Berner Fachhochschule

Technik und Informatik
EKT
Labor für technische Informatik

Manual
I2C Module zum CARME-Kit

Oktober 2010

Jürgen Schüpbach

Inhaltsverzeichnis

1 Übersicht I2C Module

Die I2C Module sind Zusatzprinte zum CARME-Kit , welche im Unterricht der technischen Informatik an der Berner Fachhochschule eingesetzt werden.

Die Module werden auf der linken Seite des CARME-Kit direkt in den I2C Stecker eingesteckt.

Die Module, mit Ausnahme von der LED Anzeige, können mit Hilfe eines 6 poligen Verbindungskabel an das Spartan FPGA Board angeschlossen werden.

In den nachfolgenden Tabellen sind die I2C Module aufgelistet.

Tabelle 1a: I2C Module

I/O Expander	8-Bit quasi-bidirection input output Erweiterungs Port
A/D D/A Converter	4-Kanal 8-Bit A/D Wandler und 1 8-Bit D/A Wandler
4-Digit Anzeige	4-Digit LED 7-Segment Anzeige
Licht Sensor	Helligkeitsensor in 4 Bereichen von 0 – 64000 Lux
EEPROM	EEPROM mit 4KBytes Speicherkapazität (0x0000-0x0FFF)

2 8-Bit I/O Expander

2.1 Beschreibung I/O Expander

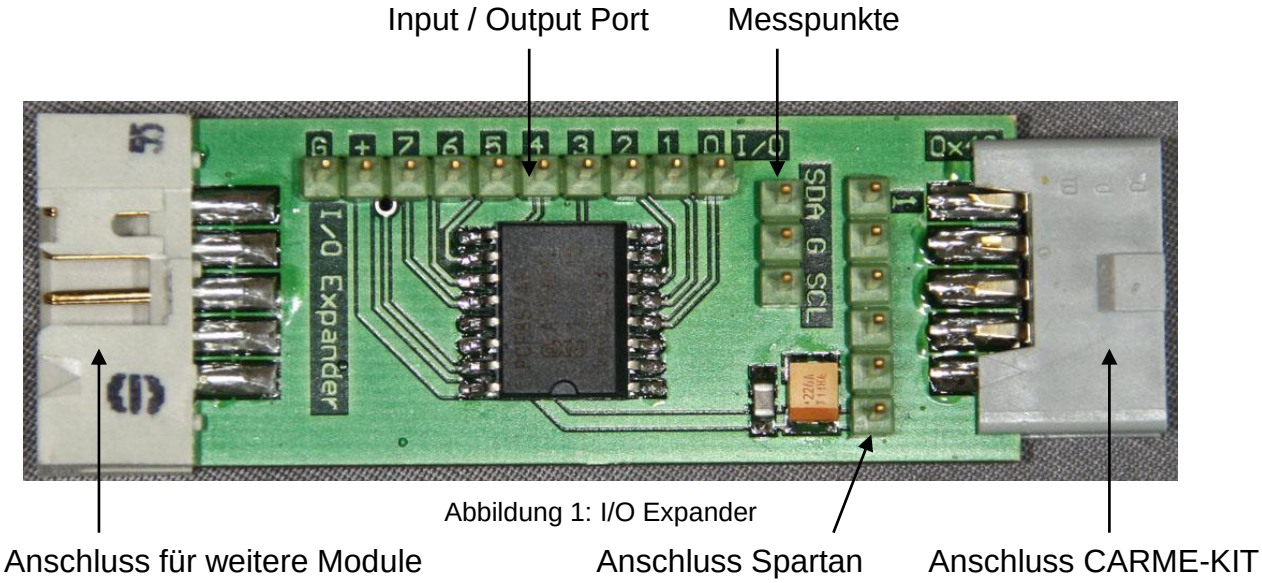

Abbildung 1: I/O Expander

Der 8-Bit I/O Expander ist ein quasi bidirektionale Port Erweiterung für den I2C Bus, welche mit dem Philips Baustein PCF8574 aufgebaut ist. Das I/O Modul wird vom CARME-Kit bzw. vom Spartan Board mit 3.3V gespeist. Das heisst, dass die Port Ein- und Ausgänge ebenfalls 3.3V Pegel aufweisen.

Die I2C Adresse ist fest verdrahtet und ist 0x40.

2.2 Anschlüsse

2.2.1.1 Anschluss I2C CARME-KIT

Die Buchse X101 wird seitlich in den I2C Stecker des CARME-Kit gesteckt.

Tabelle 2: Pinbelegung Buchse X101

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

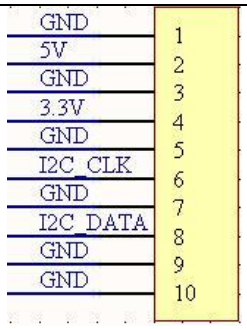

Abbildung 2: I2C_Buchse

2.2.1.2 Anschluss I2C für weitere Module

Am Stecker X102 können weitere I2C Module eingesteckt werden.

Tabelle 3: Pinbelegung Stecker X102

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

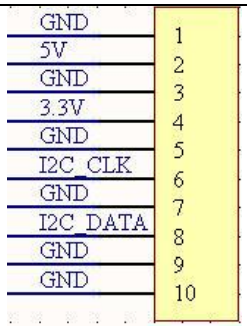

Abbildung 3: I2C_Stecker

2.2.1.3 Anschluss I2C Spartan Board

An der Stiftleiste X103 kann das I2C Modul über ein Kabel mit dem Spartan Board J18, J19 , J20 verbunden werden.

Tabelle 4: Pinbelegung Stiftleiste X103

Pin	Signal Name
1	I2C_DATA
2	I2C_CLK
3	
4	
5	GND
6	+3.3V

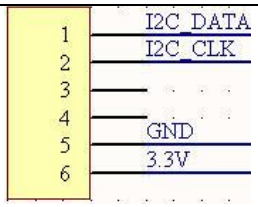

Abbildung 4: Anschluss Spartan

2.2.1.4 Anschluss I2C Messpunkte

An der Stiftleiste MP101 stehen die I2C Signale für Messzwecke zur Verfügung.

Tabelle 5: Pinbelegung Stiftleiste MP101

Pin	Signal Name
1	I2C_DATA
2	GND
3	I2C_CLK

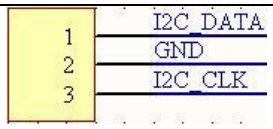

Abbildung 5: I2C_Messpunkte

2.2.1.5 Anschluss 8-Bit I/O Port

An der Buchse X104 steht das I/O Port zur Verfügung

Tabelle 6: Pinbelegung Buchse X104

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

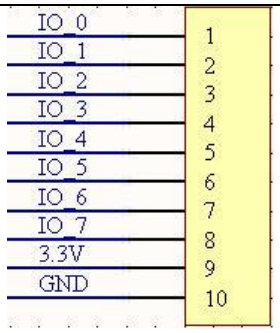

Abbildung 6: I/O Port

2.3 Beispielcode I/O Expander

```
// Beispielcode für 8-Bit I/O Expander

#include    <carme.h>
#include    <pxa270.h>
#include    <BSP_I2C_DRV.h>

/* module constant declaration  */

// Defines fuer PCF8574 Konfiguration

// I2C Adresse 0x40 / 0x41 für IO Expander
#define  I2C_PCF8574_ADDR          0x40


// Datenbyte 0x55 an I/O senden
int main(){
      INT8U daten = 0;

      daten = 0x55;

      //Daten senden
      error = BSP_I2C_sendByte(I2C_PCF8574_ADDR, daten); }


// Datenbyte empfangen
/** Byte vom IO Expander lesen  Achtung: es muss vor dem lesen immer ein FF
gesendet werden*/

int main(){
      INT8U daten = 0;

      //FF an IO senden
      error = BSP_I2C_sendByte(I2C_PCF8574_ADDR, 0xFF);
      //Daten lesen
      error = BSP_I2C_receiveByte(I2C_PCF8574_ADDR, &daten);
}
```

