

PCT Praktikum: Memory ROM

1 Einleitung

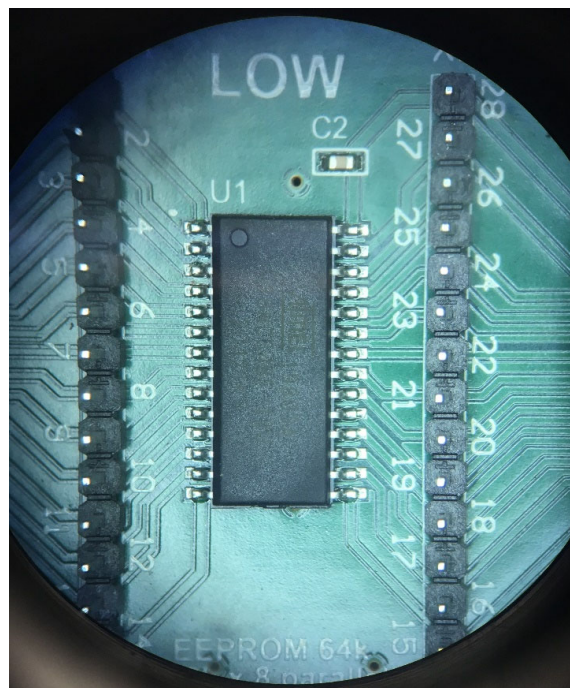
In diesem Praktikum schliessen Sie zwei vorprogrammierte EEPROMs an den externen Bus des CT Boards an. Mit der Speicheransicht in Keil uVision können Sie den Inhalt der Speicherbausteine lesen. Anschliessend schreiben Sie eigene Programme, welche auf die Bausteine zugreifen.

2 Lernziele

- Sie können Speicherbausteine korrekt an den externen Bus des CT Boards anschliessen.
- Sie können den Begriff und die Auswirkungen einer unvollständigen Adressdekodierung erklären.
- Sie können mit eigenen C-Programmen auf extern angeschlossene Speicher zugreifen.

3 Material

- 1x CT Board
- 1x CT Memory ROM Kit
 - 1 x Board
 - 3x Flachbandkabel für externen Busanschluss (noch nicht im Kit enthalten)
 - Jumper-Kabel um die beiden EEPROM Bausteine anzuschliessen
- 1x kurzes USB Kabel (USB A <-> USB mini)
um das ROM Board vom CT Board aus zu speisen



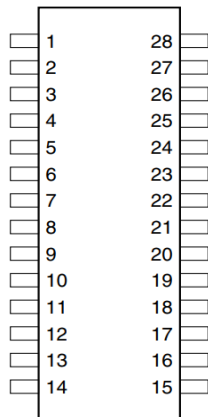
4 Grundlagen

In *Modus 2* schaltet das CT Board den externen Speicherbus auf die Schnittstellen P1 bis P4. Die genaue Belegung in diesem Modus ist auf ennis.zhaw.ch unter **External Memory Bus** beschrieben.

Für diese Funktion muss das CT Board in *Modus 2* betrieben werden!



Die EEPROM Bausteine haben *nur* einen 8 Bit Datenbus. Die zwei EEPROMs müssen deshalb parallel an den 16 Bit breiten Datenbus des CT Boards angeschlossen werden



Gegeben sind zwei EEPROMs vom Typ AT28C64B und der Bauform 28-lead SOIC. Diese haben eine Organisation von 8K * 8-Bit, also 64 KBit. Das mit "Low" bezeichnete enthält die Bytes der geraden Adressen und das mit "High" diejenigen der ungeraden.

EEPROM = electrically-erasable programmable read-only memory

Ein Baustein basierend auf Floating Gate Technologie, welche nicht flüchtig ist. Der Inhalt kann elektrisch gelöscht werden.

