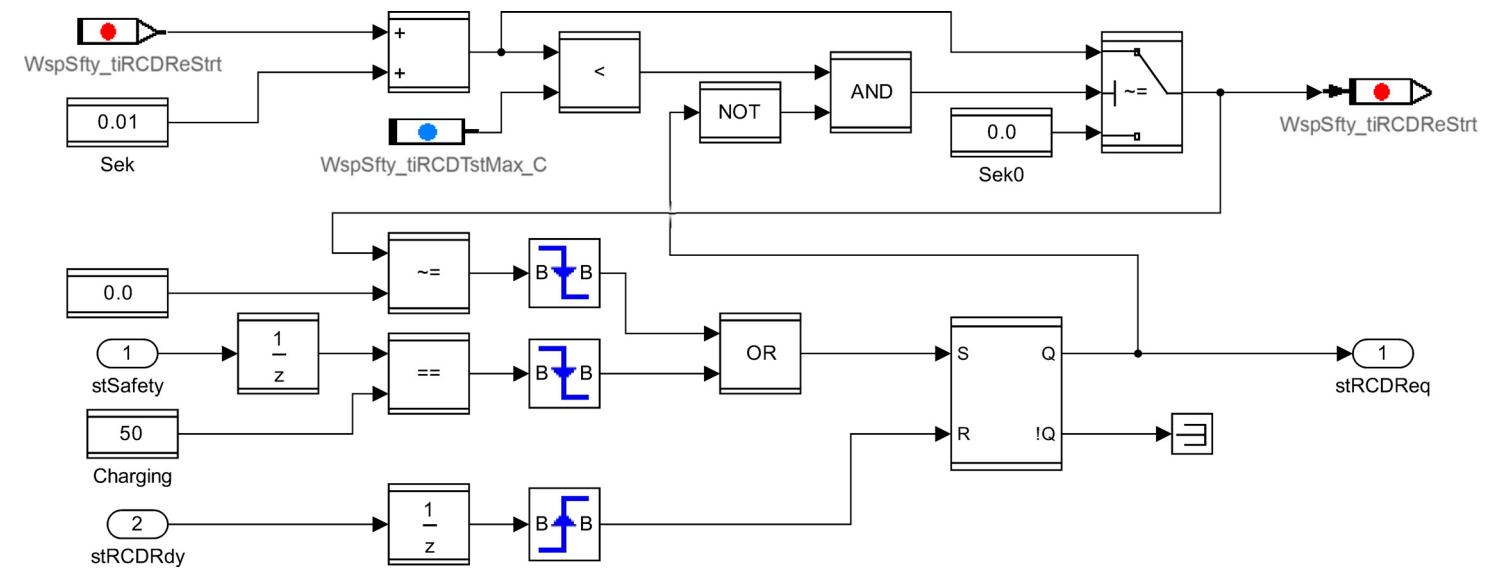
Hier ist die Anforderungslogik für den RCD Selbsttest implementiert.

Der RDC Selbsttest wird aktiviert, wenn der Ausgangsstatus *stRCDReq* (*WspSfty_stRCDReq*) den Wert *true(1)* hat.

Der Aktivierungsstatus *stRCDReq* wird auf *true* gesetzt, wenn der Status *stSafety* des Wayside Safetymanagers den State 50 (*Charging*) verlässt, oder der Timeoutcounter *WspSfty_tiRCDReStrt* die Maximale Wartezeit *WspSfty_tiRCDTstMax_C* erreicht hat.

Der Timer wird auf 0 gesetzt und angehalten, solange der Anforderungsstatus *stRCDReq* gesetzt ist.

Der Anforderungsstatus *stRCDReq* wird zurückgesetzt, wenn der Rückgabestatus *stRCDRdy* (*WspSfty_stRCDRdy*) der RDC Selbsttestfunktion auf *true* geht.



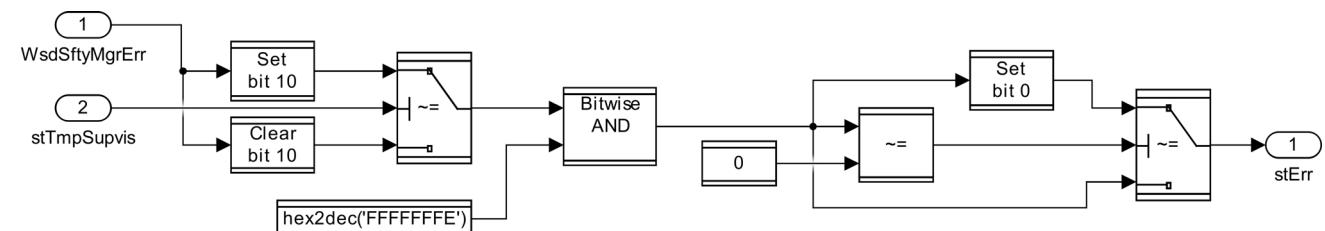
12.2.4 Block ChkErrState

Im Block *ChkErrState* werden die noch fehlenden Fehlerbits für den aktuellen Fehlerstatus *stErr* ([WspSfty_stMgrErr](#)) gesetzt.

Hierzu werden zu dem aus der Safety State Manager kommenden Fehlerstatus *WsdSftyMgrErr* noch das Bit 10 gesetzt, falls ein Temperaturüberwachungsfehler vorliegt (*stTmpSupvis = true*), anderenfalls wird Bit 10 gelöscht. Für den Fall, das irgendein Fehlerbit ausser Bit 0 gesetzt ist, wird Bit 0 gesetzt.

Die Zuordnung der Fehler ist in nachfolgender Tabelle definiert.

Fehlerbits in [WspSfty_stMgrErr](#):



Bit	Fehler
0	Es liegt ein Safety Fehler vor (in den nachfolgenden Bits näher spezifiziert)
1	tbd
2	ASC detected
3	E2E Check Fail
4	Error in Wayside System State Manager
7	Metal Object detected while Charging
10	Temperature Supervision Error
11	Pinch Error at Z Mover Up
12	Pinch Error at Z Mover Down
13 - 31	tbd

12.2.5 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

12.2.5.1 Eingangsgrößen

Name	Typ	Interface / Argument	Beschreibung
	RPort		

12.2.5.2 Ausgangsgrößen

Name	Typ	Interface / Argument	Beschreibung
	PPort		

12.2.5.3 Interfaces

12.2.6 Messgrößen

<i>WspSfty_IPosX</i>	X-Position Fahrzeug
<i>WspSfty_IPosY</i>	Y-Position Fahrzeug
<i>WspSfty_stASCDet</i>	ASC Detecion Status
<i>WspSfty_stASCReset</i>	Status ASC Erkennung zurücksetzen (um erneut starten zu können)
<i>WspSfty_stDisable</i>	Disable Status Charging Wayside
<i>WspSfty_stE2EChkFail</i>	Status E2E Fehler Eingangssignal (true = Fehler)
<i>WspSfty_stFodCheckDuringChrg</i>	Status MOD Detecion During Charging (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckStartUpChrg</i>	Status MOD Detecion at start of charging (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckZMDown</i>	Status MOD Detecion at z-Mover is down (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckZMUp</i>	Status MOD Detecion at z-Mover is up (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodDly</i>	Status FOD Verzögerung (1 = Verzögerung durch FOD aktiv)
<i>WspSfty_stFodPreCheck</i>	Status MOD Detecion Pre Check (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stMgrErr</i>	Fehlerstatus Wayside Safty Manager
<i>WspSfty_stPinchErr</i>	PinchProtection Fehler (falls true)
<i>WspSfty_stSafety</i>	Status Safteymanager WaySide
<i>WspSfty_stSysMgr</i>	Status Wsp System State Manager
<i>WspSfty_stTempSupvis</i>	Status Temperaturüberwachung ; 0 = 0k, 1 = Übertemperatur
<i>WspSfty_stVDDet</i>	Status Veh Detecion (0x96 YES_Veh, 0x5A NO_Veh)
<i>WspSfty_stZMoverEna</i>	Status Enable Z-Mover
<i>WspSfty_numASCThres</i>	Set Value for ASC Thresold PWM Value
<i>WspSfty_stASCDetInv</i>	ASC Detecion Status, Bits invertiert
<i>WspSfty_stASCResetInv</i>	Status ASC Erkennung zurücksetzen (um erneut starten zu können) BitInvertiert
<i>WspSfty_stDisableInv</i>	Disable Status Charging Wayside BitInvertiert

Softwaredokumentation

<i>WspSfty_stFodCheckDuringChrgInv</i>	Bitinvertierter Status MOD Detection During Charging (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckStartUpChrgInv</i>	Bitinvertierter Status MOD Detection at start of charging (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckZMDownInv</i>	Bitinvertierter Status MOD Detection at z-Mover is down (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodCheckZMUpInv</i>	Bitinvertierter Status MOD Detection at z-Mover is up (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stFodPreCheckInv</i>	Bitinvertierter Status MOD Detection Pre Check (0x96 YES_Metal, 0x5A NO_METAL)
<i>WspSfty_stOruSftyMgr</i>	Status Oru Safety Manager
<i>WspSfty_stPosSupervision</i>	Status POS Signal Überwachung
<i>WspSfty_stRCDRdy</i>	Status RCD Test fertig (wenn true)
<i>WspSfty_stRCDReq</i>	Status RCD Testanforderung (Angefordert wenn true)
<i>WspSfty_stSigASCDet</i>	Status Signalfehler stASCDet (true = Fehler, false = Ok)
<i>WspSfty_stSigFodChkDurChrg</i>	Status Signalfehler stFodCheckDuringCharging (true = Fehler, false = Ok)
<i>WspSfty_stSigFodChkStrtChrg</i>	Status Signalfehler stFodChckStartUpChrg (true = Fehler, false = Ok)
<i>WspSfty_stSigFodChkZMDwn</i>	Status Signalfehler stFodChckZMDown (true = Fehler, false = Ok)
<i>WspSfty_stSigFodChkZMUp</i>	Status Signalfehler stFodChckZMUp (true = Fehler, false = Ok)
<i>WspSfty_stSigFodPreChk</i>	Status Signalfehler stFodPreCheck (true = Fehler, false = Ok)
<i>WspSfty_stSigVDet</i>	Status Signalfehler stVDet (true = Fehler, false = Ok)
<i>WspSfty_stTempHous</i>	Status temperaturüberwachung PAD Gehäuse; 0 = 0k, 1 = Übertemperatur
<i>WspSfty_stTempPad</i>	Status temperaturüberwachung PAD Abdeckplatte; 0 = 0k, 1 = Übertemperatur
<i>WspSfty_stVDetInv</i>	Status Veh Detection (0x96 YES_Veh, 0x5A NO_Veh), Bits invertiert
<i>WspSfty_stZMoverDisInv</i>	Status Disable Z-Mover Bitweise Invertiert
<i>WspSfty_tDifffHous</i>	Temperaturdifferenz PAD Gehäuse zu Spulentemperatur - Max(WPC1,WPC2,WPC3)
<i>WspSfty_tDifffPad</i>	Temperaturdifferenz PAD Abdeckplatte zu Umgebungstemperatur MOD
<i>WspSfty_tHousEst</i>	Erwartete maximale PAD Gehäusetemperatur
<i>WspSfty_tHWIO</i>	Eingangswert Temperatur HWIO an Charging Pad Spulenplatte

Softwaredokumentation

primove
true e-mobility

WspSfty_tiRCDReStrt

Time Counter for RCD ReTest

12.2.7 Applikationswerte

<i>WspSfty_IFodChkPosXMin_C</i>	Minimaler x - Abstand zum Durchführen des Fod Pre Checks
<i>WspSfty_IFodChkPosYMin_C</i>	Minimaler y - Abstand zum Durchführen des Fod Pre Checks
<i>WspSfty_IPosXOffs_C</i>	Offset Position X
<i>WspSfty_IPosYOffs_C</i>	Offset Position Y
<i>WspSfty_numFodCheckDuringChrg_C</i>	zu verwendender Simulations-Wert für FOD während des Ladens, 90 = kein Metall, 150 = Metall erkannt.
<i>WspSfty_numFodCheckStartUpChrg_C</i>	zu verwendender Simulations-Wert für FOD bei Ladebeginn, 90 = kein Metall, 150 = Metall erkannt.
<i>WspSfty_numFodCheckZMDown_C</i>	zu verwendender Simulations-Wert für FOD bei z-Mover down, 90 = kein Metall, 150 = Metall erkannt.
<i>WspSfty_numFodCheckZMUp_C</i>	zu verwendender Simulations-Wert für FOD bei z-Mover up, 90 = kein Metall, 150 = Metall erkannt.
<i>WspSfty_numFodPreCheck_C</i>	zu verwendender Simulations-Wert für FOD während Pre Check, 90 = kein Metall, 150 = Metall erkannt.
<i>WspSfty_numVDDisVal_C</i>	zu verwendender Wert für Veh Detect falls Erkennung Disabled, 90 = kein Vehl, 150 = Vehicle erkannt.
<i>WspSfty.swtDisableVD_C</i>	Schaltet Veh Detectet Erkennung auf Konstanten Wert 90 - kein Metall (falls true)
<i>WspSfty.swtSimFodCheckDuringChrg_C</i>	Schaltet FOD Erkennung während des Ladens auf Konstanten Vorgabe-Wert 90 - kein Metall (falls true)
<i>WspSfty.swtSimFodCheckStartUpChrg_C</i>	Schaltet FOD bei bei Ladebeginn auf Konstanten Vorgabe-Wert 90 - kein Metall (falls true)
<i>WspSfty.swtSimFodCheckZMDown_C</i>	Schaltet FOD bei z-Mover Down up Konstanten Vorgabe-Wert 90 - kein Metall (falls true)
<i>WspSfty.swtSimFodCheckZMUp_C</i>	Schaltet FOD bei z-Mover Down auf Konstanten Vorgabe-Wert 90 - kein Metall (falls true)
<i>WspSfty.swtSimFodPreCheck_C</i>	Schaltet FOD Pre Check auf Konstanten Vorgabe-Wert 90 - kein Metall (falls true)

