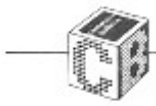


Gregorian Clock

Time from seconds to centuries

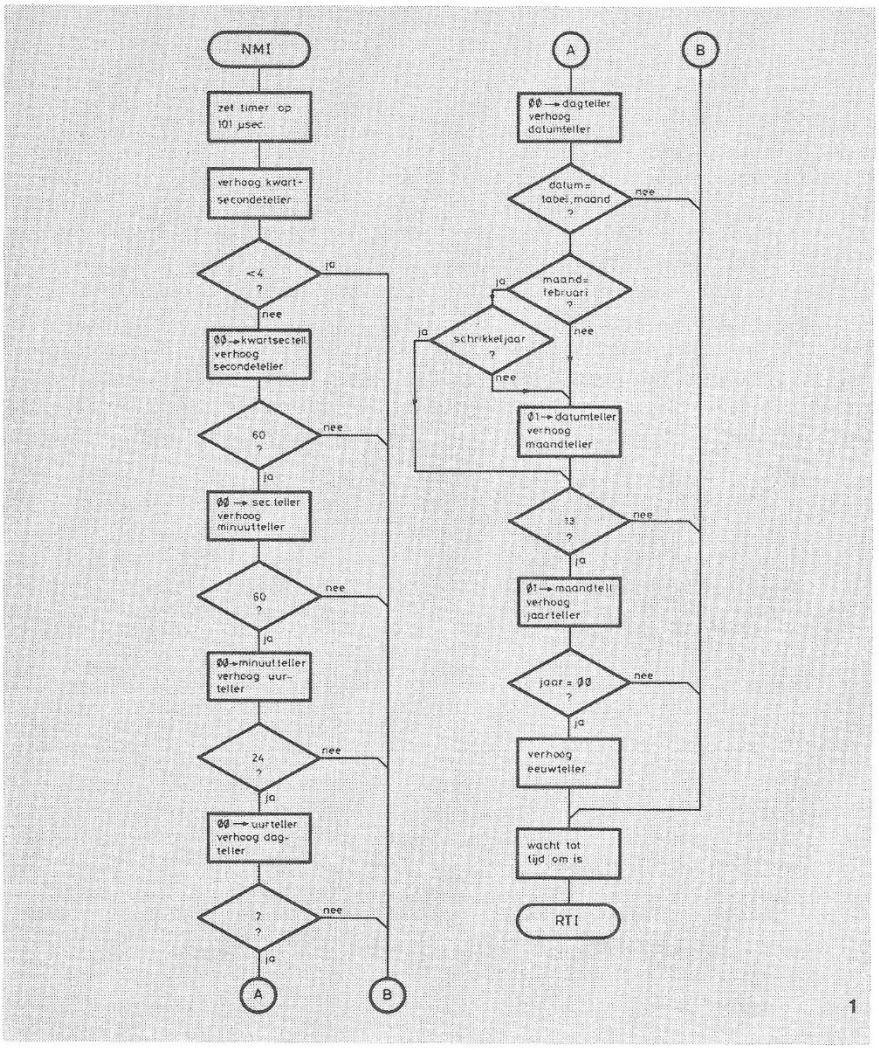
M. Dohmen
R. Koekoek

This clock program is very short and will certainly prove useful to many people. It counts in decimal and keeps track, in decimal, of quarter seconds, seconds, minutes, hours, day of the week, date, month, year and century. Leap years are taken into account!

Instructions for use

After the program has been typed in, the NMI vector must be set. (KIM: 7B at address 17FA and 03 at 17FB.) In addition, PB7 must be connected to NMI.Now, in decimal, the seconds value is set at address 0081, the minutes value at 0082, the hours value at 0083, the day of the week (0 through 6) at 0084, the date at 0085, the year at 0087, and the month at 0086, and the century at 0088. Once everything has been done correctly, "ST" is pressed. The clock should now be running. If you look at address 0081, you will see the seconds value. As long as "RS" is not pressed, the clock keeps running and is not disturbed by the execution of another program (such as the output of the monitor program). Because this clock has the advantage of continuing to work while another program is being executed, the possible applications are numerous. It is still possible to load a program from cassette into RAM memory without the clock being thrown off.