2.4 Schema I/O Expander

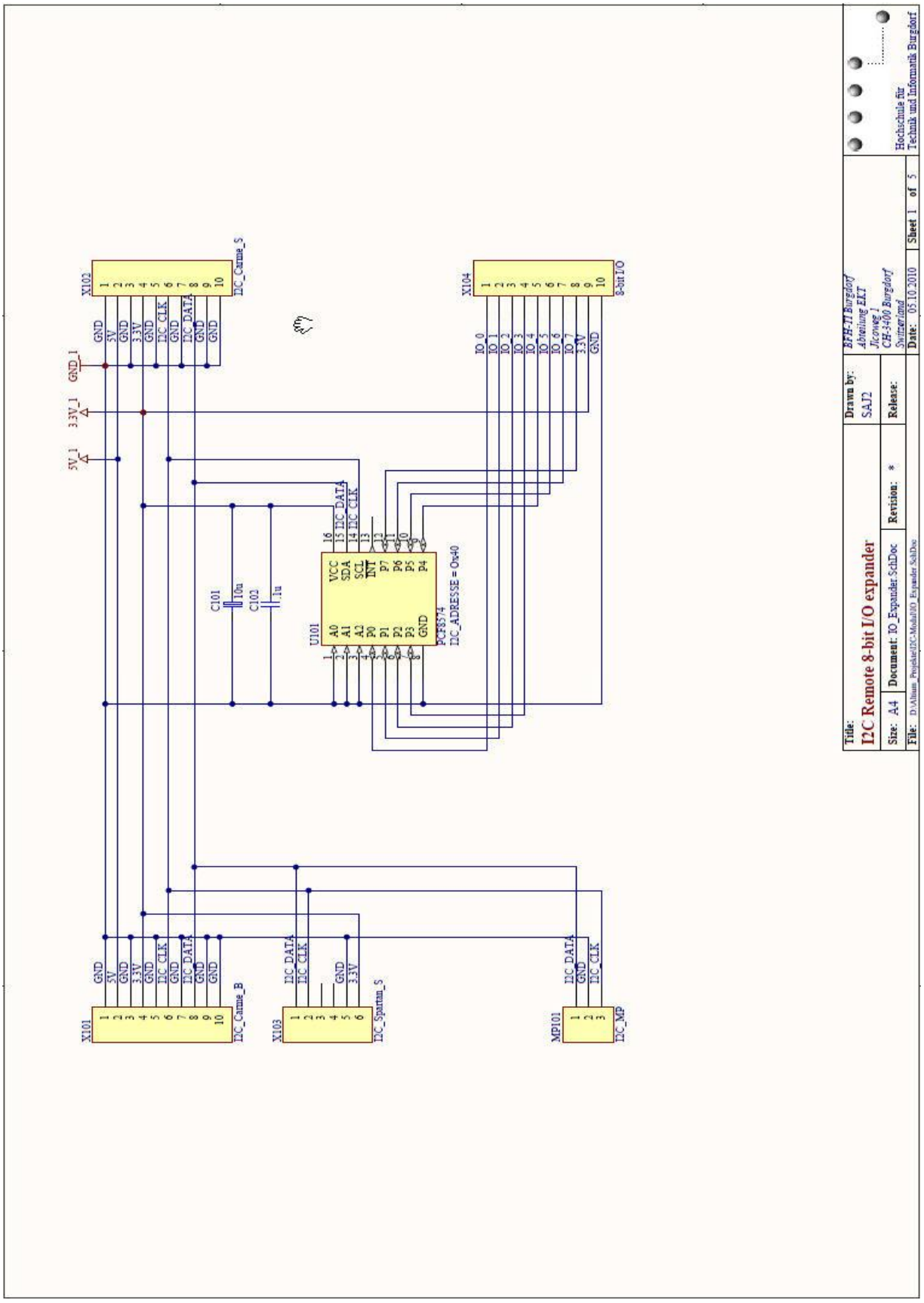

Abbildung 7: Schema I2C 8-Bit I/O Expander

3 A/D D/A Wandler

3.1 Beschreibung A/D D/A Wandler

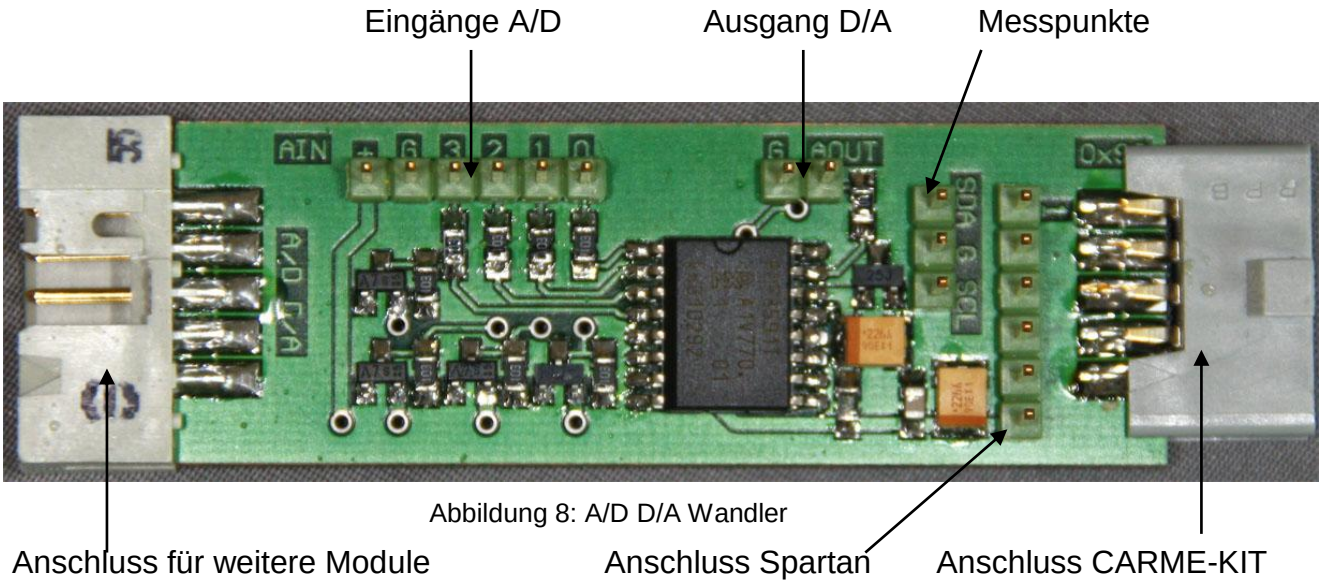

Abbildung 8: A/D D/A Wandler

Der A/D D/A Wandler besteht aus einem 4-Kanal 8-Bit A/D Wandler und aus einem 8-Bit D/A Wandler für den I2C Bus, welche mit dem Philips Baustein PCF8591 aufgebaut ist. Das Modul wird vom CARME-Kit bzw. vom Spartan Board mit 3.3V gespeist.

