2021

G.M. Tuk

X. van Rijnsoever

TIMERS

Microcontroller Programmeren 2 – Week 2

DE HAAGSE HOGESCHOOL

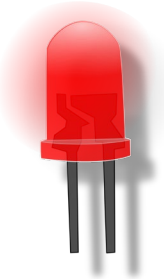
Wat gaan we doen vandaag?

- Een alternatief voor een knopje: een klokje
- Slimmer een led laten knipperen
- Willekeurige getallen genereren

Tot nu toe

- Om een bepaalde tijd te wachten, hebben we tot nu toe gebruik gemaakt van een wachtlus: **_delay_ms()**
- Dankzij de timer hoef je geen tellertje te gebruiken om te wachten: je kijkt echt “op de klok”.
- Belangrijker: je hoeft niet telkens je tellertje aan te passen als er statements en functie calls worden toegevoegd.

Wat is programmeren?



Invoer → Verwerking → Uitvoer

DE HAAGSE HOGESCHOOL

Timers

De controller van de Arduino heeft drie (of meer) timers.

- Timer0 (8 bits)
- Timer1 (16 bits)
- Timer2 (8 bits)

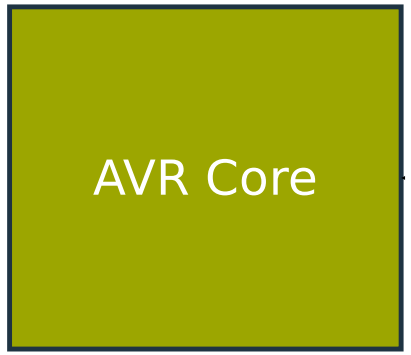
Op de Mega ook:

- Timer3, Timer4, Timer5

Let op: sommige bordjes werken op 12 MHz!

Ze maken gebruik van de systeemklok, hier 16 MHz.

Hardware



Relevante registers

- **TCNTx** - Timer/Counter Register.
  Hier wordt de timerwaarde opgeslagen.
- **OCRx** - Output Compare Register
  voor genereren PWM signalen
- **ICRx** - Input Capture Register (alleen voor 16 bit timer)
- **TIMSKx** - Timer/Counter Interrupt Mask Register.
  voor het aan- en uitzetten van timer interrupts
- **TIFRx** - Timer/Counter Interrupt Flag Register.
  Geeft aan dat er een gebeurtenis in de timer heeft plaatsgevonden.

TCCRx Registers

16.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 16-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 16-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected

temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 138.

17.11.33 TIMSK1 – Timer/Counter 1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

17.11.34 TIMSK3 – Timer/Counter 3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x71)	–	–	ICIE3	–	OCIE3C	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

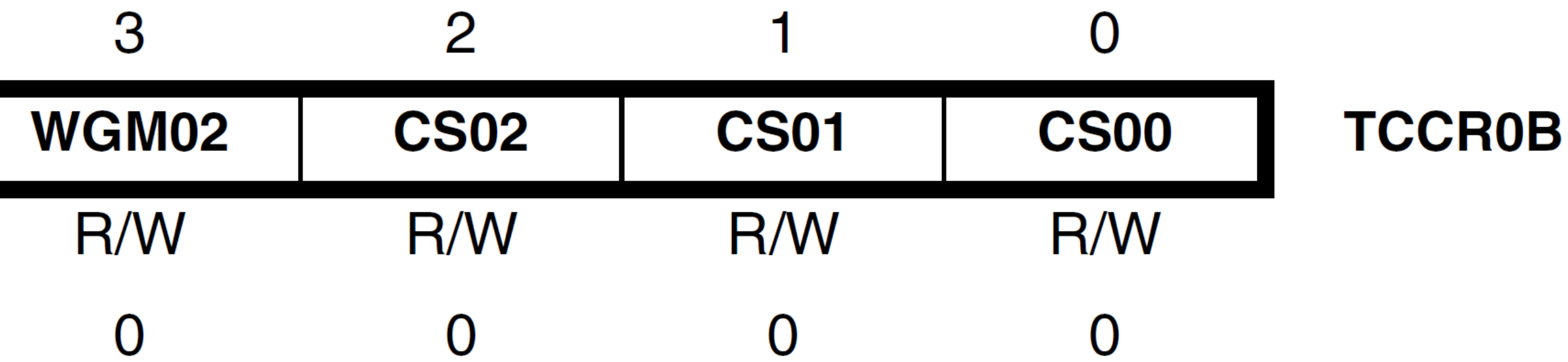
17.11.35 TIMSK4 – Timer/Counter 4 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x72)	–	–	ICIE4	–	OCIE4C	OCIE4B	OCIE4A	TOIE4	TIMSK4
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