With a bit of hardware, it is possible to, say, have the radio turn on on 12 April 2001, or to have the garden sprinkler turn on on odd-numbered days of the month. With a small software trick it is possible to have the computer figure out when the 13th next falls on a Friday.



Flowcharts

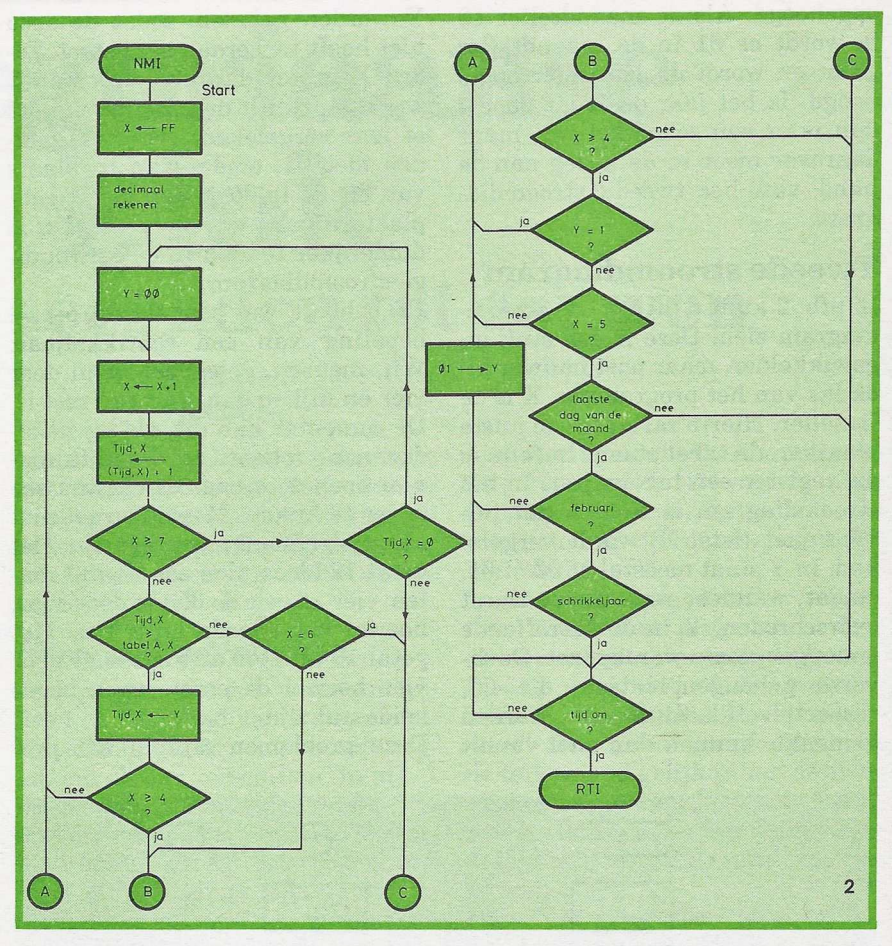
The program will be explained using 2 flowcharts, which differ considerably in complexity. Fig. 1 shows the simplified version of the flowchart. In reality the program does not consist of one long series of comparisons, but of a loop surrounded by various conditional jumps. These separately handle the various exceptions that time calculation brings with it.

The values against which the counter readings are compared are stored in a table. This table is interrupted by a small piece of program. This occurs between the value for the month of September and the value for the month of October. September is namely the 9th month (dec.) and October the 10th month (dec.). Because register X (which is the table pointer) counts in hexadecimal, it sees the number 10 (hex) instead of the number 16 (dec.), so that 6 positions are skipped. These six bytes are then put to good use.