Die 4 A/D Eingänge haben einen Spannungsbereich von 0-5V (ca 20mV/Bit).

Der D/A Ausgang hat einen Spannungsbereich von 0-2.5V (ca 10mV/Bit).

Die I2C Adresse ist fest verdrahtet und ist 0x90.

3.2 Anschlüsse A/D D/A Wandler

3.2.1.1 Anschluss I2C CARME-KIT

Die Buchse X201 wird seitlich in den I2C Stecker des CARME-Kit gesteckt.

Tabelle 7: Pinbelegung Buchse X201

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

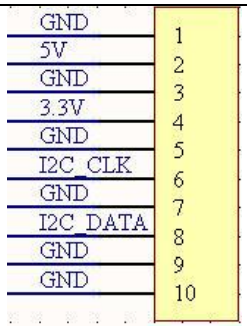

Abbildung 9: I2C_Buchse

3.2.1.2 Anschluss I2C für weitere Module

Am Stecker X202 können weitere I2C Module eingesteckt werden.

Tabelle 8: Pinbelegung Stecker X202

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

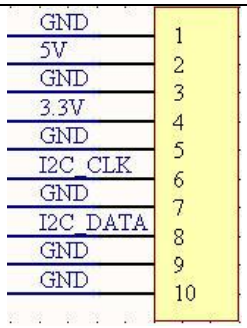

Abbildung 10: I2C_Stecker

3.2.1.3 Anschluss I2C Spartan Board

An der Stiftleiste X203 kann das I2C Modul über ein Kabel mit dem Spartan Board J18, J19 , J20 verbunden werden.

Tabelle 9: Pinbelegung Stiftleiste X203

Pin	Signal Name
1	I2C_DATA
2	I2C_CLK
3	
4	
5	GND
6	+3.3V

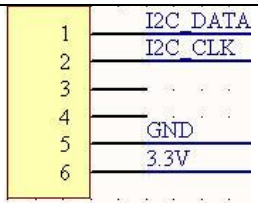

Abbildung 11: Anschluss Spartan

3.2.1.4 Anschluss I2C Messpunkte

An der Stiftleiste MP201 stehen die I2C Signale für Messzwecke zur Verfügung.

Tabelle 10: Pinbelegung Stiftleiste MP201

Pin	Signal Name
1	I2C_DATA
2	GND
3	I2C_CLK

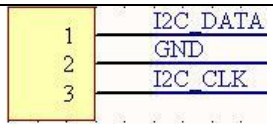

Abbildung 12: I2C_Messpunkte

3.2.1.5 Anschluss D/A Wandler Ausgang

An der Stiftleiste X202 ist das Ausgangssignal des D/A Wandlers zur Verfügung.

Tabelle 11: Pinbelegung Stiftleiste MP202

Pin	Signal Name
1	D/A OUT
2	GND

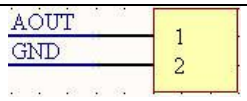

Abbildung 13: D/A Ausgang

3.2.1.6 Anschluss A/D Wandler

An der Stiftleiste X205 sind die Eingänge des A/D Wandlers (maximum 5V).

Tabelle 12: Pinbelegung Buchse X204

Pin	Signal Name
1	AIN0
2	AIN1
3	AIN2
4	AIN3
5	GND
6	+3.3V

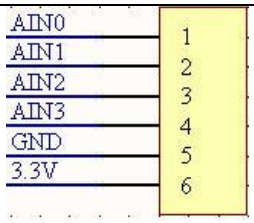

Abbildung 14: A/D Eingänge

3.3 Beispielcode A/D D/A Wandler

```
// Beispielcode für A/D D/A Wandler

#include    <carme.h>
#include    <pxa270.h>
#include    <BSP_I2C_DRV.h>

/* module constant declaration  */

// Defines fuer PCF8591 Konfiguration

//  i2C Adresse 0x90 / 0x91 für A/D D/A Wandler
#define  I2C_PCF8591_ADDR          0x90

// Choose a control byte for the PCF8591.
//    D7 = reserved (0)
//    D6 = Analog output enable Flag
//    D5 = Analog Input Programming
//    D4 = Analog Input Programming
//    D3 = reserved (0)
//    D2 = Aut increment Flag (active if 1)
//    D1 = A/D Channel Number
//    D0 = A/D Channel Number

#define  I2C_PCF8591_AOEF          (1<<6)      // Analog Output Enable Flag
#define  I2C_PCF8591_AIP_0         (00<<4)     // Analog In Prog.= 4 Single
#define  I2C_PCF8591_AIP_1         (01<<4)     // Analog In Prog.= 3 Diff
#define  I2C_PCF8591_AIP_2         (10<<4)     // Analog In Prog.= 2 S 1D
#define  I2C_PCF8591_AIP_3         (11<<4)     // Analog I Prog.= 2 Diff
#define  I2C_PCF8591_AUTO_INC      (1<<2)      // Auto Increment Flag
#define  I2C_PCF8591_CHAN_0        (00<<0)     // AD Chanel 0
#define  I2C_PCF8591_CHAN_1        (01<<0)     // AD Chanel 1
#define  I2C_PCF8591_CHAN_2        (10<<0)     // AD Chanel 2
#define  I2C_PCF8591_CHAN_3        (11<<0)     // AD Chanel 3
```

```
// Datenbyte von A/D Channel 0 lesen
int main(){
	INT8U command = 0;
	INT8U daten = 0;

	// 4 single channels, channel 0 lesen
	command = I2C_PCF8591_AOEF | I2C_PCF8591_AIP_0 | I2C_PCF8591_CHAN_0;

	// Command an A/D D/A senden
	error = BSP_I2C_sendByte(I2C_PCF8591_ADDR, command);

	// Daten Byte von A/D lesen
	error = BSP_I2C_receiveByte(I2C_PCF8591_ADDR, &daten);

}


// Datenbyte 0x55 an D/A senden
int main(){
	INT8U command [2];

	// D/A Kanal enablen
	command [0] = I2C_PCF8591_AOEF;

	// zu sendender Wert
	command [1] = 0x55;

	// Daten an D/A senden
	error = BSP_I2C_sendArray(I2C_PCF8591_ADDR, command, 2);

}
```