17.11.36 TIMSK5 – Timer/Counter 5 Interrupt Mask Register

16.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



DE HAAGSE HOGESCHOOL

Prescaling

Table 16-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

TCNTx Register

Twee bytes (twee registers) nodig voor 16 bits timer

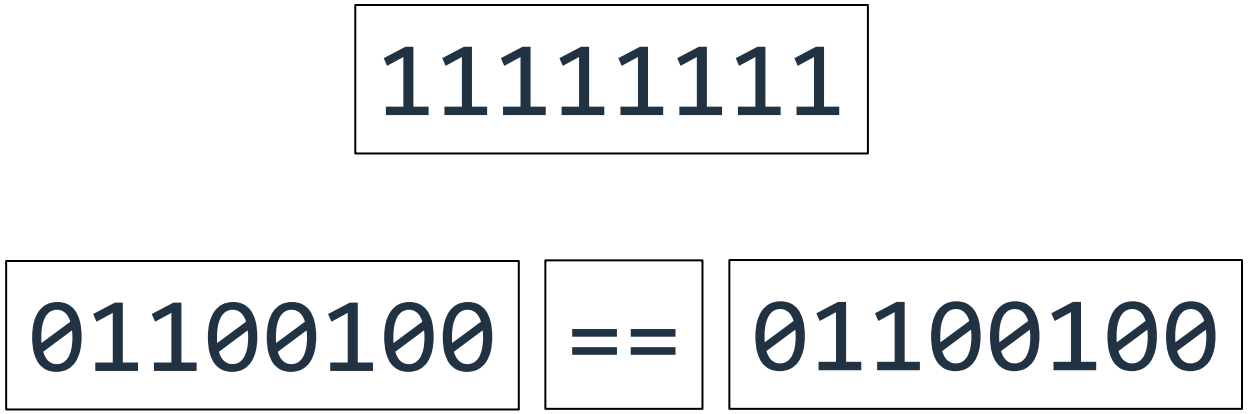
16.9.3 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

Verschillende modes

- timer overflow
- output compare match



DE HAAGSE HOGESCHOOL

Previously on Microcontroller Programmeren I...

Include files nodig

```
#include <avr/io.h>          // voor gebruik DDRx
#include <util/delay.h>      // voor delay_ms()
```

```c
int main(void)
{
    // configureer pin
    DDRB |= (1 << PB7);          // voor de Mega
    while (1)
    {
        // Toggle led
        PORTB ^= (1 << PB7);
        _delay_ms(500);
    }
}
```

```
void initTimer (void)
{
    // Timer 0 is de systeemtimer (overflow)
    TCCR0A = 0;

    // 16000000 Hz / 64 / 256 = 976.56 Hz
    // start timer 0, prescaler = 64
    TCCR0B = TCCR0B | (1 << CS01) | (1 << CS00);
}
```

```
    // enable timer 0 voor overflow interrupt flag
    TIMSK0 = TIMSK0 | (1 << TOIE0);
}

int main(void)
{
    initTimer();

    // wacht tot overflow flag geset
}
```

Random numbers

- Voor simulatie-achtige programma's is het handig om random (willekeurige) getallen te kunnen genereren.
- In C gebruik je daarvoor de functie `rand()`, die een waarde genereert tussen `0` en `RAND_MAX`.
- De modulus operator `%` kun je vervolgens gebruiken om daar de getallen van te maken die je hebben wilt.

Random numbers

```
int dobbelsteen(){
        int willekeurig = rand();
        return 1 + willekeurig % 6;
}
```

Let goed op de onder- en bovengrens van uitkomsten.

Om te voorkomen dat de functie `rand` steeds dezelfde “random” getallen geeft, heb je `srand` nodig...

Random numbers

```
#include <time.h>
#include <stdlib.h>

int willekeurig(int max){
        srand(time(NULL));  // eenmalige initialisatie
        int getal = rand(); // waarde tussen 0 and RAND_MAX
        return ...;
}
```

`time(NULL)` geeft het aantal seconden sinds 1 januari 1970

Bestaat niet voor AVR!

let’s change

YOU. US. THE WORLD.

1010 0100.