S D K - 85

E N

D A L L A S D S 1216 C

INHOUDSOPGAVE

SAMMENVATING

De SDK-85 is een micro-computer die door zijn simpele opbouw zeer geschikt is voor een introductie in micro-computers.

Het programmeren van de SDK-85 met hexadecimale code is zeker even wennen, en niet altijd even handig, toch is het programmeren op deze wijze zeer leerzaam.

Dankzij de I/O-poorten van de SDK-85 kan de programmeur met de "buitenwereld" communiceren. Daarvoor moeten de I/O-poorten wel eerst geïnitialiseerd worden. Deze I/O-poorten worden gebruikt om met de module van een LCD-display te communiceren. Dat wil zeggen het verzenden van data naar de LCD-module.

Het verzenden van deze data is ingewikkelder dan op het eerste moment gedacht, het eenvoudig verzenden van de karakters is niet genoeg. Voordat de LCD-module karakterdata kan ontvangen moet de LCD-module eerst geïnitialiseerd worden.

De LCD-display wordt gebruikt om de tijd van de DALLAS DS1216 C REAL TIME klok zichtbaar te maken voor de gebruiker. Dit programmeerbaar IC is in staat om de tijd te tellen, en dat met redelijke nauwkeurigheid. De DS1216 C is dankzij in het IC aanwezige energiecel nonvolatile. Dit betekent dat de klok gewoon door blijft tellen ook als de voedingsspanning uitgeschakeld is.

Voordat met de klok gecommuniceerd kan worden moet een code-sleutel naar het IC gestuurd worden, dit om aan de klok kenbaar te maken dat er met hem gecommuniceerd gaat worden. Als dit gebeurt is kan de inhoud van de klok "gelezen" of "beschreven" worden.

Aangezien de code van de klok in BCD is en de code van de LCD-module in ASCII is, zal de BCD-code omgezet moeten worden in ASCII-code.

HOOFDSTUK 1 INLEIDING

Eèn van mijn eerste opdrachten, tijdens de eerste stage bij VENEMA AUTOMATION, was het programmeren van een REAL TIME IC. Deze DS 1216 C REAL TIME klok werd op een microcomputer van INTEL aangesloten, de SDK-85.

Het programma om de klok uit te lezen werd dan ook op de SDK-85 geschreven. Als eerste zal dieper op deze SDK-85 in gegaan worden, zaols het toetsen bord en programmerwijze.

Om de inhoud van de DS1216 C zichtbaar te maken werd gebruik maakt van een display. In hoofdstuk drie zal dit element aan de orde komen. Het zal blijken dat eenvoudig wat wegsturen van een paar signalen niet voldoende is.

Tenslotte zal natuurlijk de DS 1216 C zelf aan de orde komen. Zowel het lezen van deze klok alsook het zetten van klok komen aan de orde met de bijbehorende programma's.

HOOFDSTUK 2 SDK-85 [INTEL system kit]

In het begin van mijn eerste stage, bij VENEMA AUTOMATION heb ik me bezig gehouden met de SDK-85. Deze micro-computer, van de Amerikaanse fabrikant INTEL, is op èèn printplaat uitgevoerd en geheel open. Door deze wijze van uitvoering is een goed overzicht op de componenten mogelijk.

Deze micro-computer wordt nog met hexadecimale code geprogrammeerd, een inmiddels achterhaalde wijze van programmeren. [Bijlage 1 toont een overzicht van assambly-code en de hex. code] Voor wie nog nooit op deze wijze geprogrammeerd heeft is het zeker wennen. De eerste gedachte bij het zien van de SDK-85 is bij de meeste stagieres moet ik hiermee werken! Toch is de SDK-85 ideaal om meer te weten te komen over de opbouw, de architectuur van een micro-computer. Door de simpele opbouw is er erg makelijk aan te meten wat de inzicht in de werking vergroot.

Door het gebruik van de hexadecimale programmingsmethode krijg je een beter inzicht in wat de micro-computer met je geschreven programma doet. De hexadecimale codes worden op een geheugenadres gezet. De gebruiker heeft zelf de keuze op welk geheugenplaats hij of zij, zijn of haar programma zet. Er zijn natuurlijk grenzen, immers de SDK-85 heeft een beperkt geheugengebied.