12.2.8 Fehlerpfade

12.2.8.1 tbd

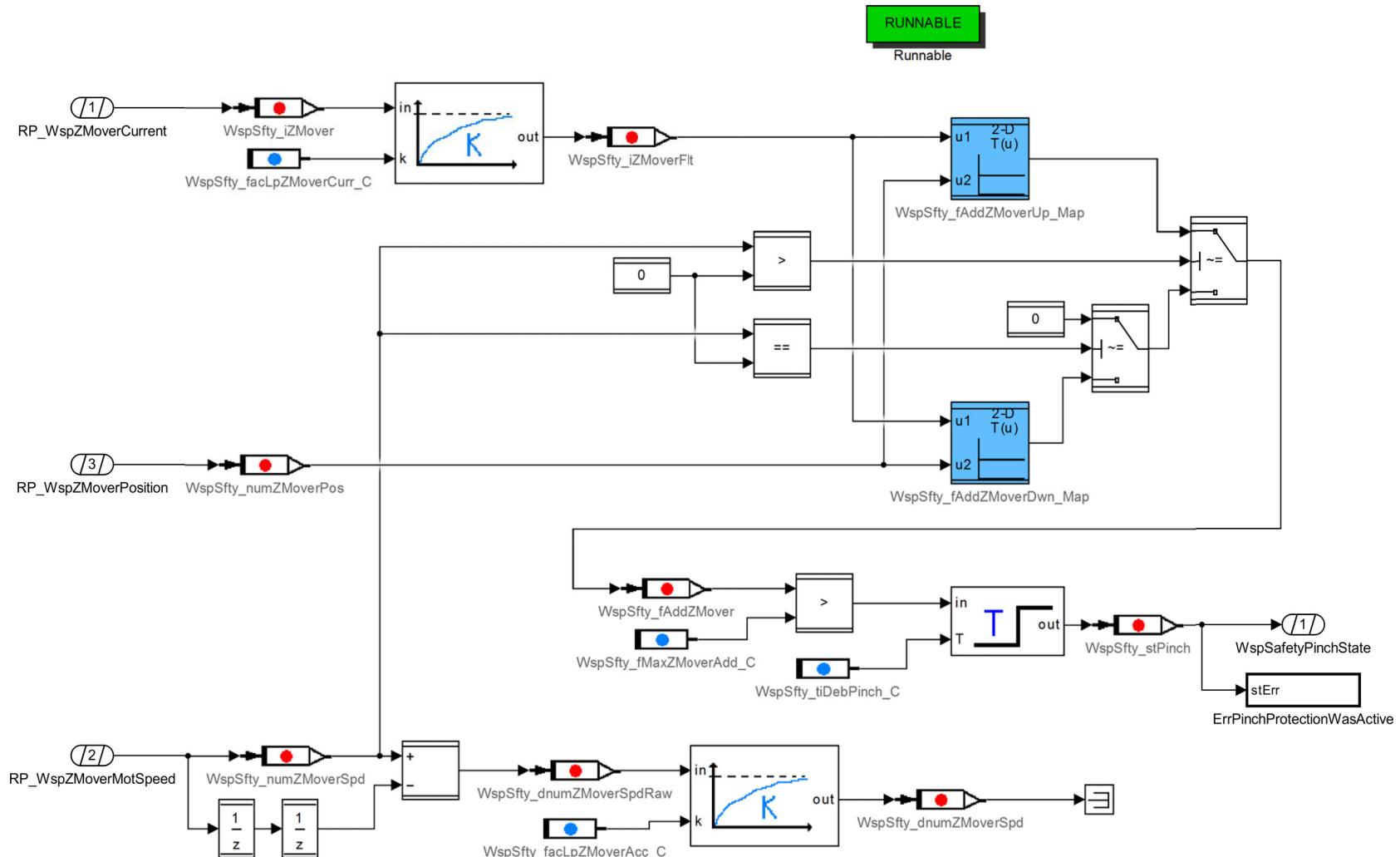
Fehler

tbd	
Fehlerindex	
Fehler Status	
Defektbedingung	
Heilbedingung	
Fehlerklasse	
Defekt Entprellzeit	
Heil Entprellzeit	
DTC	

12.2.9 Initialisierung

tbd

12.3 Komponente WspSfty_Pinch



In dieser Komponente ist der Einklemmschutz (Pinch Protection) implementiert.

Es wird zunächst der aktuelle Z-Mover Strom [WspSfty_iZMover](#) gefiltert. Die Filterkonstante für den Tiefpassfilter ist hierbei der Applikationswert [WspSfty_facLpZMoverCurr_C](#).

Der so erhaltene gefilterte Z-Mover Strom [WspSfty_iZMoverFit](#) wird dann als y-Eingang für das jeweils selektierte nachfolgende Kennfeld verwendet. Als x - Eingang wird die Z-Mover Position [WspSfty_numZMoverPos](#) verwendet.

Softwaredokumentation

primove
true e-mobility

Ist der aktuelle Z-Mover Geschwindigkeit *WspSfty_numZMoverSpd* positiv, bewegt sich also der Z-Mover nach oben, wird das Kennfeld *WspSfty_fAddZMoverUp_Map* verwendet, bei Abwärtsbewegung *WspSfty_fAddZMoverDwn_Map*.

Der Ausgang der Kennfleder ist jeweils der zusätzliche Kraftaufwand *WspSfty_fAddZMover* hinsichtlich des lastfreien Betriebs des Z-Movers. Ist die Z-Mover Geschwindigkeit 0, wird *WspSfty_fAddZMover* konstant auf 0 gesetzt. Übersteigt nun die Zusatzkraft *WspSfty_fAddZMover* den Schwellwert *WspSfty_fMaxZMoverAdd_C* länger als *WspSfty_tiDebPinch_C*, wird der Fehlerstatus *WspSfty_stPinch* gesetzt.

Der Fehlerstatus wird dann dem Fehlerpfad *PinchProtectionWasActive* zugewiesen.

Für spätere Verwendung wird noch durch zeitliche Differenzbildung der Z-Movergeschwindigkeit *WspSfty_numZMoverSpd* zunächst der Rohwert *WspSfty_dnumZMoverSpdRaw* der Z-Mover Beschleunigung berechnet. Dieser wird dann mittels der Filterkonstanten *WspSfty_facLpZMoverAcc_C* tiefpassgefiltert und ergibt so den Beschleunigungswert *WspSfty_dnumZMoverSpd*.

12.3.1 Schnittstellen (Autosar)

12.3.1.1 Eingangsgrößen

RP_WspZMoverCurrent	
RP_WspZMoverPosition	
RP_WspZMoverMotSpeed	

12.3.1.2 Ausgangsgrößen

WspSafetyPinchState	
---------------------	--

12.3.1.3 Interfaces

12.3.2 Messgrößen

<i>WspSfty_dnumZMoverSpd</i>	Zahlenwert gemessene Z-Moverbeschleunigung
<i>WspSfty_dnumZMoverSpdRaw</i>	Zahlenwert gemessene Z-Moverbeschleunigung Rohwert
<i>WspSfty_fAddZMover</i>	Additional Force on Z Mover
<i>WspSfty_iZMover</i>	aktueller Aktuatorstrom vom z-Mover
<i>WspSfty_iZMoverFilt</i>	aktueller Aktuatorstrom vom z-Mover gefiltert
<i>WspSfty_numZMoverPos</i>	Zahlenwert gemessene Z-Mover Position
<i>WspSfty_numZMoverSpd</i>	Zahlenwert gemessene Z-Movergeschwindigkeit
<i>WspSfty_stPinch</i>	Status Pinchprotection 1= Ausgelöst, 0 = alles Ok

12.3.3 Applikationswerte

<i>WspSfty_facLpZMoverAcc_C</i>	Faktor Lowpass Z-Moverbeschleunigung
<i>WspSfty_tiDebPinch_C</i>	Debouncezeit zum Setzen des Pinch States bei zu hoher Zusatzkraft am z Mover
<i>WspSfty_fMaxZMoverAdd_C</i>	Maximale zusätzliche Kraft am Z Mover für Pinch Protection
<i>WspSfty_facLpZMoverCurr_C</i>	Faktor Lowpass Z Mover- Ist Strom
<i>WspSfty_fAddZMoverUp_Map</i>	Kennfeld zusätzlicher Kraftaufwand am Z-Mover gegenüber unbelasteten Zustandes beim hochfahren
<i>WspSfty_fAddZMoverDwn_Map</i>	Kennfeld zusätzlicher Kraftaufwand am Z-Mover gegenüber unbelasteten Zustandes beim runterfahren

12.3.4 Fehlerpfade

12.3.4.1 PinchProtectionWasActive

Fehler Pinch Protection wurde ausgelöst

PinchProtectionWasActive	
Fehlerindex	<i>EHI_PINCHPROTECTIONWASACTIVE 134</i>
Fehler Status	<i>ErrHndl_stPinchProtectionWasActive</i>
Defektbedingung	Zusatz Kraftaufwand übersteigt limit
Heilbedingung	nicht heilbar
Fehlerklasse	<i>ErrHndl_Class_PinchProtectionWasActive_C</i>
Defekt Entprellzeit	<i>ErrHndl_tiDebDef_PinchProtectionWasActive_C</i>
Heil Entprellzeit	<i>ErrHndl_tiDebOk_PinchProtectionWasActive_C</i>
DTC	0x157013