3.4 Schema A/D D/A Wandler

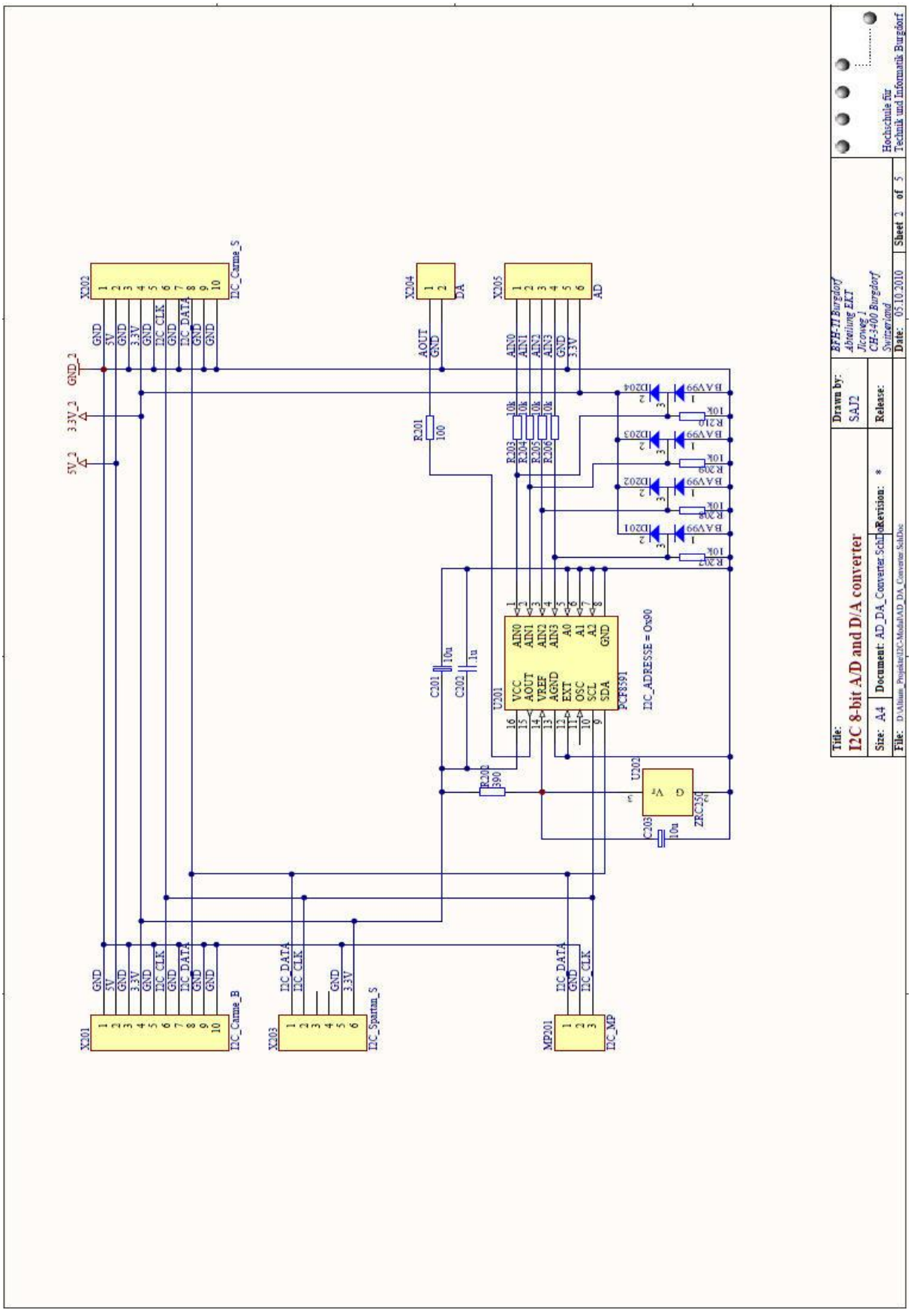

Abbildung 15: Schema I2C A/D D/A Wandler

4 4-Digit 7-Segement Anzeige

4.1 Beschreibung Anzeige

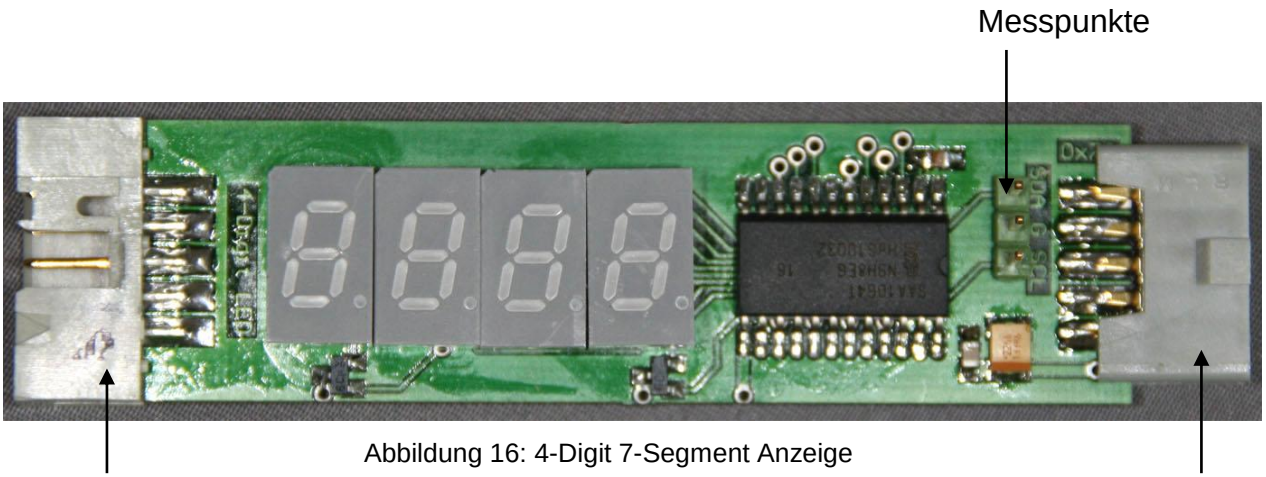

Abbildung 16: 4-Digit 7-Segment Anzeige

Die Anzeige besteht aus einem 4 7-Segment LED, welche vom Philips Baustein SAA1064 angesteuert werden. Das Modul wird vom CARME-Kit mit 5V gespeist. Eine Verwendung mit dem Spartan Board ist wegen den benötigten 5V Speisespannung nicht möglich.

Die I2C Adresse ist fest verdrahtet und ist 0x70.

4.2 Anschlüsse 4-Digit 7-Segment Anzeige

4.2.1.1 Anschluss I2C CARME-KIT

Die Buchse X301 wird seitlich in den I2C Stecker des CARME-Kit gesteckt.

Tabelle 13: Pinbelegung Buchse X301

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

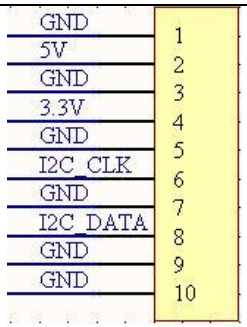

Abbildung 17: I2C_Buchse

4.2.1.2 Anschluss I2C für weitere Module

Am Stecker X302 können weitere I2C Module eingesteckt werden.

Tabelle 14: Pinbelegung Stecker X302

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

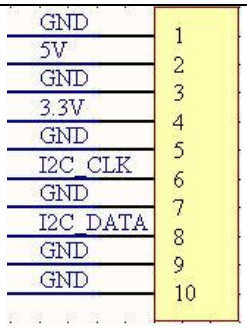

Abbildung 18: I2C_Stecker

4.2.1.3 Anschluss I2C Messpunkte

An der Stiftleiste MP301 stehen die I2C Signale für Messzwecke zur Verfügung.