Het programmeren met hexadecimale code betekent echter ook dat de programmeur zijn of haar programma vantevoren goed moet controlleren. Het opzoeken van en verbeteren van fouten als het programma eenmaal is ingevoerd, kan veel tijd vergen. Op de SDK-85 is namelijk geen INSERT functie aanwezig. Dit betekent dat een eenmaal verkeert ingevoerde code op een bepaald geheugen adres, alleen verandert kan worden door eerst het betreffende geheugenadres op te vragen en daarna de code te veranderen.

2-1 HET TOETSENBORD

Om de hexadecimale code te kunnen intoetsen is op de print een toetsenbord aangebracht. Naast het intoetsen van de programmacode, hebben deze toetsen nog een aantal andere functie's

Ten eerste is daar de RESET-toets. Deze toets wordt gebruikt om een eenmaal draaïend programma te onderbreken, maar ook als het vast komt te zitten kan deze toets gebruikt worden. Tevens is een toets aanwezig om de inhoud van de register's van de SDK-85 te bekijken. Dit is de REG VIEW-toets. Verder is er een toets om de inhoud van een geheugenadres te bekijken, de zogenaamde MEMORY-toets. Een andere toets van belang aanwezig op het toetsenbord, is de SINGLE STEP-toets. Bij het indrukken van deze toets loopt de micro-computer stap voor stap door het programma. Tenslotte is er natuurlijk een toets aanwezig om het ingevoerde programma te runnen, oftewel uit te voeren.

2-2 DE I/O-POORTEN

Wat de SDK-85 echt interesant maakt zijn zijn I/O-poorten (Input en Output poorten). Dankzij deze poorten is de programmeur in staat om met de "buitenwereld" te communiceren. Op deze manier kan bijvoorbeeld een LCD-display aangestuurd worden, maar er zijn natuurlijk tal van andere mogelijkheden.

Voordat de microcomputer met de buitenwereld kan communiceren moeten de I/O poorten eerst geïnitialiseerd worden. De microcomputer moet weten via welke poort hij communiceert. Tevens moet de status van de poort bepaald worden, INPUT of OUTPUT. Bijlage 2 toont een overzicht van de poorten en hun betekenis. Uit bijlage 2 blijkt dat poort 28 (hexadecimale waarde) de status van de poorten 29 en 2A bepaalt. Door aan poort 28 een bepaalde te geven worden poorten 29 en 2A INPUT of OUTPUT. Omdat we een LCD display willen aansturen moeten deze poorten OUTPUT's worden.

Hieronder is een programmadeel gegeven waarmee de poorten geïnitialiseerd kunnen worden. Eerst wordt Het STATUS-register geselecteerd, daarna wordt de status van de poorten bepaald.

```
LXI SP,20C2H      ;initialiseer STACK pointer
MVI A,03          ;zet waarde 3 in A-register
OUT 28            ;STATUS van de poort is OUTPUT
```

Nu is het mogelijk DATA te versturen via de poorten 29 en 2A. Zoals later zal blijken zal hiervan nog veel gebruik worden gemaakt.

HOOFDSTUK 3 DE LCD-DISPLAY

De LCD-display heeft veel toepassingen, zoals in de telecommunicatie, medische instrumenten etc. Bij VENEMA AUTOMATION wordt de display gebruikt om de gebruiker in beknopte bewoording aan te geven in welke toestand zich een apparaat bevindt.
Bij mijn eerste opdracht werd de display gebruikt om de gegevens van de REAL TIME klok zichtbaar te maken.

3-1 DE WERKING VAN DE DISPLAY

Wie denkt dat het versturen van data via de I/O poorten van de SDK-85 naar de display voldoende is, zit fout. De display moet eerst geïnitialiseerd worden. Dat wil zeggen we moeten aan de LCD-module eerst een aantal zaken mededelen, voordat tekst naar de display geschreven kan worden.

