CT Praktikum: Memory

1 Einleitung

In diesem Praktikum schliessen Sie zwei vorprogrammierte EEPROMs sowie ein asynchrones SRAM 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.

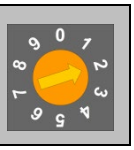
3 Material

- 1x CT Board
- 1x Indigel Board
- 1x USB Adapter für Indigel Board Speisung
- 1x kurzes USB Kabel (USB A <-> USB mini)
- 4x Flachbandkabel für externen Busanschluss
- 2x EEPROM AT28C64B (LOW/HIGH)
- 1x Asynchronous SRAM IDT6116

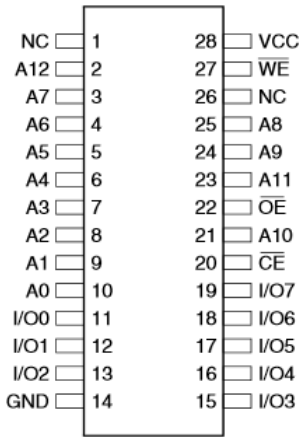
4 Grundlagen

In *Modus 2* schaltet das CT Board den externen Speicherbus auf die Schnittstellen P1 bis P4. Die genaue Belegung in diesem Modus ist im InES CT-Board Wiki 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. 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. Im Gegensatz zu einem EPROM, welches mit UV Licht gelöscht werden kann, kann ein solches EEPROM elektrisch gelöscht werden.

Abbildung 1: Pinbelegung 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.

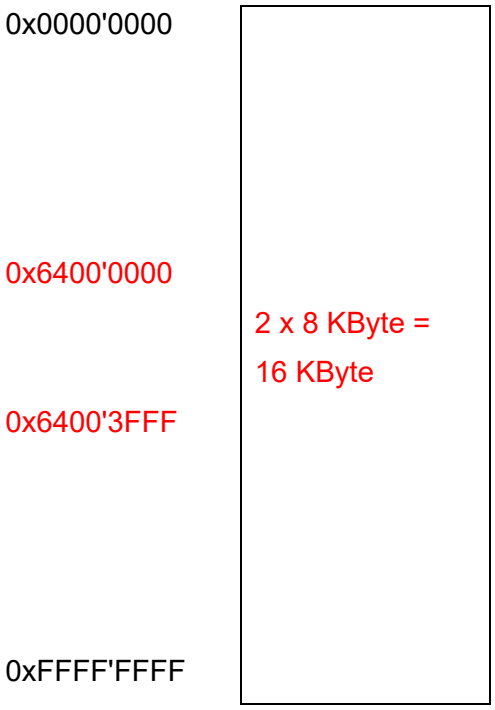


Abbildung 2: Memory Map – Speicherbereich der zwei EEPROMs

b) Zeichnen Sie in Abbildung 3 ein, wie die Bausteine an das CT Board angeschlossen werden müssen.

Abbildung 3: 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) Bauen Sie die Schaltung auf dem Indigel Board auf. Zur Speisung des Indigel Boards ist ein USB Adapter vorhanden. Mit diesem kann das Indigel Board über den USB Port des CT-Board gespeist werden. Siehe Abbildung 4.

Schliessen Sie die $\overline{\text{WE}}$-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. Verwenden Sie dazu einfach einen roten Kurzschlussstift des Indigel Boards.

