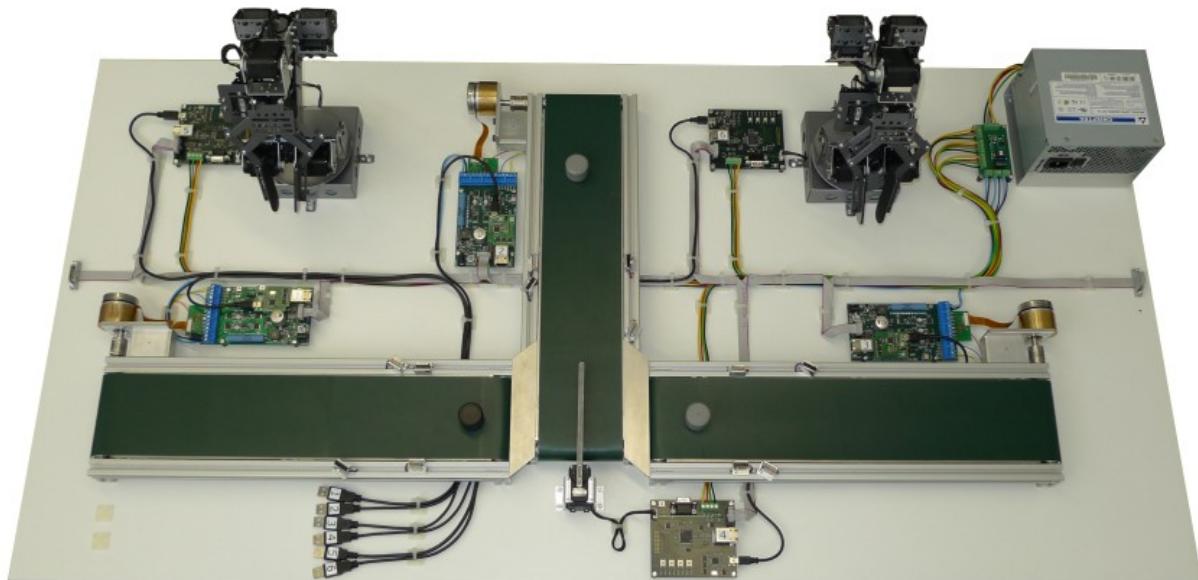


Berner Fachhochschule
Technik und Informatik
Fachbereich Elektro- und Kommunikationstechnik
Labor für Technische Informatik

Stellaris Roboter Modell



© 2011 BFT-TI / wbr1, jom4

Autor: Roger Weber, Marcel Jost
Version: 1.4.187
Datum: 13. März 2013

Inhaltsverzeichnis

1 Einführung	3
1.1 Das Modell	3
1.2 Ziel dieses Dokuments	3
2 Modellbeschreibung	4
2.1 Übersicht	4
2.2 Funktion des Modells	5
2.3 Implementation der Steuerung	5
2.4 Implementation der Knoten	5
3 Kommunikation mit CAN	6
3.1 Aufbau des CAN-Buses	6
3.2 Kurzbeschreibung des CAN-Protokolls	6
3.3 Die CAN-Bibliothek für das CARME-Kit	6
3.4 Die CAN-Bibliothek für die Knoten	7
4 Knotenbeschreibung	8
4.1 Förderband-Knoten	8
4.1.1 Förderband Links (1) und Rechts (3)	9
4.1.2 Förderband Mitte (2)	9
4.2 Servomotor-Knoten	10
4.2.1 Verteiler (4)	10
4.2.2 Roboterarm Links (5) und Rechts(6)	11
A CAN Protokoll	12
A.1 Fehler	12
A.2 Förderband Links, Mitte und Rechts	13
A.3 Verteiler	14
A.4 Roboterarm Links und Rechts	15
B Infos zu den Förderband-Knoten	17
B.1 Template	17
C Infos zu den Servomotor-Knoten	19
C.1 Verteiler-Knoten Template	19
C.2 Roboterarm-Knoten Template	19

1 Einführung

1.1 Das Modell

Das Stellaris Roboter Modell besteht aus einer Steuerung, zwei Roboterarmen, einem Verteiler und drei Förderbändern welche über einen CAN-Bus miteinander vernetzt sind. Es wurde konstruiert um den Studierenden die Möglichkeit zu geben, den Unterrichtsinhalt in praktischen Arbeiten umsetzen zu können. Das Modell wird im Fachbereich EKT in Burgdorf und Biel für verschiedene Projektarbeiten in den Unterrichtsmodulen der Technischen Informatik eingesetzt.

Die Komponenten können einzeln implementiert werden, um mit den eingesetzten Techniken Erfahrung zu sammeln. Dabei kann das programmieren auf einem Echtzeit-Betriebssystem, das vernetzen mehrerer Knoten mit einem CAN-Bus, das ansteuern von Servo- und Brushless-Motoren und anderer Peripherie geübt werden.

1.2 Ziel dieses Dokuments

Die Angaben in diesem Dokument dienen als Grundlage für die softwaremässige Implementation eines CAN-Knotens oder einer übergeordneten Steuerung.

In diesem Dokument wird der Aufbau des Stellaris Roboter Modells und der einzelnen Knoten detailliert beschrieben. Weiter wird das Protokoll, über welches die einzelnen Knoten mit der Steuerung kommunizieren, definiert.

