

## INTERRUPT-STRUCTUUR

In een computerprogramma worden de instructies één voor één na elkaar uitgevoerd. Hiervoor wordt de programma-counter (PC) elke keer met 1 verhoogd. De programma-counter kan weliswaar door de diverse BRANCH-instructies (al dan niet geconditioneerd) worden beïnvloed, maar het programmaverloop blijft sequentieel en volgt in principe zijn eigen instructievolgorde. In veel praktische situaties zal het nodig zijn dat op verzoek van bijvoorbeeld een sensor een bepaald programmadeel wordt uitgevoerd. Deze verzoeken zijn a-synchroon met het hoofdprogramma en worden interrupts genoemd. Het programma dat vervolgens wordt uitgevoerd noemt men een interrupt handler of soms een interrupt service routine. De krukhoeksensor van een auto zou hiervan een voorbeeld kunnen zijn. Op het moment dat het referentiepunt de sensor passeert, wordt een interrupt aangevraagd; de bijbehorende interrupt handler realiseert vervolgens het nieuwe ontstekingstijdstip. Nadat de interrupt handler is uitgevoerd, wordt het onderbroken hoofdprogramma weer vervolgd.

Interrupt handlers lijken veel op subroutines met dit verschil dat ze door de hardware in plaats van door een RCALL (relative call to subroutine) worden opgeroepen. Ook worden tegelijk met de programma-counter alle registers op de stack gezet. In plaats van een RET-instructie aan het einde van een subroutine, eindigt een interrupt handler met een **RETI**-instructie (ReTurn Interrupt).

Interrupts kunnen aan en uitgezet worden (enable en disable). Dit dient te gebeuren met de **SEI** (enable) en de **CLI** (clear) instructie. De SEI instructie dient als laatste instructie in de initialisatie te staan. Behalve dat dient ook de individuele enable bit te worden geset.

De interrupt-structuur werkt met behulp van vectoren. Een vector is een adres waarin zich het eerste adres van de interrupt handler zich bevindt. De vectoren zijn genummerd en hebben een vaste bron. De Reset kan worden beschouwd als een bijzondere hardware interrupt. Ze bevinden zich in de eerste adressen van de program memory space (Flash). Hierna volgt de tabel van de interrupt vectoren van de AT90CAN32/64/128:

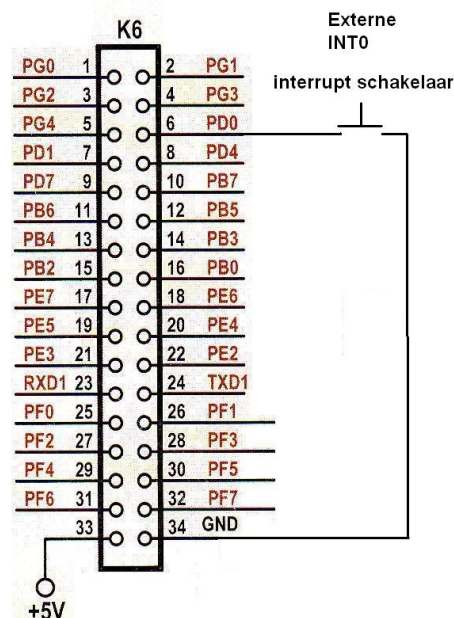


Fig. 1 Een interrupt-drukschakelaar kan worden aangesloten op pin 6 van de 34 polige K6 connector van de elektor controller set. Pin6 van de header is verbonden met pin 25 van de

controller.