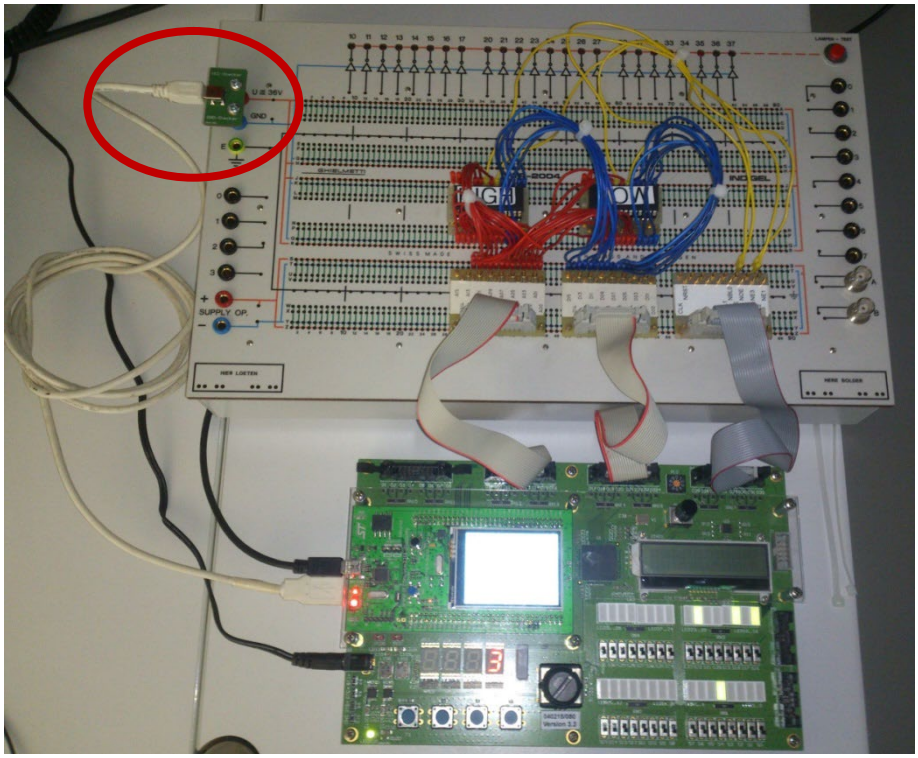
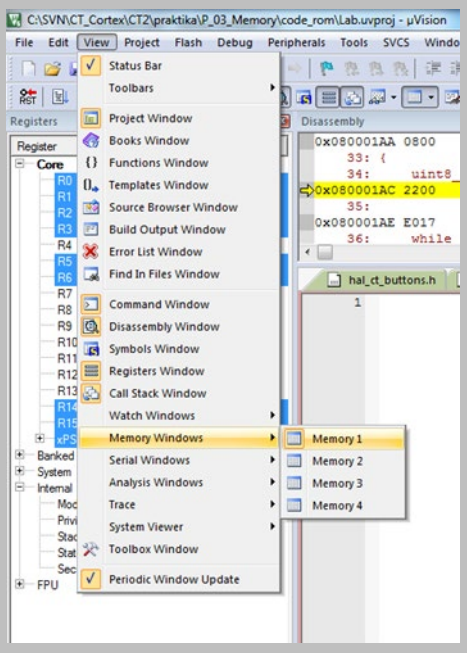


Abbildung 4: Anschluss der Speisung über USB

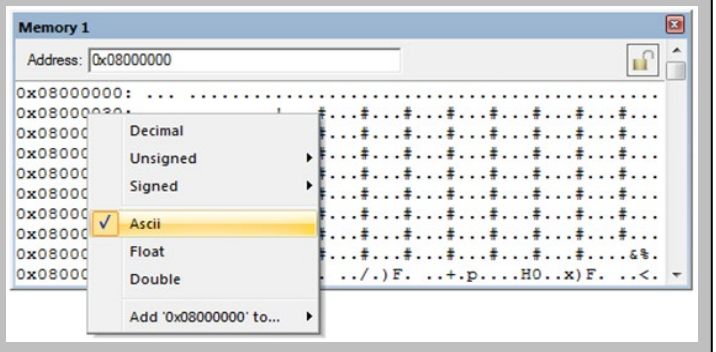
d) Lesen Sie den Text von Wilhelm Busch an der entsprechenden Speicherstelle über die Speicheransicht von Keil uVision aus. Wie lautet der Text?

„Nur probieren und schon geht's - Das heisst nicht immer, doch fast stets!“



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?

2^25 / 2^13 = 2^12 = 4‘096

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

0x6400'0000 – 0x6400'3FFF

0x6402'0000 – 0x6402'3FFF

0x670C'4000 – 0x670C'7FFF

intern_A[31:26] = 0x64 oder 0x65 oder 0x66 oder 0x67
intern_A[13:0] → Memory
intern_A[25:14] → beliebig wählbar

5.2 Memory Test

Erstellen und testen Sie ein C-Programm (verwenden Sie das Rahmenprogramm *code_rom*), 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 5.