As soon as an interrupt occurs, the timer (without interrupt) is set correctly to ensure that the interrupt routine always takes the same amount of time, regardless of the path the program follows. Thanks to the timer, the program now always takes about 101 µs. This duration also compensates for the time that the interrupt timer counts too little. After that, the quarter-second counter is incremented. As long as this is not four, the computer waits until the reserved time for the interrupt routine has elapsed. The routine that waits for the time to elapse begins at the label "end".

If the quarter-second counter does reach 4, the program resets it to zero and increments the

seconds counter. As long as this is not 60 (corresponding to the value in the table), the program jumps to "end". If the counter value equals the corresponding value from the table, this counter value is set to zero and the minutes counter is incremented.
The same applies to the hours counter, the day-of-the-week counter, and the year counter. The date and month counters are set to 01 at the moment the maximum value is exceeded.
The date counter requires some explanation. In the table, the maximum value listed for the date counter is 33. Because the date counter never actually reaches 33, we thereby prevent the date counter from being unnecessarily set to 01. As soon as the date counter has been incremented, this value is compared with the maximum value belonging to the relevant month.
Suppose the clock is now at 29 March. The computer then compares 29 with the maximum value for the month of March, which is 32. Here, 29 is less than 32, so nothing is amiss yet. As soon as the value in the date counter equals the corresponding value in table B, it is determined whether the comparison at that moment is being made against the maximum value for the month of February. If the comparison is indeed with February, it is then checked whether this is a leap year. Otherwise, 01 is placed in the date counter and the month counter is incremented. If the month counter reaches 13, 01 is placed in the month counter and the year counter is incremented. If the year is divisible by 4, then it is a leap year, but more on that in the explanation of the second flowchart.



Second flowchart
Fig. 2 shows this second flowchart. It is considerably more complicated, but corresponds more closely to the program's loop. X is the loop counter. It always holds the correct position in the table. In effect, this register is a table pointer. The flowchart shows that (something, X) is

repeatedly compared against something. Y usually holds either 00 or 01, because whenever a value is exceeded, Y is set at the relevant memory location. The various memory locations that need to be set to 00 or 01 respectively can then be quickly loaded from Y, so that the accumulator does not need to be disturbed. As soon as X becomes greater than or equal to 5 (meaning date, month, or year is being compared), Y is loaded with 01, because in that case 01 rather than 00 must be placed in the various memory locations. This is shown more clearly in the simple flowchart.
Some explanation is needed regarding the determination of a leap year. We humans divide the number by four and check whether there is a remainder. The computer can do this in the same way, if it is calculating in hexadecimal, by looking at the first two bits. But this does not work if it has to calculate in decimal. The number 12 (dec.) plus any multiple of four is also divisible by four, even though the second bit is 1. The number 10 (dec.) is not divisible by four even though the first two bits are both zero.
These problems are resolved in the program (starting at address 03CC) as follows. First, the first bit is shifted into the carry. If a carry occurs, the number is definitely not divisible by four, since it is then odd. If no carry occurs, the computer performs an AND with 09. The bits that remain used to form the number 12 (00001001 was 00010010: 12 dec.). If the number after this AND is still 09, the original number was 12 (plus some multiple of four) and thus divisible by 4.
If the number after this AND has become 00, the number before this AND was XXX0XX0X (binary), and whether the number is divisible by four then depends on the first bit. Since the carry flag was 00 before this AND was performed, the first bit was previously 00, and the number is therefore divisible by four.