2 Modellbeschreibung

2.1 Übersicht

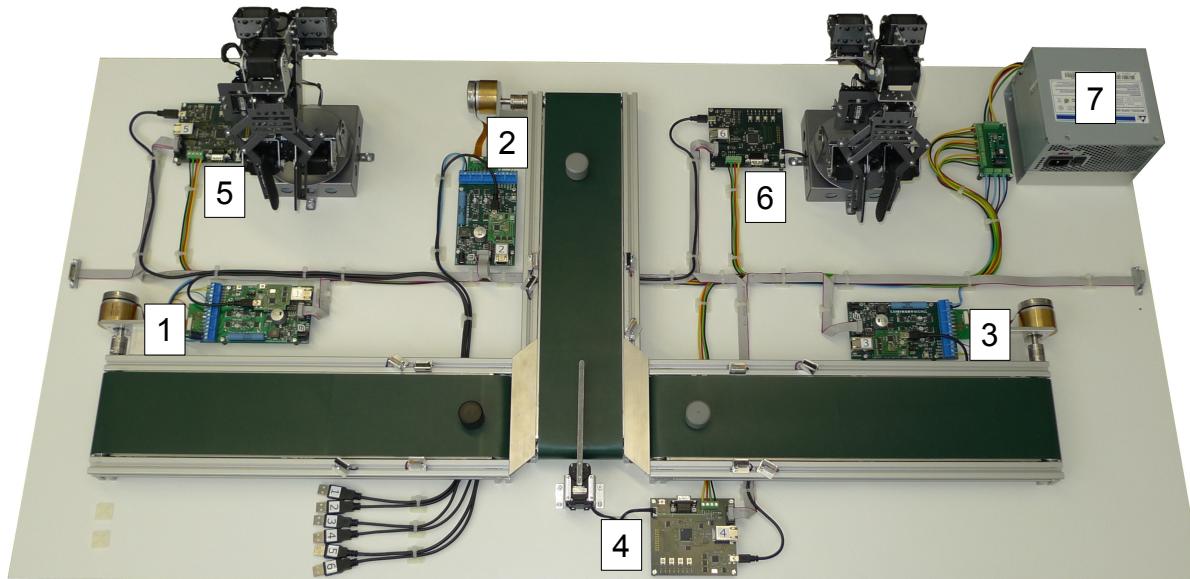


Bild 1: Übersicht der Knoten

Das Stellaris Roboter Modell besteht aus drei Förderbändern (Links 1, Mitte 2, Rechts 3), einem Verteiler (4) und zwei Roboterarmen (Links 5, Rechts 6). Die einzelnen Knoten werden im Kapitel 4 Knotenbeschreibung (S. 8) beschrieben.

Die Stromversorgung (7) besteht aus einem handelsüblichen Netzteil und einem Spannungswandler-Modul.

Jeder Knoten verfügt über einen eigenen Mikrocontroller, welcher die Steuerung der mechanischen Komponenten und das einlesen der Sensoren übernimmt. Sie sind über einem CAN-Bus mit der Steuerung verbunden.

Zum herunterladen und debuggen der Software ist jeder Knoten mit einem JTAG-Debugger ausgestattet. Dieser Debugger ist kompatibel mit dem *Texas Instruments Stellaris ICDI* Debugger der Stellaris Development-Boards. Die Knoten können damit direkt aus dem Code Composer Studio programmiert werden. Die Debugger verfügen zusätzlich über einen VCP (Virtual COM Port). Mit diesem kann eine UART-Verbindung zum Computer über die USB Schnittstelle hergestellt werden.

2.2 Funktion des Modells

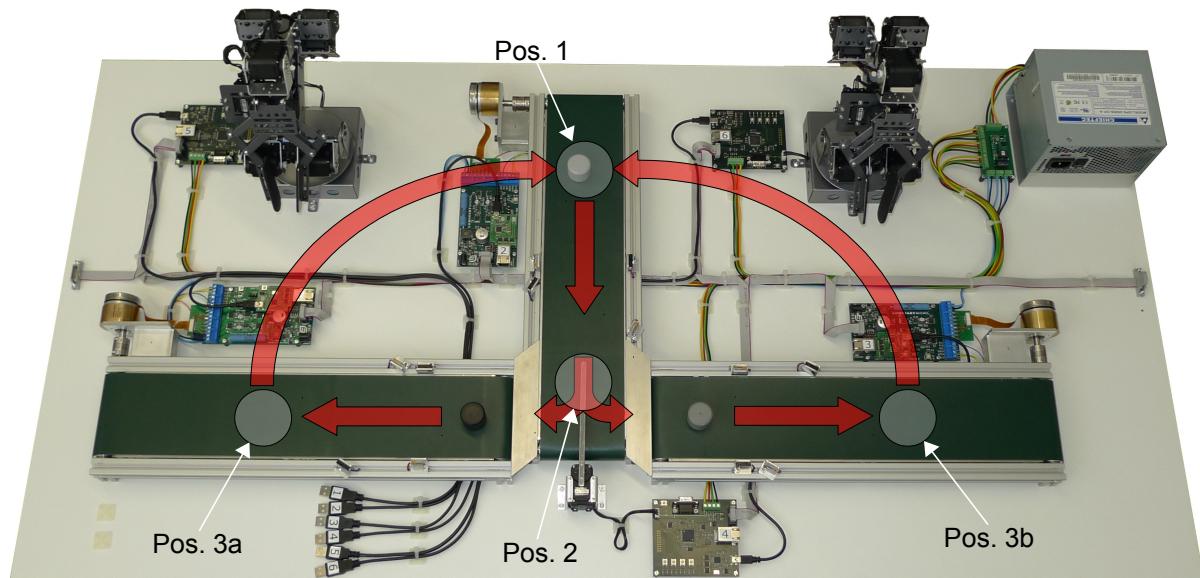


Bild 2: Übersicht der Funktion des Modells

Ziel des Modells ist es, die ECTS (Extreme Complex Transport Stone; farbiges, zylinderförmiges Element) im eingezeichneten Rundkurs zu bewegen. Die Steuerung gibt den Weg der einzelnen ECTS vor. Diese können z.B. auf dem linken oder rechten Kreis rotieren oder sich in Form einer liegenden Acht (∞) bewegen.

Die drei ECTS werden vor dem Start wie abgebildet auf das jeweilige Förderband platziert. Zu beachten ist, dass sich diese vor der ersten Lichtschranke befinden. Die genaue Position und die Farbe der ECTS ist nicht relevant.

Der Ablauf des System wird anhand des ECTS auf Pos. 1 erklärt. Dieses wird als erstes vom Förderband Mitte in Richtung Verteiler befördert. Die Lichtschranken am Förderband erkennen die genaue Position des ECTS auf dem Band. Hat das ECTS die Pos. 2 erreicht, wird das gewählte Förderband (Links oder Rechts) eingeschaltet und das ECTS vom Verteiler auf dieses Förderband geschoben. Die Lichtschranken des Förderbandes erkennen das darauf liegende ECTS und transportiert es in Richtung Pos. 3 (a oder b). Hat das ECTS die Pos. 3 erreicht, stoppt das Förderband. Der jeweilige Roboterarm ergreift das ECTS und legt es bei Pos. 1 auf das Förderband.

Beide Roboterarme können ein ECTS auf Pos. 1 legen. Dies können sie aber nicht zur gleichen Zeit tun. Pos. 1 bildet somit eine Kollisionszone, was beim Programmieren der Steuerung besonders beachtet werden muss.

2.3 Implementation der Steuerung

Die Steuerung des Stellaris Roboter Modell wird das CARME-Kit eingesetzt. Es ist über den CAN Bus an das Modell angeschlossen. Die Implementation wird mit Hilfe des in der CARME-IDE integrierten Board Support Package (BSP) und einem Realtime Kernel realisiert.

2.4 Implementation der Knoten

Um die Implementation und den Start ins Projekt zu vereinfachen, wird zu jedem Knoten ein Projekt-Template abgegeben. Dieses Template enthält die benötigten Einstellungen und einige Teile der Software. Weitere Informationen zu den Templates sind im Anhang B Infos zu den Förderband-Knoten (S. 18) und im Anhang C Infos zu den Servomotor-Knoten (S. 20) enthalten.