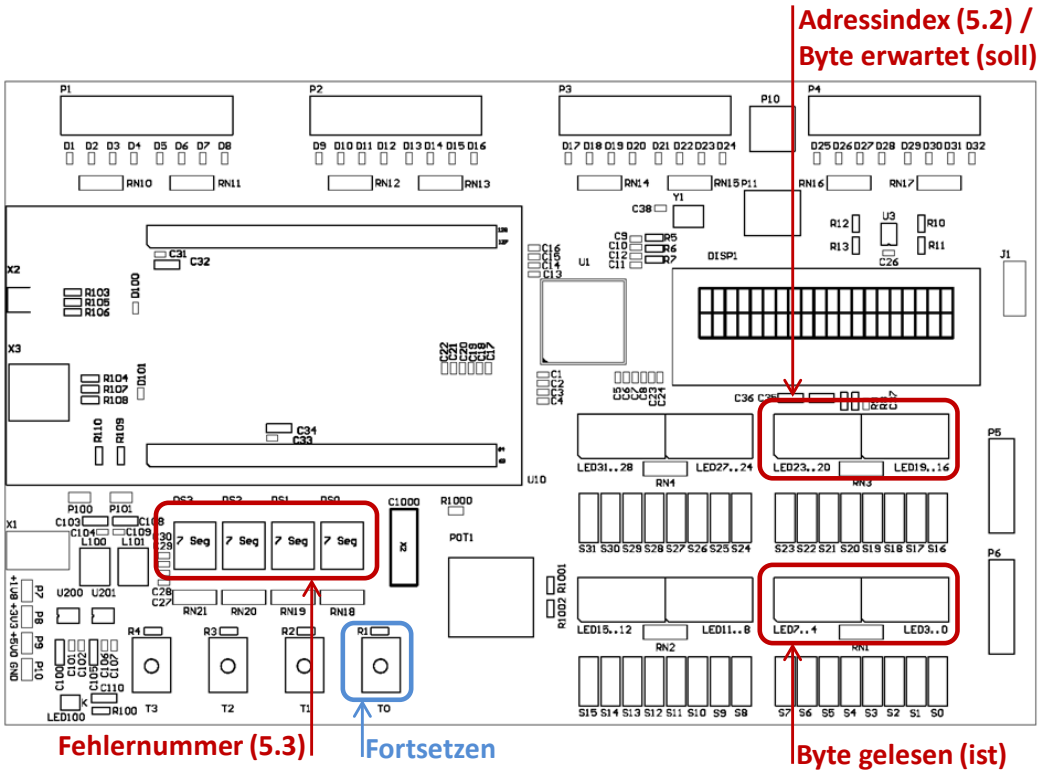


Abbildung 5: Funktion Memory Test

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
```

Fehler: 0x10 statt 0x01, 0x00 statt 0x44, 0x10 statt 0x45, 0x00 statt 0xB2, 0x10 statt 0xB3

5.3 Optional: Codewandlung mit EEPROM

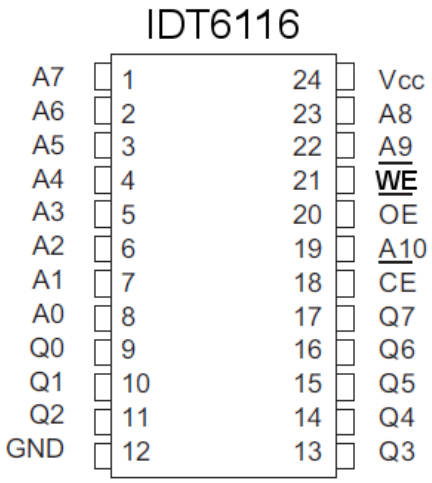
Erstellen und testen Sie eine Codewandlungsroutine, die unter Verwendung der Tabelle an der Adresse **`0x200`** binäre Daten in 7-Segment Code umwandelt. Zeigen Sie die Fehlernummer (1 bis 5) auf der 7-Segmentanzeige an.

5.4 Anschluss Asynchronous SRAM