Abbildung 1: Bauform EEPROM AT28C64B

Der Inhalt der beiden EEPROMs ist wie folgt definiert:

0x000	0x00, 0x01, 0x02, ...	Die ersten 256 Bytes enthalten Zahlenwerte von 0 bis 255 in aufsteigender Reihenfolge.
0x100	Nur ...	ASCII-Text von Wilhelm Busch.
0x200	0b1100 `0000, ...	Codewandlungstabelle: Binär → 7-Segment-Code (invertiert)
0x300	--	Unbenutzter Bereich
0x400	0x00, 0x01, 0x02, ...	0 bis 255, aber mit 5 Fehlern!
0x500	--	Unbenutzter Bereich

5 Aufgaben

5.1 Anschluss der EEPROM Bausteine

Die zwei EEPROMs mit je 8-bit breitem Datenbus sollen so an das CT Board angeschlossen werden, dass sie zusammen ein Memory mit 16-bit breitem Datenbus bilden. Die Bausteine werden so angeschlossen, dass sie von der tiefsten Adresse im Bereich SRAM – Device 2 (angesprochen mit **NE2**) an aufwärts adressiert werden können. Dabei enthält der mit 'Low' bezeichnete Baustein die Bytes, welche an geraden Adressen liegen, während der mit 'High' bezeichnete Baustein die Bytes enthält, welche an ungeraden Adressen liegen (Little Endian).

- a) Zeichnen Sie den anzuschliessenden Speicher in der Memory Map des CT Boards ein (Abbildung 2) und beschriften Sie die tiefste und die höchste Adresse.

