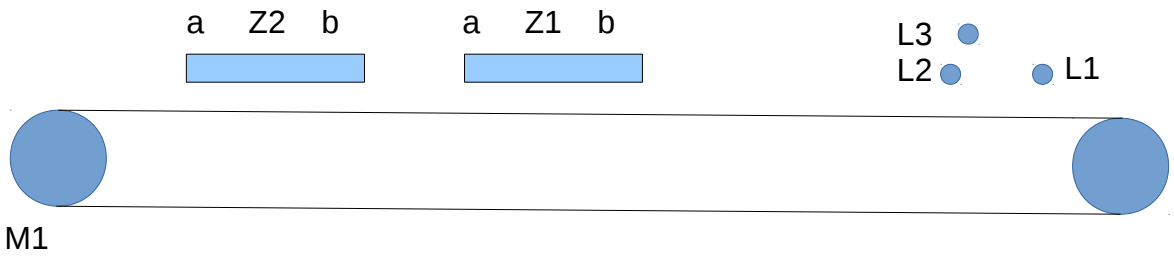
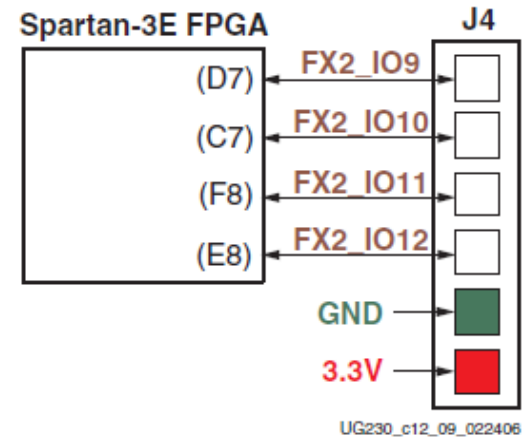
Praktikumsaufgabe zur Programmierung von Mikrocontrollern

Sortierung von 3 Paket Größen nach **Normalgröße**, **langes Paket** und langes **dickes Paket**. Zur Erkennung sind dazu 3 Lichtschranken am Förderband angebracht. Zwei Lichtschranken sind mit einem so großem Abstand angeordnet, das bei einem normalen Paket nie beide Lichtschranken gleichzeitig unterbrochen werden. Eine Dritte Lichtschranke befindet sich über der 2. Lichtschranke in der Höhe, das nur bei dicken Paketen beide Lichtschranken übereinander gleichzeitig unterbrochen werden. Damit können alle drei Sorten eindeutig unterschieden werden und an 3 verschiedenen Stellen das Band Verlassen. Einmal kann eine Sorte einfach bis an das Ende des Bandes laufen. Mit zwei pneumatisch verstellbaren Armen kann ein Paket an zwei Stellen vorzeitig vom Band geschoben werden. Dazu werden die Pneumatik Zylinder über je zwei Magnetventile angesteuert, mit denen der Hebel in die Abwurfposition oder in die normale Position gebracht werden kann. Zusätzlich ist noch der Luftdruck für die Stellzylinder zu überwachen.

Folgende Signale müssen also überwacht oder erzeugt werden:

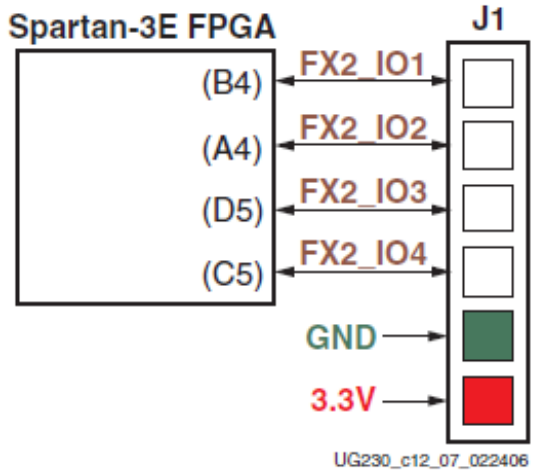
Signal	IN/OUT	Adresse	Bemerkung
L1	IN0	**PORT_IN_2_L1_3_D**	Lichtschranke am Anfang der Messstelle unten
L2	IN1		Lichtschranke am Ende der Messstelle unten
L3	IN2		Lichtschranke am Ende der Messstelle oben
D1	IN3		LEGO Kontakt als Druckschalter
Z1a	OUT0	**PORT_OUT_1_Z1Z2M1**	Magnetventil Zylinder 1a (gibt das Förderband frei)
Z1b	OUT1		Magnetventil Zylinder 1b (blockiert das Band)
Z2a	OUT2		Magnetventil Zylinder 2a (gibt das Förderband frei)
Z2b	OUT3		Magnetventil Zylinder 2b (blockiert das Band)
M1	OUT4		Förderbandmotor
(M2)	OUT5		Nicht verwendet





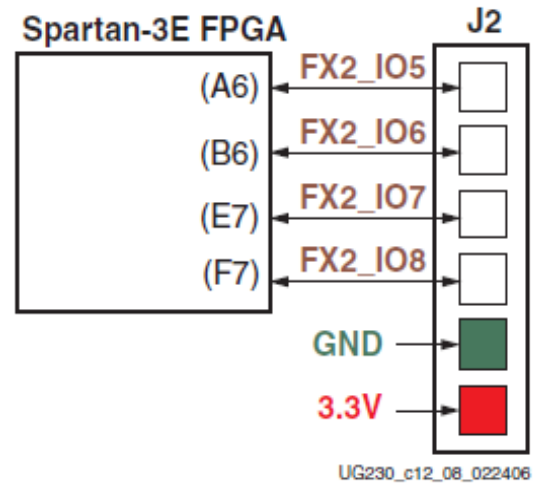
PORT_IN_2_L1_3_D

D1	Bit 3	Druckschalter
L3	Bit 2	Lichtschranke am Ende oben
L2	Bit 1	Lichtschranke am Ende unten
L1	Bit 0	Lichtschranke am Anfang unten