De LCD-module beschikt over een INSTRUCTIE-register [IR] en een DATA-register [DR]. Afhankelijk van wat naar de display verstuurt moet worden, moet het IR of het DR geselecteerd worden.
Onder instructie's wordt b.v. verstaan: CLEAR DISPLAY of RETURN HOME. Data is datgene wat uiteindelijk op de display leesbaar zal zijn. Om het IR of DR te selecteren zijn twee signalen op de LCD-module aanwezig, te weten RS [Register Select] en R/W [Read/Write]. De betekenis van beide signalen is in tabel 1 gegeven.

Tabel 1 : Betekenis RS en R/W signalen

RS	R/W	FUNCTIE
0	0	Instructie register schrijf MPU -> LCD-module
0	1	Instructie register lees MPU -> LCD-module
1	0	Data register schrijf MPU -> LCD-module
1	1	Data register lees MPU -> LCD-module

Uit het tabel blijkt welke signalen verstuurt moeten worden, om of IR of DR te selecteren.

Zoals al eerder opgemerkt moeten eerst een aantal zaken aan de display medegedeeld worden. Zo moet de display weten met hoeveel bit's er met de display gecommuniceerd wordt, vier of acht bit's. Commando's als CLEAR DISPLAY en RETURN HOME moeten ook gegeven worden voordat data verstuurt kan worden.

Dit zijn allemaal instructie's aan de LCD-module, en voordat deze instructie's naar de LCD-module gestuurt worden, moet eerst het IR geselcteerd worden. Hieronder staat het programmadeel om het IR te selecteren:

```
INSTR  MVI  A,01      ;Het sinaal om het IR te selecteren
       OUT  2A        ;Verstuur dit via I/O-poort 2A
       CALL DELAY
       MOV  A,C       ;Instructie in A-register
       OUT  29        ;Verstuur dit via I/O-poort 29
       MVI  A,00      ;Signaal voor IR
       OUT  2A
       CALL DELAY
       RET
```

Om data te versturen moet eerst het DR geslecteerd worden. Hieronder staat een programmadeel waarmee dit bereikt kan worden:

```
DATA   MVI  A,03      ;Het signaal om DR te selecteren
       OUT  2A
       CALL DELAY
       MOV  A,C
       OUT  29
       MVI  A,02
       CALL DELAY
       RET
```

Bijlage 3 toont een aantal instructie-commando's met hun code. De LCD-module werkt met de ASCII-code, waarvan bijlage 4 de karakters toont met hun bijbehorende hexadecimale code.
Het initialisatie programma ziet er nu als volgd uit:

```
INIT_LCD  MVI  A,03     ;Status van I/O-poort is OUTPUT
          OUT  28
          MVI  C,38     ;Interface lengte is 8 bit's, 2 lijnen
          CALL INSTR    ;Het deelprogramma INSTR wordt aangeroepen
          MVI  C,01     ;Clear display
          CALL INSTR
          MVI  C,02     ;Return home
          CALL INSTR
          MVI  C,0F     ;Display is ON,Cursor is displayed
          CALL INSTR
          RET
```

Na het uitvoeren van het deelprogramma INIT_LCD [en ook INSTR] kan door middel van het deelprogramma DATA gegevens naar De display gestuurd worden. N.B.: elke keer als DATA aangeroepen wordt kan maar èèn karakter op de display zichtbaar gemaakt worden. De cursor wordt hierna automatisch èèn positie naar rechts verplaats, door het opnieuw aanroepen van DATA kan de volgende karakter op de display zichtbaar gemaakt worden. Stel dat het woord "DATA" op de display zichtbaar gemaakt moet worden. De volgorde van zenden is dan:

D	eerste letter	Hexadecimale code: 44 [zie bijlage 5]
A	tweede letter	Hexadecimale code: 41
T	derde letter	Hexadecimale code: 54
A	vierde letter	Hexadecimale code: 41

HOOFDSTUK 4 DS1216 C DALLAS SMARTWATCH

De Smartwatch is een produkt van de Amerikaanse firma DALLAS. Het is een programmeerbare REAL TIME IC. De Smartwatch is in staat om zelfs honderdsten van één seconde te tellen. Verder kunnen dag, datum, maand en jaar ingesteld worden. Dankzij een in het IC aanwezige lithium energiecel, is de klok nonvolatile. Dat wil zeggen, dat de klok ook nadat de voedingsspanning is uitgeschakeld, door blijft werken.