Tabelle 15: Pinbelegung Stiftleiste MP301

Pin	Signal Name
1	I2C_DATA
2	GND
3	I2C_CLK

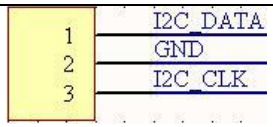

Abbildung 19: I2C_Messpunkte

4.3 Beispielcode 4-Digit 7-Segment Anzeige

```
// Beispielcode 4-Digit 7-Segment Anzeige

#include    <carme.h>
#include    <pxa270.h>
#include    <BSP_I2C_DRV.h>

/* module constant declaration  */

// Defines fuer SAA104 Konfiguration

// I2C Adressen 0x70/0x71 für SAA1064 LED Treiber
#define I2C_SAA1064_ADDR  0x70

// Choose a control byte for the SAA1064.
//    D7 = reserved
//    D6 = adds 12 mA to segment output current
//    D5 = adds  6 mA to segment output current
//    D4 = adds  3 mA to segment output current
//    D3 = segment test (all outputs are switched on)
//    D2 = enable digits 2 + 4
//    D1 = enable digits 1 + 3
//    D0 = dynamic mode (multiplex digits).}
```

```
#define  SAA1064_ADD_12mA (1<<6)            // +12mA Digitstrom
#define  SAA1064_ADD_9mA  (11<<4)           // +9mA Digitstrom
#define  SAA1064_ADD_6mA  (1<<5)            // +6mA Digitstrom
#define  SAA1064_ADD_3mA  (1<<4)            // +3mA Digitstrom
#define  SAA1064_SEGTEST  (1<<3)            // Alle Segmente an
#define  SAA1064_ENA24  (1<<2)              // Segmente 2,4 an
#define  SAA1064_ENA13  (1<<1)              // Segmente 1,3 an
#define  SAA1064_DYNMODE  (1<<0)            // Dynic Mode (multiplexed)

// instructions byte for the SAA1064
#define  SAA1064_SUBADDR_0         0x00    // Startadresse der Register


// Testmuster "1234" am Display anzeigen
int main(){
      // Displaybuffer
      INT8U ledArray[6];

      // 7-Segmentcode fuer LED. Codierungen fuer die Ziffern 0..9
      // D0..D6 Segmente A..G
      // D7 Dezimalpunkt
      INT8U int2sevenSeg[]={0x3F,0x06,0x5B,0x4F,0x66,
            0x6D,0x7D,0x07,0x7F,0x6F};


      // Testmuster "1234" in 7-Segment codiertes Array uebertragen

      //Instructions Byte
      ledArray[0]=SAA1064_SUBADDR_0;

      //Control Byte
      ledArray[1]=SAA1064_ADD_12mA | SAA1064_ENA24 | SAA1064_ENA13 |
                  SAA1064_DYNMODE;
      //Digit Tausender
      ledArray[2]=int2sevenSeg[1];

      //Digit Hunderter
      ledArray[3]=int2sevenSeg[2];

      //Digit Zehner
      ledArray[4]=int2sevenSeg[3];

      //Digit Einer
      ledArray[5]=int2sevenSeg[4];

      //Array senden
      error = BSP_I2C_sendArray(I2C_SAA1064_ADDR, ledArray, 6);

}
```

