# MICB2

Microcontrollerbesturingen 2

THRijswijk





# MICB2

Overzicht

- MICB2C1 Microcontroller besturingen college
  - 7 uur (2 uur/week in eerste 3½ weken)
  - Bd = Harry Broeders
- MICB2P1 Microcontroller besturingen begeleid practicum
  - 12 uur (2 uur/week)
  - Hv = Wim van den Heuvel
  - Bd = Harry Broeders
- MICB2Z1 Microcontroller besturingen onbegeleid practicum
  - 12 uur (2 uur/week)
- Mondelinge toets
  - 30 minuten (in week 7)

THRijswijk





# Werkvormen

C1 + P1 + Z1 = **84** SBU

- 7 uur MICB2C1 college
- 12 uur MICB2P1 practicum
- 12 uur MICB2Z1 practicum
- ½ uur toets
- 52 ½ uur zelfstudie (~1 dag/week)

# Inhoud

- Toepassingen van microcontrollers
- Embedded software voor microcontrollers in C en C++
- Interfacing: digitale in/uitgangen, seriële poort, real-time interrupt, timer output compare, timer input capture, ADC, pulse accumulator (herhaling)
- Debuggen van embedded software
- Combineren: C+assembler+libraries

THRijswijk



# Leermiddelen

**http://bd.thrijswijk.nl/micb2**

- Boek: Microcomputer Engineering van Miller
- Sheets
- 68HC11 reference manual
- 68HC11 programming reference guide
- Ontwikkelomgeving
  - GNU utilities for Win32
  - GNU toolchain voor de 68HC11
  - THRSim11 simulator
  - 68HC11 EVM (hardware)
- **Practicumopdrachten**

THRijswijk



# 68HC11

Toepassingen

- Huis, tuin en keukenproducten
  - Magneton, broodbakmachine, video, DVD speler, speelgoed, CV ketel enz...
- Computer apparatuur:
  - ZIP drive, printer, modem enz...
- Land- en tuinbouw:
  - Klimaatbeheersing, sorteermachine, weegschaal, koeherkennings-systeem enz...
- Verkeer:
  - Stoplicht, overwegbeveiliging, flitspaal, enz...
- Auto:
  - Motor management systeem, ABS, airbag, radio, route informatie-systeem enz...
- ...

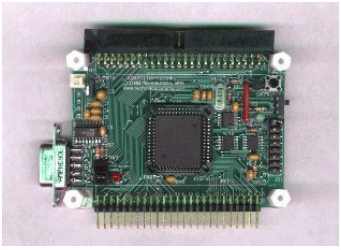
THRijswijk



# 68HC11

Ontwikkelbord

- Voorbeeld van een 68HC11 systeem:



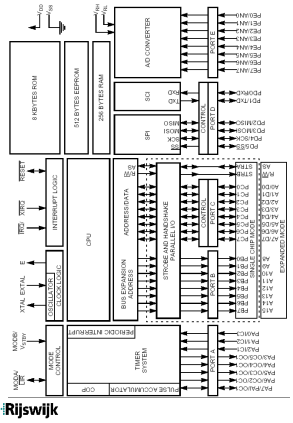
- Wij gebruiken Motorola EVM (veel groter)

THRijswijk



# 68HC11

blokschema

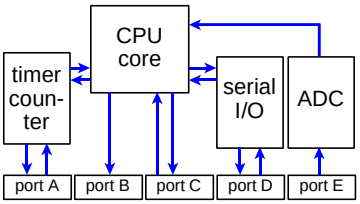


THRijswijk



# 68HC11

**eenvoudig** blokschema

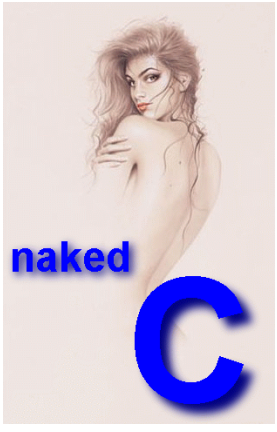



THRijswijk





# MCCA1




THRijswijk

# naked C

Hoezo naakt?

- Geen operating systeem
  - Geen threads, geen memory management, geen I/O drivers enz...
- Geen run-time environment
  - Geen printf, geen scanf, geen malloc en free enz...
- Geen directe ondersteuning voor floating point getallen.

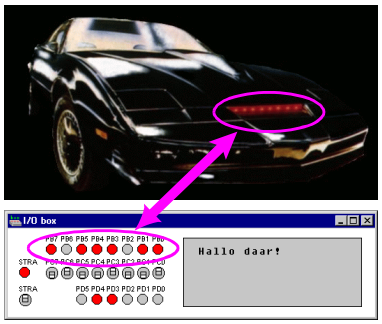
- Beperkte libraries.
- Beperkt datageheugen:
  - 256 **bytes**!
- Beperkt programmageheugen:
  - 16384 bytes.