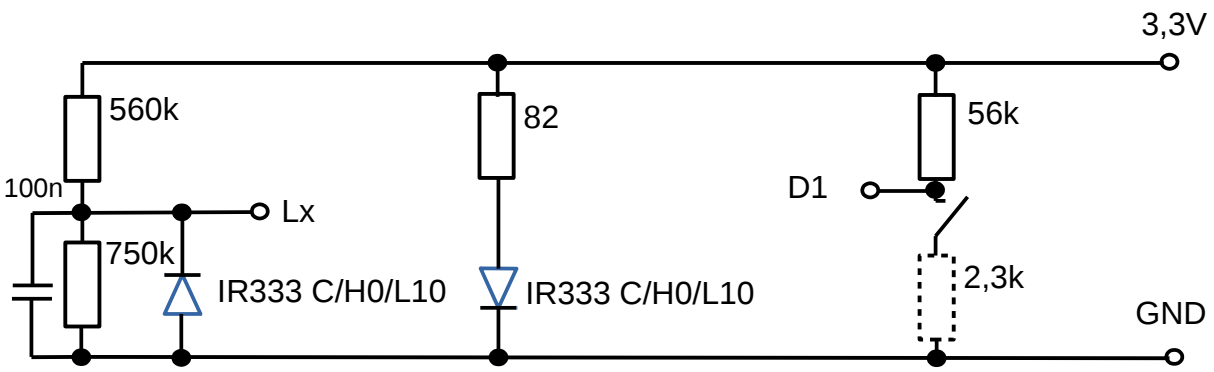


PORT_OUT_1_Z1Z2M1

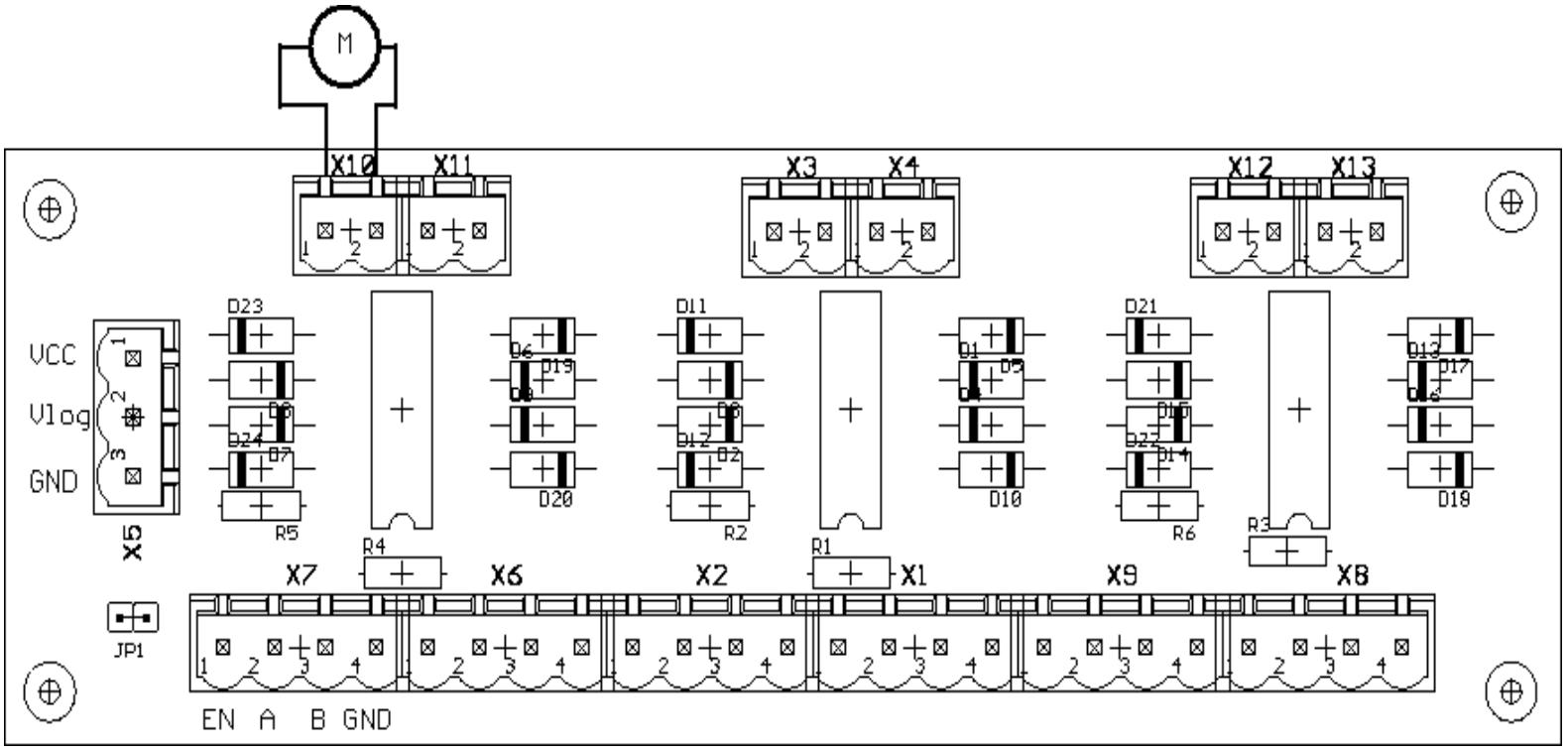
M2	Bit 5	(Pumpenmotor)
M1	Bit 4	Förderbandmotor



Z2b	Bit 3	Zylinder 2b (blockiert das Band)
Z2a	Bit 2	Zylinder 2a (gibt das Förderband frei)
Z1b	Bit 1	Zylinder 1b (blockiert das Band)
Z1a	Bit 0	Zylinder 1a (gibt das Förderband frei)



Das folgende Bild zeigt die Anschlussbelegung der zu verwendenden Motorbrücke zur Ansteuerung der Magnetventile und Motoren:



