KIM-TIMER EN KLOK

HERMAN PERK

Werkend met de KIM ben ik ooit op het probleem gestuit, dat er tijdens de loop van een programma een klok moest worden bijgehouden. Weliswaar niet voor de KIM zelf, maar voor een systeem met TIM-monitor, namelijk het op BEM-bus gebaseerde 6502-systeem van de firma Brutech, dat overigens zeer aan te bevelen is door z'n hoge kwaliteit en doordachtheid (vandaar die prijzen).
Nu ontbrak het eind 1978 bij zowel de KIM als bij het BEM-systeem nog aan 6522-VIA's met hun veelzijdige timers, in tegenstelling tot de SYM en AIM-65. Erger nog, de TIM-monitor maakt voor communicatie met de terminal gebruik van de enige timer die het systeem rijk is.

Wilde ik toch de timer van de 6530 voor de klok gebruiken, dan diende ik voor de communicatie een eigen programma te schrijven, gebaseerd op de zogenoemde software timings loops. Ik heb dat dan ook gedaan, daar ik voor het resultaat TIM zelf niet meer nodig had. Wat de timer betreft (die van de 6530 en 6532 zijn aan elkaar gelijk), de 'overduidelijke' gebruiksaanwijzing vertelt niets over een eventuele 'free-running' mode zoals dat voor een klok gebruikelijk is (en zoals wel van de 6522 is beschreven).
Het is wel mogelijk om in 'one shot' mode een klok te maken, maar dan mag er geen gebruik worden gemaakt van de Non Maskable Interrupt (NMI) zoals in mijn geval, want dan bestaat de kans dat de klok onvoorspelbaar achter gaat lopen. Er mag dan immers geen onbekende vertraging zitten tussen het aflezen van de timer en opnieuw starten ervan. (Een klok op deze wijze gemaakt is bekend bij de KIM-club in Nederland.)
Een volgende 'bijkomstigheid' was dat de terminal-communicatie (op interrupt basis), die ik zelf met timingloops had gemaakt (op 1200 baud), niet al te lang onderbroken mocht worden door de interrupt van de klok; de klok interruptservice routine moest kort wezen. Ondertussen had ik nog drie andere vrij lange IRQ-routines en een NMI-routine, die alle onderbroken moesten kunnen worden door een korte (!) IRQ van de klok.
Bij de KIM, waarvan ik gebruik maak, was bij aanschaf een boekje 'KIM Hints' bijgevoegd (ook in z'n geheel gepubliceerd in het orgaan 'KIM kenner' van de eerder genoemde KIM-club). Hierin vond ik een aanwijzing naar de oplossing, namelijk in de afgedrukte listing van het voorbeeld voor het gebruik van de interval timer staat als comment line:

WHEN THE TIMER IS READ THE DEVIDER IS RESTORED TO ITS ORIGINAL VALUE AND THE INTERRUPT IS RESET

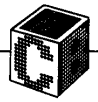
en erboven:

TRE = $170E READ TIME ENABLE INT

(Degene die moeilijkheden hebben met de timer van de KIM raad ik aan het bedoelde artikel op de kop te tikken en goed te bestuderen: hierin staat een duidelijker gebruiksaanwijzing.)

Wat mogelijk leek, was door na de interrupt alleen de timer uit te lezen (dit geeft het one's complement van de tijd in clock perioden die verstreken was sinds de interrupt) waardoor het hele proces zichzelf opnieuw start met een waarde die bekend is, namelijk het uitgelezene. Er zou dus alleen een correctie dienen te worden uitgevoerd ter grootte van de verstreken tijd tussen interrupt en uitlezen van de timer.
Na dagenlang de tijd bijgehouden te hebben bleek mij echter dat de timer nog mooier was dan ik had vermoed: correctie was niet nodig! De basistellers van de timer bleken niet te worden gereset door het lezen zoals dit door het schrijven het geval is. Ik zal op de hardware aspecten van de timer niet verder ingaan. Wel verwijs ik naar de fabrieksliteratuur van MOS (Indelec, Breda), Rockwell (Famatra, Breda), Synertek (Indelec, Brutech en MCatronics) en het boekje KIM Hints (KIM-club), omtrent de 6530 en 6532.
(In de produktbeschrijving van Rockwell van de 6532 staat nog een leuk extraatje omtrent de verstreken tijd tussen interrupt en het lezen van de timer: trek één af van het met één vermeerderde complement: dus in plaats van two's complement dient men one's complement te hanteren in de 'one-shot'-mode van de timer.)