4.4 Schema 4-Digit 7-Segment Anzeige

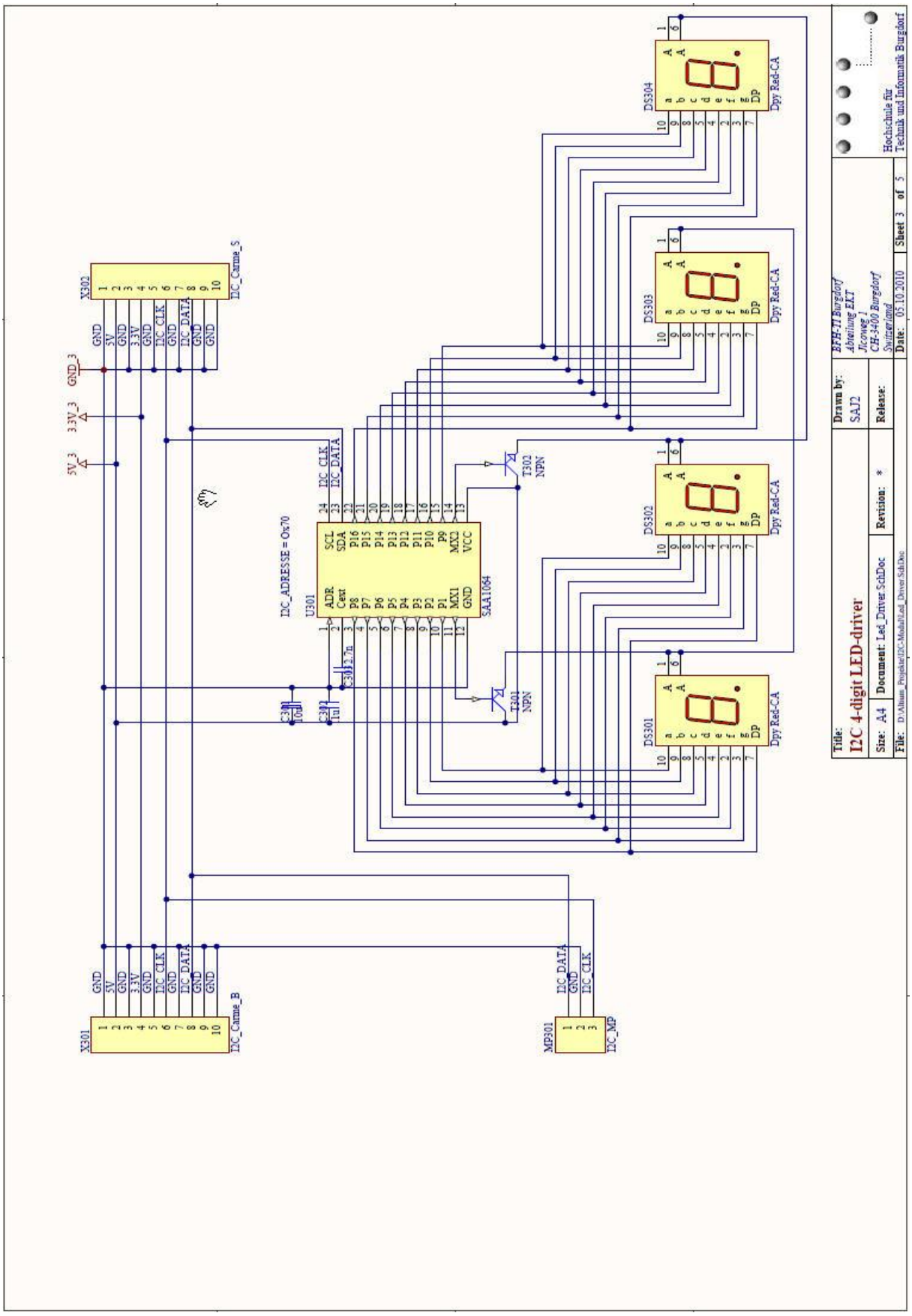

Abbildung 20: Schema 4-Digit 7-Segment Anzeige

5 Licht Sensor

5.1 Beschreibung Licht Sensor

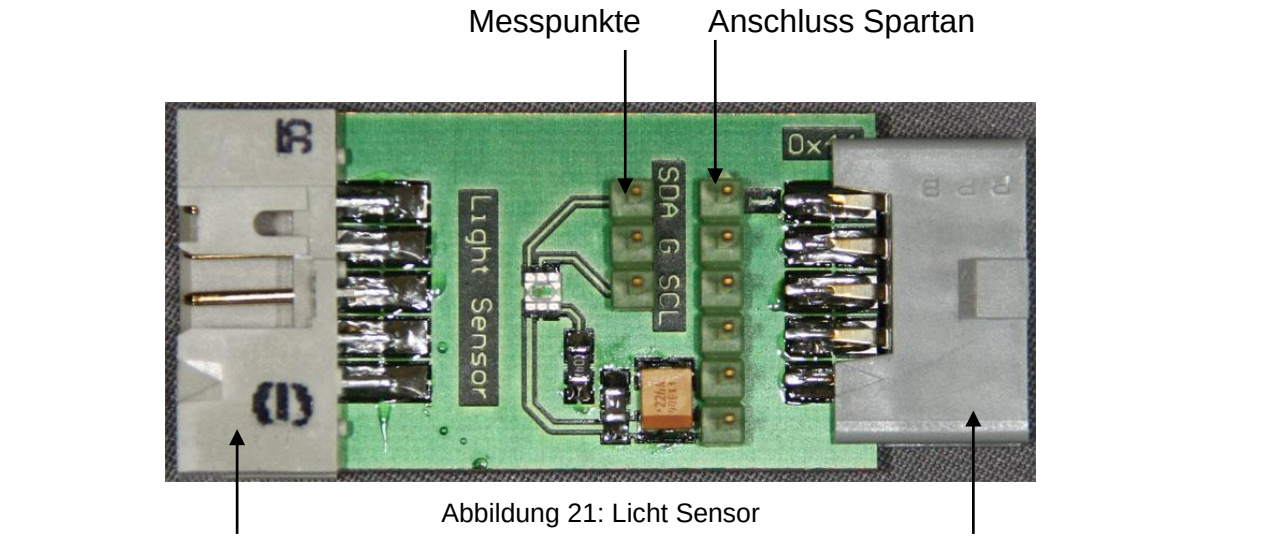

Abbildung 21: Licht Sensor

Der Licht Sensor verwendet den Light- to-Digital vom Output Sensor ISL29003 von Intersil. Der Baustein misst die Lichtstärke der Umgebung, welche dann amI2C-Bus ausgelesen werden kann. Das Modul wird vom CARME-Kit bzw. vom Spartan Board mit 3.3V gespeist.

Die I2C Adresse ist fest verdrahtet und ist 0x88.

5.2 Anschlüsse Licht Sensor

5.2.1.1 Anschluss I2C CARME-KIT

Die Buchse X401 wird seitlich in den I2C Stecker des CARME-Kit gesteckt.

Tabelle 16: Pinbelegung Buchse X401

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

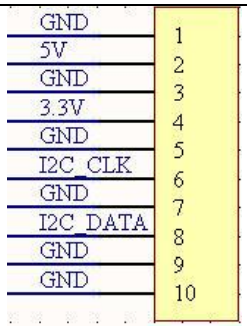

Abbildung 22: I2C_Buchse

5.2.1.2 Anschluss I2C für weitere Module

Am Stecker X402 können weitere I2C Module eingesteckt werden.

Tabelle 17: Pinbelegung Stecker X402

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

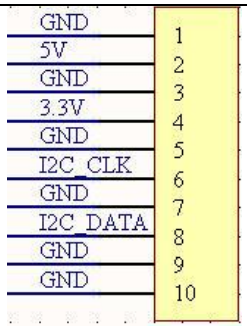

Abbildung 23: I2C_Stecker

5.2.1.3 Anschluss I2C Spartan Board

An der Stiftleiste X403 kann das I2C Modul über ein Kabel mit dem Spartan Board J18, J19 , J20 verbunden werden.

Tabelle 18: Pinbelegung Stiftleiste X403

Pin	Signal Name
1	I2C_DATA
2	I2C_CLK
3	
4	
5	GND
6	+3.3V

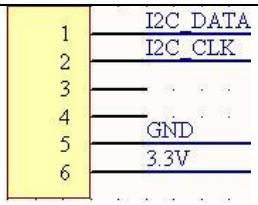

Abbildung 24: Anschluss Spartan

5.2.1.4 Anschluss I2C Messpunkte

An der Stiftleiste MP401 stehen die I2C Signale für Messzwecke zur Verfügung.

Tabelle 19: Pinbelegung Stiftleiste MP401

Pin	Signal Name
1	I2C_DATA
2	GND
3	I2C_CLK

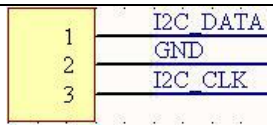

Abbildung 25: I2C_Messpunkte

5.3 Beispielcode Licht Sensor

```
// Beispielcode für Light Sensor

#include    <carme.h>
#include    <pxa270.h>
#include    <BSP_I2C_DRV.h>

/* module constant declaration  */

// Defines fuer ISL29003 Konfiguration

// I2C Adresse 0x88 / 0x89 für Light Sensor
#define  I2C_ISL29003_ADDR          0x88

// Adresse Command Register
#define      I2C_ISL29003_COMMAND   0x00

// Adresse Control Register
#define      I2C_ISL29003_CONTROL   0x01

// Adresse Datenregister LSB des Sensors
#define      I2C_ISL29003_LSB_SENSOR 0x04

// Adresse Datenregister MSB des Sensors
#define      I2C_ISL29003_MSB_SENSOR 0x05

// Choose a command byte for theISL29003.
//    D7 = Enable 0: disable ADC core, 1: enable ADC core)
//    D6 = ADCPD (0: Normal Operation, 1: Power down)
//    D5 = Timing_mode (0: Integration intern, 1: Integration extern)
//    D4 = reserved (0)
//    D3 = Mode Bit1 (0: Diode1 16Bit, 1: Diode2 16Bit)
//    D2 = Mode Bit0 (2: Differenz between Diode1&2 15Bit, 3: reserved)
//    D1 = Width Bit1 (number of clock cycles integration time)
//    D0 = Width Bit0 (0: 65536, 1:4096, 2: 256, 3:16)

#define  I2C_ISL29003_ENABLE        (1<<7)      // enable ADC-core
#define  I2C_ISL29003_MODE_16       (00<<0)     // Diode1 16-bit resolution

// Choose a control byte for theISL29003.
//    D7 = Ext_Mode (always set to 0)
//    D6 = Test_Mode (always set to 0)
//    D5 = Int_Flag (0: Interrupt cleared, 1: Interrupt triggered)
//    D4 = reserved (0)
//    D3 = Gain Bit1 (0: 0 to 1000 lux, 1: 0 to 4000 lux)
//    D2 = Gain Bit0 (2: 0 to 16000 lux, 3: 0 to 64000 lux)
//    D1 = Int_Persistant Bit1 (Interrupt is triggered after)
//    D0 = Int_Persistant Bit0 (0: 1, 1: 4, 2: 8, 3: 16 cycles)

#define  I2C_ISL29003_GAIN_1        (00<<2)     // Gain  1000Lux
#define  I2C_ISL29003_GAIN_4        (01<<2)     // Gain  4000Lux
#define  I2C_ISL29003_GAIN_16       (10<<2)     // Gain 16000Lux
#define  I2C_ISL29003_GAIN_64       (11<<2)     // Gain 64000Lux
```

```
// Lichtintensität vom ISL29003 lesen mit Gain 0-4000 lux

int main(){
	INT8U command[2];
	INT8U lightsensorByte = 0;
	INT8U lightsensorData = 0;

	// Adresse Command Register
	command[0] = I2C_ISL29003_COMMAND;

	// Enable ADC-Core, Diode1 16bit
	command[1] = I2C_ISL29003_ENABLE | I2C_ISL29003_MODE_16;

	// Command Register an Sensor senden
	error = BSP_I2C_sendArray(I2C_ISL29003_ADDR, command, 2);

	// Adresse Control Register
	command[0] = I2C_ISL29003_CONTROL;

	// Gain 4000Lux
	command[1] = I2C_ISL29003_GAIN_4;

	// Controll Register an Sensor senden
	error = BSP_I2C_sendArray(I2C_ISL29003_ADDR, command, 2);

	// LSB Daten des Sensor
	command[0] = I2C_ISL29003_LSB_SENSOR;

	// auslesen
	error = BSP_I2C_sendByte(I2C_ISL29003_ADDR, command[0]);
	error = BSP_I2C_receiveByte(I2C_ISL29003_ADDR, &lightsensorByte);
	lightsensorData = lightsensorByte;

	// MSB Daten des Sensor
	command[0] = I2C_ISL29003_MSB_SENSOR;

	// auslesen
	error = BSP_I2C_sendByte(I2C_ISL29003_ADDR, command[0]);
	error = BSP_I2C_receiveByte(I2C_ISL29003_ADDR, &lightsensorByte);
	lightsensorData = (lightsensorByte << 8 | lightsensorData);

}
```