3 Kommunikation mit CAN

3.1 Aufbau des CAN-Buses

Alle Knoten des Modells, die Förderbänder, der Verteiler und die Roboterarme sowie die Steuerung (CARME-Kit) sind über einen CAN-Bus miteinander vernetzt. Der Aufbau sieht schematisch wie folgt aus:

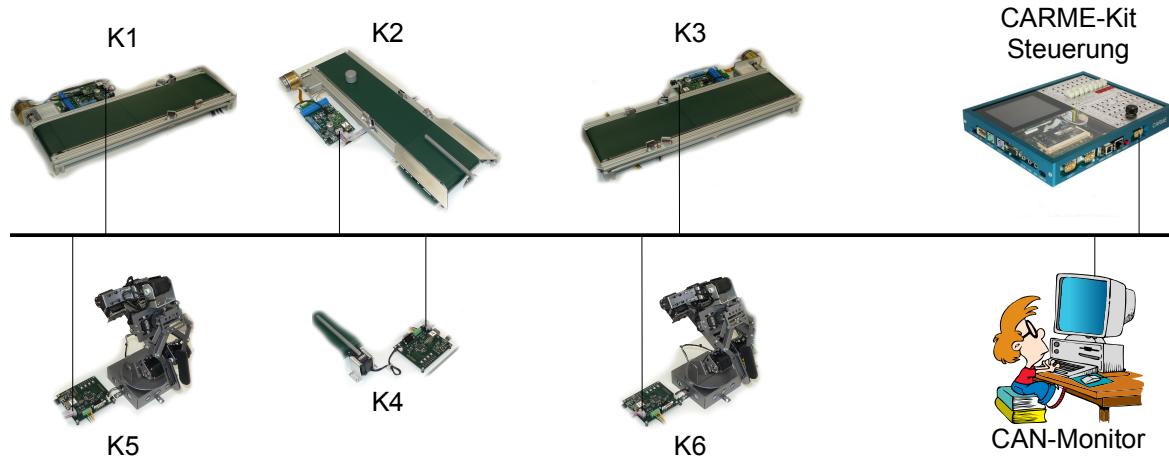


Bild 3: CAN-Bus des Modells

Mit Hilfe eines CAN-Monitors kann der Datenverkehr auf dem Bus überwacht werden. Dieser Monitor erlaubt es auch, Meldungen zu Testzwecken zu verschicken.

Der CAN-Bus wird mit einer Datenrate von 250 KBaud betrieben.

3.2 Kurzbeschreibung des CAN-Protokolls

Das CAN-Protokoll wurde für ein reines Master-Slave System definiert. Die Steuerung darf als einzige von sich aus Meldungen verschicken. Die Knoten dürfen nur auf eine Meldung der Steuerung eine Antwort zurücksenden. Diese Antwort muss sofort gesendet werden.

Es sind zwei Typen von Meldungen definiert. Zum einen sind dies Status-Meldungen, zum anderen Kommando-Meldungen. Bei einer Status-Anfrage sendet der Knoten eine Status-Antwort mit den angeforderten Daten zurück. Bei einer Kommando-Meldung erhält der Knoten einen Befehl. Er prüft diesen auf Korrektheit und führt ihn falls möglich aus. In jedem Fall sendet der Knoten eine Antwort zurück, welche aussagt, ob er den Befehl ausführt oder ob ein Fehler auftrat. Zusätzlich kann der Knoten mit einem Reset Befehl auf den Anfangszustand zurückgesetzt werden.

Das detaillierte CAN-Protokoll befindet sich im Anhang A CAN Protokoll (S. 12).

3.3 Die CAN-Bibliothek für das CARME-Kit

Im Unterrichtsmodul „Echtzeit-Betriebssystem“ wird auf dem CARME-Kit die Steuerung des Roboters mit Hilfe eines Betriebssystems implementiert. Um Ihnen das Leben etwas zu erleichtern beinhaltet das Board Support Package des CARME-Kits auch einen CAN-Treiber, mit welchem Sie die einzelnen Kommunikationsobjekte einfach senden und empfangen können.

Um die CAN-Bibliothek nutzen zu können binden Sie das untenstehende File ein:

- BSP_CAN.h

Die von der Bibliothek zur Verfügung gestellten Funktionen finden Sie in der Doxygen Dokumentation (<http://carme.bfh.ch > Downloads > Dokumentation>). Der Quelltext zur Bibliothek befindet sich auf dem

Rechner mit installierter CARME-IDE im Installationspfad der IDE (z. Bsp. C:\Program Files\CarmeIDE) im Ordner "carme\src\can".

3.4 Die CAN-Bibliothek für die Knoten

Im Unterrichtsmodul „Microcontroller in Embedded Systems“ werden die CAN-Treiber im Rahmen von Übungen und der anschliessenden Projektarbeit erstellt.

4 Knotenbeschreibung

Im Modell werden zwei Arten von Knoten eingesetzt. Dies sind erstens die Knoten für die Förderbänder und zweitens die Knoten für die Servomotoren.

4.1 Förderband-Knoten

Für die Förderbänder wird ein Kit basierend auf dem LM3S9B92 eingesetzt. Dieses Kit unterstützt dieselbe Funktionalität wie das Kit, welches für die Übungen eingesetzt wurde. Zusätzlich beinhaltet es die Leistungenlektronik und einen Adapter zum Anschluss des Motors und der Hall-Sensoren. Als Antrieb für das Förderband wird ein *maxon EC motor* (Bürstenloser DC-Motor) *EC 45 flat* und ein Getriebe 32:1 von *maxon motor* eingesetzt. Zum erkennen der ECTS auf dem Förderband werden Lichtschranken von Kodensi Corp. eingesetzt. Weitere Informationen zu den Förderband-Knoten sind im Anhang B Infos zu den Förderband-Knoten (S. 18) enthalten.

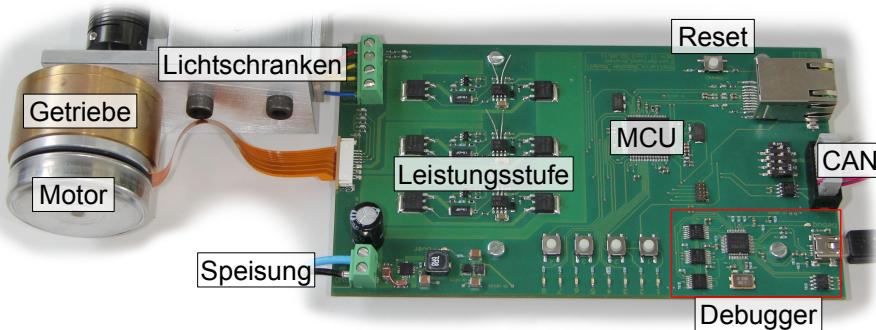


Bild 4: Förderband-Knoten

Der EC-Motor verfügt über drei Hall-Sensoren, welche für die elektronische Kommutierung des Motors verwendet werden. Mit den Sensoren kann der gefahrene Weg des ECTS auf dem Förderband bestimmt werden. Bei jedem Impuls eines Hall-Sensors wird die Funktion `CONVEYOR_UpdatePosition` aufgerufen (siehe Anhang B Infos zu den Förderband-Knoten S. 18).