Vector No.	Program adres	Equ (uit can32def)	Bron	Interrupt definitie
1	0x0000	INT0addr	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	0x0002	NT1addr	INT0	External Interrupt Request 0
3	0x0004	NT2addr	INT1	External Interrupt Request 1
4	0x0006	NT3addr	INT2	External Interrupt Request 2
5	0x0008	NT4addr	INT3	External Interrupt Request 3
6	0x000A	NT5addr	INT4	External Interrupt Request 4
7	0x000C	NT6addr	INT5	External Interrupt Request 5
8	0x000E	NT7addr	INT6	External Interrupt Request 6
9	0x0010		INT7	External Interrupt Request 7
10	0x0012		TIMER2 COMP	Timer/Counter2 Compare Match
11	0x0014		TIMER2 OVF	Timer/Counter2 Overflow
12	0x0016		TIMER1 CAPT	Timer/Counter1 Capture Event
13	0x0018		TIMER1 COMPA	Timer/Counter1 Compare Match A
14	0x001A		TIMER1 COMPB	Timer/Counter1 Compare Match B
15	0x001C		TIMER1 COMPC	Timer/Counter1 Compare Match C
16	0x001E		TIMER1 OVF	Timer/Counter1 Overflow
17	0x0020		TIMERO COMP	Timer/Counter0 Compare Match
18	0x0022		TIMERO OVF	Timer/Counter0 Overflow
19	0x0024		CANIT	CAN Transfer Complete or Error
20	0x0026		OVRIT	CAN Timer Overrun
21	0x0028		SPI, STC	SPI Serial Transfer Complete
22	0x002A		USART0, RX	USART0, Rx Complete
23	0x002C		USART0, UDRE	USART0 Data Register Empty
24	0x002E		USART0, TX	USART0, Tx Complete
25	0x0030		ANALOG COMP	Analog Comparator
26	0x0032		ADC	ADC Conversion Complete
27	0x0034		EE READY	EEPROM Ready
28	0x0036		TIMERS CAPT	Timer/Counter3 Capture Event
29	0x0038		TIMERS COMPA	Timer/Counter3 Compare Match A
30	0x003A		TIMERS COMPB	Timer/Counter3 Compare Match B
31	0x003C		TIMERS COMPC	Timer/Counter3 Compare Match C
32	0x003E		TIMERS OVF	Timer/Counter3 Overflow
33	0x0040		USART1, RX	USART1, Rx Complete
34	0x0042		USART1, UDRE	USART1 Data Register Empty
35	0x0044		USART1, TX	USART1, Tx Complete
36	0x0046		TWI	Two-wire Serial Interface

37	0x0048	SPM READY	Store Program Memory Ready
----	--------	-----------	----------------------------

Aan het begin van het programma kunnen we de vectoren als het volgt initiëren. De interrupt service routine dient dan de bijbehorende label te krijgen,

Address	Labels	Code	Comment
0x0000	jmp RESET	;Reset Handler	
0x0002	jmp EXT_INT0	;IRQ0 Handler	
0x0004	jmp EXT_INT1	;IRQ1 Handler	
0x0006	jmp EXT_INT2	;IRQ2 Handler	
0x0008	jmp EXT_INT3	;IRQ3 Handler	
0x000A	jmp EXT_INT4	;IRQ4 Handler	
0x000C	jmp EXT_INT5	;IRQ5 Handler	
0x000E	jmp EXT_INT6	;IRQ6 Handler	
0x0010	jmp EXT_INT7	;IRQ7 Handler	
0x0012	jmp TIM2_COM	;Timer2 Compare Handler	
0x0014	jmp TIM2_OVF	;Timer2 Overflow Handler	
0x0016	jmp TIM1_CAPT	;Timer1 Capture Handler	
0x0018	jmp TIM1_COMPA	;Timer1 CompareA Handler	
0x001A	jmp TIM1_COMPB	;Timer1 CompareB Handler	
0x001C	jmp TIM1_OVF	;Timer1 CompareC Handler	
0x001E	jmp TIM1_OVF	;Timer1 Overflow Handler	
0x0020	jmp TTM0_COMP	;Timer0 Compare Handler	
0x0022	jmp TIM0_OVF	;Timer0 Overflow Handler	
0x0024	jmp CAN_IT	;CAN Handler	
0x0026	jmp CTIM_OVF	;CAN Timer Overflow Handler	
0x0028	jmp SPI_STC	;SPI Transfer Complete Handler	
0x002A	jmp USART0_RXC	;USART0 RX Complete Handler	
0x002C	jmp USART0_DRE	;USART0 UDR Empty Handler	
0x002E	jmp USART0_TXC	;USART0 TX Complete Handler	
0x0030	jmp ANA_COMP	;Analog Comperator Handler	
0x0032	jmp ADC	;ADC conversion Complete Handler	
0x0034	jmp EE_RDY	;EEPROM Ready Handler	
0x0036	jmp TIM3_CAPT	;Timer3 capture Handler	
0x0038	jmp TIM3_COMPA	;Timer3 compareA Handler	
0x003A	jmp TIM3_COMPB	;Timer3 Compare B Handler	
0x003C	jmp TIM3_COMPC	;Timer3 Compare C Handler	
0x003E	jmp TIM3_OVF	;Timer3 overflow Handler	
0x0040	jmp USART1_RXC	;USART1 RX Complete Handler	
0x0042	jmp USART1_DRE	;USART1 UDR Empty Handler	
0x0044	jmp USART1_TXC	;USART1 TX Complete Handler	
0x0046	jmp TWI	;TWI interrupt handler	
0x0048	jmp SPM_RDY	;SPM Ready Handler	