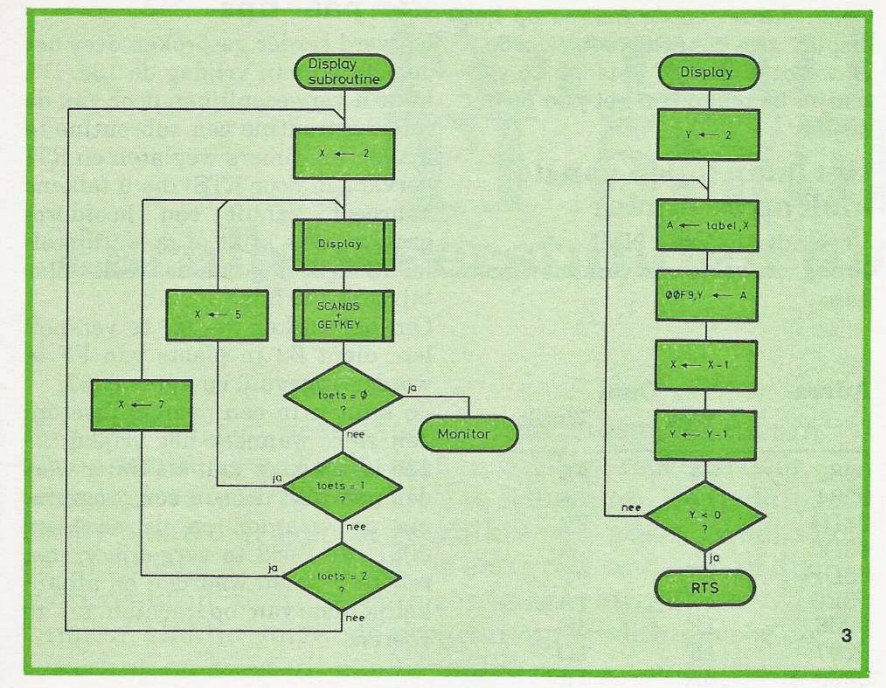
Operation of the program

As soon as an interrupt occurs, the program jumps to address 037B. Here, the values that are in the registers at that moment are first pushed onto the stack. Then the timer, which ensures that the routine always takes the same length of time, is set correctly, and the decimal flag is set. From this point on, only decimal arithmetic is used.
X is now set to FF, and this is where the actual calculation routine begins. Y is loaded with 00. The reason for this has already been explained. X is incremented and will be incremented again each time a new location needs to be changed. Next, the relevant memory location is incremented by one. X is compared with 07. This is to check whether the calculation routine has already run through 7 times. If this is the case, the program jumps to a routine that checks whether the year has become 00. If X is not 7, it is checked whether the value indicated by (0080, X) is equal to the value from (table, X). If these two values are equal, Y is placed in the relevant memory location. As long as X is not 4, this part of the program is executed as described. If these two values are not equal, X is compared with 06 (X now points to the month if X equals 06). If X equals 06, the computer exits, because no further changes need to be made and it was not the last month of the year.
If X does not equal 06 — in other words, if the program is not yet busy incrementing the month — it is checked whether X is greater than or equal to 04. If X lies between 00 and 04, or equals 00 or 04, there are no exceptions when incrementing the relevant memory locations. If X is not yet equal to 04, the program jumps to the main program. Next, it is checked whether Y equals 01. Y is only set to 01 once a month has passed. If a month has passed, the program has already gone past the point where exceptions are made, and this part of the program can be skipped.
If Y does not equal 01, we arrive at the exceptions. X is compared with 05. And if X equals 05, the computer is looking at the date. As you probably know, not every month has the same number of days. Moreover, for example, 0 December does not exist. First it must be determined which

month the computer is dealing with, and second, if the maximum value is exceeded, 01 must be placed in this memory location. Y is now loaded with the month and also serves as a table pointer. If, for example, it is 31 January, Y is loaded with 01, and via Y a pointer is also set to the first value in the table, where the number of days in the month is indicated.
The value in the table is compared with the date, and if these two values are not equal to each other, nothing is amiss. If these two values are equal, yet another problem arises. Is the comparison being made against the maximum value for the month of February (address 03BA:C0 02)? If the comparison is not against the maximum value for February, then Y is loaded with 01 and this value is placed in the date counter. This immediately resolves the second problem as well. If the comparison is indeed against the maximum value for February, it must be determined whether it is a leap year. (This routine begins at the label "schrik" [leap].) How this routine works has already been explained earlier.
Finally, at the end of the interrupt routine, the registers are retrieved from the stack, so that the computer can continue with the correct values in the registers at the point where it was interrupted.