Zum erkennen der Position des ECTS auf den Förderbändern sind je zwei Lichtschranken auf jedem Förderband installiert. Die Lichtschranke L1 ist vertikal zum Band installiert. Sie erkennt die längs gerichtete Position des ECTS auf dem Band. Die Lichtschranke L2 ist schräg zum Band installiert. Mit beiden Lichtschranken zusammen kann die Lage (vertikale Position) des ECTS auf dem Band bestimmt werden.

Die Lichtschranken dürfen nur ausgewertet werden, wenn das Förderband läuft. Ab der Position, an der die Lichtschranke L1 das ECTS erkennt, werden die gefahrenen Schritte gezählt. Erkennt die Lichtschranke L2 das ECTS, wird die Lage des ECTS gespeichert und das ECTS gilt als erkannt. Es befindet sich somit ein ECTS auf dem Band. Dieser Zustand wird beibehalten, bis das ECTS vom Förderband entfernt wird.

Wird die Lichtschranke L1 ein weiteres Mal durchbrochen oder die Lichtschranke L2 zeigt vor L1 an, ist ein Fehler aufgetreten. Im Falle eines Fehlers wird das Förderband gestoppt und der Fehler im Status des Knotens vermerkt.

Die Lichtschranke L1 ist am GPIO PC4 und die Lichtschranke L2 am GPIO PC7 des Kits angeschlossen. Für die Software ist zu berücksichtigen, dass die GPIO als weak-pullup Standard-Input konfiguriert werden. Sind die Lichtschranken offen, wird am GPIO-Eingang eine Eins eingelesen, bei geschlossener Lichtschranke eine Null (negative Logik).

Weiter können die vier Tasten (GPIO PJ0-PJ3), die acht LEDs (GPIO PH0-PH7) und der VCP (UART 2, GPIO PG0, PG1) zum Testen verwendet werden.

4.1.1 Förderband Links (1) und Rechts (3)

Die Förderbänder Links und Rechts übernehmen die ECTS vom Verteiler und befördern sie zur Position 3a respektive 3b. Damit die ECTS vom Roboterarm aufgenommen werden können, müssen sie an einer definierten Position gestoppt werden.

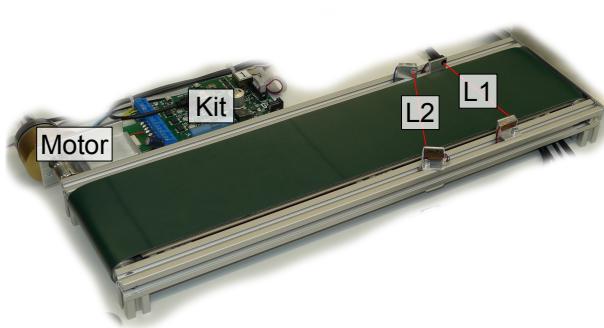


Bild 5: Förderband Links

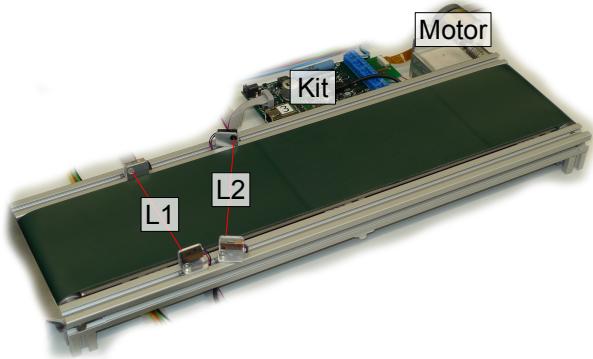


Bild 6: Förderband Rechts

Die Knoten der Förderbänder Links und Rechts unterscheiden sich nur darin, dass sie Spiegelverkehrt aufgebaut sind.

4.1.2 Förderband Mitte (2)

Auf dem Förderband Mitte werden die ECTS von den Roboterarmen abgelegt und zum Verteiler transportiert. Dort werden sie vom Verteiler entweder auf das Förderband Links oder Rechts geschoben.

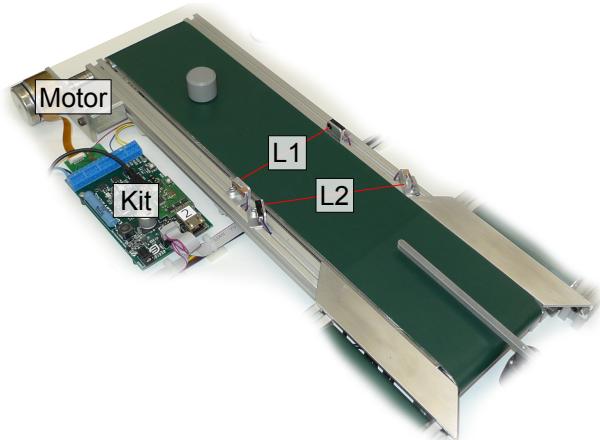


Bild 7: Förderband Mitte

4.2 Servomotor-Knoten

Für die Steuerung der Servomotoren wird das Stellaris LM3S9B92 Kit verwendet, welches bereits für die Übungen eingesetzt wurde. Als Servomotoren kommen die Dynamixel AX-12+ von Robotis zum Einsatz. Sie werden über einen seriellen Bus an den Controller angebunden. Weitere Informationen zu den Servomotoren sind im Anhang C Infos zu den Servomotor-Knoten (S. 20) enthalten.

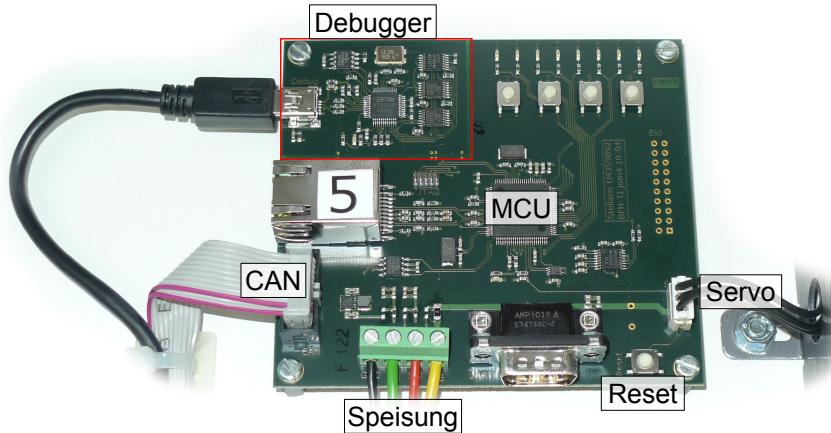


Bild 8: Servomotor-Knoten

Der serielle Bus zur Steuerung der Servomotoren ist eine einadrige Leitung, über welche die Daten per UART im halb-duplex Modus übertragen werden. Er wird auf dem Kit mit einer UART des Prozessors und einem Bustreiber realisiert. Die Richtung des Busses wird mit einem separaten GPIO Pin über den Bustreiber festgelegt. Die AX-12+ Servomotoren sind am Anschluss Servo (J6, AX12-Bus) des Stellaris LM3S9B92 Kits angeschlossen.