Men onderscheidt maskeerbare en niet maskeerbare interrupts. Een niet maskeerbare interrupt wordt altijd in behandeling genomen en een maskeerbare interrupt kan in een bepaald gedeelte van het programma worden genegeerd.

We kunnen de volgende interrupts onderscheiden:

- de externe hardware interrupts (RESET, INT0 t/m INT7);
- de interne hardware interrupts (timer, seriële communicatie e.d.);

- de software interrupt (SWI).

### Externe interrupten

De AT90CAN kent 8 externe interrupt bronnen (INT0 t/m INT7). Dit is de derde functie van poort D en poort E

PD0 INT0 pin 25  
 PD1 INT1 pin 26  
 PD2 INT2 pin 27  
 PD3 INT3 pin 28

PE4 INT4 pin 6  
 PE5 INT5 pin 7  
 PE6 INT6 pin 8  
 PE7 INT7 pin 9

Sluiten we op pin 25 van de controller bijv. een schakelaar aan die de pin met de massa verbindt dan kan, als aan een aantal voorwaarden wordt voldaan, een interrupt (INT0) worden uitgevoerd op het moment dat de schakelaar pin 25 aan massa legt (fig. 1).

Als de interrupt wordt toegestaan dan wordt de interrupt ook uitgevoerd als de poort als 'output' is geconfigureerd. Het maakt dus niet uit op de poort als input of als output is ingesteld. De voorkeur is echter 'input'.

De externe interrupten INT0 t/m INT7 worden geactiveerd wanneer de **SREG I-flag** (SEI en CLI) en de overeenkomstige bit in het EIMSK-register een logische 1 krijgt. In het volgende voorbeeld is INT0 geactiveerd.

External Interrupt Mask Register (EIMSK)

INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
0	0	0	0	0	0	0	1

**Triggering van een interrupt kan worden ingesteld op een opgaande of neergaande flank of op een laag signaal of op elke logische niveau verandering.**

We kunnen dit instellen me behulp van het **EICRA-register** (External Interrupt Control Register A) Default staan alle waarden op 0 en dit betekent een low-level interrupt.

INT3		INT2		INT1		INT0	
ISC3 1	ISC3 0	ISC2 1	ISC2 0	ISC1 1	ISC1 0	ISC0 1	ISC0 0
0	0	0	0	0	0	0	1

**In dit voorbeeld wordt een interrupt op INT0 gegenereerd op elke logische verandering**

Betekenis van de bits

- 00 low level (laag) genereert een interrupt;
- 01 elke logische veranderen genereert een interrupt;
- 10 neergaande flank genereert een interrupt;
- 11 opgaande flank genereert een interrupt.

Programma interrupt1.asm geeft een voorbeeld van het gebruik van een interrupt.