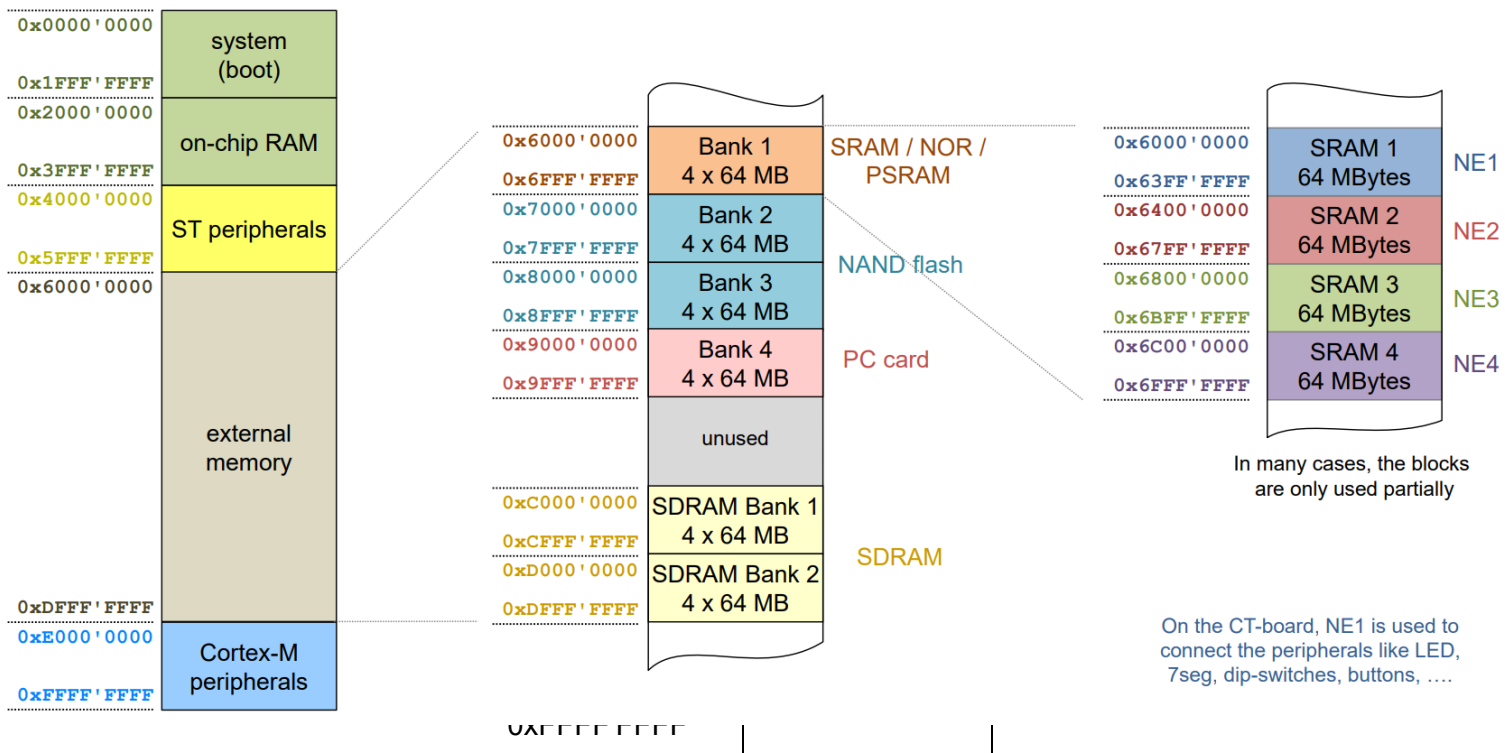


Abbildung 2: Memory Map – Speicherbereich der zwei EEPROMs

- b) Zeichnen Sie in Abbildung 3 ein, wie die Bausteine an das CT Board angeschlossen werden müssen. Abbildung 4 dient als Hilfestellung, damit Sie die Speicherbausteine pinweise beschriften können.



Abbildung 3: Anschluss von zwei 8K x 8-Bit EEPROMs an einem 16-Bit Datenbus

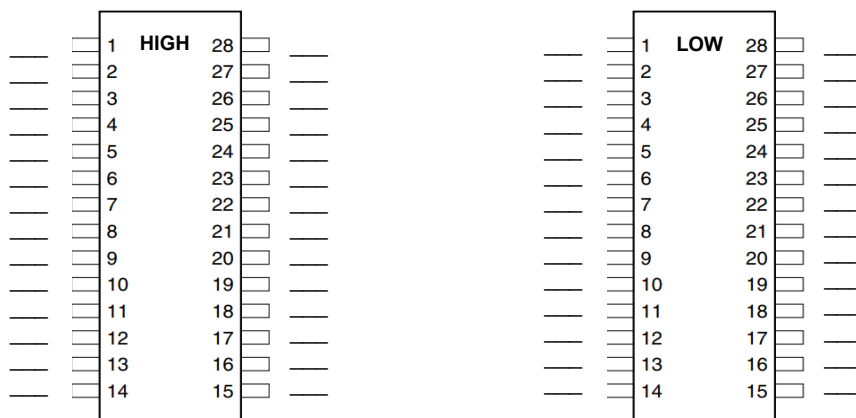


Abbildung 4: Anschluss von zwei 8K x 8-Bit EEPROMs an einem 16-Bit Datenbus

Der „Flexible Memory Controller“ (FMC) ist im gegebenen Programmrahmen bereits so konfiguriert, dass das Memory an **NE2** als 16-bit breiter Speicher angesprochen wird. Aus diesem Grund wird auf dem externen Adress-Bus nicht die auf dem internen Bus verwendete Byteadresse angelegt, sondern eine um ein Bit nach rechts verschobene half-word Adresse.

Beispiele:

Byteadresse	→ intern_A[25:0]	→ extern_A[25:0] = intern_A[25:1] >> 1
0x6922'2222	→ 0x122'2222	→ 0x091'1111
0x6922'2223	→ 0x122'2223	→ 0x091'1111

Beachten Sie dazu das Adressierschema im Anhang.

Das heisst, das Memory kann einen Zugriff auf eine gerade Adresse nicht von einem Zugriff auf die zugehörige ungerade Adresse unterscheiden. Im Lesefall spielt dies aber keine Rolle, da bei einem Bytezugriff die CPU selbständig das richtige Byte aus dem durch das Memory gelieferten half-word auswählt. Bei einem hypothetischen Write-Zugriff müsste das Memory die Signale **NBL0** und **NBL1** decodieren.

- c) Vervollständigen Sie die Schaltung auf dem ROM Board mittels Jumper-Kabel. Das ROM Board kann über den USB Port des CT-Boards gespeist werden. Siehe Abbildung 5. Die Pinbelegung der Speicherbausteine finden Sie im Schaltschema des ROM Boards (CTP_ROM_Board_Schema.pdf) oder im Datenblatt (AT28C64B.pdf).