Het uiteindelijke resultaat, aangepast voor publicatie, werkt als volgt: in een initialiseringsfase start ik de timer door hem op $170F (deelverhouding 1024) te schrijven met een 'willekeurig' getal bijv. $FF (in regel 300). Het aftellen van de timer duurt dan 256 maal 1024 clock perioden dus ruim een kwart seconde.
Dan treedt er een interrupt op, mits natuurlijk PB7 van de 6530 aan de IRQ (interrupt request line) is vastgemaakt: applicatieconnector pen 15 aan de expansieconnector pen 4. Na een onbepaalde tijd, maar korter dan 256 clock



timer en klok

perioden, lees ik de timer uit op adres $170E. Hiermee start ik de timer met het uitgelezen getal opnieuw, met deelverhouding 1024 (zoals KIM Hints aangeeft) en 'enable' de interrupt. Ik blijk nu over de informatie te beschikken hoeveel tijd er zal verstrijken tussen de interrupt die net is geweest en de volgende interrupt! (Immers de correctie vervalt.)
Deze informatie tel ik op bij een 16-bits teller 'TIMEX' en verlaat de interrupt service routine (regel 1110 t/m 1300) met RTI.
Het hoofdprogramma bestaat uit het loopje van regel 320 t/m 400 waarin het display wordt uitgelezen en de tijd wordt bijgewerkt (en in de KIM-buffer geschreven).
Subroutine TIMKEP houdt de tijd bij. In mijn situatie was dit te lang om in de interrupt service routine te stoppen, dus heb ik er een aparte subroutine van gemaakt, die ik in het hoofdprogramma elke keer moet aanroepen voordat de interrupt service routine de 16-bits teller te klein gaat vinden. Niet alle 16-bits mogen worden gebruikt, want in regel 490 t/m 530 worden er twee weggegooid.
'TIMER' is een 24-bits teller waar elke keer 1024 maal 'TIMEX' bij wordt opgeteld (490 t/m 590) en waarbij 'TIMEX' wordt gereset (420 t/m 480). Daarna wordt het vergeleken met $0F4240 (1000000), immers er gaan $0F4240 clock perioden in één seconde, en als er tenminste één seconde is verstreken wordt er $0F4240 van 'TIMER' afgetrokken (600 t/m 800). Als dit laatste gebeurt, wordt de tijd decimaal bijgehouden in seconden ('TIME+02'), minuten ('TIME+01'), uren ('TIME') en binair in dagen ('DAGNO'). Dit geschiedt in regels 810 t/m 1090. (De datum had ik ook nog nodig, vandaar dit rudiment hier.)