Calibration
When we ran the program, there was nothing that needed adjusting, but if someone wants to run the program up to, say, the year 2000 without needing to make adjustments, they will need very precise measuring instruments, a very stable crystal, a very stable power supply, and a constant temperature in the computer room. Someone who is satisfied more quickly will, after some experimentation, adjust the value at address 0381.

Setting the clock
The values at addresses 0081 through 0089 can be changed while the clock is running, by typing the correct values in at those addresses.



Example of a readout routine for the clock (listing 2)
The flowchart is shown in fig. 3. In this program, register X is the pointer that determines what appears on the display. Initially X is loaded with 02, so that the hours, minutes, and seconds become visible. First, the subroutine is called that puts the values indicated by X on the display.

Then the subroutine "SCANDS" is called.|
Next, the value of the pressed key is retrieved. If the key value is 00, the program stops; it jumps directly to the monitor, and the clock simply keeps running (no reset is needed). If the key value is 1, X is loaded with 05 and the day of the week, date, and month are put on the display. If the key value is 2, X is loaded with 07 and the month and year appear on the display.

Running the clock on the Junior

Connect 1RQ to NMI.
Make the following changes:

Address	**Data**	
	Was →	Becomes
0383	04→	F4
0384	17 →	1A
03D8	07 →	F7
03D9	17 →	1A
03DF	0F →	FF
03E0	17 →	1A
0206	1F →	8E
0207	1F →	1D
0209	6A →	F9
020A	1F →	1D
0210	22 →	33

We mentioned earlier the detection of Friday the 13th. You can achieve this by turning the interrupt routine into a subroutine (omitting the timers and replacing RTI with RTS), which you then repeatedly call from a main program that checks whether the date counter is 13 and the day-of-the-week counter is, for example, 5.
To speed up the search further, 03 should be entered at address 0387 instead of FF.
This way you can see within a few milliseconds when it is Friday the 13th. The clock can be used as a timer by having a program compare the values at addresses 0081 through 0088 with values you determine yourself, and carry out commands depending on the result.