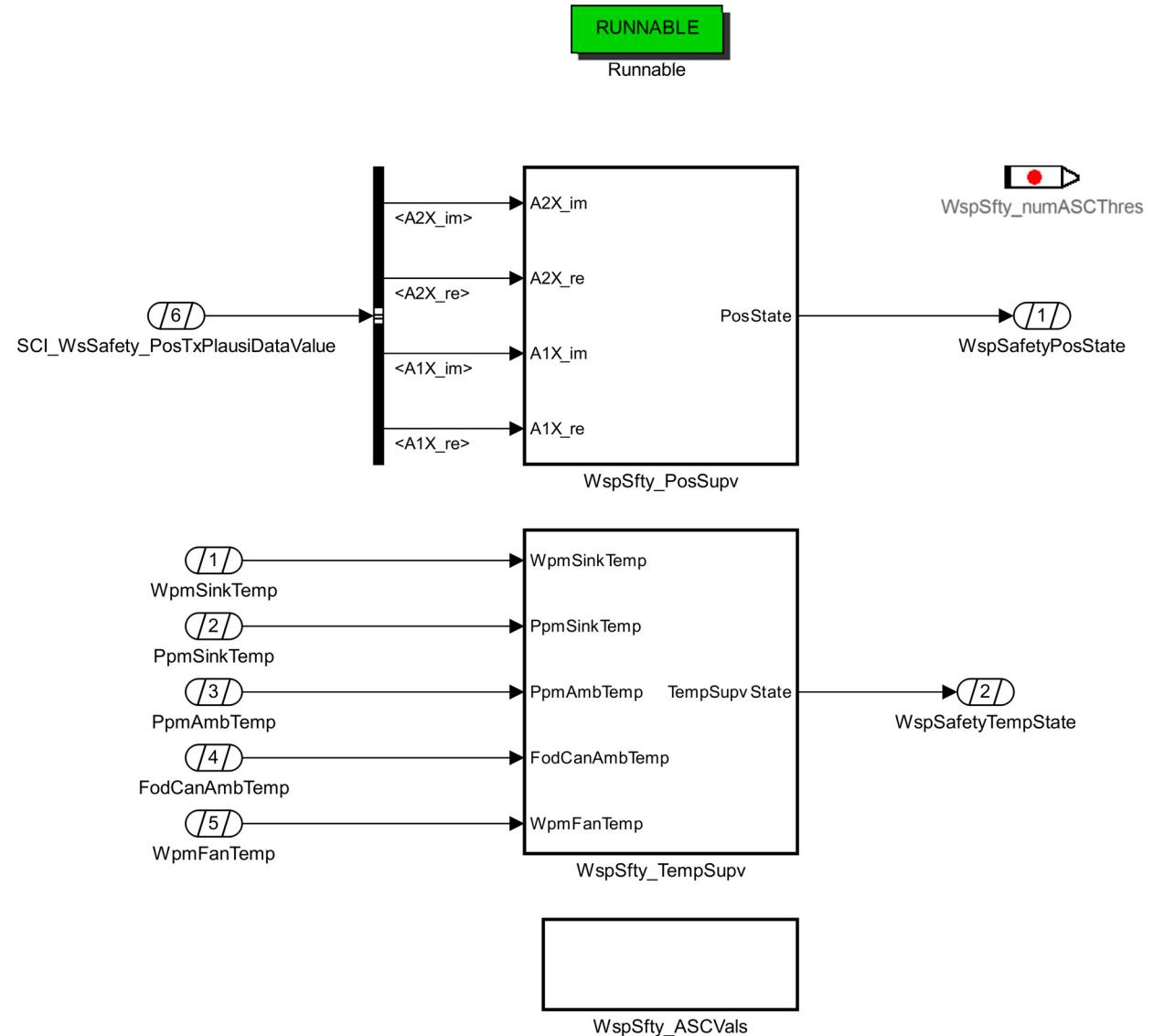
12.3.5 Initialisierung

tbd

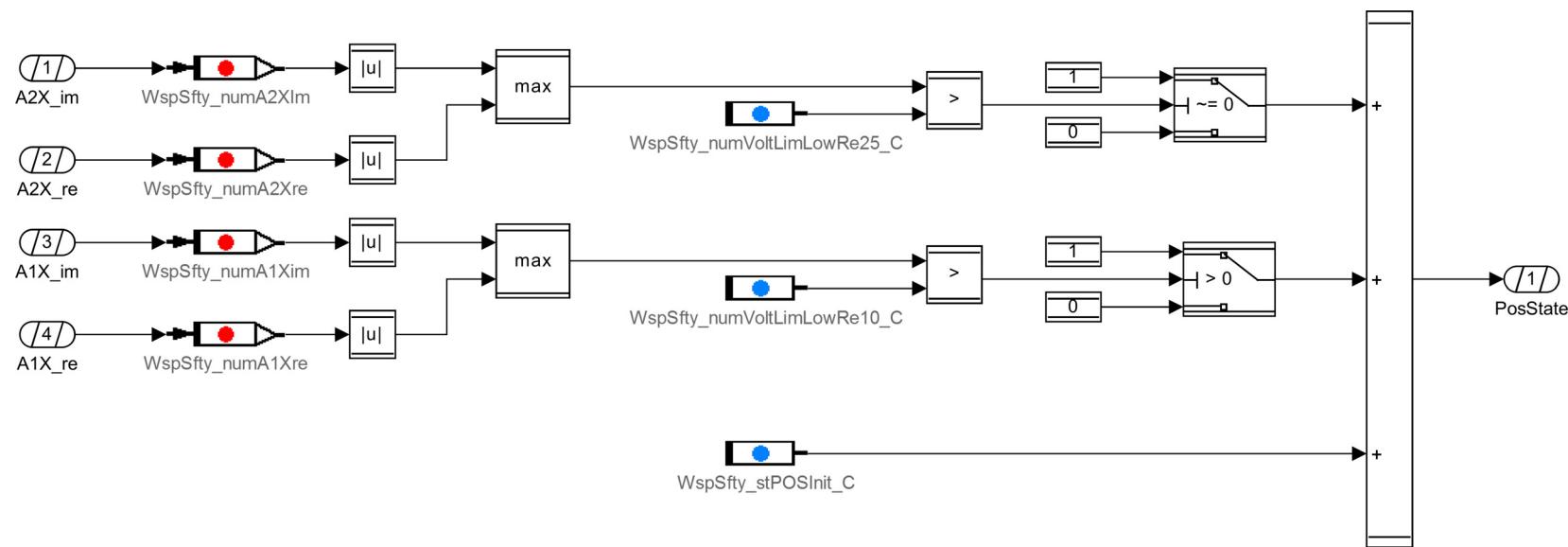
12.4 Komponente WspSfty_Calc

Das Subsystem *WspSfty_Calc* stellt im wesentlichen die Autosar IO - Schnittstellen zur Verfügung (siehe Schnittstellen).

Diese werden an die Subsysteme *WspSfty_PosSupv* (Überwachung des Oru Positionssendersignals) und *WspSfty_TempSupv* (Überwachung der erwarteten Pad / WU sowie Housingtemperaturen weitergegeben bzw. aus diesem ausgelesen.



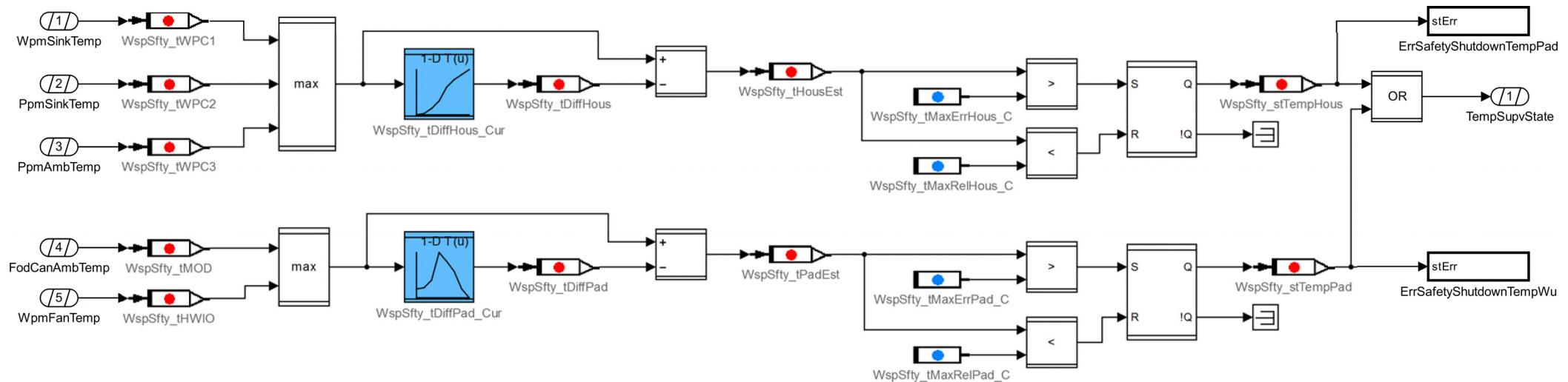
12.4.1 Block WspSfty_PosSupv



In diesem Block wird die Überwachung des Positionssendersignals der Oru durchgeführt. Hierzu wird jeweils der Maximalwert des Betrags der Fernfeld - Antennenwerte `WspSfty_numA1Xim` und `WspSfty_numA1Xre` mit der Schwelle `WspSfty_numVoltLimLowRe10_C` sowie `WspSfty_numA2Xim` und `WspSfty_numA2Xre` mit der Schwelle `WspSfty_numVoltLimLowRe25_C` verglichen.

Für jeden Maximalwert, welcher über dieser Schwelle liegt, wird die Summe `PosState` um 1 erhöht. Der Grundwert dieser Summe ist `WspSfty_stPosInit_C`. Dieser ist per Default = 1. Ist nun `PosState` > 1, geht der Wayside Sfatey Statemanager davon aus, dass der Positionssender der Oru aktiv ist.

12.4.2 Block WspSfty_TempSupv



Hier wird der Erwartungswert der Wayside - Temperaturen des Gehäuses [WspSfty_tHousEst](#) und der Pad-Abdeckung (Winding Unit) [WspSfty_tPadEst](#) bestimmt und auf zulässige Grenzen überwacht.

Für den Schätzwert [WspSfty_tHousEst](#) wird hierzu der Maximalwert der drei Temperaturen [WpmSinkTemp](#) ([WspSfty_tWPC1](#)), [PpmSinkTemp](#) ([WspSfty_tWPC2](#)) sowie [PpmAmbTemp](#) ([WspSfty_tWPC3](#)) bestimmt.

Mittels diesen Maximums wird dann über die Kennlinie [WspSfty_tDifffHous_Cur](#) die erwartete Temperaturdifferenz [WspSfty_tDifffHous](#) der Maximaltemperatur zur Gehäusetemperatur bestimmt. Die erwartete Gehäusetemperatur [WspSfty_tHousEst](#) ergibt sich dann, indem von der Maximaltemperatur die Differenztemperatur [WspSfty_tDifffHous](#) subtrahiert wird.

In gleicher Weise wird für die Erwartungstemperatur der Pad Abdeckung / WU [WspSfty_tPadEst](#) verfahren. Mit dem Maximum der FOD Temperaturen [FodCanAmbTemp](#) ([WspSfty_tMOD](#)) sowie [WpmFanTemp](#) ([WspSfty_tHWIO](#)) wird über die Kennlinie [WspSfty_tDifffPad_Cur](#) die Differenztemperatur [WspSfty_tDifffPad](#) errechnet.

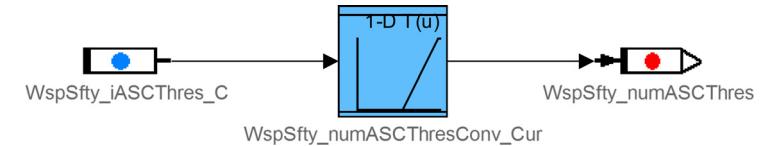
Hiermit wird dann wie zuvor die erwartete Temperatur [WspSfty_tPadEst](#) der Pad Abdeckplatte (WU) bestimmt. Übersteigt nun die erwartete Gehäusetemperatur [WspSfty_tHousEst](#) den Schwellwert [WspSfty_tMaxErrHous_C](#), wird der Fehlerstatus [WspSfty_stTempHous](#) gesetzt. Unterschreitet [WspSfty_tHousEst](#) den Wert [WspSfty_tMaxRelHous_C](#), wird der Fehlerstatus zurückgesetzt.

Der aktuelle Fehlerstatus wird dann dem Fehlerpfad [SafetyShutdownTempPad](#) übergeben. Ebenso wird für den Fehlerstatus [WspSfty_stTempPad](#) verfahren. Überschreitet er den Wert [WspSfty_tMaxErrPad_C](#) wird er gesetzt, unterschreitet er [WspSfty_tMaxRelPad_C](#) wird er zurückgesetzt. Der Fehlerstatus wird dann dem Fehlerpfad [SafetyShutdownTempWu](#) übergeben.

12.4.3 Block WspSfty_ASCVals

Hier wird die Stromschwelle `WspSfty_iASCThres_C` definiert, oberhalb welcher der ASC ausgelöst wird.

Dieser Strom wird über die Kennlinie `WspSfty_numASCThresConv_Cur` in den Zahlenwert `WspSfty_numASCThres` umgerechnet, welcher dem entsprechenden Duty Cycle für die gewünschte Stromschwelle entspricht.



12.4.4 Schnittstellen (Autosar)

Der Autosar Port Type in der Spalte Typ hat folgende Bedeutung:

RPort	Required Port - Autosar Eingangsport von anderem Modul
PPort	Provided Port - Autosar Ausgangsport zu anderem Modul
OperationCall	Aufrufschnittstelle zu andrem Modul, kann Übergabe und / oder Rückgabewerte enthalten

In der Spalte Interface / Argument steht für RPorts und PPorts der Name des Portintefaces und für Operation Calls der Name der Operation Arguments.

12.4.4.1 Eingangsgrößen

SCI_WsSafety_PosTxPlausiDataValue	
WpmSinkTemp	
PpmSinkTemp	
PpmAmbTemp	
FodCanAmbTemp	
WpmFanTemp	

12.4.4.2 Ausgangsgrößen

WspSafetyPosState	
WspSafetyTempState	

12.4.4.3 Interfaces

12.4.5 Messgrößen

<i>WspSfty_numA1Xim</i>	Zahlenwert Imaginärteil aktuellen Amplitude Positions-Antenne 1
<i>WspSfty_numA1Xre</i>	Zahlenwert Realteil aktuellen Amplitude Positions-Antenne 1
<i>WspSfty_numA2Xim</i>	Zahlenwert Imaginärteil aktuellen Amplitude Positions-Antenne 2
<i>WspSfty_numA2Xre</i>	Zahlenwert Realteil aktuellen Amplitude Positions-Antenne 2
<i>WspSfty_stTempHous</i>	Status temperaturüberwachung PAD Gehäuse; 0 = 0k, 1 = Übertemperatur
<i>WspSfty_stTempPad</i>	Status temperaturüberwachung PAD Abdeckplatte; 0 = 0k, 1 = Übertemperatur
<i>WspSfty_tDifffHous</i>	Temperaturdifferenz PAD Gehäuse zu Spulentemperatur - Max(WPC1,WPC2,WPC3)
<i>WspSfty_tDifffPad</i>	Temperaturdifferenz PAD Abdeckplatte zu Umgebungstemperatur MOD
<i>WspSfty_tHousEst</i>	Erwartete maximale PAD Gehäusetemperatur
<i>WspSfty_tPadEst</i>	Erwartete maximale PAD-Abdeckplattentemperatur
<i>WspSfty_tWPC1</i>	Eingangswert Temperatur an WPC1
<i>WspSfty_tWPC2</i>	Eingangswert Temperatur an WPC2
<i>WspSfty_tWPC3</i>	Eingangswert Temperatur an WPC3

12.4.6 Applikationswerte

<i>WspSfty_numVoltLimLowRe10_C</i>	Dezimalwert Realteil unteres Spannungslimit Antenne 10
<i>WspSfty_numVoltLimLowRe25_C</i>	Dezimalwert Realteil unteres Spannungslimit Antenne 25
<i>WspSfty_stPOSInit_C</i>	Initialstatus Pos Supervision if No Signal is detected
<i>WspSfty_tDifffHous_Cur</i>	Temperaturdifferenz Wsp Gehäuse zu WPC Temperaturen
<i>WspSfty_tDifffPad_Cur</i>	Temperaturdifferenz PAD Abdeckplatte zu MOD Umgebungstemperatur
<i>WspSfty_tMaxErrHous_C</i>	Maximal zulässige Gehäusetemperatur PAD Gehäuse
<i>WspSfty_tMaxRelHous_C</i>	Maximal zulässige Freigabetemperatur PAD Gehäuse um Übertemperaturfehler zurück zu setzen.
<i>WspSfty_tMaxErrPad_C</i>	Maximal zulässige Abdeckungstemperatur PAD Gehäuse
<i>WspSfty_tMaxRelPad_C</i>	Maximal zulässige Freigabetemperatur PAD Abdeckung um Übertemperaturfehler zurück zu setzen.