4-1 DE WERKING

De Smartwatch werkt samen met een stuk RAM-geheugen [ook nonvolatile]. Communicatie met de klok gaat via dit RAM-geheugen.

Communicatie met de REAL TIME klok is gebaseert op het versturen van een uniek bitpatroon, en wel serieel.
Het versturen van dit uniek bitpatroon is nodig omdat communicatie in principe via elk geheugenadres mogelijk is. Om nu aan de Smartwatch duidelijk te maken dat met hem gecommuniceerd gaat worden is het versturen van het uniek bitpatroon nodig. Elke keer dat klok "gelezen" of "beschreven" wordt moet het bitpatroon verstuurt worden. Als het herkeningsregister, wat zich in de Smartwatch bevindt, het bitpatroon herkent heeft kan de klok "gelezen" of "beschreven" worden. Bijlage 6 toont de opbouw van het uniek bitpatroon.

Het versturen van het bitpatroon op serïeele basis vereist een apart stukje programma, wat hieronder is gegeven:

```
SEND  MVI  B,08       ;Byte teller wordt geset
      LXI  H,2A00     ;Start adres waar bitpatroon zich bevindt
LOOP1 MOV  A,M        ;Inhoud geh.adres in A-register
      MVI  E,08       ;Bit teller wordt geset
LOOP2 STA  4000       ;Inhoud A-register in Smartwatch
      RRC
      DCR  E
      JNZ  LOOP2
      INX  H
      DCR  B
      JNZ  LOOP1
      RET
```

Het geheugenadres 4000 is gekozen als communicatieadres met de Smartwatch.

4-2 HET PROGRAMMEREN VAN DE KLOK

Als het bitpatroon verzonden is en goed ontvangen door de Smartwatch, kan de klok gelezen of beschreven worden. Als ervan uitgegaan wordt dat de klok net de fabriek verlaten heeft, moet de klok eerst ingesteld worden op onze tijd. Tevens moet de oscillator, door de fabrikant uit gezet, opgestart worden. Zonder deze oscillator werkt de REAL TIME klok namelijk niet!

De Smartwatch beschikt over acht register's die allemaal een gedeelte van de tijd voor hun rekening nemen. Bijlage 7 toont precies welke informatie de registers bevatten en in welke code. Voor het zetten van de tijd, datum etc. kunnen registers één en twee zonder meer overgeslagen worden. Het zetten van de 0,01 sec. en de sec. heeft natuurlijk weinig zin. Het zetten van de andere registers heeft meer zin.

Zoals uit bijlage 7 blijkt bevatten sommige registers behalve informatie over tijd, ook informatie over een aantal andere zaken, zoals in register 3. Dit register bevat naast de uren ook informatie over de AM/PM status, indien voor de 12 uurs mode gekozen wordt. Register 4 bevat naast de dagen nog informatie over de RESET en OSCillator. De RESET moet '1' zijn en OSC moet '0' gemaakt worden. In een aantal registers zijn bits met '0' als inhoud. Bij het setten van de klok mag hier zowel een '1' als een '0' geschreven worden.

Hieronder is het programmadeel gegeven waarmee data naar de klok gestuurd kan worden:

```
SET    MVI  B,08
       LXI  H,2600      ;beginadres met data voor de klok
LOOP1  MOV  A,M         ;Schuif inhoud geh.adr. in A-reg
       MVI  E,08
LOOP2  STA  4000        ;Store inhoud A-register op geh.adr. 4000
       RRC
       DCR  E
       JNZ  LOOP2
       INX  H
       DCR  B
       JNZ  LOOP1
       RET
```

Uit bovenstaand programma blijkt dat het verzenden van data naar de Smartwatch hetzelfde is als het verzenden van het uniek bit-patroon. Alleen het beginadres van het buffer waar vanaf de in-formatie gehaald wordt verschild.

4-3 HET UITLEZEN VAN DE KLOK

Als de klok geset is met de juiste informatie, de juiste tijd dus, kan de klok uitgelezen worden. Om de tijd voor de gebruiker zichtbaar te maken, zeten we de informatie van de klok op de display van de LCD.