Lijst 1

```
        MICRO-WARE ASSEMBLER 65NX-1.0 PAGE 01

0010:
0020:                   ****************************************
0030:                   *                                      *
0040:                   *          GREGORIAANSE KLOK           *
0050:                   *              6502                    *
0060:                   *       AUTEURS: M. DOHMEN             *
0070:                   *                R. KOEKOEK            *
0080:                   *                                      *
0090:                   ****************************************
0100:
0110:
0120: 037B                      ORG   $037B
0130:
0140:
0150:                   ZERO-PAGE ADRESSEN
0160:
0170: 037B              KWSEC   *     $0080
0180: 037B              SEC     *     $0081
0190: 037B              MINUT   *     $0082
0200: 037B              UREN    *     $0083
0210: 037B              WEEKDG  *     $0084
0220: 037B              DATUM   *     $0085
0230: 037B              MAAND   *     $0086
0240: 037B              JAAR    *     $0087
0250: 037B              EEUW    *     $0088
0260: 037B              INH     *     $00F9
0270: 037B              ADRESL  *     $00FA
0280: 037B              ADRESH  *     $00FB
0290:
0300:                   TIMER
0310:
0320: 037B              TIMER   *     $1704
0330: 037B              TIMEH   *     $1707
0340: 037B              TIMEB   *     $170F
0350:
0360:                   SUBROUTINES UIT DE KIM-MONITOR
0370:
0380: 037B              MONITR  *     $1C22
0390: 037B              SCANDS  *     $1F1F
0400: 037B              GETKEY  *     $1F6A
0410:
0420:
0430:                   KLOK HOOFDPROGRAMMA
0440:
0450:
0460: 037B 48           START   PHA           ZET A OP DE STACK
0470: 037C 8A                   TXA
0480: 037D 48                   PHA           ZET X OP DE STACK
0490: 037E 98                   TYA
0500: 037F 48                   PHA           ZET Y OP DE STACK
0510: 0380 A9 65                LDAIM #65
0520: 0382 8D 04 17             STA   TIMER   TIMER 101 MICROSEC.
0530: 0385 F8                   SED           DECIMAAL REKENEN
0540: 0386 A2 FF                LDXIM #FF     LOOP TELLER
0550: 0388 A0 00        RETURN  LDYIM $00     WAARDEN VOOR 80-84
0560: 038A E8           BEGIN   INX
0570: 038B 18                   CLC           VERHOOG DE WAARDE DIE DOOR
0580: 038C B5 80                LDAZX KWSEC   (80 + X) WORDT AANGEWEZEN
0590: 038E 69 01                ADCIM #01     MET 1 (DECIMAAL)...
0600: 0390 95 80                STAZX KWSEC   EN VERGELIJK MET DE
0610: 0392 E0 07                CPXIM #07     BIJBEHORENDE TABELWAARDE
0620: 0394 B0 61                BCS   PRGR    KLEINER DAN MAX.:TERUG
0630: 0396 DD E7 03             CMPAX TABELA  VERGELIJK MET TABEL
0640: 0399 B0 06                BCS   VERDER
0650: 039B E0 06                CPXIM #06
0660: 039D F0 38                BEQ   END
0670: 039F D0 06                BNE   VERDTW  ONCONDITIONEEL
0680: 03A1 94 80        VERDER  STYZX KWSEC   KWSEC + X WORDT 0
0690: 03A3 E0 04                CPXIM #04     X WIJST NAAR WEEKDG
0700: 03A5 90 E3                BCC   BEGIN   X NIET GROTER, GEEN UITZONDERING
0710: 03A7 E0 04        VERDTW  CPXIM #04     X=4?
0720: 03A9 90 2C                BCC   END     NEEN, DAN NAAR HOOFDPROGRAMMA
0730: 03AB C0 01                CPYIM #01
0740: 03AD F0 DB                BEQ   BEGIN
0750: 03AF E0 05                CPXIM #05     VERGELIJKEN AANTAL DAGEN IN MAAND?
0760: 03B1 D0 D7                BNE   BEGIN   NEEN, TERUG NAAR HOOFDPROGRAMMA
0770: 03B3 A4 86                LDYZ  MAAND   WELKE MAAND?
0780: 03B5 D9 ED 03             CMPAY TABELB  IS DE MAAND VOL?
0790: 03B8 90 1D                BCC   END
0800: 03BA C0 02                CPYIM #02     MAAND FEBRUARI?
0810: 03BC F0 06                BEQ   SCHRIK  JA, DAN KAN ER EEN SCHRIKKELJAAR ZIJN
0820: 03BE A0 01        RESET   LDYIM #01     Y=01.0 IS GEEN DAG
0830: 03C0 84 85                STYZ  DATUM
0840: 03C2 D0 C6                BNE   BEGIN   ONCONDITIONEEL
0850: 03C4 C9 30        SCHRIK  CMPIM #30     WAS HET 29 FEBRUARI?
0860: 03C6 F0 F6                BEQ   RESET   JA, GEEN AANPASSING
0870: 03C8 A5 87                LDAZ  JAAR    NEEM JAAR
0880: 03CA EA                   NOP
0890: 03CB EA                   NOP
0900: 03CC 4A                   LSRA          SCHUIF BIT 0 IN CARRY
0910: 03CD B0 EF                BCS   RESET   ONEVEN, GEEN SCHRIKKELJAAR
0920: 03CF 29 09                ANDIM #09     MASKEER
0930: 03D1 F0 04                BEQ   END     00, GEEN SCHRIKKELJAAR
0940: 03D3 C9 09                CMPIM #09     ZIJN ZE "1", DAN
0950: 03D5 D0 E7                BNE   RESET   SCHRIKKELJAAR
0960: 03D7 2C 07 17     END     BIT   TIMEH   WACHT TOT DE
0970: 03DA 10 FB                BPL   END     TIJD OM IS
0980: 03DC A9 F4                LDAIM #F4
0990: 03DE 8D 0F 17             STA   TIMEB
1000: 03E1 68                   PLA

0440: 03E2 A8                   TAY           ZET DE STACK
0450: 03E3 68                   PLA           WEER GOED
0460: 03E4 AA                   TAX           EN HERSTEL DE WAARDEN
0470: 03E5 68                   PLA           VAN A, X EN Y
0480: 03E6 40                   RTI           TERUG NAAR HOOFDPROGRAMMA
0490: 03E7 04           TABELA  =     $04
0500: 03E8 60                   =     $60     SECONDEN
0510: 03E9 60                   =     $60     MINUTEN
0520: 03EA 24                   =     $24     UREN
0530: 03EB 07                   =     $07     DAGEN
0540: 03EC 33                   =     $33     DAGEN IN EEN MAAND
0550: 03ED 13           TABELB  =     $13     MAANDEN
0560: 03EE 32                   =     $32     JANUARI
0570: 03EF 29                   =     $29     FEBRUARI
0580: 03F0 32                   =     $32     MAART
0590: 03F1 31                   =     $31     APRIL
0600: 03F2 32                   =     $32     MEI
0610: 03F3 31                   =     $31     JUNI
0620: 03F4 32                   =     $32     JULI
0630: 03F5 32                   =     $32     AUGUSTUS
0640: 03F6 31                   =     $31     SEPTEMBER
0650: 03F7 B5 80        PRGR    LDAZX KWSEC
0660: 03F9 F0 8D                BEQ   RETURN
0670: 03FB D0 DA                BNE   END
0680: 03FD 32                   =     $32     OKTOBER
0690: 03FE 31                   =     $31     NOVEMBER
0700: 03FF 32                   =     $32     DECEMBER
```