12.4.7 Fehlerpfade

12.4.7.1 SafetyShutdownTempWu

Fehler Temperatur Wsp Abdeckplatte Winding Unit zu hoch

SafetyShutdownTempWu	
Fehlerindex	EHI_SAFETYSHUTDOWNTEMPWU 14
Fehler Status	ErrHndl_stSafetyShutdownTempWu
Defektbedingung	Temperatur in WU > Abschaltschwelle Safety
Heilbedingung	Temperatur in WU <= Abschaltschwelle Safety
Fehlerklasse	ErrHndl_Class_SafetyShutdownTempWu_C
Defekt Entprellzeit	ErrHndl_tiDebDef_SafetyShutdownTempWu_C
Heil Entprellzeit	ErrHndl_tiDebOk_SafetyShutdownTempWu_C
DTC	0x150004

12.4.7.2 SafetyShutdownTempPad

Fehler Temperatur Wsp Gehäuse hoch

SafetyShutdownTempPad	
Fehlerindex	EHI_SAFETYSHUTDOWNTEMPPAD 15
Fehler Status	ErrHndl_stSafetyShutdownTempPad
Defektbedingung	Temperatur in PAD > Abschaltschwelle Safety
Heilbedingung	Temperatur in PAD <= Abschaltschwelle Safety
Fehlerklasse	ErrHndl_Class_SafetyShutdownTempPad_C
Defekt Entprellzeit	ErrHndl_tiDebDef_SafetyShutdownTempPad_C
Heil Entprellzeit	ErrHndl_tiDebOk_SafetyShutdownTempPad_C
DTC	0x15000C

12.4.8 Initialisierung

tbd

Softwaredokumentation

primove
true e-mobility

13 Index DTC's

Autor: Generiert

Softwaredokumentation

AirGapCtrl 21
AirGapCtrl 25
AirGapCtrl_erFilter_C 54
AirGapCtrl_iErrorFilt 51
AirGapCtrl_iErrorRaw 50
AirGapCtrl_iErrorRaw 51
AirGapCtrl_iErSwitchOFF_C 51
AirGapCtrl_iErSwitchON_C 51
AirGapCtrl_iPrimEstim 50
AirGapCtrl_iPrimEstim 50
AirGapCtrl_iPrimEstim_Map 50
AirGapCtrl_Kp_C 50
AirGapCtrl_Kp_C 54
AirGapCtrl_maxNegSpeed_C 54
AirGapCtrl_maxPosSpeed_C 54
AirGapCtrl_psErrorFilt 53
AirGapCtrl_psErrorRaw 53
AirGapCtrl_psErSwitchOFF_C 54
AirGapCtrl_psErSwitchON_C 54
AirGapCtrl_psMeasured 50
AirGapCtrl_psMeasured 53
AirGapCtrl_psRequest 53
AirGapCtrl_vReqZmover 50
AirGapCtrl_vReqZmover 53
B6cdd_uAvrg 50
ChkZeroCrossState 23
Diagnostic 110
DioConf_DioChannel_G3_D_ZM_FAULT_B_WPM_CM_05 150
DioConf_DioChannel_I2_D_ZM_CCW_CM_WPM_01 150
DioConf_DioChannel_I3_D_ZM_CW_CM_WPM_01 150
ErrHndl_Class_B6TempScToGnd_C 165
ErrHndl_Class_B6TempScToSup_C 164
ErrHndl_Class_B6TempTooHigh_C 189
ErrHndl_Class_B6TempTooLow_C 191
ErrHndl_Class_BoostTempTooHigh_C 190
ErrHndl_Class_BoostTempTooLow_C 193

Softwaredokumentation

ErrHndl_Class_BoostVoltTooHigh_C 181
ErrHndl_Class_BoostVoltTooLow_C 181
ErrHndl_Class_CapsTempTooHigh_C 189
ErrHndl_Class_CapsTempTooLow_C 192
ErrHndl_Class_FODOverTemp_C 191
ErrHndl_Class_FODUnderTemp_C 193
ErrHndl_Class_GRIDInputCurrentHigh_C 179
ErrHndl_Class_GRIDInputVoltHigh_C 180
ErrHndl_Class_GRIDInputVoltLow_C 180
ErrHndl_Class_IncorrectCodingResistorDetected_C 167
ErrHndl_Class_OpenLoopCodingResistor_C 168
ErrHndl_Class_P12V0VoltOverVolt_C 160
ErrHndl_Class_P12V0VoltSctoGnd_C 162
ErrHndl_Class_P12V0VoltSctoSup_C 163
ErrHndl_Class_P12V0VoltUnderVolt_C 160
ErrHndl_Class_P3V3SVoltCtoSup_C 162
ErrHndl_Class_P3V3SVoltOverVolt_C 159
ErrHndl_Class_P3V3SVoltSctoGnd_C 162
ErrHndl_Class_P3V3SVoltUnderVolt_C 159
ErrHndl_Class_P5V0VoltOverVolt_C 158
ErrHndl_Class_P5V0VoltSctoGnd_C 161
ErrHndl_Class_P5V0VoltSctoSup_C 161
ErrHndl_Class_P5V0VoltUnderVolt_C 158
ErrHndl_Class_PFCMFTempScToGnd_C 167
ErrHndl_Class_PFCMFTempScToSup_C 166
ErrHndl_Class_PinchProtectionWasActive_C 226
ErrHndl_Class_RFTempScToGnd_C 166
ErrHndl_Class_RFTempScToSup_C 165
ErrHndl_Class_SBCTempScToGnd_C 163
ErrHndl_Class_SBCTempScToSup_C 164
ErrHndl_Class_ShortCircuitCodingResistor_C 168
ErrHndl_Class_WPCTinputPowerHigh_C 179
ErrHndl_Class_WPCTempHigh_C 190
ErrHndl_Class_WPCTempLow_C 192
ErrHndl_FirstShOffErr 17
ErrHndl_LastError 17

Softwaredokumentation

ErrHndl_numClrOne_C 18
ErrHndl_numClrOne_C 18
ErrHndl_numErrors 17
ErrHndl_stB6TempScToGnd 165
ErrHndl_stB6TempScToSup 164
ErrHndl_stB6TempTooHigh 189
ErrHndl_stB6TempTooLow 191
ErrHndl_stBoostTempTooHigh 190
ErrHndl_stBoostTempTooLow 193
ErrHndl_stBoostVoltTooHigh 181
ErrHndl_stBoostVoltTooLow 181
ErrHndl_stCapsTempTooHigh 189
ErrHndl_stCapsTempTooLow 192
ErrHndl_stClrAll_C 18
ErrHndl_stFODOverTemp 191
ErrHndl_stFODUnderTemp 193
ErrHndl_stGlobal 17
ErrHndl_stGRIDInputCurrentHigh 179
ErrHndl_stGRIDInputVoltHigh 180
ErrHndl_stGRIDInputVoltLow 180
ErrHndl_stIncorrectCodingResistorDetected 167
ErrHndl_stOpenLoopCodingResistor 168
ErrHndl_stP12V0VoltOverVolt 159
ErrHndl_stP12V0VoltSctoGnd 162
ErrHndl_stP12V0VoltSctoSup 163
ErrHndl_stP12V0VoltUnderVolt 160
ErrHndl_stP3V3SVoltCtoSup 162
ErrHndl_stP3V3SVoltOverVolt 159
ErrHndl_stP3V3SVoltSctoGnd 161
ErrHndl_stP3V3SVoltUnderVolt 159
ErrHndl_stP5V0VoltOverVolt 158
ErrHndl_stP5V0VoltSctoGnd 160
ErrHndl_stP5V0VoltSctoSup 161
ErrHndl_stP5V0VoltUnderVolt 158
ErrHndl_stPFCMFTempScToGnd 167
ErrHndl_stPFCMFTempScToSup 166

Softwaredokumentation

ErrHndl_stPinchProtectionWasActive 226
ErrHndl_stRFTempScToGnd 166
ErrHndl_stRFTempScToSup 165
ErrHndl_stSBCTempScToGnd 163
ErrHndl_stSBCTempScToSup 164
ErrHndl_stShortCircuitCodingResistor 168
ErrHndl_stWPCInputPowerHigh 179
ErrHndl_stWPCTempHigh 190
ErrHndl_stWPCTempLow 192
ErrHndl_tiDebDef_B6TempScToGnd_C 165
ErrHndl_tiDebDef_B6TempScToSup_C 164
ErrHndl_tiDebDef_B6TempTooHigh_C 189
ErrHndl_tiDebDef_B6TempTooLow_C 191
ErrHndl_tiDebDef_BoostTempTooHigh_C 190
ErrHndl_tiDebDef_BoostTempTooLow_C 193
ErrHndl_tiDebDef_BoostVoltTooHigh_C 181
ErrHndl_tiDebDef_BoostVoltTooLow_C 181
ErrHndl_tiDebDef_CapsTempTooHigh_C 189
ErrHndl_tiDebDef_CapsTempTooLow_C 192
ErrHndl_tiDebDef_FODOverTemp_C 191
ErrHndl_tiDebDef_FODUnderTemp_C 193
ErrHndl_tiDebDef_GRIDInputCurrentHigh_C 179
ErrHndl_tiDebDef_GRIDInputVoltHigh_C 180
ErrHndl_tiDebDef_GRIDInputVoltLow_C 180
ErrHndl_tiDebDef_IncorrectCodingResistorDetected_C 167
ErrHndl_tiDebDef_OpenLoopCodingResistor_C 168
ErrHndl_tiDebDef_P12V0VoltOverVolt_C 160
ErrHndl_tiDebDef_P12V0VoltSctoGnd_C 163
ErrHndl_tiDebDef_P12V0VoltSctoSup_C 163
ErrHndl_tiDebDef_P12V0VoltUnderVolt_C 160
ErrHndl_tiDebDef_P3V3SVoltCtoSup_C 162
ErrHndl_tiDebDef_P3V3SVoltOverVolt_C 159
ErrHndl_tiDebDef_P3V3SVoltSctoGnd_C 162
ErrHndl_tiDebDef_P3V3SVoltUnderVolt_C 159
ErrHndl_tiDebDef_P5V0VoltOverVolt_C 158
ErrHndl_tiDebDef_P5V0VoltSctoGnd_C 161

Softwaredokumentation

ErrHndl_tIDebDef_P5V0VoltSctoSup_C 161
ErrHndl_tIDebDef_P5V0VoltUnderVolt_C 158
ErrHndl_tIDebDef_PFCMFTempScToGnd_C 167
ErrHndl_tIDebDef_PFCMFTempScToSup_C 166
ErrHndl_tIDebDef_PinchProtectionWasActive_C 226
ErrHndl_tIDebDef_RFTempScToGnd_C 166
ErrHndl_tIDebDef_RFTempScToSup_C 165
ErrHndl_tIDebDef_SBCTempScToGnd_C 163
ErrHndl_tIDebDef_SBCTempScToSup_C 164
ErrHndl_tIDebDef_ShortCircuitCodingResistor_C 168
ErrHndl_tIDebDef_WPCTempHigh_C 179
ErrHndl_tIDebDef_WPCTempHigh_C 190
ErrHndl_tIDebDef_WPCTempLow_C 192
ErrHndl_tIDebOk_B6TempScToGnd_C 165
ErrHndl_tIDebOk_B6TempScToSup_C 164
ErrHndl_tIDebOk_B6TempTooHigh_C 189
ErrHndl_tIDebOk_B6TempTooLow_C 191
ErrHndl_tIDebOk_BoostTempTooHigh_C 190
ErrHndl_tIDebOk_BoostTempTooLow_C 193
ErrHndl_tIDebOk_BoostVoltTooHigh_C 181
ErrHndl_tIDebOk_BoostVoltTooLow_C 181
ErrHndl_tIDebOk_CapsTempTooHigh_C 189
ErrHndl_tIDebOk_CapsTempTooLow_C 192
ErrHndl_tIDebOk_FODOverTemp_C 191
ErrHndl_tIDebOk_FODUnderTemp_C 193
ErrHndl_tIDebOk_GRIDInputCurrentHigh_C 179
ErrHndl_tIDebOk_GRIDInputVoltHigh_C 180
ErrHndl_tIDebOk_GRIDInputVoltLow_C 180
ErrHndl_tIDebOk_IncorrectCodingResistorDetected_C 167
ErrHndl_tIDebOk_OpenLoopCodingResistor_C 168
ErrHndl_tIDebOk_P12V0VoltOverVolt_C 160
ErrHndl_tIDebOk_P12V0VoltSctoGnd_C 163
ErrHndl_tIDebOk_P12V0VoltSctoSup_C 163
ErrHndl_tIDebOk_P12V0VoltUnderVolt_C 160
ErrHndl_tIDebOk_P3V3SVoltCtoSup_C 162
ErrHndl_tIDebOk_P3V3SVoltOverVolt_C 159