Der AX12-Bus ist an der UART 1 auf den GPIO Pins PB0 (U1Rx) und PB1 (U1Tx) angeschlossen. Der Busstreiber wird über den GPIO PB2 (U1Mode) umgeschaltet. Logisch Null (0) bedeutet Senden, logisch Eins (1) bedeutet Empfangen.

Weiter können die vier Tasten (GPIO PJ0-PJ3), die acht LEDs (GPIO PH0-PH7) und der VCP (UART 2, GPIO PG0, PG1) zum Testen verwendet werden.

4.2.1 Verteiler (4)

Der Verteiler-Knoten ist dafür zuständig, die ECTS vom mittleren Förderband auf das rechte oder linke Förderband zu befördern. Er besitzt nur ein AX-12+ Servomotor mit der Nr. 1. Der an diesem Servomotor befestigte Stab kann die ECTS durch eine Drehbewegung auf das gewünschte Förderband schieben.

Der Verteiler-Knoten erhält von der Steuerung die Position, an welche er den Verteiler bewegen soll. Der Wertebereich liegt zwischen 0 und 255.

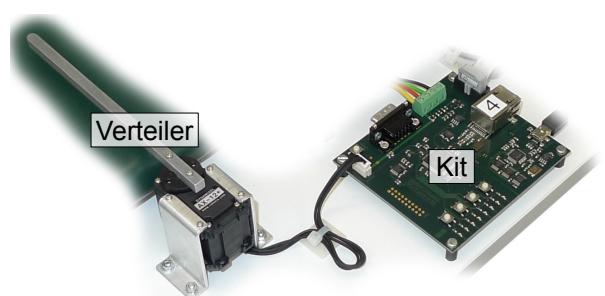


Bild 9: Verteiler

4.2.2 Roboterarm Links (5) und Rechts(6)

Die Roboterarme nehmen die ECTS vom jeweiligen (linken oder rechten) Förderband und legen es auf das mittlere Förderband ab. Die Roboterarme Links und Rechts unterscheiden sich nicht.

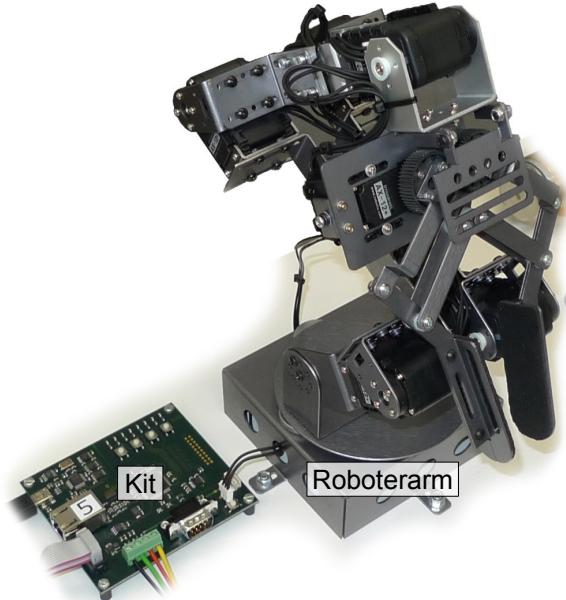


Bild 10: Roboterarm Links

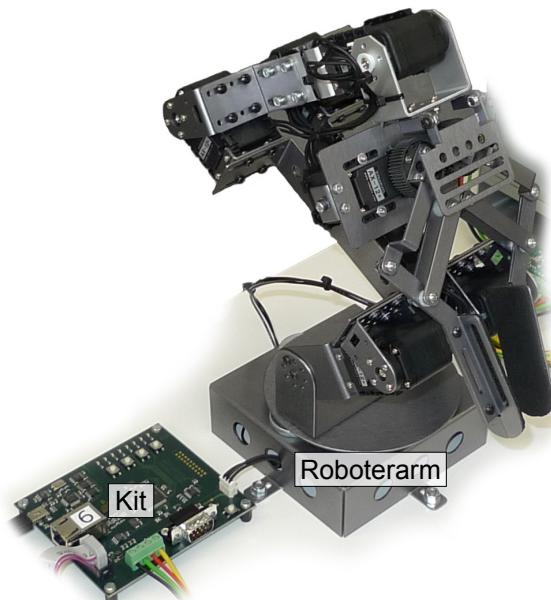


Bild 11: Roboterarm Rechts

Sie bestehen aus einem Arm aus Aluminium mit sieben Servomotoren als Antrieb. Die Achsen werden von den Servomotoren mit den im Bild 12 angegeben Nummern angetrieben. Die Schulter und der Ellbogen bestehen aus je zwei Servomotoren, die spiegelverkehrt montiert sind. Diese müssen synchron betrieben werden.

Der Roboterarm-Knoten erhält von der Steuerung die Position und Geschwindigkeit einer einzelnen Achse oder die Positionen für den gesamten Arm. Er muss die Servomotoren mit diesen Daten an die gewünschte Position bewegen.

