

LA1 - Labor

Abteilung Elektronik u. technische Informatik

#Title#

**Nico Zehetner und Tizian Wackerle**

3cHel; Jahrgang 2023/24

**Betreuer\*In:** #Name#

Ausgeführt am: 29.02.2024 Abgegeben am: 29.02.2024

Inhaltsverzeichnis:

[18 Dez.: Aufgabenstellung & Erster Test mit LEDs 2](#_Toc187308906)

[Anforderungen: 2](#_Toc187308907)

[Function Test: 2](#_Toc187308908)

[25 Dez: Timing Check mit FreeSoC2 3](#_Toc187308909)

# 18 Dez.: Aufgabenstellung & Erster Test mit LEDs

Der Z80 ist ein bis heute viel verwendeter 8bit-Prozessor der Firma Zilog. Um diesen einflussreichen und (für heutige Standards) simplen Prozessor besser verstehen zu können ist es unser Plan, ein Z80-Shield für einen LearnSoc! oder andere Boards mit Arduino kompatiblen Headers

## Anforderungen:

* Passt auf den Arduino-Shield-Formfaktor
* Clock Signal wird vom Host geeneriert. Dadurch „Single-Step“ auch möglich
* Instructions können vom Mikrocontroller gesendet werden (µC simuliert ROM)
* RAM ist verbaut, kann aber von µC ausgelesen werden

## Function Test:

Als erster Test wollten wir eine minimale Beschaltung eines Z80, welcher dauerhaft NOPs ausführt, testen. Dabei orientierten wir uns an folgender Schematic dieses Blogs:

<https://maker.pro/pic/projects/z80-computer-project-part-1-the-cpu>

Ein Bild, das Text, Diagramm, Plan, technische Zeichnung enthält.

Automatisch generierte Beschreibung

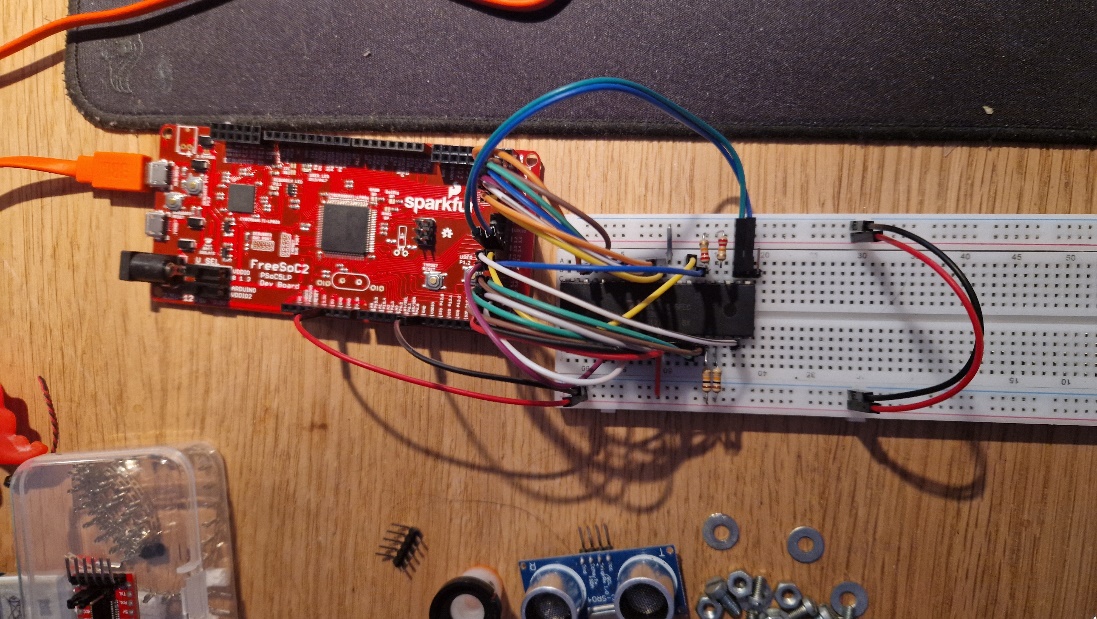
Für den CLK-Eingang verwendeten wir einen Funktionsgenerator, welcher im einstelligen Hz Bereich läuft. Der DIP-Switch ist hierbei auf 0x00 gestellt, da dies einer der Hex-Code für NOP ist. Zu erwarten ist damit, dass der Z80 dann nur den M1 Cycle durchläuft, da keine weitere Arbeit außer der OPCode-Fetch vollbracht wird. Dies ist im Datenblatt des Z80 wie folgt dargestellt:

Ein Bild, das Text, Diagramm, Plan, Reihe enthält.

Automatisch generierte Beschreibung

Funktioniert hat dies einwandfrei, wie in [diesem Video](https://htlanichstrasse-my.sharepoint.com/personal/nzehetner_office_htlinn_ac_at/Documents/Dokumente/Schule/Werkstaette/SJ2024_25/PsoC_Z80Board/20241218_155240.mp4) gezeigt wird. Die weißen LEDs sind A0-A3, die roten LEDs sind an RD, WR, M1 und MREQ angeschlossen.

# 25 Dez: Timing Check mit FreeSoC2



Eine visuelle Kontrolle des Adress- und Datenbusses ist im weiteren verlauf recht aufwendig. Deshalb wurde mit Hilfe eines FreeSoC2 simples Skript realisiert. Dieser gibt einerseits die Daten am Adressbus, Datenbus und einiger Steuersignale über UART aus und steuert die Clock des Z80. Dabei kann durch senden einiger ASCII-Zeichen entweder zwischen einer 10Hz Clock oder einem Single-step Modus gewechselt werden. Im nachfolgenden Bild ist die Ausgabe diese Skripts mit HTerm aufgezeichnet:

Ein Bild, das Text, Screenshot, Zahl, parallel enthält.

Automatisch generierte Beschreibung

Hier kann man nochmals, da der Z80 konstant NOPs ausführt, den M1-Zyklus beobachten.

# 8. Jän: Erster Versuch Programmspeicher & PCB

Der generelle Aufbau des Boards sieht vor, dass der PSoc den Code, welchen er über UART empfängt, in den RAM des Z80 speichert. Von diesem Speicherchip aus führt der Z80 aus den Programmcode dann aus. Damit müssen wir uns nicht darum sorgen, dass der Code effizient genug ist um mit der Betriebsgeschwindigkeit des Z80 mithalten kann.

Die Beschaltung des RAM Chips (hier ein 2kByte HM6112) ist komplett mit Jumper auf den FreeSoc2 gelegt, da wir so via Software alle Kontrollsignale und Busse steuern können und sieht wie folgt aus:

[Bild]

[PCB]

# 15.Jän: Programmcode in den RAM bringen

Als ersten Test gilt hier das Projekt „PSoC\_to\_RAM“ in welchem vorgezeigt wird, wie der PSoC die Daten, die er über UART erhält, in den RAM-IC schreibt. Die Timings sind mit einem einzigen 1us-Delay realisiert (sowohl in Read als auch in Write). Dies ist zwar nicht genau in Spec mit den Timing Diagrammen, aber funktioniert trotzem Einwandfrei, da immer nur von Minimum Zeiten die Rede ist.

Ein Bild, das Diagramm, Entwurf, technische Zeichnung, Reihe enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Zahl, Quittung, Schrift enthält.

Automatisch generierte Beschreibung