Lijst 2

```
0320:
0330:                   VOORBEELD VAN EEN PROGRAMMA
0340:                   OM DE KLOK UIT TE LEZEN
0350:
0360: 0200                      ORG   $0200
0370:
0380: 0200 A2 02        RUN     LDXIM #02     HAAL DE TIJD OP
0390: 0202 20 23 02     TERUG   JSR   DISPL   VUL DE UITLEZING
0400: 0205 20 1F 1F             JSR   SCANDS  LICHT DE UITLEZING OP
0410: 0208 20 6A 1F             JSR   GETKEY  HAAL DE INGEDRUKTE
0420: 020B C9 00                CMPIM #00     TOETS "0" INGEDRUKT?
0430: 020D D0 05                BNE   GEENN   NEEN, VERDER
0440: 020F 4C 22 1C             JMP   MONITR  JA, NAAR MONITOR
0450: 0212 C9 01        GEENN   CMPIM #01     TOETS "1" INGEDRUKT?
0460: 0214 D0 05                BNE   GEENE   NEEN, VERDER
0470: 0216 A2 05                LDXIM #05     JA, HAAL DAG EN MAAND
0480: 0218 4C 02 02             JMP   TERUG   EN DISPLAY
0490: 021B C9 02        GEENE   CMPIM #02     TOETS "2" INGEDRUKT?
0500: 021D D0 E1                BNE   RUN     NEEN, OPNIEUW
0510: 021F A2 07                LDXIM #07     JA, HAAL DATUM
0520: 0221 D0 DF                BNE   TERUG   ONCONDITIONEEL
0530: 0223 A0 02        DISPL   LDYIM #02     KIES DISPLAY
0540: 0225 B5 81        DIS     LDAZX SEC     HAAL INHOUD
0550: 0227 99 F9 00             STAAY INH     ZET DIT IN HET DISPLAY
0560: 022A CA                   DEX
0570: 022B 88                   DEY
0580: 022C 10 F7                BPL   DIS
0590: 022E 60                   RTS
```