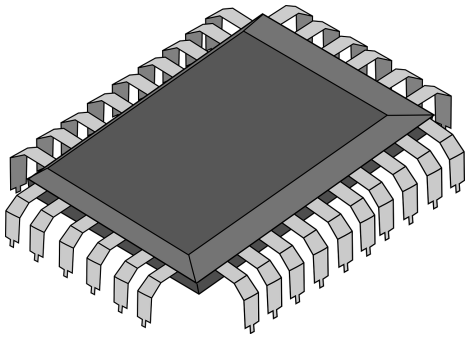
# NISSE32 2021 v0.1

## 1 Overblikk



- ARM 32-bit Cortex M4 CPU
- 512 Kb flash minne
- Flash- og debug låsemekanisme
- Trådløs factory reset

## 2 Beskrivelse

For å imøtekomme stadig økende krav til robusthet, intelligens og smidighet i moderne leketøy trengte vi en mikrokontroller som kan holde tritt med de nye tidene i mange år fremover. Den nyeste brikken fra alvdelingen for silisium ved nissens verksted *er* denne revolusjonen; **N**ok en **I**ntelligent **S**tyreenhet for **S**mart **E**lektronikk **32** oppfyller alle krav en måtte ha for moderne leketøysdesign.

- Lavt strømforbruk for effektiv utnyttelse av batteri
- 16-bit innebygget timer
- Flere GPIOer enn du kan telle! (Så lenge du bruker fingrene til å telle...)
- Støtte for audio via I2S
- CRC-enhet
- JTAG debug interfjes for fastvareutviklere
- USART kommunikasjon for applikasjonsspesifikke bruksområder
- Gjennomgående testet for trygghet (svært få barn ble skadet under testing av denne brikken)

## 3 Tekniske spesifikasjoner

| | Enhet | Verdi |
|---|---|---|
| Spenningstilførsel | $V$ | 1,7 til 3,8 |
| Temperatur | °C | -40 til +120 |
| Dimensjoner | $mm * mm * mm$ | 14*7*4 |
| Weight | $g$ | 13 |

# 4 Sikkerhet og målinger

## 4.1 Funksjonelle målinger

Får å beregne effektiviteten under kodeeksekvering har vi utført komplekse og robuste analyser av klokkefrekvens mot instruksjonshastighet. Våre analyser viser at *samtlige* ARM v8 instruksjoner kjøres på *nøyaktig* 2 klokkesykluser på NISSE32. Disse *nærmest utrolige* tallene dokumenteres herved av hensyn til fremtidige undersøkelser rundt kodeeffektivitet.

## 4.2 Sikring mot fault injection

| Type feil | Resultat | Kommentar |
|---|---|---|
| Elektromagnetisk fault injection | **OK** | Ingen observerte glitcher |
| Voltage fault injection | **OK** | Tryner bare under test, trolig greit |
| Clock fault injection | **OK** | Hvilket år er det egentlig? LMAO! |
| Laser fault injection | **OK** | Har ikke råd til testutstyr |

# 5 Programmeringsguide

Koden din vil bli kjørt direkte fra flash fra **baseaddresse 0x08000000** , så sørg for å flashe binærfilen din til denne addressen. Herfra vil bootROM gjøre en avsjekk mot produksjonslåsen på addresse 0x1FFF7800 for å avgjøre om JTAG interfjeset skal åpnes. Det er dermed viktig å ikke sette denne låsen under utvikling for å ikke bli låst ute av debug.

Hvis dette skulle skje, men du har implementert de riktige kommandoene fra nisSDKen sitt command_handler eksempel, kan du fremdeles tilbakestille brikken ved å bruke factory_reset over USART, eller trådløst via julemagi.

## 5.1 Eksempel på bruk av nødkommandoer

Nødkommandoene i eksempelimplementasjonen fra nisSDKen (som du gjerne må bruke i produksjonsklar fastvare!) lar oss både tilbakestille brikken til ikke-produksjonsklar modus , og dumpe innholdet av den kritiske konfigurasjonsdataen fra flash.

Se under for eksempel på hvordan man kan dumpe den kritiske konfigurasjonsdataen fra et kjørende system over USART:

```
>>> dump_flash Password123!
Entering command handler
Commencing flash dump:
OVERRIDE_PASSWORD=Password123!
```

I akkurat dette eksempelet er det kun selve passordet som ligger på flash, men poenget er illustrert.

For å tilbakestille en brikke til opprinnelig tilstand kan vi bruke factory_reset, men vær obs på at dette vil slette hele flashen, inkludert den kjørende fastvaren! Ettersom denne funksjonen er svært kritisk har den blitt ekstra herdet mot fault injection angrep:

```
>>> factory_reset Password123!
Entering command handler
Factory reset complete. Resetting MCU.
```

Dette interfjeset kan også nås trådløst via julemagi, dersom tilkobling med USART skulle være upraktisk, eller ved store batch-jobber.

# 6 CPU informasjon

| Kategori | Beskrivelse |
|---|---|
| Arkitektur | 32-bit ARM v8 |
| Endianness | Little Endian |
| Klokkefrekvens | 100 MHz |

# 7 Addresserbare områder

## 7.1 Flash

| Sektor | Addresser | Bruksområde |
|---|---|---|
| 0 | 0x0800 0000 - 0x0800 3FFF | Fastvare |
| 1 | 0x0800 4000 - 0x0800 7FFF | Fastvare |
| 2 | 0x0800 8000 - 0x0800 BFFF | Ukategorisert |
| 3 | 0x0800 C000 - 0x0800 FFFF | Ukategorisert |
| 4 | 0x0801 0000 - 0x0801 FFFF | Kritisk konfigurasjonsdata |
| 5 | 0x0802 0000 - 0x0803 FFFF | Applikasjonsdata |
| 6 | 0x0804 0000 - 0x0805 FFFF | Reservert for påskeharen |
| 7 | 0x0806 0000 - 0x0807 FFFF | Reservert for fremtidig bruk |

## 7.2 Øvrig

### 7.2.1 Minne

| Addresser | Bruksområde |
|---|---|
| 0x1FFF 0000 - 0x1FFF 77FF | Systemminne |
| 0x1FFF 7800 - 0x1FFF 7A0F | Produksjonslås mm. |
| 0x1FFF C000 - 0x1FFF C00F | Øvrig opsjonsområde |

### 7.2.2 Perfiere enheter

| Addresser | Bruksområde |
|---|---|
| 0x4000 3800 - 0x4000 3BFF | I2S |
| 0x4000 4400 - 0x4000 47FF | USART |
| 0x4001 0000 - 0x4001 03FF | Timer |
| 0x4002 0000 - 0x4002 03FF | GPIOA |
| 0x4002 0400 - 0x4002 07FF | GPIOB |
| 0x4002 0800 - 0x4002 0BFF | GPIOC |
| 0x4002 0C00 - 0x4002 0FFF | GPIOD |
| 0x4002 1000 - 0x4002 13FF | GPIOE |
| 0x4002 1C00 - 0x4002 1FFF | GPIOH |