5.4 Schema Licht Sensor

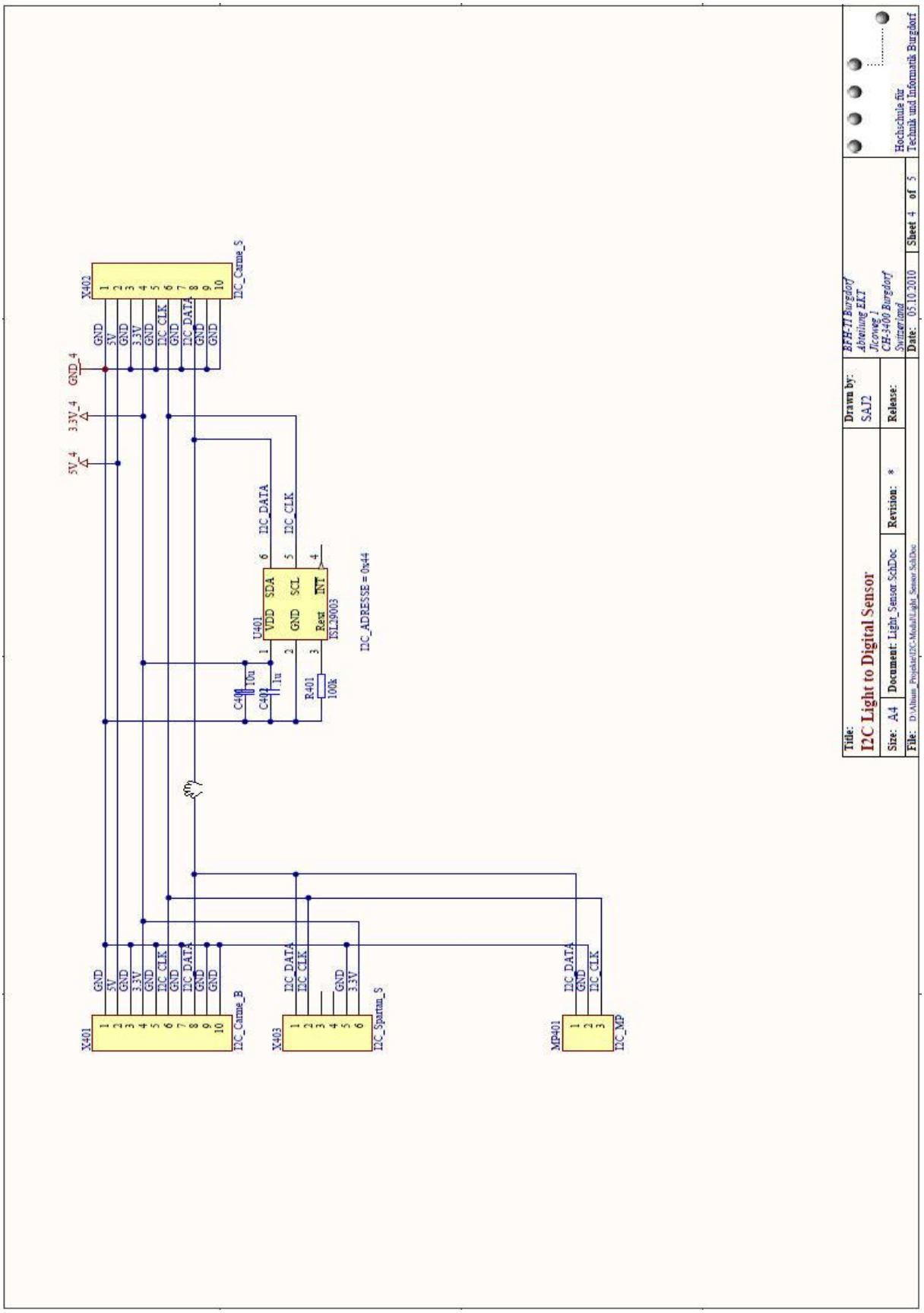

Abbildung 26: Schema Licht Sensor

6 EEPROM

6.1 Beschreibung EEPROM

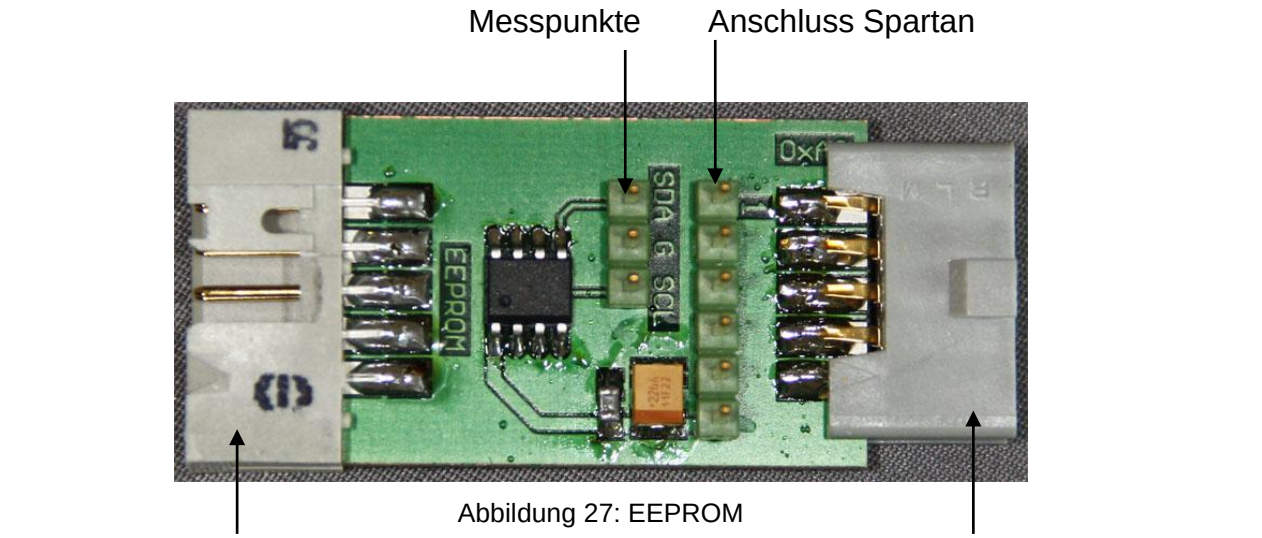

Abbildung 27: EEPROM

Das EEROM 24LC32A von Microchip ist 32Kbit (4KByte) gross, welches als single Block organisiert ist. Der Adressenraum ist 0x0000 bis 0x0FFF. Das Modul wird vom CARME-Kit bzw. vom Spartan Board mit 3.3V gespeist.

Die I2C Adresse ist fest verdrahtet und ist 0xA2.

6.2 Anschlüsse EEPROM

6.2.1.1 Anschluss I2C CARME-KIT

Die Buchse X501 wird seitlich in den I2C Stecker des CARME-Kit gesteckt.