Als eerste moeten de registers van de klok gelezen worden. Dit gebeurt serïeel via het geheugenadres 4000. Eerst moet echter het bitpatroon weer verzonden worden, hiervoor zorgt het programmadeel SEND. Nadat dit gedaan is kunnen we de informatie van de registers opvragen, met onderstaand programma

```
BYTE  LXI  H,2800      ;bgeinadres van het buffer
      MVI  B,08
LOOP  PUSH B           ;zet B-register op stack
      CALL BIT         ;subroutine voor ophalen 8 bit's
      MOV  M,A
      POP  B           ;Haal B-register van Stack
      INX  H
      DCR  B
      RZ
      JMP  LOOP
```

De subroutine BIT zorgt voor het op juiste wijze ophalen van een byte.

```
BIT   MVI  B,08
      MVI  C,00
LOOP  LDA  4000
      ANI  01
      ORA  C
      RRC
      MOV  C,A
      JNZ  LOOP
      MOV  A,C
      RET
```

Echter, de informatie die de registers bevat is in BCD-code terwijl de display met ASCII-code werkt. Dat betekent dat de informatie van de registers eerst omgezet moet worden van BCD naar ASCII. Om dit te bereiken is het volgende programma nodig.

```
OMZ  ANI  F0     ;Maskeer MSB van byte
     RRC         ;bit 7 wordt in totaal
     RRC         ;vier positie's naar
     RRC         ;rechts verschoven
     RRC
     ADI  30     ;Hier wordt hex. 30 bij opgeteld
     MOV  C,A    ;Het Meest significante gedeelte in C-register
     MOV  A,M
     ANI  0F     ;Maskeer LSB
     ADI  30     ;Minst significante gedeelte in A-register
     RET
```

Van èèn byte in BCD-code wordt dus twee bytes in ASCII-code gemaakt, het meest significante gedeelte in het C-register. Het minst significante gedeelte in het A-register.

Als De klokregisters uitgelezen zijn, en de code is omgezet

kan de data naar de display gestuurd worden. Dit proces wordt constant herhaald zodat de klok REAL TIME uitgelezen wordt.

HOOFDSTUK 5 SLOT

Het programmeren met hexadecimale code van een micro-computer is inmiddels achterhaald. Toch is het programmeren op deze wijze leerzaam, zeker om meer te weten te komen over de micro-computer zelf. Tegenwoordig programmeert men met een editor waar de code als karakter[zie bijlage 1] wordt ingevoerd. Programmeren op deze wijze bespaart veel tijd, in het bedrijfsleven een belangrijk aspect.

Het gebruik van de LCD-module kan een apparaat een stuk gebruiksvriendelijker maken, wat een efficienter gebruik mogelijk maakt. Het aansturen is een stuk eenvoudiger dan bijvoorbeeld een monitor, en tevens prijsgunstiger.

Display's zijn tegenwoordig in alle vormen en maten te verkrijgen ook display's met grafische mogelijkheden zijn leverbaar.

Het grootte voordeel van de DALLAS DS1216 C is dat het later in een bestaande schakeling kan worden ingebouwd, andere REAL TIME klokken bieden dit voordeel niet. Daarnaast wordt deze klok samen met een non-volatile RAM geheugen gebruikt. Hierdoor is het mogelijk grootte delen van het programma, om de klok te lezen, in dit geheugen te zetten. Een nadeel is het versturen van het uniek bitpatroon wat de snelheid vertraagt en serieel gebaseerde communicatie, wat extra programma's vereist.

Bijlagen

BIJLAGE 1 : Overzicht assambly-code en de bijbehorende hexadecimale code.

BIJLAGE 2 : Overzicht I/O-poorten van de SDK-85.

BIJLAGE 3 : Instructie's voor LCD-module.

BIJLAGE 4 : ASCII-karakterset voor display.

Bijlage 6 : Uniek bitpatroon DS1216 C.

BIJLAGE 7 : Overzicht Smartwatch registers.

8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE

Table 5-2

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	–
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	–
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	–	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	–	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	–	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	–
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	–	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	–
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	–	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity.