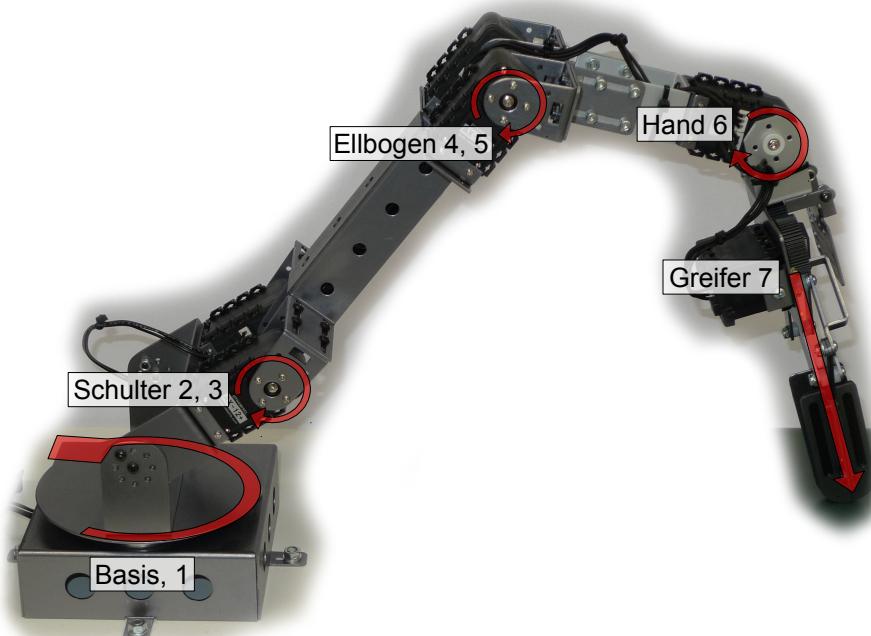


Bild 12: Roboterarm mit Achsen

Die möglichen Positionen und Geschwindigkeiten der verschiedenen Achsen sind im Anhang A.4 Roboterarm Links und Rechts (S. 16) aufgelistet.

A CAN Protokoll

Das erste Datenbyte der CAN-Meldung, falls vorhanden, definiert den Typ der Meldung. Die restlichen Datenbytes werden in der Spalte DB1-7 erklärt. Datenfelder über 2 Bytes werden als Big-Endian (MSB zuerst) gesendet. Jede Meldung hat immer die selbe Länge. Nicht benötigte Datenbytes sind undefiniert und müssen nicht beachtet werden.

Für eine einfache Programmierung der Knoten wurde ein Protokoll definiert, welches die Positionen der einzelnen Servomotoren oder die Vorgaben für die Förderbänder von der Steuerung zu den Knoten sendet.

Jeder Knoten unterstützt das Kommando Reset. Dieses führt auf dem Knoten (Microcontroller) einen Software-Reset durch. In der Software wird dieser Reset durch die Funktion SysCtlReset() ausgelöst. Auf das Kommando Reset sendet der Knoten kein Acknowledge. Das Kommando Reset führt auch dazu, dass der Debugger angehalten wird.

Ein Knoten führt nach einem Reset keine Bewegungen durch. So gehen beispielsweise die Roboterarme nie von sich aus in die Nullposition.

A.1 Fehler

Die Erläuterung der mit * markierten Fehler:

Fehler	Wert	Beschreibung
System	0x00	Das System hat einen Fehler und ist nicht mehr Einsatzbereit.
M. Typ Unbekannt	0x01	Der Typ der Meldung ist unbekannt.
M. Länge	0x02	Die Länge der Meldung ist zu kurz.
Daten Ungültig	0x03	Die in der Meldung enthaltenen Daten sind ungültig. z. B. Ausserhalb des Wertebereichs.

Tabelle 1: Fehler

A.2 Förderband Links, Mitte und Rechts

Das Förderband Links hat die ID's 0x11n, das Förderband Mitte die ID's 0x12n, das Förderband Rechts die ID's 0x13n.

Name	ID	DLC	DB0, Typ	DB1-7	Beschreibung
Status Anfrage	0x110 0x120 0x130	1	1 = Zustand	--	Anfrage des Zustands.
Status Antwort	0x111 0x121 0x131	7	0 = Fehler	1: Fehler* 2-6: undef.	Die Anfrage ist Fehlerhaft.
			1 = Zustand	1: Status 2: ECTS 3-4: Position 5-6: Lage	Der Zustand des Förderbandes.
Kommando	0x112 0x122 0x132	3	1 = Start	1-2: undef.	Startet das Förderband.
			2 = Stopp	1-2: undef.	Stoppt das Förderband.
			3 = Stopp Position	1-2: Position	Stoppt das Förderband, wenn sich das ECTS an einer bestimmten Position befindet. Position: Anzahl Schritte von L1 zur Position
			4 = Fertig	1-2: undef.	Das ECTS wurde vom Förderband entfernt. Stoppt das Förderband falls es noch läuft.
Kommando Antwort	0x113 0x123 0x133	2	0 = Fehler	1: Fehler*	Die Anfrage ist Fehlerhaft.
			1 = Start 2 = Stopp 3 = Stopp Position 4 = Fertig	1: undef.	Das Kommando wird ausgeführt.
Reset	0x11F 0x12F 0x13F	0	--	--	Führt ein Software-Reset des System durch.

Tabelle 2: CAN-Protokoll der Förderbänder Links, Mitte und Rechts

Der Zustand des Förderbandes enthält folgende Werte:

Feld	Wert Wertebereich	Name	Beschreibung
Status	1	Aus	Das Förderband läuft nicht.
	2	Ein	Das Förderband läuft
ECTS	0	ECTS Erkennungsfehler	Es wurde mehr als ein ECTS erkannt oder die Lichtschranken wurden in anderer Reihenfolge durchbrochen.
	1	Kein ECTS	Es wurde noch kein ECTS auf dem Förderband erkannt.
	2	ECTS mit Position	Ein ECTS und deren Position wurde erkannt. Die Lage ist noch nicht bekannt. (Durchbrechen von L1)
	3	ECTS mit Lage	Die Lage des ECTS wurde erkannt. (Durchbrechen von L2)
Position	0 - 10000		Die Position des ECTS auf dem Band. Anzahl Schritte von L1. Ist die Position unbekannt, wird 0 zurückgegeben.
Lage	0, 800 - 1750		Die Lage des ECTS auf dem Band. Anzahl Schritte von L1 zu L2. Ist die Lage unbekannt, wird 0 zurückgegeben.

Tabelle 3: CAN-Protokoll, Zusätzliche der Förderbänder

Die Steuerung hat folgende Varianten, um das Förderband zu betätigen:

Variante 1: Die Steuerung wird dem Förderbandknoten die Kommandos in folgender Reihenfolge schicken:

- 1) Kommando Stopp Position

- 2) Kommando Start, das ECTS muss nun bis an die definierte Position befördert werden.
- 3) Kommando Fertig, der Roboterarm hat das ECTS vom Band weggenommen.

Die Steuerung kann zwischendurch mehrfach den Status des Förderbandes abfragen.

Variante 2: Die Steuerung wird dem Förderbandknoten die Kommandos in folgender Reihenfolge schicken:

- 1) Kommando Start, das ECTS wird befördert.
- 2) Kommando Stopp, das Förderband wird gestoppt.
- 3) Kommando Fertig, der Roboterarm hat das ECTS vom Band weggenommen.

Variante 3: Die Steuerung wird dem Förderbandknoten die Kommandos in folgender Reihenfolge schicken:

- 1) Kommando Start, das ECTS wird befördert.
- 2) Kommando Stopp, das Förderband wird gestoppt.
- 3) Kommando Start, das ECTS wird befördert.
- 4) Kommando Stopp Position, das ECTS muss nun bis an die definierte Position befördert werden.
- 5) Kommando Fertig, der Roboterarm hat das ECTS vom Band weggenommen.