Tabelle 20: Pinbelegung Buchse 5401

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

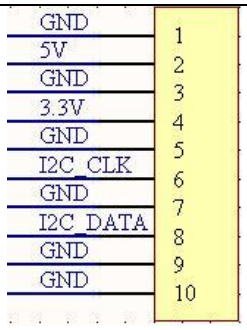

Abbildung 28: I2C_Buchse

6.2.1.2 Anschluss I2C für weitere Module

Am Stecker X502 können weitere I2C Module eingesteckt werden.

Tabelle 21: Pinbelegung Stecker X502

Pin	Signal Name
1	GND
2	+5V
3	GND
4	+3.3V
5	GND
6	I2C_CLK
7	GND
8	I2C_DATA
9	GND
10	GND

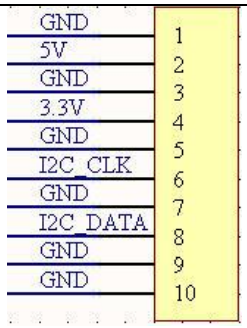

Abbildung 29: I2C_Stecker

6.2.1.3 Anschluss I2C Spartan Board

An der Stiftleiste X503 kann das I2C Modul über ein Kabel mit dem Spartan Board J18, J19 , J20 verbunden werden.

Tabelle 22: Pinbelegung Stiftleiste X503

Pin	Signal Name
1	I2C_DATA
2	I2C_CLK
3	
4	
5	GND
6	+3.3V

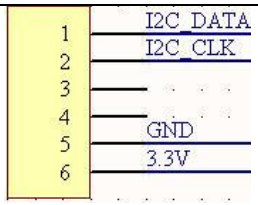

Abbildung 30: Anschluss Spartan

6.2.1.4 Anschluss I2C Messpunkte

An der Stiftleiste MP501 stehen die I2C Signale für Messzwecke zur Verfügung.

Tabelle 23: Pinbelegung Stiftleiste MP501

Pin	Signal Name
1	I2C_DATA
2	GND
3	I2C_CLK

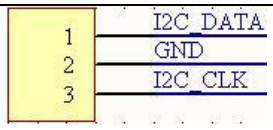

Abbildung 31: I2C_Messpunkte

6.3 Beispielcode EEPROM

```
// Beispielcode für EEPROM

#include    <carme.h>
#include    <pxa270.h>
#include    <BSP_I2C_DRV.h>

/* module constant declaration  */

// Defines fuer 24LC32A Konfiguration

// I2C Adresse 0xA2 / 0xA3 für externes EEPROM
#define  I2C_EEPROM_ADDR            0xA2


// Datenbyte 0x55 an Adresse 0x01aa ins EEPROM schreiben

int main(){

      INT8U addressLowByte = 0xaa;
      INT8U addressHighByte = 0x=1;
      INT8U daten = 0x55;

      err = BSP_I2C_connect(I2C_EEPROM_ADDR, I2C_TX);
      err = BSP_I2C_send(addressHighByte, 0);
      err = BSP_I2C_send(addressLowByte, 0);
      err = BSP_I2C_send(daten, 1);
      err = BSP_I2C_disconnect();
}


// Datenbyte von Adresse 0x01aa vom EEPROM lesen

int main(){

      INT8U addressLowByte = 0xaa;
      INT8U addressHighByte = 0x=1;
      INT8U daten = 0;

    err = BSP_I2C_connect(I2C_EEPROM_ADDR, I2C_TX);
    err = BSP_I2C_send(addressHighByte, 0);
    err = BSP_I2C_send(addressLowByte, 0);
    err = BSP_I2C_connect(I2C_EEPROM_ADDR, I2C_RX);
      err = BSP_I2C_receive(1, &daten, 1);
      err = BSP_I2C_disconnect();
}
```

6.4 Schema EEPROM

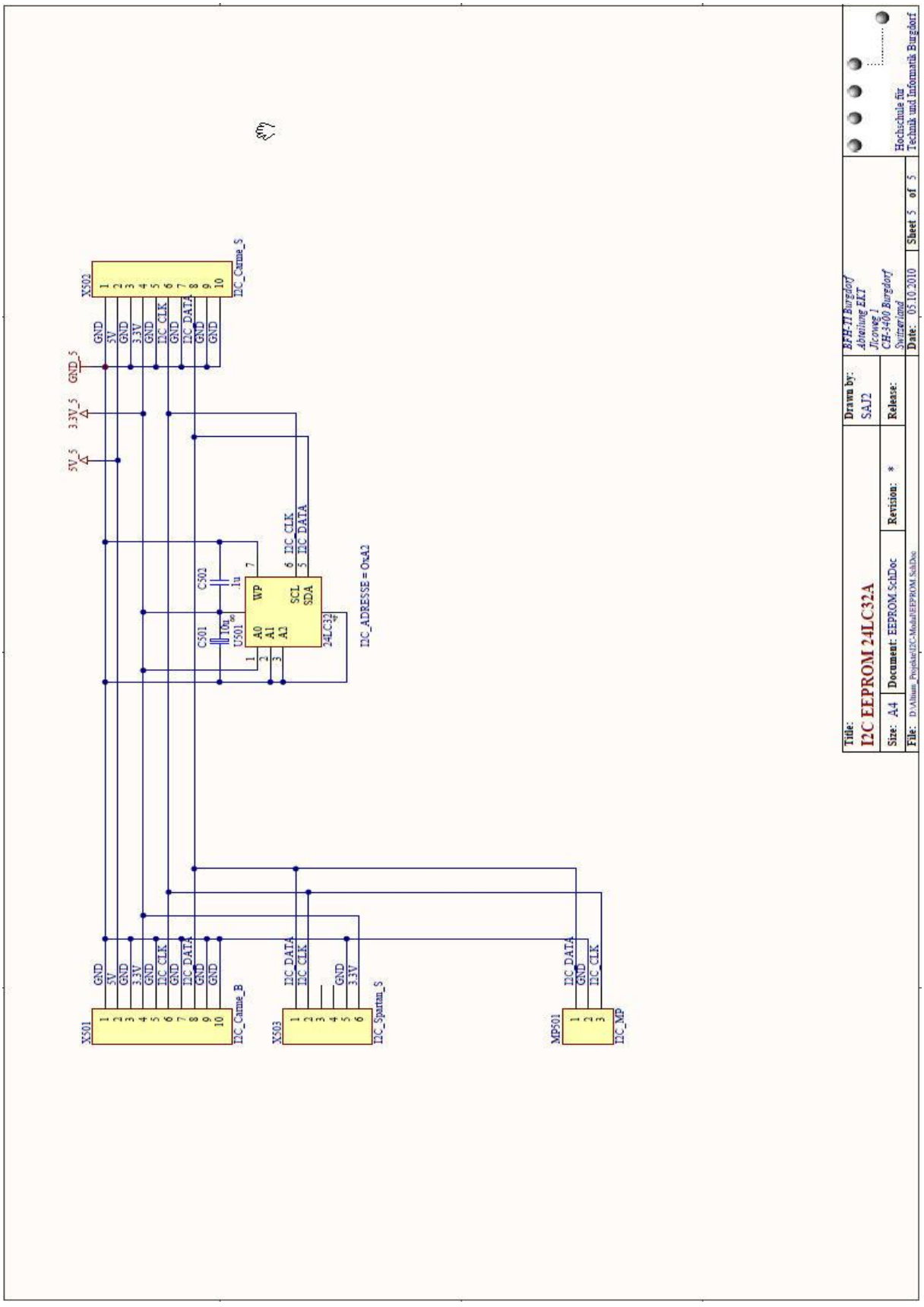

Abbildung 32: Schema EEPROM