Folgende Jumper-Kabel sind im ROM-Kit enthalten:

Stückzahl	Farbe	Verwendung
24	Weiss	Adressleitung
16	Grün	Datenleitung
6	Gelb	Steuerleitung
2	Rot	VCC (+5V)
2	Schwarz	GND

Achtung! Schliessen Sie die \overline{WE} -Pins (27) der EEPROMs an die Speisespannung (VCC 5V) an, damit das Signal nicht undefiniert ist! Ansonsten könnte der Inhalt des EEPROMs unbrauchbar gemacht werden.

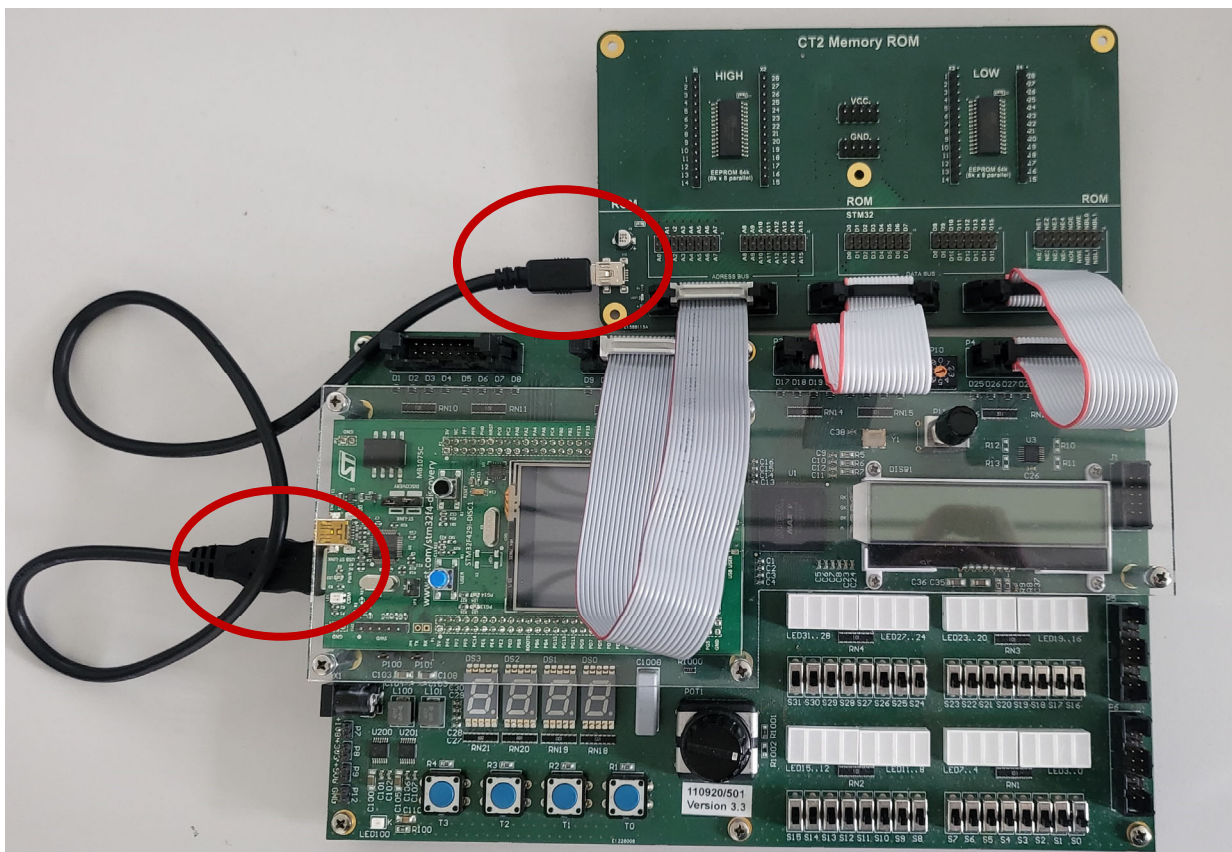
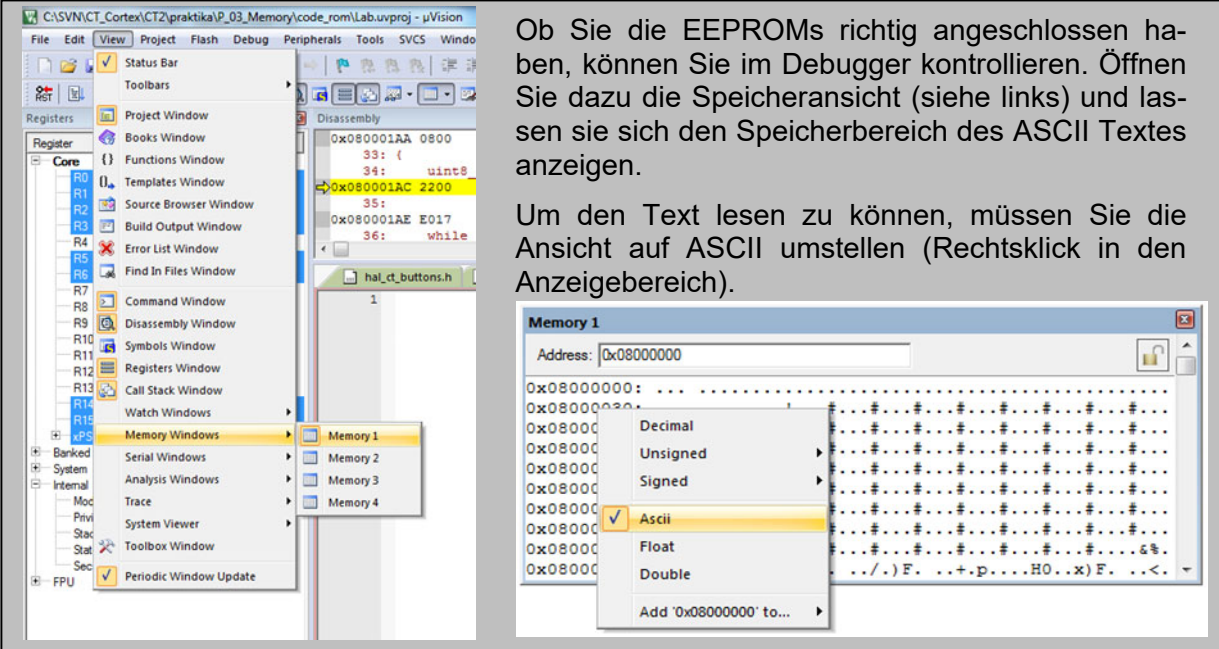