Adr = 16-bit address.

Bylage 2

PORT	FUNCTION
00	Monitor ROM PORT A
01	Monitor ROM PORT B
02	Monitor ROM PORT A Data Direction Register
03	Monitor ROM PORT B Data Direction Register
08	Expansion ROM PORT A
09	Expansion ROM PORT B
0A	Expansion ROM PORT A Data Direction Register
0B	Expansion ROM PORT B Data Direction Register
20	BASIC RAM COMMAND/STATUS Register
21	BASIC RAM PORT A
22	BASIC RAM PORT B
23	BASIC RAM PORT C
24	BASIC RAM Low Order Byte of Timer Count
25	BASIC RAM High Order Byte of Timer Count
28	EXPANSION RAM COMMAND/STATUS Register
29	EXPANSION RAM PORT A
2A	EXPANSION RAM PORT B
2B	EXPANSION RAM PORT C
2C	EXPANSION RAM Low Order Byte of Timer Count
2D	EXPANSION RAM High Order Byte of Timer Count

Bylage 3

INSTRUCTION DESCRIPTION

• OUTLINE

Two registers of the HD44780, the Instruction Register (IR) and the Data Register (DR) only can be controlled by MPU directly. Control information is temporarily stored in these registers, prior to internal operation start, to allow interface to various types of MPUs which operate in different speeds from HD44780 internal operation or to allow interface to peripheral control ICs. The HD44780 internal operation is determined by signals sent from the MPU, these signals including register selection signals (RS), read/write signals (R/W) and data bus signals (DB_0 ~ DB_7), are called instructions in this paragraph. Table on page 10 shows the instructions and the execution time of the instructions. Details are explained in the following sections. The instructions can be divided into the following 4 types:

(1) Instructions that designate the HD44780 functions such as display format, data length, etc.
(2) Instructions that give internal RAM addresses.
(3) Instructions that perform data transfer with internal RAM.
(4) Other instructions

In the normal use, instructions of category (3), which sends display data, is used most frequently. However, since the HD44780 internal RAM addresses are configured to be automatically incremented (or decremented) by +1 after each data write, MPU program load is lessened. Especially, display shift is performed concurrently with display data write, and this enables the user to develop systems with minimum time and maximum efficiency of programming. When an instruction is being executed (during internal operation), the busy flag DB_7 is active high. This must be monitored when high speed operation is planned (≈50KHz).

• CLEAR DISPLAY

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	0	0	0	0	1

Writes space code "20" (hexadecimal) into all the DD RAM addresses. The cursor returns to Address 0 (A_{dd} = "80") and display, if it has been shifted, returns to the original position. In other words, display disappears and the cursor goes to the left edge of the display (the first line if 2 lines are displayed).

• RETURN HOME

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	0	0	0	1	*

*(Don't Care)

Returns the cursor to Address 0 (A_{DD} = "80") and display, if it has been shifted, to the original position. The DD RAM contents remain unchanged.

• ENTRY MODE SET

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increments (I/D = 1) or decrements (I/D = 0) the DD RAM address by one upon writing into or reading from the DD RAM a character code. The cursor moves to the right when incremented by one. The same applies to writing and reading of CG RAM.

S: Shifts the entire display to either the right or the left when S is 1; to the left when I/D = 1 and to the right when I/D = 0. Therefore, the cursor looks as if it stood still with the display only moved. Display is not shifted when reading from the DD RAM. Display is not shifted when S = 0.

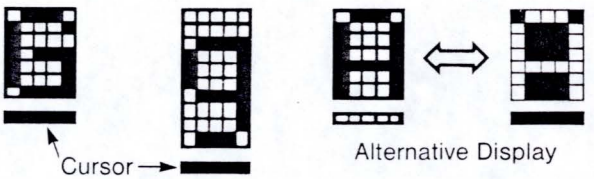
• DISPLAY ON/OFF CONTROL

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	0	1	D	C	B

D: Display is turned ON when D = 1 and OFF when D = 0. When display is turned off due to D = 0, the display data remains in the DD RAM and it can be displayed immediately by setting D = 1.