Ersetzen Sie die EEPROMs durch das gegebene SRAM mit *8-bit* breitem Datenbus, so dass es von der tiefsten Adresse im Bereich SRAM – Bank 1 Device 2 (angesprochen mit **`NE2`**) an aufwärts adressiert werden kann. Der FMC ist im gegebenen Programmrahmen (verwenden Sie diesmal *code_ram*) bereits so konfiguriert, dass das Memory an **`NE2`** als 8-bit breiter Speicher angesprochen wird.

Schreiben Sie ein Programm, welches alle Bytes des SRAMs mit einem fallenden Zählerwert von **`0xFF`** bis **`0x00`** beschreibt und danach auf einen Tastendruck wartet.

Lesen Sie nach dem Tastendruck die Werte aus und prüfen Sie diese. Das Programm soll bei einem gefundenen Fehler anhalten und diesen mit Adresse und Daten anzeigen. Prüfen Sie Ihr Programm, indem Sie nach dem ersten Schritt und vor dem Drücken der Taste, Adressen- und/oder Datenleitungen umstecken. Sie können auch in der Speicheransicht im Debugger den Speicherinhalt durch doppelklicken verändern. Der Debugger muss dazu angehalten sein.



Gegeben ist ein SRAM vom Typ IDT6116. Dieses hat eine Organisation von 2K * 8-Bit, also 16 KBit.

Schauen Sie sich die Belegung des RAMs und der EEPROMs genau an. Der EEPROM LOW Baustein lässt sich mit wenig Aufwand gegen den RAM Baustein austauschen!