Abbildung 5: Anschluss der Speisung über USB

- d) Lesen Sie den Text von Wilhelm Busch an der entsprechenden Speicherstelle über die Speicheransicht von Keil uVision aus. Wie lautet der Text?

Nur probieren und schon gehts



Ob Sie die EEPROMs richtig angeschlossen haben, können Sie im Debugger kontrollieren. Öffnen Sie dazu die Speicheransicht (siehe links) und lassen sie sich den Speicherbereich des ASCII Textes anzeigen.

Um den Text lesen zu können, müssen Sie die Ansicht auf ASCII umstellen (Rechtsklick in den Anzeigebereich).

- e) Da nicht alle externen Adresslinien A[25:0] decodiert werden, ist das Memory unter mehreren Adressen ansprechbar (Partial Address Decoding). Wie viele Adressbereiche sind es?

Der FMC liest nur die hinteren 13 bits. Die Anderen kann er nicht lesen und beachtet sie dementsprechend nicht.

- f) Nennen Sie drei dieser Adressbereiche? Verifizieren Sie Ihre Lösung über die Speicheransicht im Debugger.

6410 0000
6401 0000
6403 0000

die geben immer die gleiche Ausgabe also kann man den Satz immer lesen.

6400 1000
gibt einen anderen Text

- g) In der Teilaufgabe d) haben Sie den Debugger verwendet, um Speicherbereiche direkt auslesen zu können. Nennen Sie zwei weitere Situationen, in denen das direkte Auslesen von Speicherbereichen nützlich sein kann. 0x6400'0000 – 0x6400'3FFF

Debugging von Fehler
Capture the flag

- h) Nennen Sie zwei Anwendungsbeispiele, wobei die Verwendung von externem EEPROM nützlich sein könnte. 0x6402'0000 – 0x6402'3FFF

permanenter Speicher non-volatile

externe Speicher wie bei der Übung.