Softwaredokumentation

ErrHndl_tIDebOk_P3V3SVoltSctoGnd_C 162
ErrHndl_tIDebOk_P3V3SVoltUnderVolt_C 159
ErrHndl_tIDebOk_P5V0VoltOverVolt_C 158
ErrHndl_tIDebOk_P5V0VoltSctoGnd_C 161
ErrHndl_tIDebOk_P5V0VoltSctoSup_C 161
ErrHndl_tIDebOk_P5V0VoltUnderVolt_C 158
ErrHndl_tIDebOk_PFCMFTempScToGnd_C 167
ErrHndl_tIDebOk_PFCMFTempScToSup_C 166
ErrHndl_tIDebOk_PinchProtectionWasActive_C 226
ErrHndl_tIDebOk_RFTempScToGnd_C 166
ErrHndl_tIDebOk_RFTempScToSup_C 165
ErrHndl_tIDebOk_SBCTempScToGnd_C 164
ErrHndl_tIDebOk_SBCTempScToSup_C 164
ErrHndl_tIDebOk_ShortCircuitCodingResistor_C 168
ErrHndl_tIDebOk_WPCTinputPowerHigh_C 179
ErrHndl_tIDebOk_WPCTempHigh_C 190
ErrHndl_tIDebOk_WPCTempLow_C 192
GridCtrl 21
GridCtrl_flagACok 43
GridCtrl_flagACok 46
GridCtrl_flagPwrGood 43
GridCtrl_flagPwrGood 46
GridCtrl_iGrid 43
GridCtrl_iGrid 46
GridCtrl_iLimGrid 43
GridCtrl_iLimGrid 43
GridCtrl_iLimGrid 43
GridCtrl_iLimGrid 46
GridCtrl_iLimGrid_C 47
GridCtrl_kiGrid_C 43
GridCtrl_kiGrid_C 47
GridCtrl_pwrLimGrid 43
GridCtrl_pwrLimGrid 43
GridCtrl_pwrLimGrid 46
GridCtrl_pwrLimGridCurrent 43
GridCtrl_pwrLimGridCurrent 46

Softwaredokumentation

GridCtrl_pwrLimGridVotage 43
GridCtrl_pwrLimGridVotage 46
GridCtrl_resCable 46
GridCtrl_uGrid 43
GridCtrl_uGrid 46
LedMgr_MainFunction 94
OpCtrl 21
PinchProtectionWasActive 200
PinchProtectionWasActive 222
PwmChannel_PWM_ZM_SPEED_CM_WPM_01 150
PwrCtrl 21
PwrCtrl 22
PwrCtrl_effMagnFilt 27
PwrCtrl_effMagnFilt 31
PwrCtrl_effMagnFTC_C 27
PwrCtrl_effMagnFTC_C 32
PwrCtrl_effMagnMax_C 27
PwrCtrl_effMagnMax_C 32
PwrCtrl_effMagnMin_C 27
PwrCtrl_effMagnMin_C 32
PwrCtrl_effMagnRaw 27
PwrCtrl_effMagnRaw 27
PwrCtrl_effMagnRaw 31
PwrCtrl_effMagnSat 23
PwrCtrl_effMagnSat 31
PwrCtrl_effSystemRaw 27
PwrCtrl_effSystemRaw 31
PwrCtrl_frlInverter_Cur 22
PwrCtrl_frlInverter_Cur 32
PwrCtrl_frlReqB6 31
PwrCtrl_iMeas 31
PwrCtrl_iMeasORU 23
PwrCtrl_iMeasORU 31
PwrCtrl_iReqORU 23
PwrCtrl_iReqORU 31
PwrCtrl_pwrGrid 27

Softwaredokumentation

PwrCtrl_pwrGrid 31
PwrCtrl_pwrLoss 31
PwrCtrl_pwrLossKi_C 32
PwrCtrl_pwrLossMax_C 32
PwrCtrl_pwrLossMin_C 32
PwrCtrl_pwrMeasORU 23
PwrCtrl_pwrMeasORU 31
PwrCtrl_pwrOutputMaxEstim 22
PwrCtrl_pwrOutputMaxEstim 23
PwrCtrl_pwrOutputMaxEstim 31
PwrCtrl_pwrReqB6 22
PwrCtrl_pwrReqB6 23
PwrCtrl_pwrReqB6 23
PwrCtrl_pwrReqB6 31
PwrCtrl_pwrReqORU 23
PwrCtrl_pwrReqORU 23
PwrCtrl_pwrReqORU 31
PwrCtrl_pwrReqVariation 31
PwrCtrl_uBattOru 25
PwrCtrl_uBattOru 25
PwrCtrl_uBooster_Cur 23
PwrCtrl_uBooster_Cur 32
PwrCtrl_uMeas 31
PwrCtrl_uMeasORU 23
PwrCtrl_uMeasORU 23
PwrCtrl_uMeasORU 31
PwrCtrl_uReqB6 23
PwrCtrl_uReqB6 31
PwrTrf 21
PwrTrfStMgr_erInhibitMask_C 59
PwrTrfStMgr_erPwrTrf 57
PwrTrfStMgr_erPwrTrf 58
PwrTrfStMgr_erPwrTrf 58
PwrTrfStMgr_pwrDecreaseThr_C 62
PwrTrfStMgr_pwrIncreaseThr_C 62
PwrTrfStMgr_pwrReqB6 25

Softwaredokumentation

PwrTrfStMgr_stPwrTrf 57
PwrTrfStMgr_stPwrTrf 58
PwrTrfStMgr_stPwrTrf 58
PwrTrfStMgr_timeoutDPD_C 61
SafetyShutdownTempPad 230
SafetyShutdownTempWu 230
stLocalErr 83
stPwrTrf 26
stSwtIndicator 23
SwitchCtrl_diRmpDwnOpt_C 26
SwitchCtrl_diRmpUpOpt_C 26
SwitchCtrl_dpwrMaxRmpDwn_C 29
SwitchCtrl_dpwrMaxRmpUp_C 29
SwitchCtrl_facZC_C 24
SwitchCtrl_iMaxOpt_C 25
SwitchCtrl_iMinOpt_C 25
SwitchCtrl_iOptStrt_C 26
SwitchCtrl_iOptStrt_C 26
SwitchCtrl_iPrimPwrReq_Cur 25
SwitchCtrl_iPrimRmpOpt 25
SwitchCtrl_iPrimRmpOpt 26
SwitchCtrl_iPrimSet 22
SwitchCtrl_iPrimSet 25
SwitchCtrl_iPrimSet 25
SwitchCtrl_iPrimSet 50
SwitchCtrl_iPrimSet 50
SwitchCtrl_iPrimVoltBatt 25
SwitchCtrl_iPrimZero_C 25
SwitchCtrl_pwrMax_C 29
SwitchCtrl_pwrMaxFromZC 29
SwitchCtrl_pwrMaxFromZC 29
SwitchCtrl_uBattZero_C 25
SwitchCtrl_zcMeasU 24
SwitchCtrl_zcMeasV 24
SwitchCtrl_zcMeasW 24
SwitchCtrl_zcMinFilt 24

Softwaredokumentation

SwitchCtrl_zcSwitchingInd_C 24
SwitchCtrl_zcSwitchingIndicator 24
SwitchCtrl_zcSwitchingIndicator 24
SwitchCtrl_zcSwitchingIndicator 24
SwitchCtrl_zcSwitchingIndicator 24
SwitchCtrl_zcSwitchingIndicator 24
SwitchCtrl_zcSwitchingOptim_C 24
ThermCtrl 21
ThermCtrl 35
ThermCtrl_errFlagOverTemp 39
ThermCtrl_numFanVolt 35
ThermCtrl_numFanVolt 39
ThermCtrl_numFanVolt_Cur 35
ThermCtrl_numFanVolt_Cur 40
ThermCtrl_pwrLimFodCanAmbTemp 37
ThermCtrl_pwrLimFodCanAmbTemp 39
ThermCtrl_pwrLimFodCanAmbTemp_Cur 37
ThermCtrl_pwrLimFodCanAmbTemp_Cur 40
ThermCtrl_pwrLimPpmAmbTemp 37
ThermCtrl_pwrLimPpmAmbTemp 39
ThermCtrl_pwrLimPpmAmbTemp_Cur 37
ThermCtrl_pwrLimPpmAmbTemp_Cur 40
ThermCtrl_pwrLimPpmSinkTemp 37
ThermCtrl_pwrLimPpmSinkTemp 39
ThermCtrl_pwrLimPpmSinkTemp_Cur 37
ThermCtrl_pwrLimPpmSinkTemp_Cur 40
ThermCtrl_pwrLimTemp 35
ThermCtrl_pwrLimTemp 36
ThermCtrl_pwrLimTemp 36
ThermCtrl_pwrLimTemp 39
ThermCtrl_pwrLimWpmAmbTemp 37
ThermCtrl_pwrLimWpmAmbTemp 39
ThermCtrl_pwrLimWpmAmbTemp_Cur 37
ThermCtrl_pwrLimWpmAmbTemp_Cur 40
ThermCtrl_pwrLimWpmFanTemp 37
ThermCtrl_pwrLimWpmFanTemp 39

Softwaredokumentation

ThermCtrl_pwrLimWpmFanTemp_Cur 37
ThermCtrl_pwrLimWpmFanTemp_Cur 40
ThermCtrl_pwrLimWpmSinkTemp 37
ThermCtrl_pwrLimWpmSinkTemp 39
ThermCtrl_pwrLimWpmSinkTemp_Cur 37
ThermCtrl_pwrLimWpmSinkTemp_Cur 40
ThermCtrl_rateFan 39
ThermCtrl_tFodCanAmb 36
ThermCtrl_tFodCanAmb 39
ThermCtrl_thrOverTemp_C 36
ThermCtrl_thrOverTemp_C 40
ThermCtrl_tMax 35
ThermCtrl_tMax 36
ThermCtrl_tMax 39
ThermCtrl_tPpmAmb 35
ThermCtrl_tPpmAmb 39
ThermCtrl_tPpmSink 35
ThermCtrl_tPpmSink 39
ThermCtrl_tWpmAmb 35
ThermCtrl_tWpmAmb 39
ThermCtrl_tWpmFan 35
ThermCtrl_tWpmFan 39
ThermCtrl_tWpmSink 35
ThermCtrl_tWpmSink 39
VariableWsploHwAbs_stRCDReq 143
WsploHwAb_Calc 141
WsploHwAbs 141
WsploHwAbs 141
WsploHwAbs_facVoltP12V0_C 144
WsploHwAbs_facVoltP12V0_C 156
WsploHwAbs_facVoltP5V0_C 144
WsploHwAbs_facVoltP5V0_C 156
WsploHwAbs_iZMov_Cur 143
WsploHwAbs_iZMov_Cur 156
WsploHwAbs_iZMovRaw 143
WsploHwAbs_iZMovRaw 143

Softwaredokumentation

WsploHwAbs_iZMovRaw 153
WsploHwAbs_numCodeResistor 149
WsploHwAbs_numCodeResistor 153
WsploHwAbs_numDtyZMovVolt_Cur 150
WsploHwAbs_numDtyZMovVolt_Cur 157
WsploHwAbs_numMotorTicks 155
WsploHwAbs_numPeriodVoltZMov_C 150
WsploHwAbs_numPeriodVoltZMov_C 157
WsploHwAbs_prcCpuCurrLoad 153
WsploHwAbs_prcCpuPeakLoad 153
WsploHwAbs_stCodeResErr 149
WsploHwAbs_stCodeResErr 153
WsploHwAbs_stCodeResErr 167
WsploHwAbs_stCodeResErr 167
WsploHwAbs_stCodeResOL 149
WsploHwAbs_stCodeResOL 153
WsploHwAbs_stCodeResSC 148
WsploHwAbs_stCodeResSC 153
WsploHwAbs_stP12V0VoltHi 145
WsploHwAbs_stP12V0VoltHi 153
WsploHwAbs_stP12V0VoltLo 145
WsploHwAbs_stP12V0VoltLo 153
WsploHwAbs_stP12V0VoltSCGnd 153
WsploHwAbs_stP12V0VoltSCSup 153
WsploHwAbs_stP3V3VoltHi 145
WsploHwAbs_stP3V3VoltHi 153
WsploHwAbs_stP3V3VoltLo 145
WsploHwAbs_stP3V3VoltLo 153
WsploHwAbs_stP3V3VoltSCGnd 153
WsploHwAbs_stP3V3VoltSCSup 153
WsploHwAbs_stP5V0VoltHi 145
WsploHwAbs_stP5V0VoltHi 153
WsploHwAbs_stP5V0VoltLo 145
WsploHwAbs_stP5V0VoltLo 153
WsploHwAbs_stP5V0VoltSCGnd 153
WsploHwAbs_stP5V0VoltSCSup 153