Achtung: data ist als uint8_t deklariert. Dadurch entsteht ein wrap-around in der Schreibschleife. i=254 -> data=0x01, i=255 -> data=0x00, i=256 -> data=0xFF. Dadurch schreiben wir bei einem 2k x 8bit RAM total 8-mal den gleichen Datenblock. D.h. wie die Adressbits 10:8 gesetzt sind, spielt keine Rolle. Es wird immer der gleiche Wert ausgelesen, wenn auch von unterschiedlichen Speicherstellen.

Abbildung 6: Anschluss Asynchronous SRAM

5.5 Bewertung

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

Bewertungskriterien	**Gewichtung**
Lösung der Aufgabe 5.1: Hardwareaufbau und Beantwortung der Fragen.	1/3
Das Programm erfüllt die in Aufgabe 5.2 geforderte Funktionalität.	1/3
Das Programm erfüllt die in Aufgabe 5.4 geforderte Funktionalität.	1/3

6 Anhang: Indigel-Board

Das Indigel-Board in Abbildung 7 erlaubt einen einfachen, lötfreien Aufbau von Schaltungen.

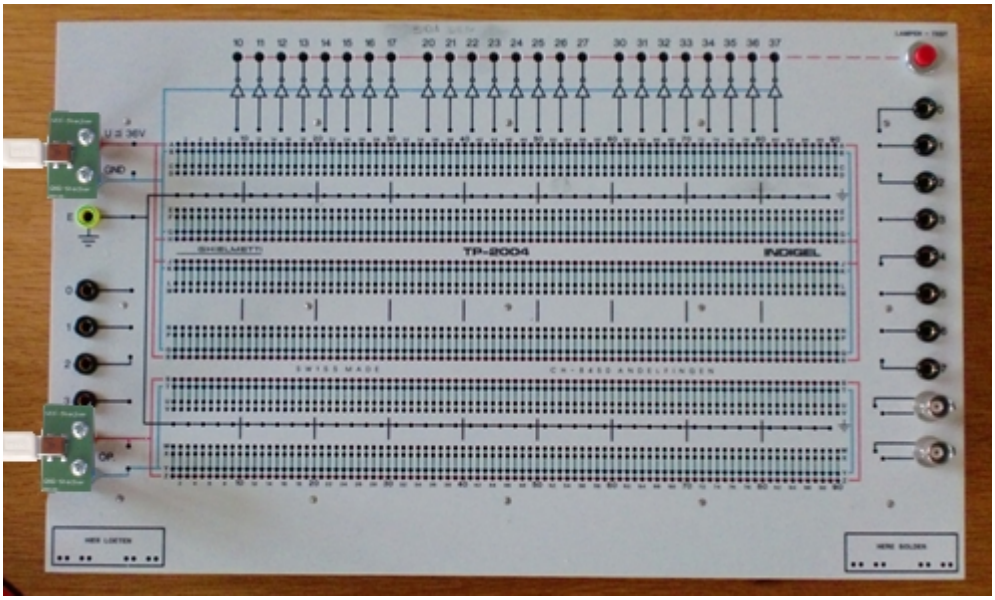


Abbildung 7: Indigel-Board

Für ein einfaches Verbinden der Bauteile untereinander und mit der Speisespannung sorgt ein Kreuzschienenverteiler. Dieser besteht aus den vertikal angeordneten Signalschienen und den horizontal angeordneten tiefer liegenden Speisungsschienen. Diese können wie in Abbildung 8 dargestellt, mit Hilfe von Kurzschlussstiften miteinander verbunden werden.