Es sind insgesamt 6 Motoren/Magnetventile anschließbar. Als Motorbrückenschaltkreis wurde der L293 von Texas Instruments verwendet. Das Datenblatt und der Schaltplan geben nähere Auskünfte für Interessierte. Zu den einzelnen Klemmen: Links an der Seite sind die Versorgungsspannungen anzuklemmen. Die entsprechende Klemme heißt X5. V_{CC} ist der Anschluss für die Motoren, V_{log} ist der Anschluss für die Logik-Teile in den Schaltkreisen. Für unseren Versuch ist V_{log} mit den 5,0 V vom Board zu verbinden und V_{CC} muss mit einem 9V Netzteil für die Motoransteuerung und bei einer 2. Brücke mit 12V für die Magnetventile verbunden werden. Der Jumper **JP1** darf **nicht geschlossen** werden, da er V_{log} und V_{CC} verbindet. Der dritte Anschluss ist Masse. **Die Lego-Motoren vertragen höchstens 9V Spannung.** Aus diesem Grund schließen Sie bitte an dem Versorgungs-spannungsanschluss V_{CC} für die Motoren nie mehr als 9V an, die Motoren nehmen sonst Schaden! Zudem sollten Sie wissen, dass Sie die **Versorgungsspannungsanschlüsse nie verpolen** dürfen, die Schaltkreise nehmen sonst unwiderruflich Schaden! Oben finden Sie die Anschlüsse für die 6 Motoren. Das Bild sollte eigentlich selbst erklärend sein, wie die Motoren anzuschließen sind. Unten finden Sie die Steueranschlüsse für die Motoren. Hier müssen Sie den SpartanMC anschließen. Der Eingang **EN** ist der Enable-Eingang. Wenn er auf Low gesetzt ist, ist der Motor in jedem Fall abgeschaltet, egal welcher Pegel an den Richtungseingängen **A** und **B** anliegt. Auf der Leiterplatte ist der Enable Eingang durch einen Pullup-Widerstand auf High gesetzt, damit sind die Richtungssteuereingänge standardmäßig wirksam. A und B sind also die Richtungssteuereingänge. Wenn A auf High und B auf Low gesetzt werden, dann dreht der Motor in eine Richtung, wenn A auf Low und B auf High gesetzt werden, dann dreht der Motor in die andere Richtung. Wenn A und B auf gleiche Pegel gesetzt werden, dann ist die Motorbremse aktiv. Die anderen Anschlüsse an der Klemme sind Masseanschlüsse. Bitte vergessen Sie nicht, einen Masseanschluss vom SpartanMC-Board mit einem Masseanschluss an der Motorbrückenplatine zu verbinden. Für den Versuch könnte EN mit A und B mit GND verbunden werden. Die Brücke von **EN und A** muss dann mit einem der **M*** oder **Z**** Signalen verbunden werden. Bei falscher Drehrichtung der Motoren müssen die Anschlüsse an den oberen Klemmen vertauscht werden. **Offene Eingänge A oder B** wirken bei der Schaltung wie ein **high Pegel**! In der Konfiguration des SpartanMC sollten diese Signale mit PULLDOWN Widerständen versehen werden.

File

New | Load | Save | Validate | Build Incrementally | Show schematic | Auto-Script

System components

- test1_step_foerderband
  - spartanmc_0
    - spartanmc_mem_local_0
    - intctrl_0
    - port_bi_0_lcd
    - port_in_0_sw3_0
    - port_in_1_o_w
    - port_in_2_l1_3_d
    - port_out_0
    - port_out_1_z1z2m1
    - rot_0
    - rti_0
    - rti_1
    - timer_0
    - timer_1
    - timer_2
    - watchdog_0
    - compare_0
    - uart_0
    - uart_1
    - port_out_2_LED
    - timer_3
  - clk_xilinx_0

Configuration: test1_step_foerderband

General | Target | Connections | I/O-Configuration | Debug

Target Options

Target	3E-Starter

User Constraints

UC_DATA	
UC_FILE	project.ucf

Device and Toolchain Options

Device	XC3S500E-FG320-4
Family	Spartan-3
JTAG-ID	1
Toolchain	ise
PAR	-ol high -xe n
TRANSLATE	-nt on -p XC3S500E-FG320-4
SYNTHESIS	-p XC3S500E-FG320-4 -ifmt mixed -ofmt NGC -opt_mode Speed -opt_level 1 -iuc NO -glob_opt ALLClockNets -rtlview YES -read_cores YES -write_timing_constraints NO -cross_clock_analysis NO -hierarchy_separator / -bus_delimiter [] -case maintain -slice_utilization_ratio 100 -fsm_extract YES -fsm_encoding Auto -fsm_style lut -ram_extract Yes -ram_style Auto -rom_extract Yes -rom_style Auto -shreg_extract YES -resource_sharing YES -mult_style auto -iobuf YES -max_fanout 500 -bufg 8 -register_duplication YES -equivalent_register_removal YES -register_balancing No -optimize_primitives No -iob auto -slice_utilization_ratio_maxmargin 5
BITGEN	-m -l
MAP	-cm speed -timing -p XC3S500E-FG320-4 -logic_opt on -pr b -register_duplication -xe n -ol high
NETGEN	-s 4
PROGRAM	

Errors | Terminal

Document opened

Abbildung 1: Verwendetes Board im Förderbandprojekt

File

New | Load | Save | Validate | Build Incrementally | Show schematic | Auto-Script

System components

- test1_step_foerderband
  - spartanmc_0
    - spartanmc_mem_local_0
    - intctrl_0
    - port_bi_0_lcd
    - port_in_0_sw3_0
    - port_in_1_o_w
    - port_in_2_l1_3_d
    - port_out_0
    - port_out_1_z1z2m1
    - rot_0
    - rti_0
    - rti_1
    - timer_0
    - timer_1
    - timer_2
    - watchdog_0
    - compare_0
    - uart_0
    - uart_1
    - port_out_2_LED
    - timer_3
  - clk_xilinx_0

