AN01 Implementering av Watchdog Timer i Interrupt og System Reset modus

ATmega328P

Petter Wandsvik

I. Introduksjon

DETTE applikasjonsnotatet tar for seg karakteristikk, funksjonalitet og implementering av periferienheten WDT - Watchdog Timer, for mikrokontrolleren ATmega48A/PA/88A/PA/168A/PA/328/P.

WDT er en periferienhet som ved bruk av separat on-chip 128kHz oscillator fungerer som en timer/counter. For hver klokkesyklus øker telleren frem til den når en gitt time-out verdi. Når telleren når den gitte verdien, og systemet ikke har nullstilt telleren, vil WDT gi et avbrudd (eng. interrupt) og/eller restart av systemet, **avhengig av valgt modus**.

For mikrokontrolleren vil WDT være nyttig med bakgrunn i at den vil forhindre at mikrokontrolleren forblir i en ugyldig tilstand - eksempelvis at kontrolleren kjører seg fast i en løkke på bakgrunn av en bug. I kritiske system kan en slik bug være katastrofal og er dermed en viktig komponent i et komplett system.

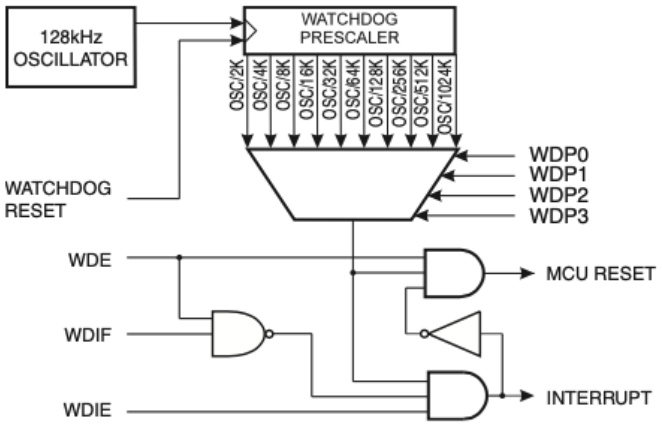

Fig. 1. Watchdog Timer, hentet fra ATmega328p datablad.

II. Teori

WDT kan operere i tre moduser: Interrupt, System Reset, Interrupt og System Reset.

Interrupt: Når telleren når den gitte verdien satt i programvaren, vil WDT gi en interrupt. Denn funksjonaliteten kan brukes som en måte å vekke kontrolleren fra sleep-modes, eller begrense tiden kontrolleren kan bruke på gitte oppgaver.

System Reset: Når telleren når den gitte verdien satt i programvaren, vil WDT gi en system reset. WDT i System Reset-modus er typis brukt som en metode for å forhindre at systemet henger seg opp som et resultat av unormal programflyt.

Interrupt og System Reset: Kombinasjonen vil tillate programmet å først gi et avbrudd, for deretter en system reset. Avbruddet i forkant vil tillate en trygg omstart ved at kritiske parameter eventuelt kan lagres før systemet starter på nytt.

A. Endring av register

Eksempelvis valg av modus skjer ved å endre periferienhetens register. Ved å bla opp i databladet for ATmega328P, under 11.9.2 WDTCSR - Watchdog Timer Control Register, finner vi registeret som kan endres etter applikasjonens behov. Se referanse [1] for datablad og figur 3 for WDTCSR.

11.9.2 WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

Fig. 2. Kontrollregister for WDT - WDTCSR, hentet fra ATmega328p datablad.

Det eksisterer et system for å endre registrene. Ved å inkludere avr/io.h, kan vi bruke pre-defines som avr allerede har laget.
Studerer vi figur 3 kan vi se at registeret heter WDTCSR og videre navnene på de respektive bitposisjonene, eksempelvis WDIF, WDIE etc.

Går vi nå inn i databladet for å lese beskrivelsen av funksjonaliteten til bitposisjonene, kan vi for eksempel se at ved å skrive 1 til bitposisjon 3, altså WDE: Watchdog System Reset Enable, vil vi sette WDT i System Reset Mode. Beskrivelse av bitposisjonene er også under 11.9.2 i databladet.

```
WDTCSR = (1 << WDE);
// Leses som: Skriv 1 til bitposisjon WDE
```

Hvis vi ønsker kan vi skrive det binært, men som raskt kan bli uoversiktelig og vanskelig å tolke jo mer kode som kommer. Linjen med kode under viser hvordan det eventuelt ville sett ut:

```
WDTCSR = (1 << 0b00001000);
```

Men for forståelsens del så tilsvarer hver bit i linjen over en plass i registeret, WDTCSR, og kan leses som skriv 1

til bitposisjon 3. WDE er definert som 3 eller 0b00001000 i avr/io.h.

Ønsker vi videre å legge til bits, for eksempel sette WDT i **Interrupt og System Reset** kan vi bruke or-operatoren kjent fra digitalteknikken. I c blir syntaxen:

```
WDTCSR = (1 << WDIE) | (1 << WDE);
```

Setter vi opp en sannhetstabell for dette vil vi fort se at det blir 0b01001000.