Die Steuerung kann zwischendurch mehrfach den Status des Förderbandes abfragen. Der Knoten muss nicht prüfen, ob das ECTS über das Band hinaus geschoben wird. Dies ist alleine Sache der Steuerung.

A.3 Verteiler

Name	ID	DLC	DB0, Typ	DB1-7	Beschreibung
Status Anfrage	0x140	1	1 = Position	--	Anfrage der Position des Verteilers.
Status Antwort	0x141	2	0 = Fehler	1: Fehler*	Die Anfrage ist Fehlerhaft.
			1 = Position	1: Wert	Die Position des Verteilers.
Kommando	0x142	3	1 = Position	1: Wert 2: Geschw.	Bewegt den Verteiler an die Position mit dem angegebenen Wert und der angegebenen Geschwindigkeit. Wird bei der Geschwindigkeit Null angegeben, wird mit der normalen Geschwindigkeit bewegt.
Kommando Antwort	0x143	2	0 = Fehler	1: Fehler*	Die Anfrage ist Fehlerhaft.
			1 = Position	1: undef.	Das Kommando wird ausgeführt.
Reset	0x14F	0	--	--	Führt ein Software-Reset des System durch.

Tabelle 4: CAN-Protokoll des Verteilers

Folgende Werte sind für die Position des Verteilers festgelegt:

Name	Wert
Mitte	127
Startposition Links	150
Startposition Rechts	105
Maximum / Links	205
Minimum / Rechts	50

Tabelle 5: CAN-Protokoll: Werte für die Positionen

Die Geschwindigkeit ist von 5 bis 255 begrenzt. Die Normale Geschwindigkeit beträgt 255.

A.4 Roboterarm Links und Rechts

Der Roboterarm Links hat die ID's 0x15n, der Roboterarm Rechts die ID's 0x16n.

Name	ID	DLC	DB0, Typ	DB1-7	Beschreibung
Status Anfrage	0x150 0x160	2	1 = Achse	1: Nr.	Anfrage der Position der Achse mit der angegebenen Nr..
			2 = Arm	1: undef.	Anfrage der Positionen aller Achsen des Roboterarms.
Status Antwort	0x151 0x161	6	0 = Fehler	1: Fehler* 2-5: undef.	Die Anfrage ist Fehlerhaft.
			1 = Achse	1: Nr. 2: Wert 3-5: undef.	Die Position der Achse mit der angegebenen Nr.
			2 = Arm	1: Basis 2: Schulter 3: Ellbogen 4: Hand 5: Greifer	Der Zustand des Armes.
Kommando	0x152 0x162	6	1 = Achse	1: Nr. 2: Wert 3: Geschw. 4-5: undef.	Bewegt die Achse mit der angegebenen Nr. an die Position mit dem angegebenen Wert und der angegebenen Geschwindigkeit. Wird bei der Geschwindigkeit Null angegeben, wird mit der normalen Geschwindigkeit bewegt.
			2 = Arm	1: Basis 2: Schulter 3: Ellbogen 4: Hand 5: Greifer	Bewegt alle Achsen an die Positionen mit den angegebenen Werten. Als Geschwindigkeit wird die normale Geschwindigkeit verwendet.
Kommando Antwort	0x153 0x163	2	0 = Fehler	1: Fehler*	Die Anfrage ist Fehlerhaft.
			1 = Achse 2 = Arm	1: undef.	Das Kommando wird ausgeführt.
Reset	0x15F 0x16F	0	--	--	Führt ein Software-Reset des System durch.

Tabelle 6: CAN-Protokoll der Roboterarme Links und Rechts

Der Roboterarm hat folgende Achsen:

Nr.	Name	Wertebereich		Geschwindigkeit		Normale Geschw.
		Min.	Max.	Min.	Max.	
1	Basis	0	255	5	50	25
2	Schulter	30	225	5	50	25
4	Ellbogen	30	225	5	50	37
6	Hand	30	225	5	50	50
7	Greifer	135	170	5	75	75

Tabelle 7: Achsen der Roboterarme Links und Rechts

Bemerkung: Eine Kombination von Minimalwerten kann insofern ein Problem sein, dass die Roboterarme mit anderen Elementen kollidieren. Die Steuerung ist dafür verantwortlich, dass die Roboterarme keine kritischen Positionen einnehmen!

Der linke Roboterarm nimmt folgende Werte für die verschiedenen Positionen an (Bemerkung: Die einzelnen Positionen können je nach Modell leicht ändern):

Basis	Schulter	Ellbogen	Hand	Greifer	Position
126, 0x7E	105, 0x69	225, 0xE1	210, 0xD2	135, 0x87	Grundposition
126, 0x7E	145, 0x91	162, 0xA2	166, 0xA6	135, 0x87	Zwischenschritt
126, 0x7E	173, 0xAD	162, 0xA2	175, 0xAF	135, 0x87	Zwischenschritt
126, 0x7E	173, 0xAD	162, 0xA2	175, 0xAF	170, 0xAA	Greifen
126, 0x7E	145, 0x91	162, 0xA2	166, 0xA6	170, 0xAA	Zwischenschritt
205, 0xCD	130, 0x82	200, 0xC8	166, 0xA6	170, 0xAA	Drehen
205, 0xCD	172, 0xAC	170, 0xAA	164, 0xA4	170, 0xAA	Zwischenschritt
205, 0xCD	172, 0xAC	170, 0xAA	164, 0xA4	135, 0x87	Ablegen
205, 0xCD	130, 0x82	200, 0xC8	166, 0xA6	135, 0x87	Zwischenschritt

Tabelle 8: Positionen Roboterarme links

Der rechte Roboterarm nimmt folgende Werte für die verschiedenen Positionen an (Bemerkung: Die einzelnen Positionen können je nach Modell leicht ändern):

Basis	Schulter	Ellbogen	Hand	Greifer	Position
129, 0x81	105, 0x69	225, 0xE1	210, 0xD2	135, 0x87	Grundposition
129, 0x81	145, 0x91	164, 0xA4	168, 0xA8	135, 0x87	Zwischenschritt
129, 0x81	170, 0xAA	164, 0xA4	167, 0xA7	135, 0x87	Zwischenschritt
129, 0x81	170, 0xAA	164, 0xA4	167, 0xA7	170, 0xAA	Greifen
129, 0x81	145, 0x91	168, 0xA8	166, 0xA6	170, 0xAA	Zwischenschritt
50, 0x32	130, 0x82	168, 0xA8	166, 0xA6	170, 0xAA	Drehen
50, 0x32	168, 0xA8	168, 0xA8	166, 0xA6	170, 0xAA	Zwischenschritt
50, 0x32	168, 0xA8	168, 0xA8	166, 0xA6	135, 0x87	Ablegen
50, 0x32	130, 0x82	200, 0xC8	166, 0xA6	135, 0x87	Zwischenschritt

Tabelle 9: Positionen Roboterarme rechts

B Infos zu den Förderband-Knoten

Die Informationen über die Förderband-Knoten befinden sich im Unterordner „Förderband-Knoten“ der abgegebenen Daten zum Modell.