Configuration: test1_step_foerderband

General | Target | Connections | I/O-Configuration | Debug

		Connected	Inv	Direction	Standard	Drive	Slew	Termination	User Contraints	Clock	
SW											
#SW_0	=>	/spartanmc_0/port_in_0_sw3_0.pin	☐	Input	LVTTL			PullUp			Delete
#SW_1	=>	/spartanmc_0/port_in_0_sw3_0.pin	☐	Input	LVTTL			PullUp			Delete
#SW_2	=>	/spartanmc_0/port_in_0_sw3_0.pin	☐	Input	LVTTL			PullUp			Delete
#SW_3	=>	/spartanmc_0/port_in_0_sw3_0.pin	☐	Input	LVTTL			PullUp			Delete
BTN											
#BTN_NORTH	=>	/spartanmc_0/intctrl_0.intr_in	☐	Input	LVTTL			PullDown			Delete
#BTN_SOUTH	=>	/clk_xilinx_0.reset /spartanmc_0/watchdog_0.reset_lowactiv	☑	Input	LVTTL			PullDown			Delete
#BTN_EAST	=>	/spartanmc_0/port_in_1_o_w.pin	☐	Input	LVTTL			PullDown			Delete
#BTN_WEST	=>	/spartanmc_0/port_in_1_o_w.pin	☐	Input	LVTTL			PullDown			Delete
ROT											
#ROT_A	=>	/spartanmc_0/rot_0.rot_a	☐	Input	LVTTL			PullUp			Delete
#ROT_B	=>	/spartanmc_0/rot_0.rot_b	☐	Input	LVTTL			PullUp			Delete
#ROT_CENTER	=>	/spartanmc_0/rot_0.rot_p	☐	Input	LVTTL			PullDown			Delete
LED											
#LED_0	<=	/spartanmc_0/port_out_2_LED.pin	☐	Output	LVTTL	2		None			Delete
#LED_1	<=	/spartanmc_0/port_out_2_LED.pin	☐	Output	LVTTL	2		None			Delete
#LED_2	<=	/spartanmc_0/port_out_2_LED.pin	☐	Output	LVTTL	2		None			Delete
#LED_3	<=	/spartanmc_0/port_out_2_LED.pin	☐	Output	LVTTL	2		None			Delete