5.2 Memory Test

Erstellen und testen Sie ein C-Programm, das fortlaufend den Inhalt des EEPROM-Bereichs von $0x400$ bis $0x4FF$ ausliest. Falls der Inhalt nicht mit dem erwarteten Wert übereinstimmt, soll das Programm anhalten und den Adressindex ($0x00$ bis $0xFF$) sowie den fehlerhaften Wert an den LEDs anzeigen. Mit einem Druck auf Taste T0 soll das Programm fortfahren. Siehe Abbildung .

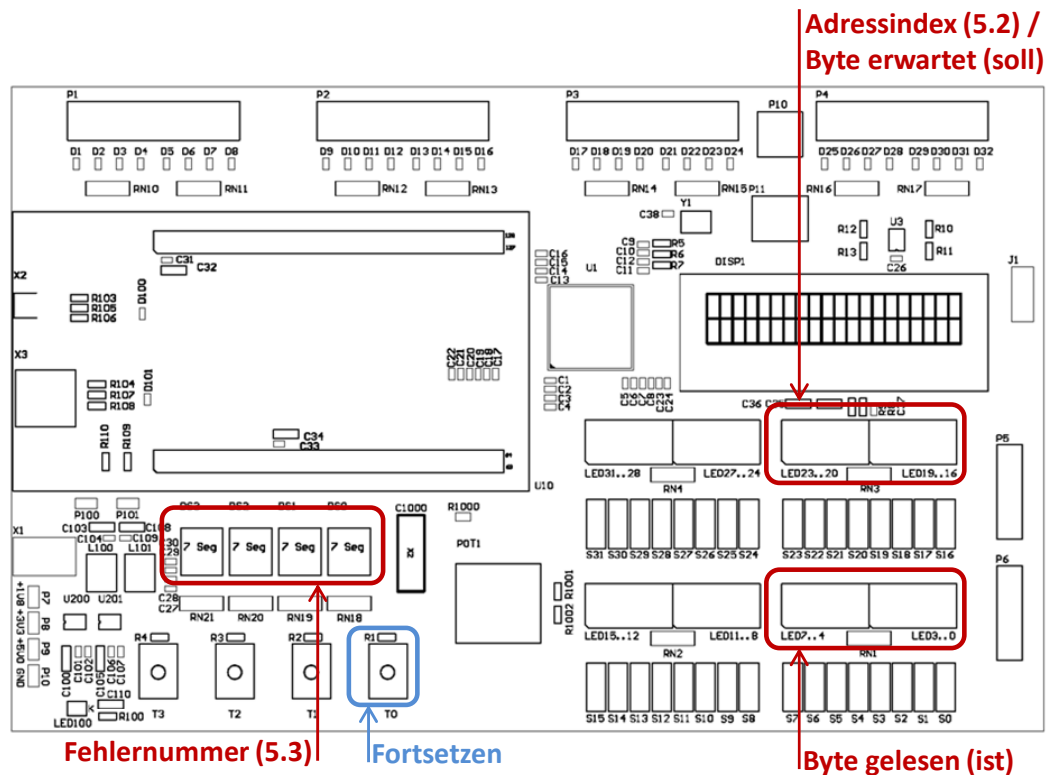


Abbildung 6: Funktion Memory Test

Verwenden Sie das Modul `hal_ct_buttons` (`#include "hal_ct_buttons.h"`) um auf einen Tastendruck zu warten:

```
while (!hal_ct_button_is_pressed(HAL_CT_BUTTON_T0));
```

Um Werte auf die LEDs auszugeben, verwenden Sie die vordefinierten Makros (`#include <reg_ctboard.h>`) z.B.

```
CT_LED->BYTE.LED23_16
```