Het gelijkzetten gebeurt door met de monitor 'TIME' te vullen met het gewenste uur, minuut en seconde als een gewone digitale 24-uurs klok, en vervolgens bij TEST te starten (adres $0200) of op de knop ST van het KIM-toetsenbord te drukken. Dit laatste echter alleen als regels 150 t/m 220 tenminste eens zijn doorlopen, want deze zetten de vectoren op hun waarden die ze voor dit gebruik behoeven.
In de meeste gevallen zal de klok na verloop van tijd niet meer gelijk blijken te lopen (ik had 13 seconden per dag achterstand). Dit is ten gevolge van het niet exact afgeregeld zijn van het kristal van de processor-clock.
Het testloopje in de interrupt routine (1140 t/m 1200) zorgt voor een regelmatige verandering van de seconden (het getal dat gelezen wordt, en waarmee de timer dus wordt geset, wordt kleiner). Hier heb ik een routine zitten, die test van welk randapparaat de interrupt afkomstig is: device polling. Voor mij was het niet belangrijk dat de seconden onregelmatig leken te tikken. Dit kan op een andere wijze worden veranderd door een andere deelverhouding te kiezen. Dan moeten ook regels 490 t/m 590 worden veranderd (deze tellen immers 1024 maal 'TIMEX' bij 'TIMER' op).
De listing is gemaakt met behulp van MICRO-ADE, een bij de Muiderkring verkrijgbare Assembler, Dissembler, Editor-combinatie, die 8K extra geheugen en een terminal vereist boven de kale KIM. (Famatra heeft bijv. MEMORY-PLUS, of Brutech z'n BEM-systeem voor die extra 8K byte.)
Het klokprogramma werkt echter zelf op de kale KIM zonder toevoegingen. Dan dient men dit wel hexadecimaal in te tikken en wel vanaf locatie $0200 t/m $02CF volgens kolommen 3, 4 en 5 van de listing (0200 D8 A9 B5 8D FE 17 A9 02 8D ...).

Lijst

MICRO-WARE ASSEMBLER 65XX–1.0

```
0010: 0200                TIJD    ORG     $0200
0020:
0030: 0200                TIMKEN  *       $170F
0040: 0200                TIMTRE  *       $170E
0050:
0060: 0200                SCANS   *       $1F1F
0070:
0080: 0200                TIME    *       $0000
0090: 0200                DAGNO   *       $0003
0100: 0200                TIMER   *       $0005
0110: 0200                TIMTMP  *       $0008
0120: 0200                TIMEX   *       $0009
0130:
0140: 0200 D8             TEST    CLD
0150: 0201 A9 B5          LDAIM   INTERR
0160: 0203 8D FE 17       STA     $17FE
0170: 0206 A9 02          LDAIM   INTERR  /
0180: 0208 8D FF 17       STA     $17FF
0190: 020B A9 00          LDAIM   TEST
0200: 020D 8D FA 17       STA     $17FA
0210: 0210 A9 02          LDAIM   TEST    /
0220: 0212 8D FB 17       STA     $17FB
0230: 0215 A2 00          LDXIM   $00
0240: 0217 86 05          STX     TIMER
0250: 0219 86 06          STX     TIMER   +01
0260: 021B 86 07          STX     TIMER   +02
0270: 021D 86 09          STX     TIMEX
0280: 021F 86 0A          STX     TIMEX   +01
0290: 0221 CA             DEX
0300: 0222 8E 0F 17       STX     TIMKEN
0310: 0225 58             CLI
0320: 0226 20 1F 1F LOOP  JSR     SCANS
0330: 0229 20 3B 02       JSR     TIMKEP
0340: 022C A5 00          LDA     TIME
0350: 022E 85 FB          STA     $00FB
0360: 0230 A5 01          LDA     TIME    +01
0370: 0232 85 FA          STA     $00FA
0380: 0234 A5 02          LDA     TIME    +02
0390: 0236 85 F9          STA     $00F9
0400: 0238 4C 26 02       JMP     LOOP
0410:
0420: 023B A0 00    TIMKEP LDYIM  $00
0430: 023D 78             SEI
0440: 023E A5 0A          LDA     TIMEX   +01
0450: 0240 A6 09          LDX     TIMEX
0460: 0242 84 0A          STY     TIMEX   +01
0470: 0244 84 09          STY     TIMEX
0480: 0246 58             CLI
0490: 0247 86 08          STX     TIMTMP
0500: 0249 0A             ASLA
0510: 024A 26 08          ROL     TIMTMP
0520: 024C 0A             ASLA
0530: 024D 26 08          ROL     TIMTMP
0540: 024F 18             CLC
0550: 0250 65 06          ADC     TIMER   +01
0560: 0252 85 06          STA     TIMER   +01
```