Errors | Terminal

Document opened

Abbildung 2: Verwendete I/O-Signale des 3e Starterkit1

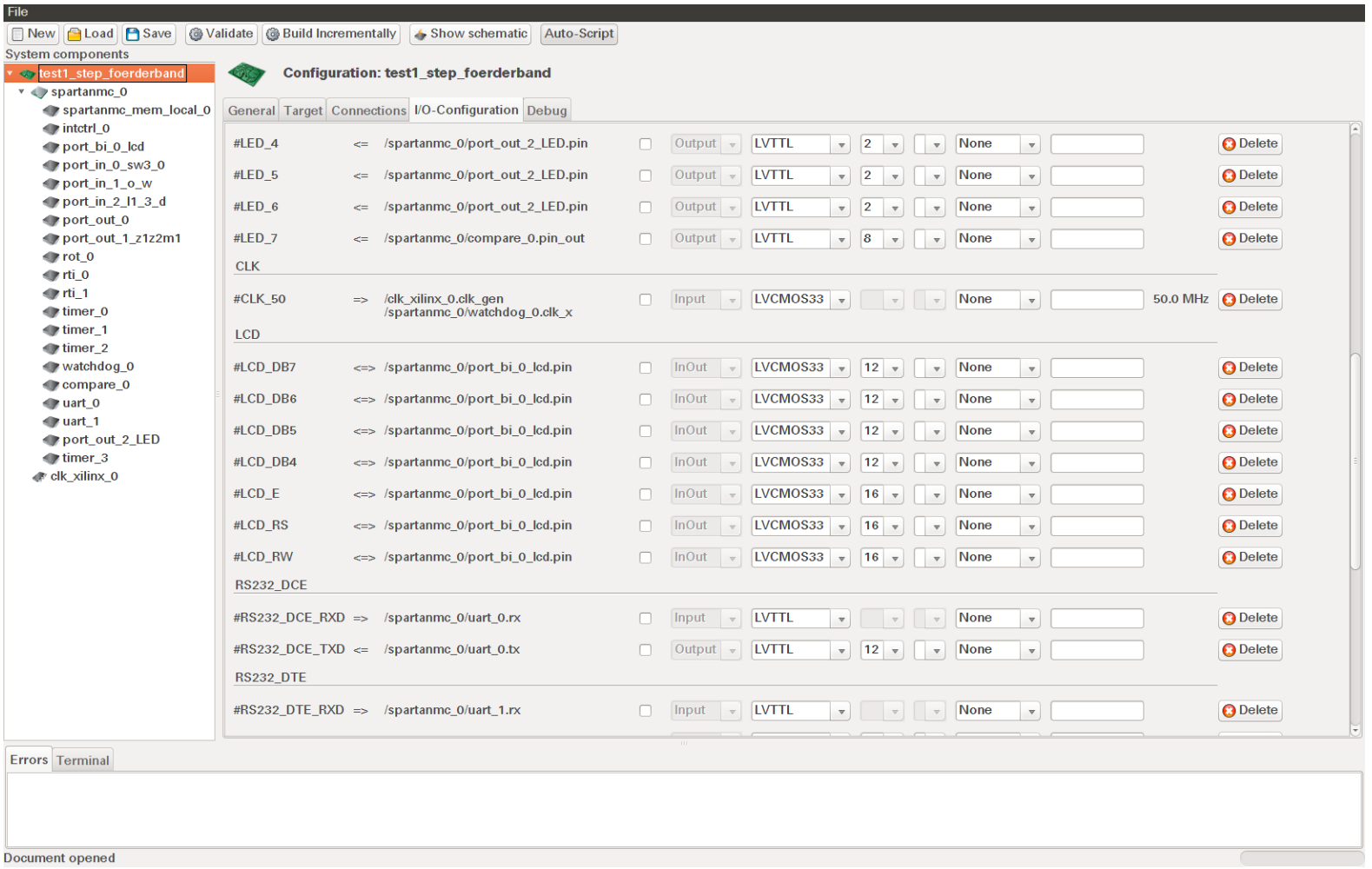


Abbildung 3: Festlegung der Datenrichtung, der Signalpegel und weitere Parameter

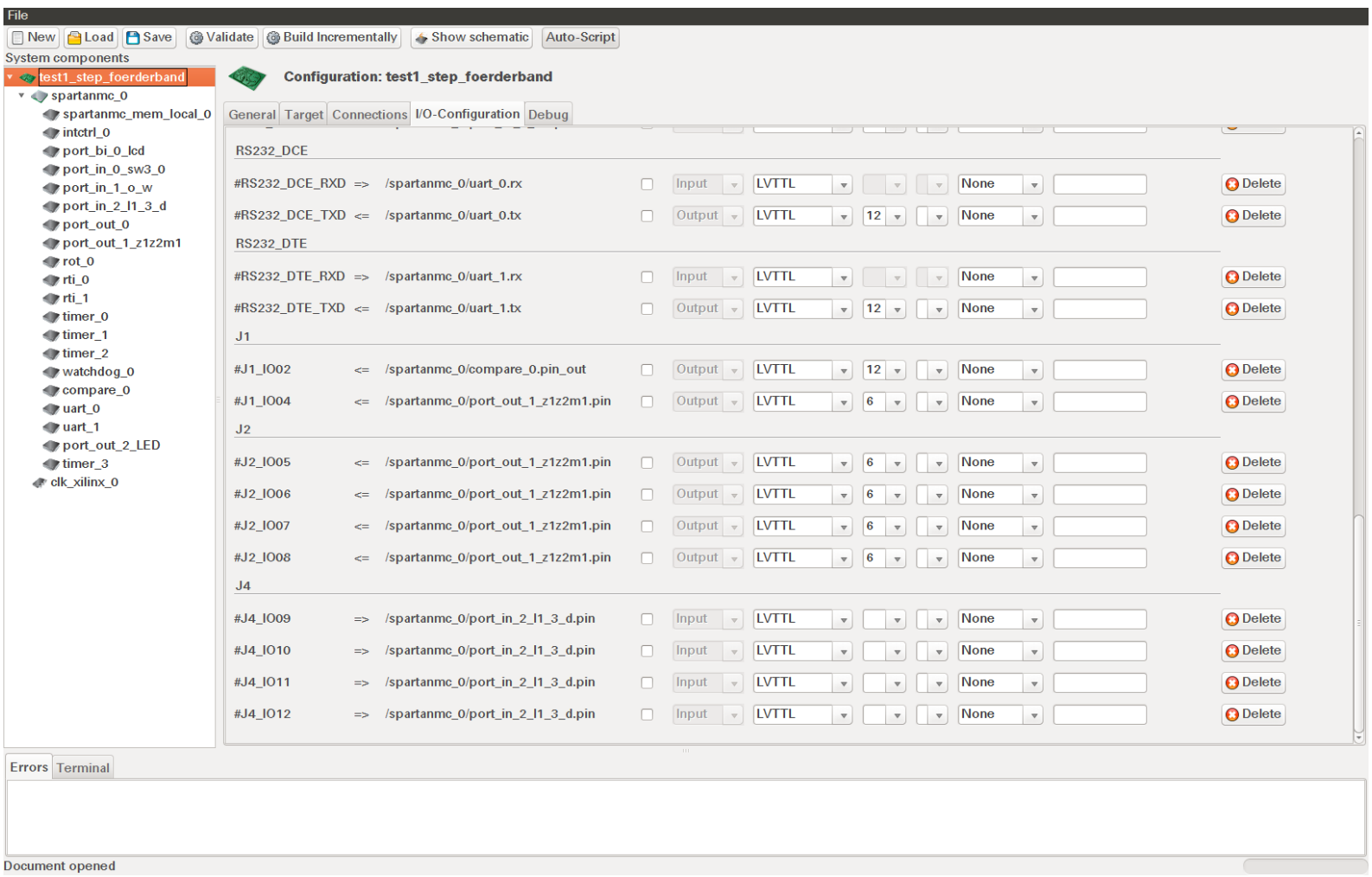