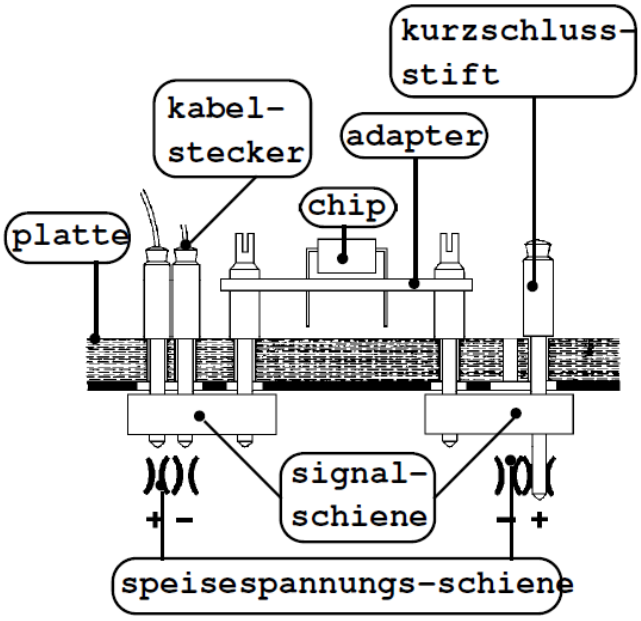
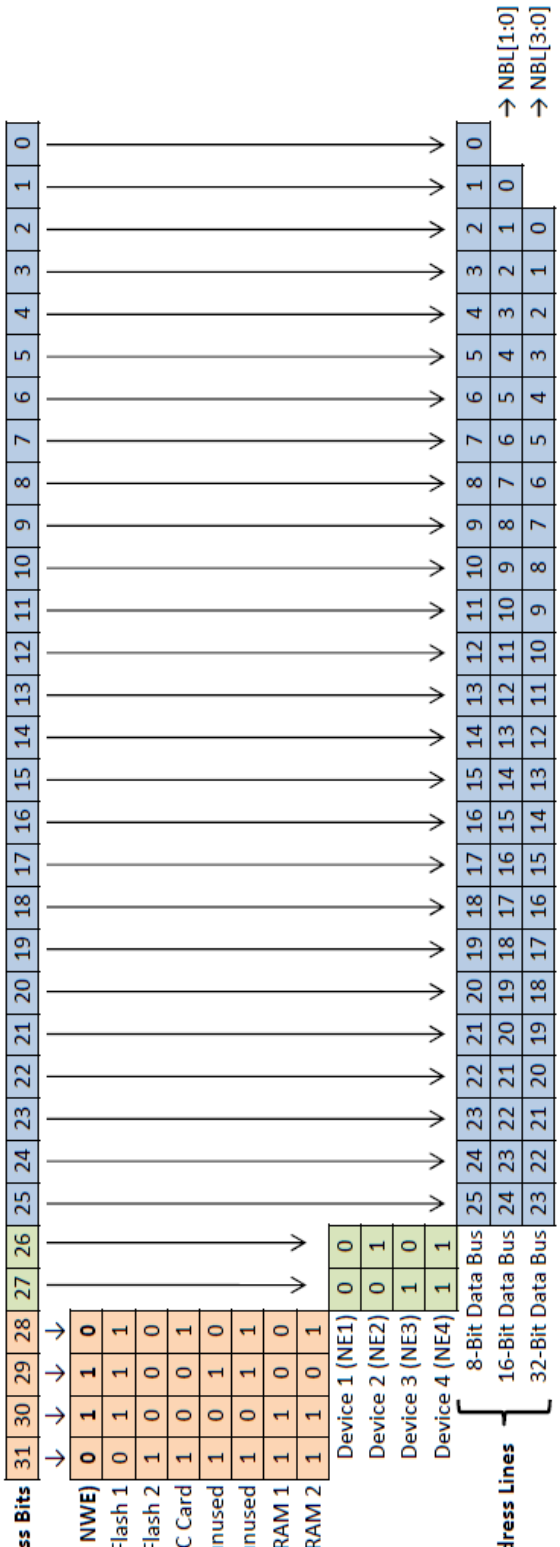


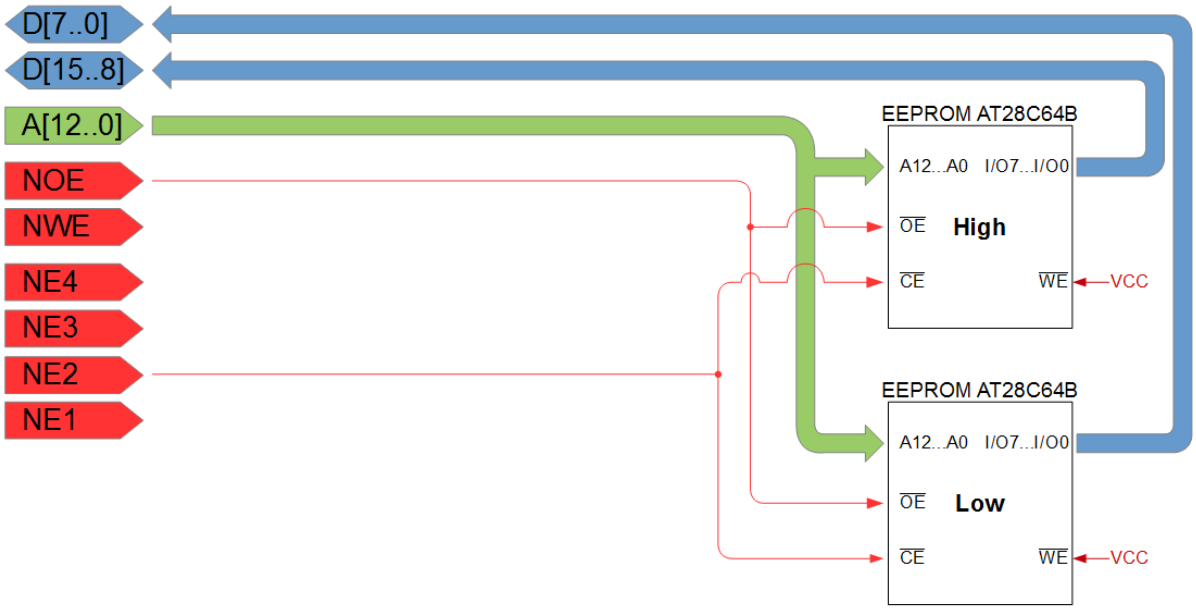
Abbildung 8: Verbinden von Signal- und Speisungsschienen auf dem Indigel-Board

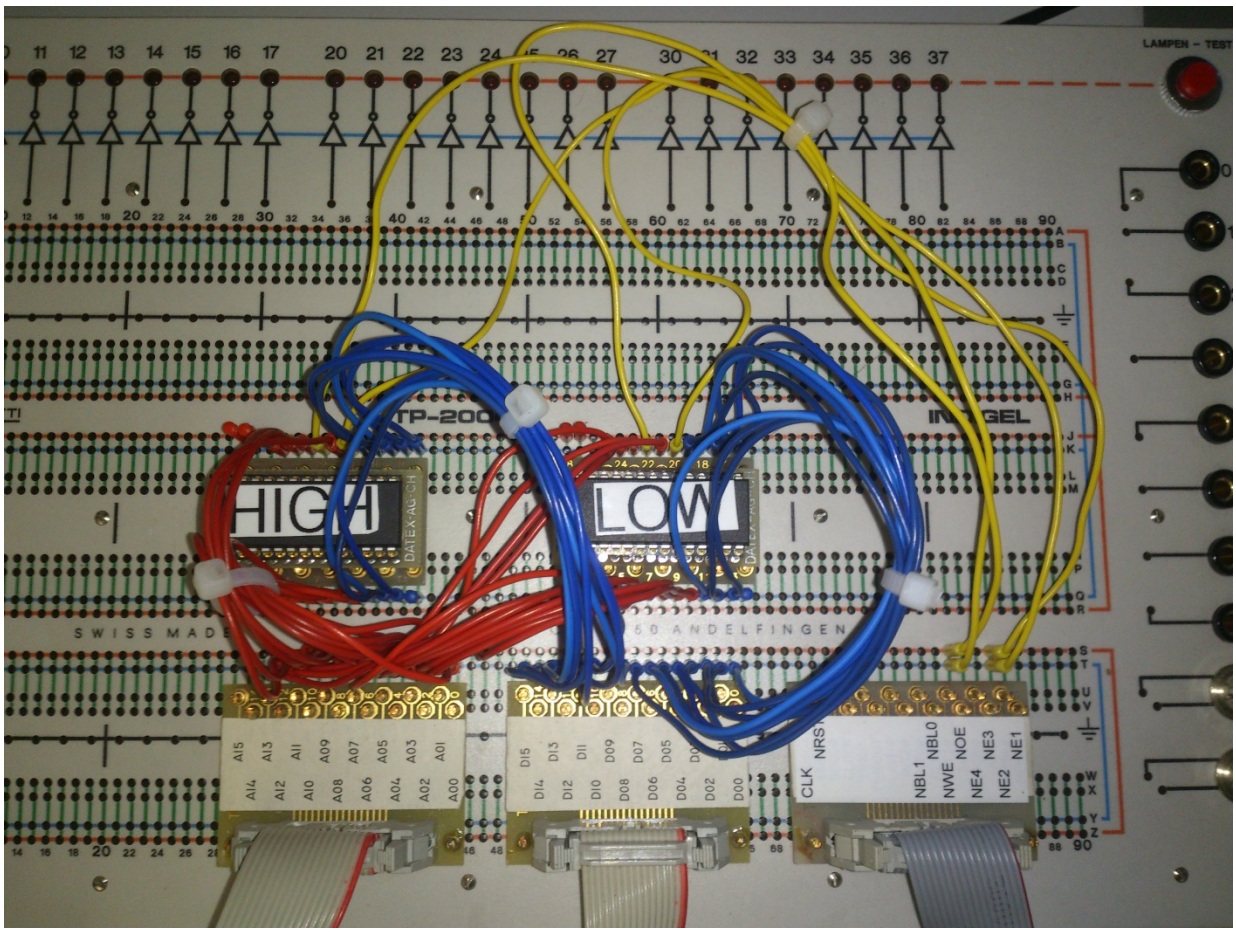
7 Anhang: Adressierschema



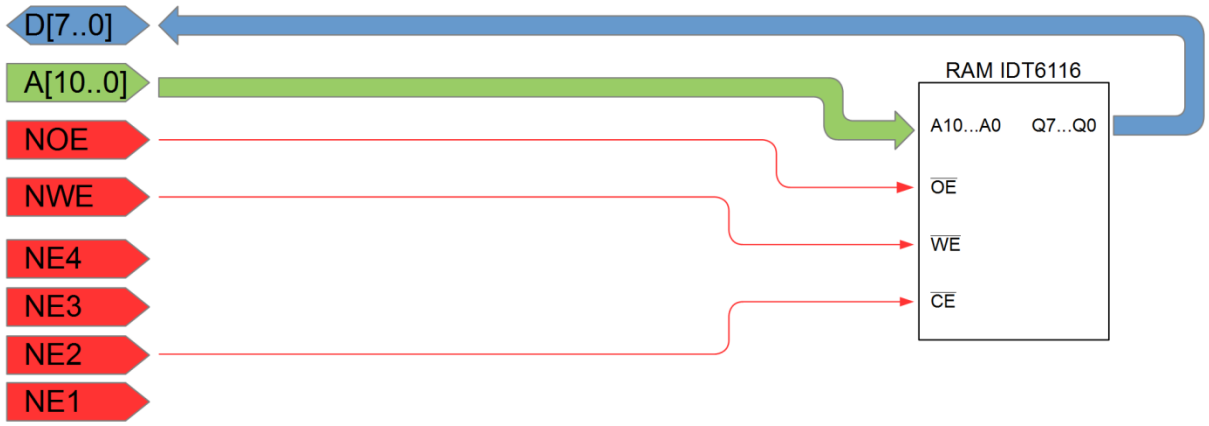
8 Anhang: Lösungen

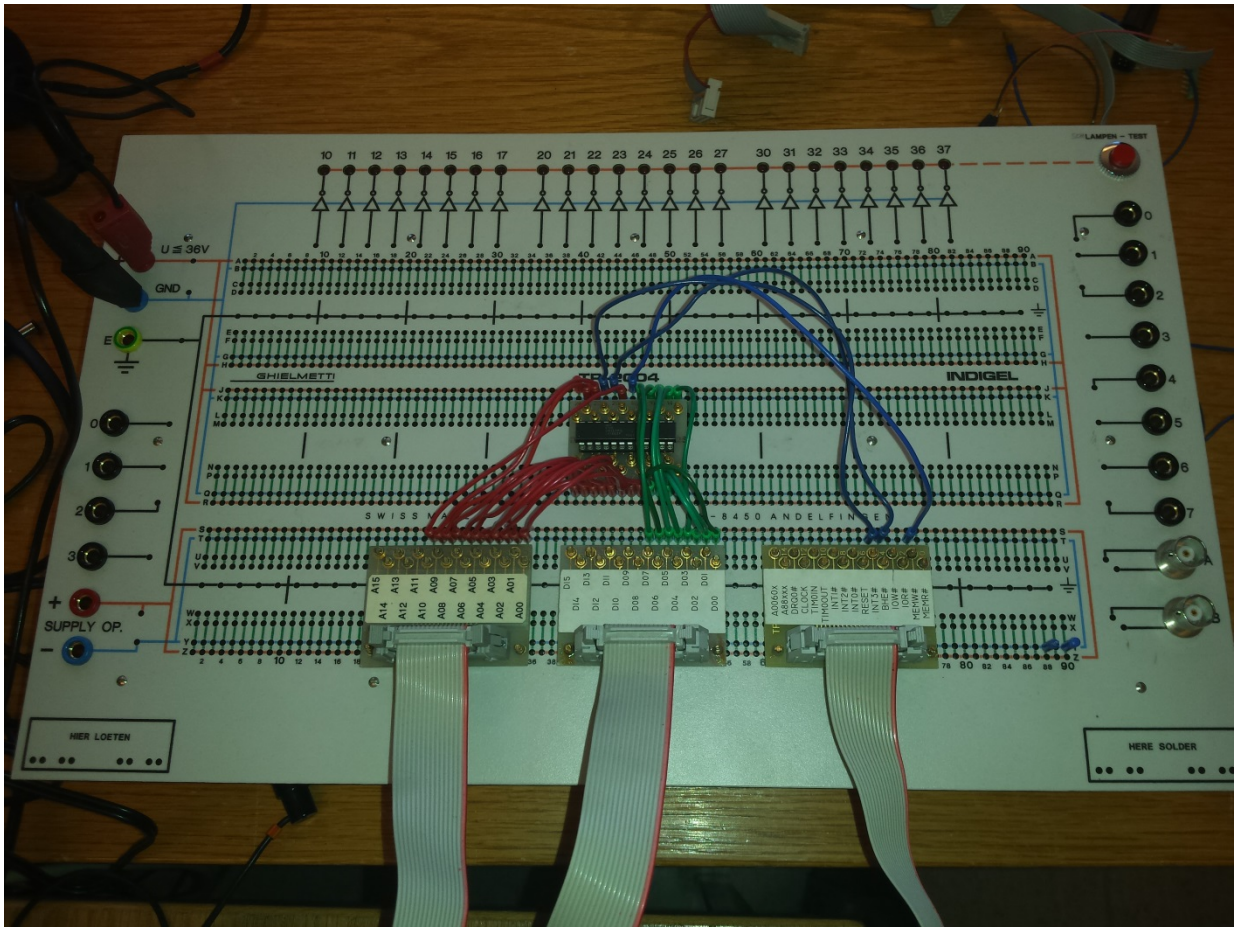
Aufgabe 5.1





Aufgabe 5.4


D[7..0]
A[10..0]
NOE
NWE
NE4
NE3
NE2
NE1
RAM IDT6116
A10...A0
Q7...Q0
OE
WE
CE


U ≤ 36V
GND
SUPPLY OP.
HIER LOETEN
HERE SOLDER
INDIGEL