Softwaredokumentation

WsploHwAbs_stPFCVoltTempHi 146
WsploHwAbs_stPFCVoltTempHi 153
WsploHwAbs_stPFCVoltTempLo 146
WsploHwAbs_stPFCVoltTempLo 153
WsploHwAbs_stRCDReq 144
WsploHwAbs_stRCDReq 145
WsploHwAbs_stRCDReq 146
WsploHwAbs_stRCDReq 148
WsploHwAbs_stRectVoltTempHi 146
WsploHwAbs_stRectVoltTempHi 153
WsploHwAbs_stRectVoltTempLo 146
WsploHwAbs_stRectVoltTempLo 153
WsploHwAbs_stSBCVoltSCGnd 144
WsploHwAbs_stSBCVoltSCGnd 154
WsploHwAbs_stSBCVoltSCSup 144
WsploHwAbs_stSBCVoltSCSup 154
WsploHwAbs_stWPCVoltTempHi 147
WsploHwAbs_stWPCVoltTempHi 154
WsploHwAbs_stWPCVoltTempLo 147
WsploHwAbs_stWPCVoltTempLo 154
WsploHwAbs_stZMovDirCCW 155
WsploHwAbs_stZMovDirCW 155
WsploHwAbs_stZMovErr 150
WsploHwAbs_stZMovErr 155
WsploHwAbs_tAmb 146
WsploHwAbs_tAmb 154
WsploHwAbs_tAmbPos_Cur 146
WsploHwAbs_tAmbPos_Cur 157
WsploHwAbs_tPfc 146
WsploHwAbs_tPfc 154
WsploHwAbs_tPfcPos_Cur 146
WsploHwAbs_tPfcPos_Cur 157
WsploHwAbs_tRect 146
WsploHwAbs_tRect 154
WsploHwAbs_tRect_Cur 146
WsploHwAbs_tRect_Cur 157

Softwaredokumentation

WsploHwAbs_tWPC 147
WsploHwAbs_tWPC 154
WsploHwAbs_tWpc_Cur 147
WsploHwAbs_tWpc_Cur 157
WsploHwAbs_tWPM2 147
WsploHwAbs_tWPM2 154
WsploHwAbs_tWPM2_Cur 147
WsploHwAbs_tWPM2_Cur 157
WsploHwAbs_uAdcCurrAscThresNeg 154
WsploHwAbs_uAdcCurrAscThresPos 154
WsploHwAbs_uAdcCurrWPCRef 154
WsploHwAbs_uAdcCurrZmWPM 143
WsploHwAbs_uAdcCurrZmWPM 154
WsploHwAbs_uAdcTempAmbPos 146
WsploHwAbs_uAdcTempAmbPos 154
WsploHwAbs_uAdcTempPfcPos 146
WsploHwAbs_uAdcTempPfcPos 154
WsploHwAbs_uAdcTempPfcPos 166
WsploHwAbs_uAdcTempPfcPos 166
WsploHwAbs_uAdcTempPfcPos 167
WsploHwAbs_uAdcTempPfcPos 167
WsploHwAbs_uAdcTempRectPos 146
WsploHwAbs_uAdcTempRectPos 154
WsploHwAbs_uAdcTempRectPos 165
WsploHwAbs_uAdcTempRectPos 165
WsploHwAbs_uAdcTempRectPos 166
WsploHwAbs_uAdcTempRectPos 166
WsploHwAbs_uAdcTempWpcPos 147
WsploHwAbs_uAdcTempWpcPos 154
WsploHwAbs_uAdcTempWpcPos 164
WsploHwAbs_uAdcTempWpcPos 164
WsploHwAbs_uAdcTempWpcPos 165
WsploHwAbs_uAdcTempWpcPos 165
WsploHwAbs_uAdcTempWPM2Pos 147
WsploHwAbs_uAdcTempWPM2Pos 154
WsploHwAbs_uAdcVoltMuxSBC 144

Softwaredokumentation

WsploHwAbs_uAdcVoltMuxSBC 154
WsploHwAbs_uAdcVoltP12V 144
WsploHwAbs_uAdcVoltP12V 154
WsploHwAbs_uAdcVoltP12V 162
WsploHwAbs_uAdcVoltP12V 162
WsploHwAbs_uAdcVoltP12V 163
WsploHwAbs_uAdcVoltP12V 163
WsploHwAbs_uAdcVoltP3V3 154
WsploHwAbs_uAdcVoltP3V3 161
WsploHwAbs_uAdcVoltP3V3 161
WsploHwAbs_uAdcVoltP3V3 162
WsploHwAbs_uAdcVoltP3V3 162
WsploHwAbs_uAdcVoltP5V0 144
WsploHwAbs_uAdcVoltP5V0 154
WsploHwAbs_uAdcVoltP5V0 160
WsploHwAbs_uAdcVoltP5V0 161
WsploHwAbs_uAdcVoltP5V0 161
WsploHwAbs_uAdcVoltP5V0 161
WsploHwAbs_uAdcVoltRNetExtPos 148
WsploHwAbs_uAdcVoltRNetExtPos 148
WsploHwAbs_uAdcVoltRNetExtPos 154
WsploHwAbs_uAdcVoltRNetExtPos 168
WsploHwAbs_uAdcVoltRNetExtPos 168
WsploHwAbs_uAdcVoltRNetExtPos 168
WsploHwAbs_uAdcVoltRNetExtPos 168
WsploHwAbs_uCodeRes1Low_C 149
WsploHwAbs_uCodeRes1Low_C 156
WsploHwAbs_uCodeRes1Up_C 149
WsploHwAbs_uCodeRes1Up_C 156
WsploHwAbs_uCodeRes2Low_C 156
WsploHwAbs_uCodeRes2Up_C 156
WsploHwAbs_uCodeRes3Low_C 156
WsploHwAbs_uCodeRes3Up_C 156
WsploHwAbs_uCodeRes4Low_C 156
WsploHwAbs_uCodeRes4Up_C 156
WsploHwAbs_uCodeRes5Low_C 149

Softwaredokumentation

WsploHwAbs_uCodeRes5Low_C 156
WsploHwAbs_uCodeRes5Up_C 149
WsploHwAbs_uCodeRes5Up_C 156
WsploHwAbs_uCodeResOL_C 149
WsploHwAbs_uCodeResOL_C 156
WsploHwAbs_uCodeResOL_C 168
WsploHwAbs_uCodeResOL_C 168
WsploHwAbs_uCodeResSC_C 148
WsploHwAbs_uCodeResSC_C 156
WsploHwAbs_uCodeResSC_C 168
WsploHwAbs_uCodeResSC_C 168
WsploHwAbs_uP12V0 144
WsploHwAbs_uP12V0 155
WsploHwAbs_uP12V0Hi_C 145
WsploHwAbs_uP12V0Hi_C 156
WsploHwAbs_uP12V0Lo_C 145
WsploHwAbs_uP12V0Lo_C 156
WsploHwAbs_uP12V0SCGnd_C 156
WsploHwAbs_uP12V0SCGnd_C 162
WsploHwAbs_uP12V0SCGnd_C 162
WsploHwAbs_uP12V0SCSup_C 156
WsploHwAbs_uP12V0SCSup_C 163
WsploHwAbs_uP12V0SCSup_C 163
WsploHwAbs_uP3V3 155
WsploHwAbs_uP3V3Hi_C 145
WsploHwAbs_uP3V3Hi_C 156
WsploHwAbs_uP3V3Lo_C 145
WsploHwAbs_uP3V3Lo_C 156
WsploHwAbs_uP3V3SCGnd_C 156
WsploHwAbs_uP3V3SCGnd_C 161
WsploHwAbs_uP3V3SCGnd_C 161
WsploHwAbs_uP3V3SCSup_C 156
WsploHwAbs_uP3V3SCSup_C 162
WsploHwAbs_uP3V3SCSup_C 162
WsploHwAbs_uP5V0 144
WsploHwAbs_uP5V0 155

Softwaredokumentation

WsploHwAbs_uP5V0Hi_C 145
WsploHwAbs_uP5V0Hi_C 157
WsploHwAbs_uP5V0Lo_C 145
WsploHwAbs_uP5V0Lo_C 157
WsploHwAbs_uP5V0SCGnd_C 157
WsploHwAbs_uP5V0SCGnd_C 160
WsploHwAbs_uP5V0SCGnd_C 161
WsploHwAbs_uP5V0SCSup_C 157
WsploHwAbs_uP5V0SCSup_C 161
WsploHwAbs_uP5V0SCSup_C 161
WsploHwAbs_uPFCTempHi_C 146
WsploHwAbs_uPFCTempHi_C 157
WsploHwAbs_uPFCTempHi_C 166
WsploHwAbs_uPFCTempHi_C 166
WsploHwAbs_uPFCTempLo_C 146
WsploHwAbs_uPFCTempLo_C 157
WsploHwAbs_uPFCTempLo_C 167
WsploHwAbs_uPFCTempLo_C 167
WsploHwAbs_uRectTempHi_C 146
WsploHwAbs_uRectTempHi_C 157
WsploHwAbs_uRectTempHi_C 165
WsploHwAbs_uRectTempHi_C 165
WsploHwAbs_uRectTempLo_C 146
WsploHwAbs_uRectTempLo_C 157
WsploHwAbs_uRectTempLo_C 166
WsploHwAbs_uRectTempLo_C 166
WsploHwAbs_uSBC 144
WsploHwAbs_uSBC 155
WsploHwAbs_uSBCGnd_C 144
WsploHwAbs_uSBCGnd_C 157
WsploHwAbs_uSBCSCSup_C 144
WsploHwAbs_uSBCSCSup_C 157
WsploHwAbs_uSetZMov 155
WsploHwAbs_uWPCTempHi_C 147
WsploHwAbs_uWPCTempHi_C 157
WsploHwAbs_uWPCTempHi_C 164

Softwaredokumentation

WspLoHwAbs_uWPCTempHi_C 164
WspLoHwAbs_uWPCTempLo_C 147
WspLoHwAbs_uWPCTempLo_C 157
WspLoHwAbs_uWPCTempLo_C 165
WspLoHwAbs_uWPCTempLo_C 165
WspLed_color 102
WspLed_colorBlue_C 103
WspLed_colorGreen_C 103
WspLed_colorRed_C 103
WspLed_colorWhite_C 103
WspLed_colorYellow_C 103
WspLed_dcBlue 102
WspLed_dcBlue 97
WspLed_dcGreen 102
WspLed_dcGreen 97
WspLed_dcRed 102
WspLed_dcRed 97
WspLed_intensity 102
WspLed_mode 102
WspLed_period 102
WspLed_reqBapColor 102
WspLed_reqBapColor 99
WspLed_reqBapControl 102
WspLed_reqBapControl 99
WspLed_reqBapMode 102
WspLed_reqBapMode 99
WspLed_reqDiagCtrFreeze 102
WspLed_reqDiagCtrFreeze 96
WspLed_reqDiagRGB 102
WspLed_stPfcErrors 102
WspLed_stPfcErrors 98
WspLed_stPwrTrf 102
WspLed_stSafetyErrors 102
WspLed_stSSM 102
WspLed_taskCounter 102
WspLed_time 102

Softwaredokumentation

WspProtec_100ms 170
WspProtec_10ms 170
WspProtec_Grid 172
WspProtect 170
WspProtect 170
WspProtect_frqGrid 173
WspProtect_frqGridHi_CA 173
WspProtect_frqGridHi_CA 178
WspProtect_frqGridLo_CA 173
WspProtect_frqGridLo_CA 178
WspProtect_iGridAC 172
WspProtect_iGridAC 172
WspProtect_iGridAC 172
WspProtect_iGridAC 177
WspProtect_iGridAC 179
WspProtect_iGridAC 179
WspProtect_iGridMaxOff_C 172
WspProtect_iGridMaxOff_C 179
WspProtect_iGridMaxOff_CA 178
WspProtect_iGridMaxOn_C 172
WspProtect_iGridMaxOn_C 179
WspProtect_iGridMaxOn_CA 178
WspProtect_iGridUnPlaus_C 172
WspProtect_iZMover 175
WspProtect_iZMover 177
WspProtect_iZMoverMaxOff_C 175
WspProtect_iZMoverMaxOff_C 178
WspProtect_iZMoverMaxOn_C 175
WspProtect_iZMoverMaxOn_C 178
WspProtect_numCodeResistor 172
WspProtect_numCodeResistor 177
WspProtect_pwrGridAC 172
WspProtect_pwrGridAC 177
WspProtect_pwrGridAC 179
WspProtect_pwrGridAC 179
WspProtect_pwrGridMaxOff_C 179