Abbildung 4: Festlegung der Datenrichtung, der Signalpegel und weitere Parameter

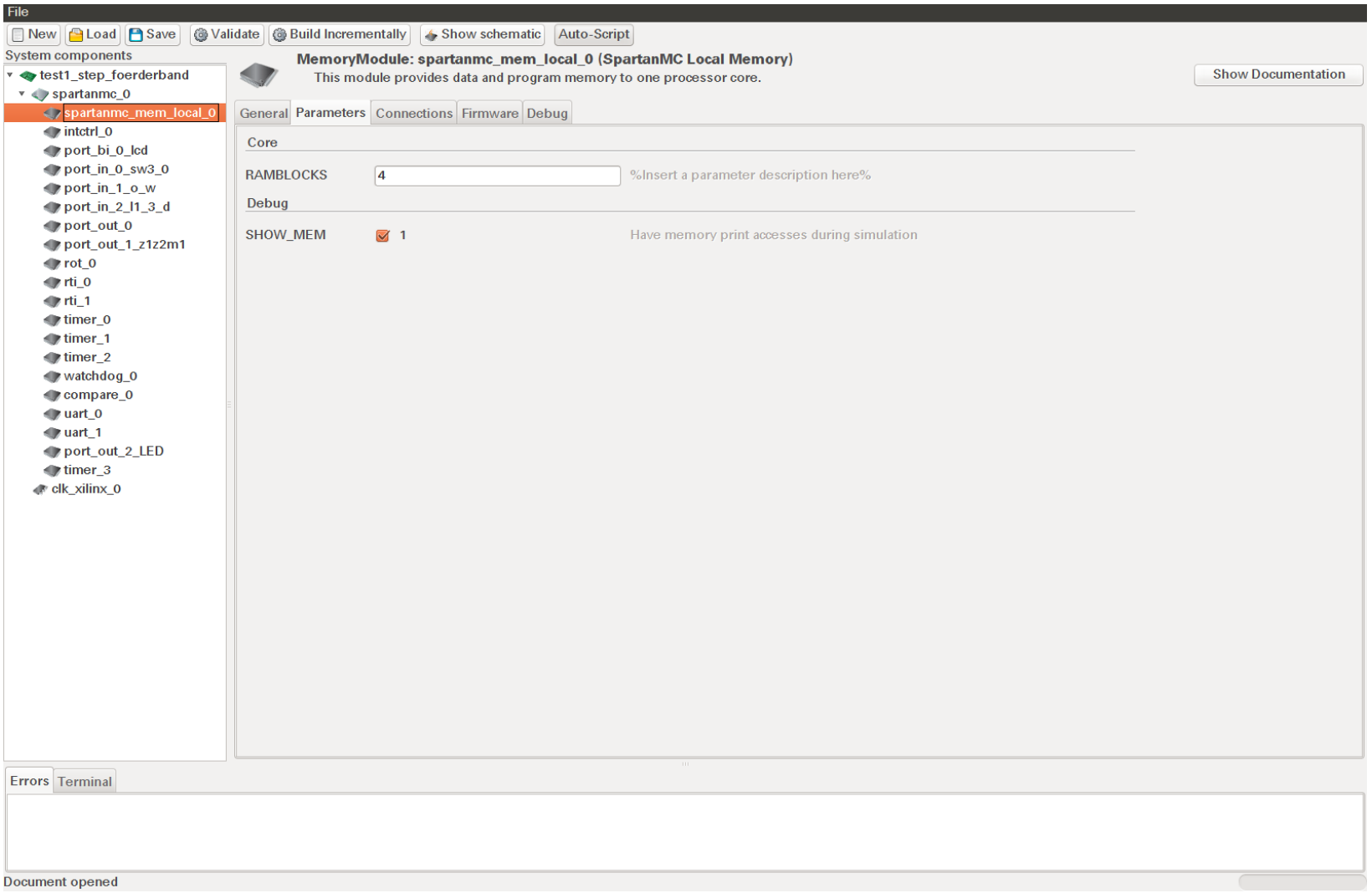


Abbildung 5: Anzahl der Speicherblöcke einstellen

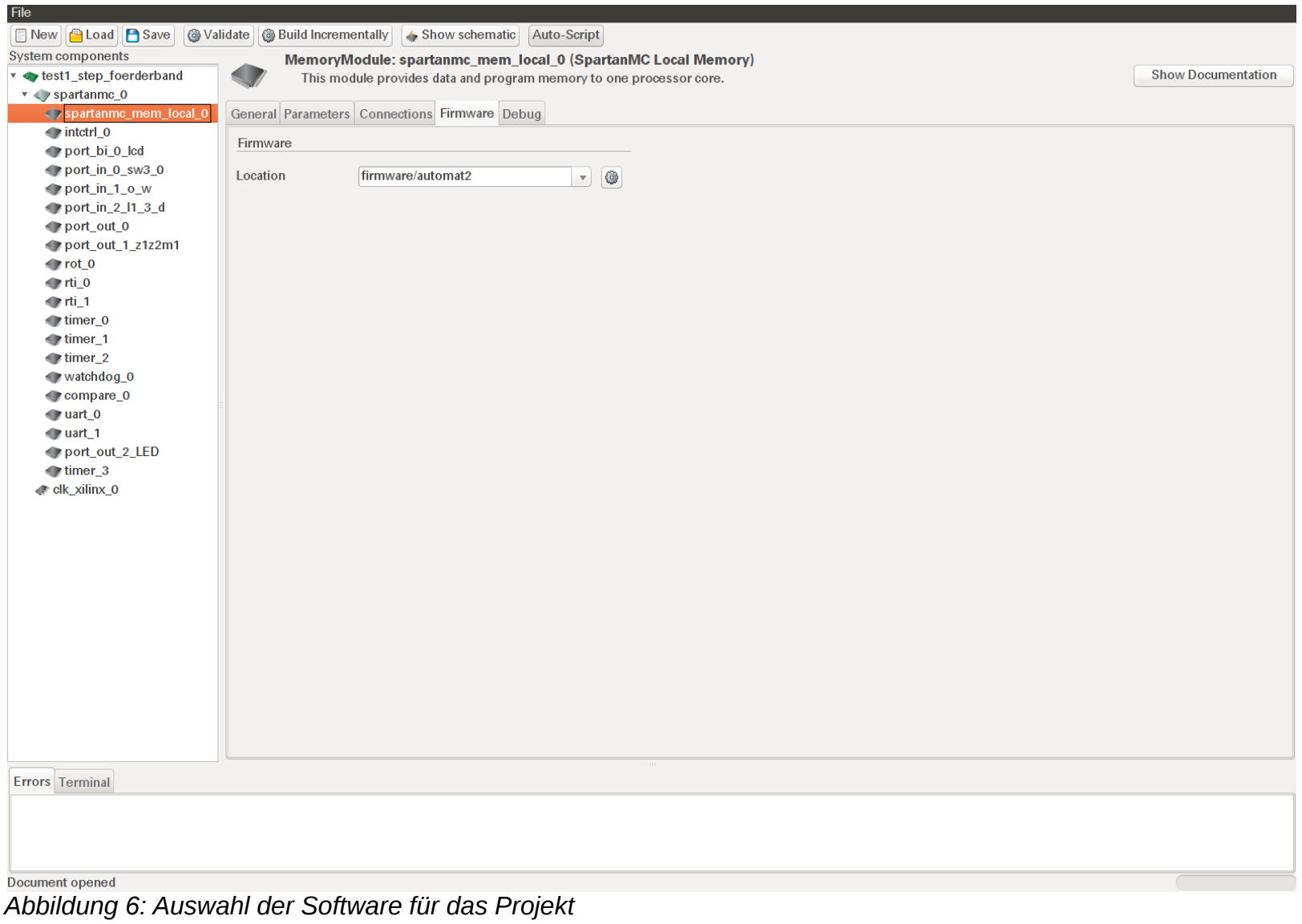
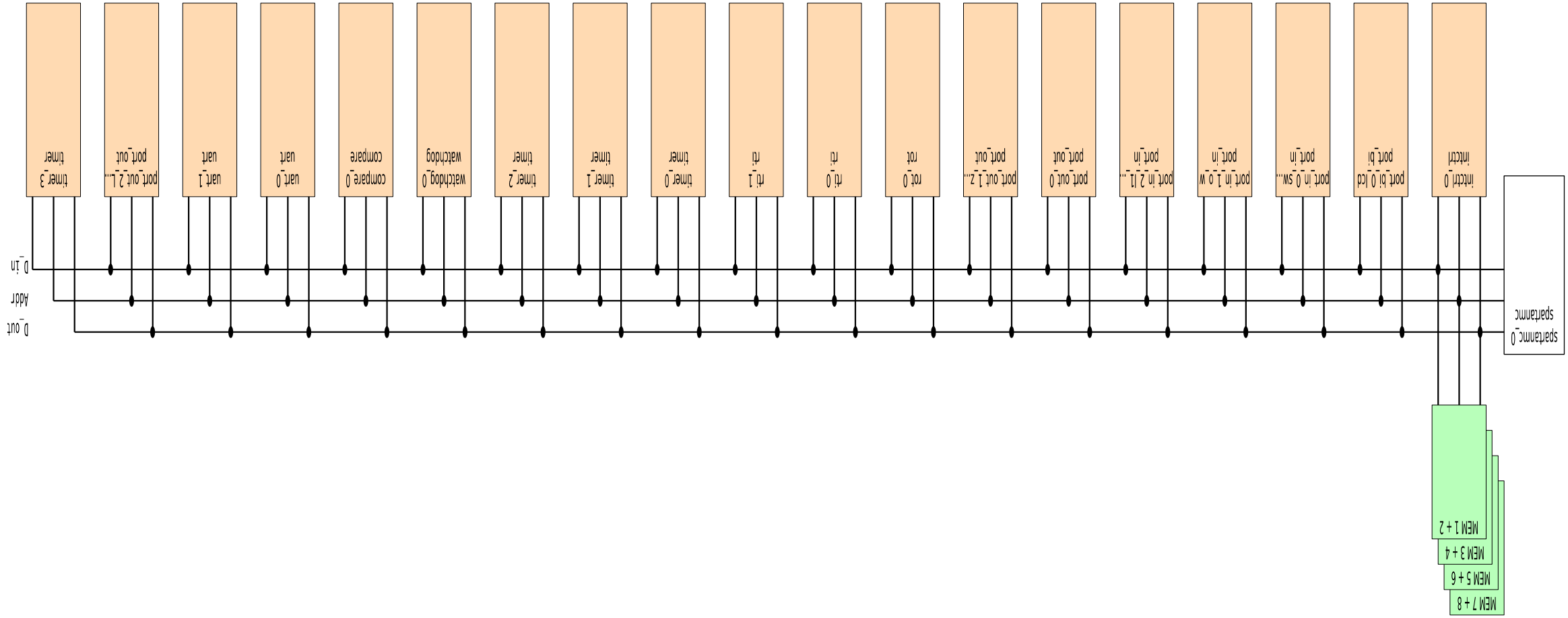