THRijswijk

10

# naked C

Voorbeeld




THRijswijk



11

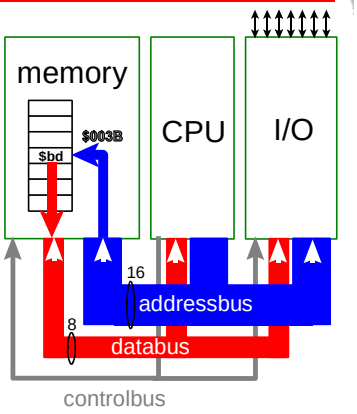
# Voorbeeld

```
typedef unsigned short word;
typedef unsigned char byte;

void wait() {
   volatile word i;
   for (i=0; i<10000; ++i)
      /*empty*/;
}

int main() {
   void wait();
   byte c1, c2;
   volatile byte* p=(byte*)0x1004;
   byte i;
   while (1) {
      c1=0x80; c2=0x01;
      for (i=0;i<4;i++) {
         wait();
         *p=c1|c2;
         c1>>=1; c2<<=1;
      }
   }
   return 0;
}
```

THRijswijk

12

# µcontroller




THRijswijk

13

# Bits en bytes

- Byte = groepje van **8** bits
  - C: unsigned char / signed char
- Word (bij de HC11) = **16** bits
  - C: unsigned short / signed short
  - bestaat uit MSB en LSB.
- **32** bits
  - C: int / unsigned int

- Decimaal
  - C: 17, 1022
  - Assembler: **&**17, **&**1022
- Hexadecimaal
  - C: **0x**11, **0x**03FE
  - Assembler: **$**11, **$**03FE
- Binair
  - C: ? (gebruik hex)
  - Assembler: **%**00010001, **%**0000001111111110

THRijswijk

14

# Byte

betekenis

Wat betekent **$bd** ?

- 189 (unsigned binary)
- -67 (signed binary = two's complement)
- C (7 bits ASCII met even pariteitsbit)
- ½ (8 bits ASCII)
- JSR (68HC11 instructie: jump to subroutine)
- ... (Wat jij wil !!!)

THRijswijk

15

# memory map

indeling van geheugen

Memory Map of the EVM.

| Memory Address | Function |
|---|---|
| $0000 - $00FF | RAM memory (256 bytes) |
| $0100 - $0FFF | unused |
| $1000 - $103F | special registers (64 bytes) |
| $1040 - $B5FF | unused |
| $B600 - $B7FF | EEPROM memory (512 bytes) |
| $B800 - $BFFF | unused |
| $C000 - $FFFF | ROM memory (16384 bytes) |

THRijswijk

16

# JSR en RTS

subroutine

- Terugkeeradres wordt opgeslagen op de stack (stukje RAM aangewezen door SP).
- SP wijst (bij de 68HC11) naar de eerste lege plaats en "groeit" naar adres $0000 toe.

```
_start  lds  #$00FF
        ...
main    ...
        jsr  wait
        ...
        jsr  wait
        ...

wait    ...
        rts
```

THRijswijk



17

# Interrupts

- Onderbreking van "normale" programma.
- Verschillende redenen:
  - Karakter ontvangen via seriele poort.
  - Timer die afloopt.
  - Bepaald ingangssignaal veranderd.
  - Enz...
- Bij optreden interrupt:
  - Alle registers gesaved (op de stack).
  - Spring naar een bij de interrupt behorende **interrupt service routine** (**ISR**).
  - ISR eindigt met RTI instructie die alle registers restored.
  - Onderbroken programma gaat verder.

THRijswijk

18

# Interrupts

! Tijdens ISR wordt niet op andere interrupts gereageerd.
" ISR moet dus **snel** zijn.
! Interrupt **vector** wijst naar begin van ISR.
! 68HC11 heeft 32 interrupt vectors $FFC0 -$FFFF.

| | |
|---|---|
| $FFC0 | reserved |
| ... | ... (zie reference guide) |
| $FFEE | timer input capture 1 |
| $FFF0 | **real time interrupt** |
| $FFF2 | extrernal pin or parallel IO (IRQ) |
| $FFF4 | non maskable interrupt (XIRQ) |
| $FFF6 | software interrupt (SWI) |
| $FFF8 | illegal opcode trap |
| $FFFA | COP failure |
| $FFFC | COP clock monitor fail |
| $FFFE | **system reset (RESET)** |





# Real-time interrupt

! Soort wekker die telkens na een bepaalde tijd afloopt.
! Tijd wordt ingesteld met RTR1 en RTR0 bits in PACTL register. (Klokfrequentie EVM = 8 MHz).
! Na ingestelde tijd wordt bit RTIF in het TFLG2 register geset.
! RTIF bit kan gereset worden door er een 1 naar toe te schrijven (raar maar waar)!
! Als je wilt kun je dan een real-time interrupt opwekken: bit RTII in TMSK2 register aanzetten.







# Real-time interrupt

```
volatile byte* tmsk2=(byte*)0x1024;
volatile byte* tflg2=(byte*)0x1025;
void rtti_isr(void) __attribute__((interrupt));
void rtti_isr(void) {
   /* ... */
   *tflg2=0x40;
}
int main() {
   *tmsk2|=0x40;
   /* ... */
}

typedef void (*isr)(void);
#define E (isr)0xffff
isr vectors[32] __attribute__ ((
                        section (".vector"))) = {
   E, E, E, E, E, E, E, E,
   E, E, E, E, E, E, E, E,
   E, E, E, E, E, E, E, E,
   rtti_isr, E, E, E, E, E, (isr)0xc000
};
```





# Reset vector initialiseren

" Je wilt niet het “harde” adres 0xc000 gebruiken maar de identifier _start
" _start is gedefinieerd in de file **crt1.o** die automatisch wordt meegelinkt.

```
typedef void (*isr)(void);
#define E (isr)0xffff
#ifdef __cplusplus
extern "C" void _start(void);
#else
extern void _start(void);
#endif
isr vectors[32] __attribute__
((section(".vector"))) = {
   E, E, E, E, E, E, E, E,
   E, E, E, E, E, E, E, E,
   E, E, E, E, E, E, E, E,
   rtti_isr, E, E, E, E, E, E, _start
};
```