WspProtect_pwrGridMaxOff_CA 172
WspProtect_pwrGridMaxOff_CA 178
WspProtect_pwrGridMaxOn_C 179
WspProtect_pwrGridMaxOn_CA 172
WspProtect_pwrGridMaxOn_CA 178
WspProtect_stBoostMaxTempErr 184
WspProtect_stBoostMaxTempErr 187
WspProtect_stBoostMinTempErr 185
WspProtect_stBoostMinTempErr 187
WspProtect_stBoostVoltMaxErr 174
WspProtect_stBoostVoltMaxErr 177
WspProtect_stBoostVoltMinErr 174
WspProtect_stBoostVoltMinErr 177
WspProtect_stFODMaxTempErr 184
WspProtect_stFODMaxTempErr 187
WspProtect_stFODMinTempErr 185
WspProtect_stFODMinTempErr 187
WspProtect_stGridCurMaxErr 172
WspProtect_stGridCurMaxErr 177
WspProtect_stGridCurUnPlaus 172
WspProtect_stGridFreqMaxErr 173
WspProtect_stGridFreqMaxErr 177
WspProtect_stGridFreqMinErr 173
WspProtect_stGridFreqMinErr 177
WspProtect_stGridPwrMaxErr 172
WspProtect_stGridPwrMaxErr 177
WspProtect_stGridVoltMaxErr 173
WspProtect_stGridVoltMaxErr 177
WspProtect_stGridVoltMinErr 173
WspProtect_stGridVoltMinErr 177
WspProtect_stStrtUp 177
WspProtect_stStrtUpDly 174
WspProtect_stStrtUpDly 177
WspProtect_stWPC1MaxTempErr 184
WspProtect_stWPC1MaxTempErr 187
WspProtect_stWPC1MinTempErr 185

Softwaredokumentation

WspProtect_stWPC1MinTempErr 187
WspProtect_stWPC2MaxTempErr 184
WspProtect_stWPC2MaxTempErr 187
WspProtect_stWPC2MinTempErr 185
WspProtect_stWPC2MinTempErr 187
WspProtect_stWPC3MaxTempErr 184
WspProtect_stWPC3MaxTempErr 187
WspProtect_stWPC3MinTempErr 185
WspProtect_stWPC3MinTempErr 187
WspProtect_stZMoverCurMaxErr 175
WspProtect_stZMoverCurMaxErr 177
WspProtect_tBoost 184
WspProtect_tBoost 185
WspProtect_tBoost 187
WspProtect_tBoost 190
WspProtect_tBoost 190
WspProtect_tBoost 193
WspProtect_tBoost 193
WspProtect_tBoostMaxOff_C 184
WspProtect_tBoostMaxOff_C 188
WspProtect_tBoostMaxOff_C 190
WspProtect_tBoostMaxOn_C 184
WspProtect_tBoostMaxOn_C 188
WspProtect_tBoostMaxOn_C 190
WspProtect_tBoostMinOff_C 174
WspProtect_tBoostMinOff_C 185
WspProtect_tBoostMinOff_C 188
WspProtect_tBoostMinOff_C 193
WspProtect_tBoostMinOn_C 174
WspProtect_tBoostMinOn_C 185
WspProtect_tBoostMinOn_C 188
WspProtect_tBoostMinOn_C 193
WspProtect_tFOD 184
WspProtect_tFOD 185
WspProtect_tFOD 187
WspProtect_tFOD 191

Softwaredokumentation

WspProtect_tFOD 191
WspProtect_tFOD 193
WspProtect_tFOD 193
WspProtect_tFODMaxOff_C 184
WspProtect_tFODMaxOff_C 188
WspProtect_tFODMaxOff_C 191
WspProtect_tFODMaxOn_C 184
WspProtect_tFODMaxOn_C 188
WspProtect_tFODMaxOn_C 191
WspProtect_tFODMinOff_C 185
WspProtect_tFODMinOff_C 188
WspProtect_tFODMinOff_C 193
WspProtect_tFODMinOn_C 185
WspProtect_tFODMinOn_C 188
WspProtect_tFODMinOn_C 193
WspProtect_tiGrid 173
WspProtect_tiGrid 177
WspProtect_tiGridNorm 173
WspProtect_tiGridNorm 177
WspProtect_tiStrtUpDly_C 174
WspProtect_tiStrtUpDly_C 178
WspProtect_tWPC1 184
WspProtect_tWPC1 185
WspProtect_tWPC1 187
WspProtect_tWPC1 189
WspProtect_tWPC1 189
WspProtect_tWPC1 191
WspProtect_tWPC1 191
WspProtect_tWPC1MaxOff_C 184
WspProtect_tWPC1MaxOff_C 188
WspProtect_tWPC1MaxOff_C 189
WspProtect_tWPC1MaxOn_C 184
WspProtect_tWPC1MaxOn_C 188
WspProtect_tWPC1MaxOn_C 189
WspProtect_tWPC1MinOff_C 185
WspProtect_tWPC1MinOff_C 188

Softwaredokumentation

WspProtect_tWPC1MinOff_C 191
WspProtect_tWPC1MinOn_C 185
WspProtect_tWPC1MinOn_C 188
WspProtect_tWPC1MinOn_C 191
WspProtect_tWPC2 184
WspProtect_tWPC2 185
WspProtect_tWPC2 187
WspProtect_tWPC2 189
WspProtect_tWPC2 189
WspProtect_tWPC2 192
WspProtect_tWPC2 192
WspProtect_tWPC2MaxOff_C 184
WspProtect_tWPC2MaxOff_C 188
WspProtect_tWPC2MaxOff_C 189
WspProtect_tWPC2MaxOn_C 184
WspProtect_tWPC2MaxOn_C 188
WspProtect_tWPC2MaxOn_C 189
WspProtect_tWPC2MinOff_C 185
WspProtect_tWPC2MinOff_C 188
WspProtect_tWPC2MinOff_C 192
WspProtect_tWPC2MinOn_C 185
WspProtect_tWPC2MinOn_C 188
WspProtect_tWPC2MinOn_C 192
WspProtect_tWPC3 184
WspProtect_tWPC3 185
WspProtect_tWPC3 187
WspProtect_tWPC3 190
WspProtect_tWPC3 190
WspProtect_tWPC3 192
WspProtect_tWPC3 192
WspProtect_tWPC3MaxOff_C 184
WspProtect_tWPC3MaxOff_C 188
WspProtect_tWPC3MaxOff_C 190
WspProtect_tWPC3MaxOn_C 184
WspProtect_tWPC3MaxOn_C 188
WspProtect_tWPC3MaxOn_C 190

Softwaredokumentation

WspProtect_tWPC3MinOff_C 185
WspProtect_tWPC3MinOff_C 188
WspProtect_tWPC3MinOff_C 192
WspProtect_tWPC3MinOn_C 185
WspProtect_tWPC3MinOn_C 188
WspProtect_tWPC3MinOn_C 192
WspProtect_uBoost 174
WspProtect_uBoost 177
WspProtect_uBoost 181
WspProtect_uBoost 181
WspProtect_uBoost 181
WspProtect_uBoost 181
WspProtect_uBoostMaxOff_C 174
WspProtect_uBoostMaxOff_C 178
WspProtect_uBoostMaxOff_C 181
WspProtect_uBoostMaxOn_C 174
WspProtect_uBoostMaxOn_C 178
WspProtect_uBoostMaxOn_C 181
WspProtect_uBoostMinOff_C 178
WspProtect_uBoostMinOff_C 181
WspProtect_uBoostMinOn_C 178
WspProtect_uBoostMinOn_C 181
WspProtect_uGridAC 172
WspProtect_uGridAC 173
WspProtect_uGridAC 177
WspProtect_uGridAC 180
WspProtect_uGridAC 180
WspProtect_uGridAC 180
WspProtect_uGridMaxOff_C 173
WspProtect_uGridMaxOff_C 180
WspProtect_uGridMaxOff_CA 178
WspProtect_uGridMaxOn_C 173
WspProtect_uGridMaxOn_C 180
WspProtect_uGridMaxOn_CA 178
WspProtect_uGridMinOff_C 180

WspProtect_uGridMinOff_CA 173
WspProtect_uGridMinOff_CA 178
WspProtect_uGridMinOn_C 180
WspProtect_uGridMinOn_CA 173
WspProtect_uGridMinOn_CA 178
WspSfMgr 71
WspSfMgr 71
WspSfMgr 71
WspSfty 196
WspSfty 196
WspSfty_Calc 196
WspSfty_Calc 228
WspSfty_dnumZMoverSpd 222
WspSfty_dnumZMoverSpd 224
WspSfty_dnumZMoverSpdRaw 222
WspSfty_dnumZMoverSpdRaw 224
WspSfty_facLpZMoverAcc_C 222
WspSfty_facLpZMoverAcc_C 225
WspSfty_facLpZMoverCurr_C 221
WspSfty_facLpZMoverCurr_C 225
WspSfty_fAddZMover 222
WspSfty_fAddZMover 222
WspSfty_fAddZMover 222
WspSfty_fAddZMover 224
WspSfty_fAddZMoverDwn_Map 222
WspSfty_fAddZMoverDwn_Map 225
WspSfty_fAddZMoverUp_Map 222
WspSfty_fAddZMoverUp_Map 225
WspSfty_fMaxZMoverAdd_C 222
WspSfty_fMaxZMoverAdd_C 225
WspSfty_iASCThres_C 231
WspSfty_iZMover 221
WspSfty_iZMover 224
WspSfty_iZMoverFit 221
WspSfty_iZMoverFit 224
WspSfty_lFodChkPosXMin_C 204

Softwaredokumentation

WspSfty_lFodChkPosXMin_C 205
WspSfty_lFodChkPosXMin_C 218
WspSfty_lFodChkPosYMin_C 205
WspSfty_lFodChkPosYMin_C 205
WspSfty_lFodChkPosYMin_C 218
WspSfty_lPosX 204
WspSfty_lPosX 205
WspSfty_lPosX 215
WspSfty_lPosXOffs_C 218
WspSfty_lPosY 205
WspSfty_lPosY 205
WspSfty_lPosY 215
WspSfty_lPosYOffs_C 218
WspSfty_numA1Xim 229
WspSfty_numA1Xim 233
WspSfty_numA1Xre 229
WspSfty_numA1Xre 233
WspSfty_numA2Xim 229
WspSfty_numA2XIm 233
WspSfty_numA2Xre 229
WspSfty_numA2Xre 233
WspSfty_numASCThres 215
WspSfty_numASCThres 231
WspSfty_numASCThresConv_Cur 231
WspSfty_numFodCheckDuringChrg_C 200
WspSfty_numFodCheckDuringChrg_C 218
WspSfty_numFodCheckStartUpChrg 218
WspSfty_numFodCheckZMDown_C 218
WspSfty_numFodCheckZMUp_C 218
WspSfty_numFodPreCheck 200
WspSfty_numFodPreCheck_C 218
WspSfty_numVDDisVal_C 201
WspSfty_numVDDisVal_C 218
WspSfty_numVoltLimLowRe10_C 229
WspSfty_numVoltLimLowRe10_C 234
WspSfty_numVoltLimLowRe25_C 229

Softwaredokumentation

WspSfty_numVoltLimLowRe25_C 234
WspSfty_numZMoverPos 221
WspSfty_numZMoverPos 224
WspSfty_numZMoverSpd 222
WspSfty_numZMoverSpd 222
WspSfty_numZMoverSpd 224
WspSfty_Pinch 196
WspSfty_PosSupv 228
WspSfty_stASCDet 209
WspSfty_stASCDet 215
WspSfty_stASCDetInv 215
WspSfty_stASCReset 203
WspSfty_stASCReset 209
WspSfty_stASCReset 215
WspSfty_stASCResetInv 215
WspSfty_State 196
WspSfty_State 197
WspSfty_StateTL 197
WspSfty_stDisable 203
WspSfty_stDisable 209
WspSfty_stDisable 215
WspSfty_stDisableInv 215
WspSfty_stE2EChkFail 215
WspSfty_stFodCheckDuringChrg 215
WspSfty_stFodCheckDuringChrgInv 216
WspSfty_stFodCheckStartUpChrg 208
WspSfty_stFodCheckStartUpChrg 208
WspSfty_stFodCheckStartUpChrg 215
WspSfty_stFodCheckStartUpChrgInv 216
WspSfty_stFodCheckZMDown 206
WspSfty_stFodCheckZMDown 215
WspSfty_stFodCheckZMDownInv 216
WspSfty_stFodCheckZMDwn 206
WspSfty_stFodCheckZMUp 207
WspSfty_stFodCheckZMUp 207
WspSfty_stFodCheckZMUp 215

Softwaredokumentation

WspSfty_stFodCheckZMUpInv 216
WspSfty_stFodDly 204
WspSfty_stFodDly 205
WspSfty_stFodDly 215
WspSfty_stFodPreCheck 205
WspSfty_stFodPreCheck 205
WspSfty_stFodPreCheck 215
WspSfty_stFodPreCheckInv 216
WspSfty_stMgrErr 213
WspSfty_stMgrErr 213
WspSfty_stMgrErr 215
WspSfty_stOruSftyMgr 204
WspSfty_stOruSftyMgr 216
WspSfty_stPinch 222
WspSfty_stPinch 224
WspSfty_stPinchErr 200
WspSfty_stPinchErr 207
WspSfty_stPinchErr 208
WspSfty_stPinchErr 215
WspSfty_stPosInit_C 229
WspSfty_stPOSInit_C 234
WspSfty_stPosSupervision 204
WspSfty_stPosSupervision 216
WspSfty_stRCDRdy 212
WspSfty_stRCDRdy 216
WspSfty_stRCDReq 212
WspSfty_stRCDReq 216
WspSfty_stSafety 199
WspSfty_stSafety 199
WspSfty_stSafety 203
WspSfty_stSafety 215
WspSfty_stSigASCDet 216
WspSfty_stSigFodChkDurChrg 198
WspSfty_stSigFodChkDurChrg 216
WspSfty_stSigFodChkStrtChrg 216
WspSfty_stSigFodChkZMDwn 216