Abbildung 6: Auswahl der Software für das Projekt


spartanmc_0
spartanmc
D_out
Addr
D_in
intctrl_0
intctrl
port_bi_0_lcd
port_bi
port_in_0_sw...
port_in
port_in_1_0_w
port_in
port_in_2_l1_...
port_in
port_out_0
port_out
port_out_1_z...
port_out
rst_0
rst
rti_0
rti
rti_1
rti
timer_0
timer
timer_1
timer
timer_2
timer
watchdog_0
watchdog
compare_0
compare
uart_0
uart
uart_1
uart
port_out_2_L...
port_out
timer_3
timer
MEM 1 + 2
MEM 3 + 4
MEM 5 + 6
MEM 7 + 8

Die Ports der Konfiguration „foerderband5.jc3“:

Die Freigabe aller Interrupts erfolgt durch die Konstante SB_INTCTRL_0_IR_SOURCES automatisch, die zusammen mit allen Konstanten in der Datei „moduleParameters.h“ zur Verfügung stehen. Die Namen aller Ports stehen in der Datei „peripherals.h“.

1. Für **Motoren** und **Pneumatik Zylinder** des Förderband Versuches

Das Port hat in der Konfiguration die Bezeichnung „**PORT_OUT_1_Z1Z2M1**“. Um die 5 Ausgangssignale nutzen zu können, muss erst eine Freigabe der Bits im PORT_OUT_1_Z1Z2M1.oe Register durch schreiben einer 0x1f erfolgen. Danach haben die folgenden Ausgaben auf das PORT_OUT_1_Z1Z2M1.data Register die Funktion:

- **0x10** Förderbandmotor **M1** eingeschaltet.
- **0x08** Zylinder **Z2b** (blockiert das Band)
- **0x04** Zylinder **Z2a** (gibt das Förderband frei)
- **0x02** Zylinder **Z1b** (blockiert das Band)
- **0x01** Zylinder **Z1a** (gibt das Förderband frei)
- **0x00** alles **aus**geschaltet

2. Für die **Lichtschranken** und den **Drucksensor**

Das Port hat in der Konfiguration die Bezeichnung „**PORT_IN_2_L1_3_D**“. Am Register PORT_IN_2_L1_3_D.data kann sofort der Zustand aller 4 Sensoren abgefragt werden. Soll ein Sensor Interrupt auslösen, dann muss das Bit im PORT_IN_2_L1_3_D.ie Register gesetzt werden. Im PORT_IN_2_L1_3_D.edgsel Register kann dann noch die Flanke zur Interrupt Auslösung festgelegt werden. Die folgenden Bit Belegungen am Port haben dann die Bedeutung:

- **0x00** Soll Druck ist erreicht, keine Lichtschranke ist unterbrochen
- **0x01** Die Lichtschranke **L1** ist unterbrochen.
- **0x02** Die Lichtschranke **L2** ist unterbrochen.
- **0x04** Die Lichtschranke **L3** ist unterbrochen.
- **0x08** Der **Druck** ist zu **niedrig**.
- **0x02 kurzes** Paket und genügend Druck.
- **0x0a kurzes** Paket und zu **wenig** Druck.
- **0x03 langes** Paket und genügend Druck.
- **0x0b langes** Paket und zu **wenig** Druck.
- **0x07 langes**, **dickes** Paket und genügend Druck.
- **0x0f langes**, **dickes** Paket und zu **wenig** Druck.

Das Port ist mit dem Signal „intr[9]“ verbunden. Ein Interrupt muss damit in "**ISR(9)**“ im C-Programm behandelt werden.