Folgende Dokumente und Ordner sind darin enthalten:

Name	Beschreibung
Antrieb	Informationen und Datenblätter zum Motor und Getriebe von maxon motor.
Foerderband_Knoten.pdf	Schema des Förderband-Knotens
Stellaris LM3S9B92 Microcontroller Data Sheet (Rev. L).pdf	Datenblatt des Mikrocontrollers des Förderband-Knotens
Stellaris LM3S9B92 Rev C3_C5 Errata (Rev. I).pdf	Fehlerverzeichnis des Mikrocontrollers
ConveyorTemplate.zip	Das Template für den Förderband-Knoten

Tabelle 10: Dateien zum Förderband-Knoten

B.1 Template

Das Template zum Förderband-Knoten beinhaltet die Initialisierung des Controllers sowie die Ansteuerung des EC-Motors.

Zu beachten ist, dass die Ansteuerung des EC-Motors folgende Peripherie des Controllers verwendet:

- ADC 0
- GPIO A0, A1, A2, C4, C7, D2, D3, D5, D6, D7, E0, E1, E7, F0, F1
- PWM 0, 1, 2
- SysTick
- Timer 1
- Watchdog 0
- Interrupts
 - ADC0
 - GPIOB
 - PWM0
 - PWM1
 - PWM2
 - SysTick
 - Timer1
 - Watchdog0

Diese Peripherie darf nicht für andere Zwecke gebraucht werden und die Einstellungen dafür dürfen nicht geändert werden. Andernfalls könnte dies zur Zerstörung der Hardware führen.

Die Funktionalität des Knotens kann in die `conveyor.c` Datei implementiert werden. Folgende Funktionen sind darin schon enthalten:

Funktion	Beschreibung
<code>void CONVEYOR_Main(void)</code>	Die Hauptfunktion. Wird von der <code>main()</code> Funktion nach der Initialisierung des Konten und der Motoransteuerung aufgerufen. Die Funktion muss eine Endlos-Schleife beinhalten und darf nicht verlassen werden.
<code>void CONVEYOR_UpdatePosition(void)</code>	Die Funktion wird bei jedem Impuls eines Hall-Sensors aufgerufen. Damit können die gefahrenen Schritte gezählt werden.

Tabelle 11: Funktionen der `conveyor.c` Datei

Folgende Funktionen zur Ansteuerung des EC-Motors können verwendet werden:

Funktion	Beschreibung
void MainRun(void)	Startet den EC-Motor.
void MainStop(void)	Stoppt den EC-Motor.
void MainEmergencyStop(void)	Stoppt den EC-Motor sofort. Wird nur im Fehlerfall gebraucht.
void MainSetSpeed(unsigned long ulSpeed)	Setzt die Geschwindigkeit des Motors. In 1/min.
void MainSetDirection(tBoolean bForward)	Setzt die Fahrtrichtung des Motors. TRUE vorwärts, FALSE rückwärts
unsigned long MainIsRunning(void)	Gibt true zurück, falls der Motor läuft, sonst false.
unsigned long MainIsReverse(void)	Gibt true zurück, falls der Motor vorwärts läuft, sonst false.
unsigned long MainIsStartup(void)	Gibt true zurück, falls der Motor am starten ist, sonst false.

Tabelle 12: Funktionen des Förderband Templates

C Infos zu den Servomotor-Knoten

Die Informationen über die Servomotor-Knoten befinden sich im Unterordner „Servomotor-Knoten“ der abgegebenen Daten zum Modell.

Folgende Dokumente und Ordner sind darin enthalten:

Name	Beschreibung
AX-12.pdf	Datenblatt der Robotis Dynamixel AX-12+ Servomotoren
Servo Controller.pdf	Schema des Servomotor-Knotens
Stellaris LM3S9B92 Microcontroller Data Sheet (Rev. L).pdf	Datenblatt des Mikrocontrollers des Servomotor-Knotens
Stellaris LM3S9B92 Rev C1 Errata (Rev. G).pdf	Fehlerverzeichnis des Mikrocontrollers
ArmTemplate.zip	Template für den Roboterarm-Knoten
SplitterTemplate.zip	Template für den Verteiler-Knoten

Tabelle 13: Dateien zum Servomotor-Knoten

C.1 Verteiler-Knoten Template

Das Template des Verteiler-Knotens beinhaltet nur die Initialisierung des Controllers.

Die Ansteuerung des Servomotors über die halb-duplex UART Schnittelle ist nicht implementiert. Die Informationen dazu sind im Datenblatt des Dynamixel AX-12+ Servomotors sowie im Schema des Stellaris Boards zu finden. Der angeschlossene Servomotor hat die ID 1.

C.2 Roboterarm-Knoten Template

Das Template zum Roboterarm-Knoten beinhaltet die Initialisierung des Controllers sowie die Ansteuerung des Roboterarmes.

Die Ansteuerung des Roboterarmes wurde so implementiert, dass nur der erste Servomotor (kleinere Nr.) der Achse angesteuert werden muss, der zweite Servomotor (größere Nr.) wird abhängig vom ersten betrieben. Die gültigen Nr. entsprechen denen aus Tabelle 7 Achsen der Roboterarme Links und Rechts (S. 16).

Zu beachten ist, dass die Ansteuerung des Roboterarmes neben der benötigten GPIOs und der UART auch den System-Tick benötigt.

Folgende Funktionen zur Ansteuerung des Roboterarmes können verwendet werden:

Funktion	Beschreibung
void ArmInit()	Initialisiert den Roboterarm. Diese Funktion muss während der Initialisierung des Systems aufgerufen werden.
void ArmProcess()	Führt wiederkehrende Funktionen der Roboterarm Ansteuerung aus. Diese Funktion muss im Main-Loop des Programms regelmässig ausgeführt werden. Es dürfen keine größeren Pausen oder Berechnungen im Main-Loop gemacht werden.
int ArmGetGoalPosition(int id)	Gibt die gewünschte Position einer Achse zurück.
int ArmGetCurrentPosition(int id)	Gibt die aktuelle Position einer Achse zurück.
void ArmMove(int id, int position, int speed)	Bewegt eine Achse an die angegebenen Position mit der angegebenen Geschwindigkeit. Die möglichen Werte sind der Tabelle 7 zu entnehmen.

Tabelle 14: Funktionen des Roboterarm Templates