Softwaredokumentation

WspSfty_stSigFodChkZMUp 216
WspSfty_stSigFodPreChk 216
WspSfty_stSigVDet 216
WspSfty_stSysMgr 204
WspSfty_stSysMgr 215
WspSfty_stTempHous 216
WspSfty_stTempHous 230
WspSfty_stTempHous 233
WspSfty_stTempPad 216
WspSfty_stTempPad 230
WspSfty_stTempPad 233
WspSfty_stTempSupvis 204
WspSfty_stTempSupvis 215
WspSfty_stVDDet 208
WspSfty_stVDDet 215
WspSfty_stVDetInv 216
WspSfty_stZMoverDisInv 216
WspSfty_stZMoverEna 204
WspSfty_stZMoverEna 215
WspSfty_swtDisableVD_C 201
WspSfty_swtDisableVD_C 218
WspSfty_swtSimFodCheckDuringChrg_C 200
WspSfty_swtSimFodCheckDuringChrg_C 218
WspSfty_swtSimFodCheckStartUpChrg_C 200
WspSfty_swtSimFodCheckStartUpChrg_C 218
WspSfty_swtSimFodCheckZMDown_C 200
WspSfty_swtSimFodCheckZMDown_C 218
WspSfty_swtSimFodCheckZMUp_C 200
WspSfty_swtSimFodCheckZMUp_C 218
WspSfty_swtSimFodPreCheck_C 200
WspSfty_swtSimFodPreCheck_C 218
WspSfty_tDifffHous 216
WspSfty_tDifffHous 230
WspSfty_tDifffHous 230
WspSfty_tDifffHous 233
WspSfty_tDifffHous_Cur 230

Softwaredokumentation

WspSfty_tDiffHous_Cur 234
WspSfty_tDifPad 216
WspSfty_tDifPad 230
WspSfty_tDifPad 233
WspSfty_tDifPad_Cur 230
WspSfty_tDifPad_Cur 234
WspSfty_TempSupv 228
WspSfty_tHousEst 216
WspSfty_tHousEst 230
WspSfty_tHousEst 230
WspSfty_tHousEst 230
WspSfty_tHousEst 230
WspSfty_tHousEst 230
WspSfty_tHWIO 216
WspSfty_tHWIO 230
WspSfty_tiDebPinch_C 222
WspSfty_tiDebPinch_C 225
WspSfty_tiDebSigErr_C 198
WspSfty_tiRCDReStrt 212
WspSfty_tiRCDReStrt 217
WspSfty_tiRCDTstMax_C 212
WspSfty_tMaxErrHous_C 230
WspSfty_tMaxErrHous_C 234
WspSfty_tMaxErrPad_C 230
WspSfty_tMaxErrPad_C 234
WspSfty_tMaxRelHous_C 230
WspSfty_tMaxRelHous_C 234
WspSfty_tMaxRelPad_C 234
WspSfty_tMOD 230
WspSfty_tPadEst 230
WspSfty_tPadEst 230
WspSfty_tPadEst 230
WspSfty_tPadEst 233
WspSfty_tWPC1 230
WspSfty_tWPC1 233

WspSfty_tWPC2 230
WspSfty_tWPC2 233
WspSfty_tWPC3 230
WspSfty_tWPC3 233
WspStMgr 72
WspStMgr_10ms 72
WspStMgr_dbRSSILvl 90
WspStMgr_dbRSSILvlFlt 90
WspStMgr_dbRSSILvlMax 90
WspStMgr_dbRSSIThresMin_C 91
WspStMgr_dbZMvInitThres_C 91
WspStMgr_facLpRSSI_C 91
WspStMgr_numModCmdOn_C 80
WspStMgr_numModCmdOn_C 91
WspStMgr_numReqZMovPos 90
WspStMgr_numReqZMovRate 90
WspStMgr_stAcReqStMgr 79
WspStMgr_stAcReqStMgr 84
WspStMgr_stAcReqStMgr 90
WspStMgr_stActvCharging 85
WspStMgr_stActWS 74
WspStMgr_stActWS 74
WspStMgr_stActWS 78
WspStMgr_stActWS 78
WspStMgr_stActWS 79
WspStMgr_stActWS 90
WspStMgr_stBLE 78
WspStMgr_stBLE 90
WspStMgr_stBLESim_C 91
WspStMgr_stEna12VFOD 78
WspStMgr_stEna12VFOD 90
WspStMgr_stError 73
WspStMgr_stError 75
WspStMgr_stError 76
WspStMgr_stError 90
WspStMgr_stMODCmd 78

Softwaredokumentation

WspStMgr_stMODCmd 78
WspStMgr_stMODCmd 90
WspStMgr_stParked 85
WspStMgr_stParked 90
WspStMgr_stProtInh 79
WspStMgr_stProtInh 90
WspStMgr_stPwrTrf 76
WspStMgr_stPwrTrf 79
WspStMgr_stPwrTrf 90
WspStMgr_stReqWs 76
WspStMgr_stReqWs 78
WspStMgr_stReqWs 90
WspStMgr_stRssiPlaus 84
WspStMgr_stRssiPlaus 90
WspStMgr_stSfty 76
WspStMgr_stSfty 90
WspStMgr_stZMOpCtrl 90
WspStMgr_stZMvInitPos 84
WspStMgr_stZMvInitPos 90
WspStMgr_swtBLE2Sim_C 91
WspStMgr_swtDisInhb_C 91
WspStMgr_tiDebDefRssiPlaus_C 91
WspStMgr_tiDebOkRssiPlaus_C 91
WspStMgr_vReqStMgr 84
WspStMgr_vReqStMgr 90
WspZMovCdd_dirCCW 150
WspZMovCdd_dirCW 150
ZMControl 107
ZMDrive 110
ZMDrive_i 123
ZMDrive_i 123
ZMDrive_i 133
ZMDrive_iFTC_C 123
ZMDrive_iFTC_C 135
ZMDrive_iOffset 123
ZMDrive_iOffset 123

Softwaredokumentation

ZMDrive_iOffset 133
ZMDrive_iOffsetTooHigh_C 124
ZMDrive_iOffsetTooHigh_C 135
ZMDrive_iRaw 123
ZMDrive_iRaw 123
ZMDrive_iRaw 133
ZMDrive_iTooHigh_C 124
ZMDrive_iTooHigh_C 135
ZMDrive_J_C 125
ZMDrive_J_C 135
ZMDrive_k1ProtTherm_C 135
ZMDrive_k2ProtTherm_C 135
ZMDrive_kAntiW_C 135
ZMDrive_kF_C 125
ZMDrive_kF_C 135
ZMDrive_kl_C 127
ZMDrive_kl_C 135
ZMDrive_kMotor_C 125
ZMDrive_kMotor_C 135
ZMDrive_kP_C 127
ZMDrive_kP_C 135
ZMDrive_loadEstim 125
ZMDrive_loadEstim 133
ZMDrive_loadMaxNeg_C 125
ZMDrive_loadMaxNeg_C 135
ZMDrive_loadMaxPos_C 125
ZMDrive_loadMaxPos_C 135
ZMDrive_stCCW 127
ZMDrive_stCCW 127
ZMDrive_stCCW 127
ZMDrive_stCCW 133
ZMDrive_stCW 127
ZMDrive_stCW 127
ZMDrive_stCW 133
ZMDrive_stMotorON 127

Softwaredokumentation

ZMDrive_stMotorON 127
ZMDrive_stMotorON 127
ZMDrive_stMotorON 133
ZMDrive_stNegOverload 125
ZMDrive_stNegOverload 126
ZMDrive_stNegOverload 133
ZMDrive_stPosOverload 125
ZMDrive_stPosOverload 126
ZMDrive_stPosOverload 133
ZMDrive_tErResetThr_C 135
ZMDrive_tErSetThr_C 135
ZMDrive_tEstrm 133
ZMDrive_ticks 120
ZMDrive_ticks 133
ZMDrive_TicksMax 121
ZMDrive_ticksMax 121
ZMDrive_ticksMax 133
ZMDrive_ticksMin 121
ZMDrive_ticksMin 121
ZMDrive_ticksMin 133
ZMDrive_trobsK1_C 135
ZMDrive_trobsK2_C 135
ZMDrive_u 127
ZMDrive_u 127
ZMDrive_u 133
ZMDrive_uMaxNeg_C 135
ZMDrive_uMaxNegOverload_C 126
ZMDrive_uMaxNegOverload_C 135
ZMDrive_uMaxOverTemp_C 135
ZMDrive_uMaxPos_C 135
ZMDrive_uMaxPosOverload_C 126
ZMDrive_uMaxPosOverload_C 135
ZMDrive_vFilt 120
ZMDrive_vFilt 133
ZMDrive_vManualOperation_C 122
ZMDrive_vManualOperation_C 133

ZMDrive_vRaw 120
ZMDrive_vRaw 133
ZMDrive_vRawStoppedThr_C 123
ZMDrive_vRawStoppedThr_C 135
ZMDrive_vReq 127
ZMDrive_vReq 133
ZMManual_stON_C 135
ZMManual_stReqDiag_C 136
ZMManual_stReqPwrTrf_C 136
ZMManual_stReqStMng_C 136
ZMManual_stVoltageDirect_C 127
ZMManual_stVoltageDirect_C 127
ZMManual_stVoltageDirect_C 136
ZMManual_u_C 127
ZMManual_u_C 136
ZMManual_vReqDiag_C 136
ZMManual_vReqPwrTrf_C 136
ZMManual_vReqStMng_C 136
ZMMechanics 110
ZMMechanics_Cur 129
ZMMechanics_Cur 136
ZMMechanics_hOffsetPAD_C 129
ZMMechanics_hOffsetPAD_C 136
ZMMechanics_hPad 129
ZMMechanics_hPAD 133
ZMMechanics_stParked 129
ZMMechanics_stParked 129
ZMMechanics_stParked 133
ZMMechanics_stPos1 129
ZMMechanics_stPos1 133
ZMMechanics_stPos2 133
ZMMotor_vFiltStoppedThr_C 136
ZMMotor_vFltStoppedThr_C 123
ZMNechanics_stPos2 129
ZMOpCtrl 110
ZMOpCtrl_cntr 133

ZMOpCtrl_cntr 66
ZMOpCtrl_erComponent 133
ZMOpCtrl_erComponent 66
ZMOpCtrl_erDrive 121
ZMOpCtrl_erDrive 122
ZMOpCtrl_erDrive 124
ZMOpCtrl_erDrive 126
ZMOpCtrl_erDrive 126
ZMOpCtrl_st 112
ZMOpCtrl_st 114
ZMOpCtrl_st 115
ZMOpCtrl_st 116
ZMOpCtrl_st 116
ZMOpCtrl_st 117
ZMOpCtrl_st 117
ZMOpCtrl_st 118
ZMOpCtrl_st 118
ZMOpCtrl_stReqDiag 112
ZMOpCtrl_stReqDiag 134
ZMOpCtrl_stReqDiag 66
ZMOpCtrl_stReqPwrTrf 134
ZMOpCtrl_stReqPwrTrf 66
ZMOpCtrl_stReqStMgr 116
ZMOpCtrl_stReqStMgr 134
ZMOpCtrl_stReqStMgr 66
ZMOpCtrl_vReqDiag 118
ZMOpCtrl_vReqDiag 134
ZMOpCtrl_vReqDiag 66
ZMOpCtrl_vReqPwrTrf 117
ZMOpCtrl_vReqPwrTrf 134
ZMOpCtrl_vReqPwrTrf 66
ZMOpCtrl_vReqStMgr 110
ZMOpCtrl_vReqStMgr 118
ZMOpCtrl_vReqStMgr 134
ZMOpCtrl_vReqStMgr 66

Softwaredokumentation

primove
true e-mobility

ZMover 107
ZMover_delayPowerUp 114
ZMover_stCalib 134
ZMover_timeoutPark 114
ZMover_uSet 134

1 Index DTC's

Autor: Generiert

Softwaredokumentation

104
137
219
33
41
48
55
68
91
0x100005 191
0x100006 193
0x120000 193
0x120001 190
0x120004 191
0x120005 189
0x120006 192
0x120007 189
0x120008 165
0x120009 165
0x12000D 166
0x12000E 165
0x120012 167
0x120013 166
0x12001C 190
0x12001D 192
0x121000 181
0x121001 181
0x121004 180
0x121006 180
0x12200F 179
0x128000 179
0x150001 164
0x150002 164
0x150004 235
0x15000C 235
0x151002 163

Softwaredokumentation

0x151003 163
0x151005 160
0x151006 160
0x151012 161
0x151013 161
0x151015 158
0x151016 158
0x15101A 159
0x15101B 159
0x151027 162
0x151028 162
0x153022 167
0x153023 168
0x153024 168
0x157013 226