5.3 Optional: Codewandlung mit EEPROM

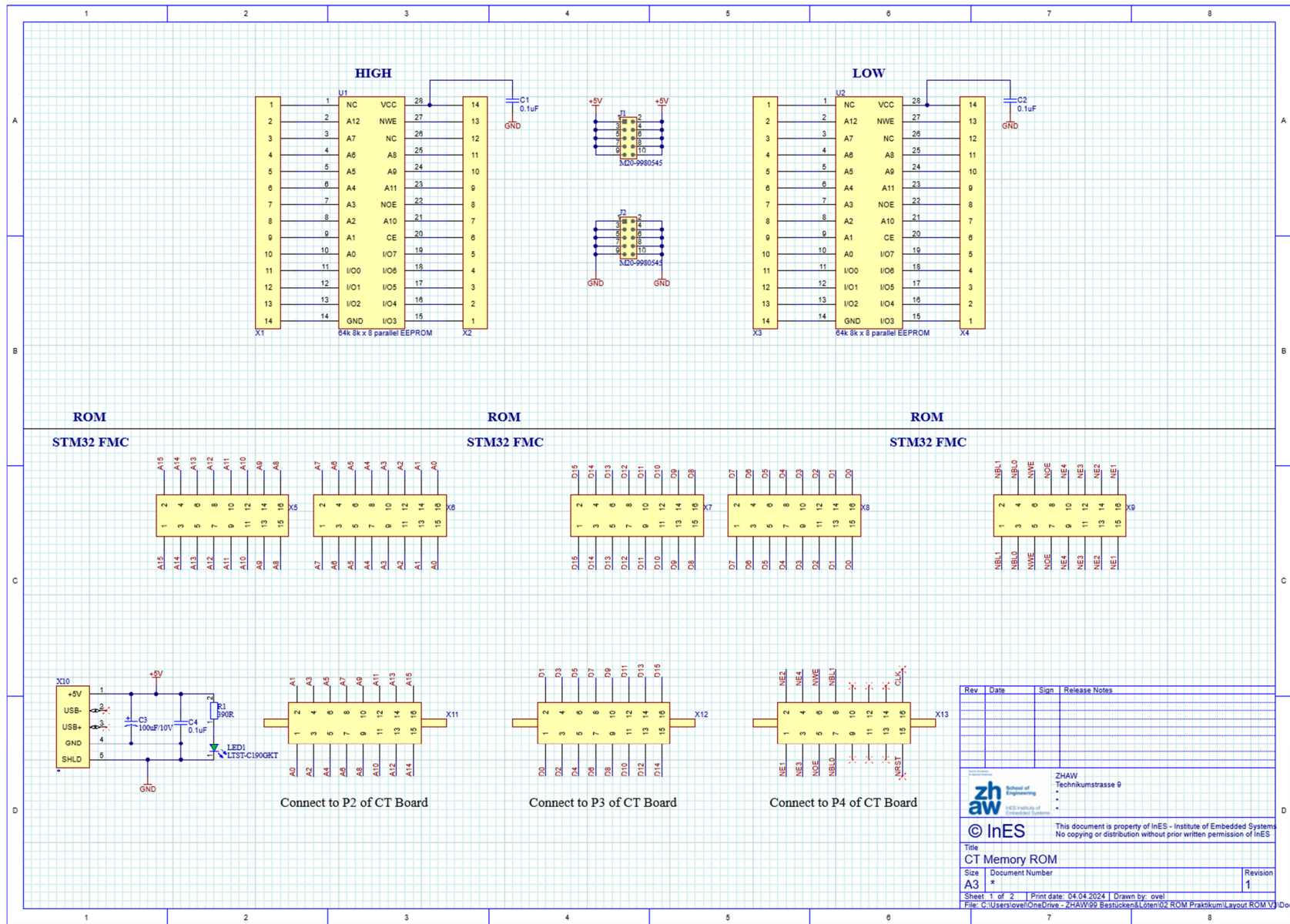
Erstellen und testen Sie eine Codewandlungsroutine, die unter Verwendung der Tabelle an der Adresse $0x200$ binäre Daten in 7-Segment Code umwandelt. Zeigen Sie die Fehlernummer (1 bis 5) auf der 7-Segmentanzeige an.

5.4 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Bewertungskriterien	Gewichtung
Der EEPROM Baustein wurde angeschlossen und die Fragen dazu beantwortet.	2/4
Memory Test wurde implementiert.	2/4

6 Anhang: Schaltschema des CT ROM Boards



Anhang: Adressierschema

STM32F429 Flexible Memory Controller (FMC) Decoding