3. **Timer Modul 0** für die Wartezeiten bei der LCD Ansteuerung

Das Port hat in der Konfiguration die Bezeichnung „**TIMER_0**“. Die Verwendung ist zusammen mir dem „PORT_BI_0_LCD“ zur Ansteuerung der LCD mit den Funktionen in **lcd.h** vorgesehen.

4. **bidirektionales Port** mit 7 Bit zur Ansteuerung der LCD

Das Port hat in der Konfiguration die Bezeichnung „**PORT_BI_0_LCD**“. Die Verwendung ist zusammen mir dem „timer0“ zur Ansteuerung der LCD mit den Funktionen in **lcd.h** vorgesehen. Das Port ist mit dem Signal „intr[2]“ verbunden. Ein Interrupt muss damit in "**ISR(2)**“ im C-Programm behandelt werden. Die LCD Funktionen verwenden keinen Interrupt!

5. **Timer Modul 1** für die Erzeugung des Takt am Eingang von RTI0

Das Port hat in der Konfiguration die Bezeichnung „**TIMER_1**“. Das Bit 14 des Ausgangssignales „ti1_out[17:0]“ ist mit dem Takt Eingang von „timer_rti0“ verbunden. RTI0 kann damit Interrups in Sekunden Größe erzeugen. Die Verwendung ist in den Programmen „timer_led“ und „timer_led_pr“ zu sehen.

6. **RTI 0** für Interrupts in Sekunden Größe

Das Port hat in der Konfiguration die Bezeichnung „**RTI_0**“. Der Takt Eingang ist mit „ti1_out[14]“ vom „TIMER_1“ verbunden. RTI0 kann damit Interrupts in Sekunden Größe erzeugen. Die Verwendung ist in den Programmen „timer_led“ und „timer_led_pr“ zu sehen. Das Port ist mit dem Signal „intr[0]“ verbunden. Ein Interrupt muss damit in "**ISR(0)**“ im C-Programm behandelt werden.

7. **RTI 1** für Interrupts mit mehreren Sekunden Abstand

Das Port hat in der Konfiguration die Bezeichnung „**RTI_1**“. Der Takt Eingang ist mit „rti_out0“ vom „timer_rti0“ verbunden. RTI1 kann damit Interrupts mit einem Vielfachem von RTI0 erzeugen. Die Verwendung ist im Programm „app/timer_led_pr“ zu sehen. Das Port ist mit dem Signal „intr[1]“ verbunden. Ein Interrupt muss damit in "**ISR(1)**“ im C-Programm behandelt werden.

8. **Rotationstaster Port**

Das Port hat in der Konfiguration die Bezeichnung „**ROT_0**“. Es wertet die rechts/links Bewegung und das Drücken des Moduls aus. Die Verwendung ist in den Programmen „app/rot_ta“ und „app/rot_ta2“ zu sehen. Das Port ist mit dem Signal „intr[4]“ verbunden. Ein Interrupt muss damit in "**ISR(4)**“ im C-Programm behandelt werden.

9. **Port für die Schalter SW[3:0]**

Das Port hat in der Konfiguration die Bezeichnung „**PORT_IN_0_SW3_0**“. Mit ihm kann der Zustand der 4 Schiebeschalter sw[3:0] abgefragt werden. Das Port ist mit dem Signal „intr[5]“ verbunden. Ein Interrupt muss damit in "**ISR(5)**“ im C-Programm behandelt werden.

10. **Port für die Taster OST und WEST**

Das Port hat in der Konfiguration die Bezeichnung „**PORT_IN_1_O_W**“. Mit ihm können Eingaben durch die Taster OST und WEST realisiert werden. Der Taster OST ist dabei mit Bit 1 und der Taster WEST mit Bit 0 verbunden. Das Port ist mit dem Signal „intr[6]“ verbunden. Ein Interrupt muss damit in "**ISR(6)**“ im C-Programm behandelt werden.

11. **Port zur Ansteuerung der LED‘s**

Das Port hat in der Konfiguration die Bezeichnung „**PORT_OUT_2_LED**“. Mit ihm kann die LED[6:0] angesteuert werden.

12. **UART** zur Kommunikation mit einem Terminal Programm

Das Port hat in der Konfiguration die Bezeichnung „**UART_0**“. Über diese UART werden alle Monitorfunktionen ausgeführt. Die UART ist mit dem Steckverbinder J9 verbunden. Das Port ist mit dem Signal „intr[3]“ verbunden. Ein Interrupt muss damit in "**ISR(3)**“ im C-Programm behandelt werden. **Die Standard Ein- Ausgabefunktionen verwenden keinen Interrupt!**

13. **UART** zur Kommunikation mit einem Terminal Programm

Das Port hat in der Konfiguration die Bezeichnung „**UART_1**“. Über diese UART kann frei verfügt werden. Sie ist mit dem 9 poligen Steckverbinder J10 verbunden. Das Port ist mit dem Signal „intr[7]“ verbunden. Ein Interrupt muss damit in "**ISR(7)**“ im C-Programm behandelt werden. Die Verwendung ist in dem Programm „uart2“ zu sehen.

14. **Interrupt Taster** – Taster NORD

Dieser Taster ist an den Interrupt mit der höchsten Priorität angeschlossen. In dieser Konfiguration ist er mit „intr[10] verbunden. Der Interrupt muss damit in "**ISR(10)**“ im C-Programm behandelt werden.

Dieser Taster ist für die Fehlersuche geeignet. Man kann damit herausfinden, was das Programm gerade macht. Die Nutzung in der ISR(10) erfolgt mit folgenden Programmzeilen:

```
// Interrupt Taster (Taste Nord auf dem 3e Board)
ISR(10)(int reg12, int reg13, int reg14, int reg15) {
        printf("isr_10 PC= %05x\r\n",reg12);
}
```

Beim Betätigen der Taste werden dann ständig die Adresse des nächsten Befehls angezeigt. Damit ist zusammen mit der *.lst Datei Ihres Programms eine genaue Verfolgung der Aktivitäten des Programms möglich.

15. **RESET Taster** – Taster SÜD

Beim betätigen dieser Taste werden alle Hardware Module zurückgesetzt und die Firmware des **SpartanMC** neu gestartet.