B. Skalering av timer

Skaleringen av timeren bestemmes av bitposisjonene WDP[3:0]. I tabell 11-2 fra databladet til ATmega328P, finner vi de forskjellige prescale-verdiene samt deres periodetider.
For å gjøre endringer i forbindelse med skalering av timer eller å sette bit WDE lav, må bitposisjon WDCE - Watchdog Change Enable, settes høy. Denne biten vil settes lav igjen av hardware etter fire klokkesykluser som betyr at endringen av prescale-bits og WDE må skje innen den tid.

Table 11-2. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16ms
0	0	0	1	4K (4096) cycles	32ms
0	0	1	0	8K (8192) cycles	64ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s

Fig. 3. WDP[3:0] bestemmer WDT skaleringen av oscillatoren. Figuren viser forskellige skaleringsverdier, samt deres korresponderende time-out perioder.Figuren er hentet fra ATmega328P datablad.

C. Aktivering av Interrupt, ISR og bruk av interrupt-vektor

Avbrudd, eller interrupts er et signal til prosessoren om at det er en handling som krever umiddelbar oppmerksomhet. Avbrudd kan være styrt av både software og hardware og vil bryte normal programflyt der hvor avbruddet oppstår.
Globale Interrupts må aktiveres før det kan oppstå et avbrudd. Funksjonene cli() og sei(), som henholdsvis deaktiverer og aktiverer globale avbrudd, må skrives inn i koden. Ved initialisering av WDT bør globale interrupts deaktiveres for å unngå at et avbrudd skjer midt i initialiseringen. Etter linjene med initialiserings-kode må globale avbrudd aktiveres igjen.

ISR - Interrupt Service Routine, er veldig likt vanlige funksjoner i c, men kalles av avbrudd og terminerer etter utført ISR-kode. ISR kan aksessere globale variabler samt ha lokale variabler, men globale variabler brukt av ISR bør deklareres som **volatile**. Dette indikerer at variabelen kan endres utenfor normal programflyt.

I datablad for ATmega328P finner vi en tabell med Reset og Interrupt-vektorer. En interrupt-vektor er en adresse i programminnet som forteller interrupt-handleren hvor den finner ISR tilknyttet det spesifiserte avbruddet. Syntax for ISR og avbrudds-vektor knyttet til WDT:

```
ISR(WDT_vect){
/* Kode som skal utføres ved WDT-avbrudd.
   Eksempelvis lagring av kritisk data
   til EEPROM. */
}
```

Går vi inn i databladet for ATmega328P, under 12.4 Interrupt Vectors in ATmega328 and ATmega328P, finner vi en tabell med kontrollerens Reset og Interrupt vektorer. For mer informasjon rundt ISR og interrupt vektorer, se vedlegg [3] og [4].

D. Nullstilling av timer

For å unngå at programvaren gjennomfører en system-reset uavhengig av systemtilstand, må WDT-timeren i normal modus nullstilles. Ved å inkludere avr/wdt.h vil en kunne bruke wdt_reset()-funksjonen for nullstilling av timeren. Hvis systemet ikke nullstiller timeren vil et avbrudd eller en system-reset bli utstedt.

III. Funksjonell beskrivelse

A. Initialisering

For initialisering av WDT i Interrupt og System Reset modus, gjør følgende:

1) Inkluder avr/io.h, avr/wdt.h, avr/interrupt.h
2) Nullstill WDT-timer
3) Sett WDCE, WDE og WDIE i WDTCSR registeret
4) Innen fire klokkesykluser, sett time-out perioden i WDP[3:0]

Merk! En endring i WDP[3:0] kan resultere i en time-out når time-out perioden endres til en kortere periode og timeren bør dermed nullstilles før endring i WDP[3:0].

IV. Konklusjon

Applikasjonsnotatet gir en enkel måte å initialisere og implementere Watchdog Timer for ATmega328P i Interrupt og System Reset modus. Implementeringen gir utgangspunkt til en sikkerhetsmekanisme som kan brukes i kritiske systemer hvor programvarefeil og ugyldige tilstander kan være katastrofal. Modusen tillater programvaren å lagre kritisk data i forkant av system-nullstillingen dersom viktig data må lagres.

APPENDIX A
VEDLEGG

[1] megaAVR® Data Sheet - ATmega48A/PA/88A/PA/168A/PA/328/P. Microchip Technology Inc. 2020, hentet fra: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[2] Microchip Technology, avr_insights/Ep_7_Watchdog_Timer/avr_wdt/main.c, Chandler, AZ, 2019, hentet fra: https://github.com/MicrochipTech/avr_insights/blob/master/Ep_7_Watchdog_Timer/avr_wdt/main.c

[3] More C and the wider C environment,Tim Wilmshurst, ScienceDirect, 2020, hentet fra: https://www.sciencedirect.com/topics/computer-science/interrupt-service-routine

[4] Microprocessors, Peng Zhang, ScienceDirect, 2020, hentet fra: https://www.sciencedirect.com/topics/engineering/interrupt-vector