C: The cursor is displayed when C = 1 and not displayed when C = 0. Even if the cursor disappears, function of I/D, etc. does not change during display data write. The cursor is displayed using 5 dots in the 8th lines when the 5×7 dot character font is selected and in the 11th line when 5×10 dot character font is selected.

B: The character residing at the cursor position blinks when B = 1. The blink is done by switching between all the black dots and display characters at 0.4 second interval. The cursor and the blink can be set concurrently.



(a) Cursor Display Example

(b) Blink Display Example

• CURSOR OR DISPLAY SHIFT

	RS	R/W	DB_7							DB_0
Code	0	0	0	0	0	1	S/C	R/L	*	*

*(Don't Care)

Shifts the cursor position or display to the right and the left without writing or reading the display data. This function is used for correction or search of display.

S/C R/L

0 0 Shifts the cursor position to the left. (AC is decremented by one.)
0 1 Shifts the cursor position to the right. (AC is incremented by one.)
1 0 Shifts the entire display to the left. The corsor follows the display shift.
1 1 Shifts the entire display to the right. The cursor follows the display shift.

Bijlage 3

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	1	DL	N	F	*	*

*(Don't Care)

DL: Sets interface data length. Data is sent or received in 8 bit length ($DB_7 \sim DB_0$) when DL = 1 and 4 bit length ($DB_7 \sim DB_4$) when DL = 0. When 4 bit length is selected, data must be sent or received twice.
N: Sets number of display lines.
F: Sets character font.

N	F	No. of Display Lines	Character Font	Duty Facotr	Remarks
0	0	1	5×7 dots	1/8	
0	1	1	5×10 dots	1/11	
1	*	2	5×7 dots	1/16	Cannot display 2 lines with 5×10 dot character font.

*(Don't Care)

• SET CG RAM ADDRESS

	RS	R/W	DB_7						DB_0
Code	0	0	0	1	A	A	A	A	A	A

← Higher Order Bits — Lower Order Bits →

Sets the CG RAM address in a binary number of AAAAAA to the address counter, and data is written or read from the MPU related to the CG RAM after this.

• SET DD RAM ADDRESS

	RS	R/W	DB_7						DB_0
Code	0	0	1	A_N	A	A	A	A	A	A

← Higher Order Bits — Lower Order Bits →

Sets the DD RAM address in a binary number of A_NAAAAAA to the address counter, and data is written or read from the MPU related to the DD RAM after this. However, when N = 0 (1-line display), A_NAAAAAA is "00"→"4F" (hexadecimal). When N = 1 (2-line display), A_NAAAAAA is "00"→"27" (hexadecimal) for the first line, and "40"→"67" (hexadecimal) for the second line.

• READ BUSY FLAG AND ADDRESS

	RS	R/W	DB-							DB_0
Code	0	1	BF	A	A	A	A	A	A	A

← Higher Order Bits — Lower Order Bits →

Reads Busy Flag (BF) that indicates the system is internally operating on an instruction received before. When BF = 1, it indicates that internal operation is going on and the next instruction is not accepted until BF is set to "0." Check the BF status before the next write operation. At the same time, this value of the address counter expressed in a binary number of AAAAAA. The address counter is used by both of CG and DD RAM address, and its value is determined by the previous instruction. Address contents are those of CG RAM and DD RAM previously shown.

	RS	R/W	DB_7							DB_0
Code	1	0	D	D	D	D	D	D	D	D

← Higher Order Bits — Lower Order Bits →

Writes binary 8 bit data DDDDDDDD to the CG or the DD RAM. Whether the CG or the DD RAM is to be written is determined by the previous designation (CG RAM address setting or DD RAM address setting). After write, the address is automatically incremented or decremented by one according to entry mode. Display shift also follows the entry mode.

• READ DATA FROM CG OR DD RAM

	RS	R/W	DB_7							DB_0
Code	1	1	D	D	D	D	D	D	D	D

← Higher Order Bits — Lower Order Bits →

Reads binary 8 bit data DDDDDDDD from the CG or the DD RAM. Whether the CG RAM or the DD RAM is to be read is determined by the previous designation. *Prior to inputting this read instruction, either the CG RAM address set instruction or the DD RAM address set instruction must be executed. If it is not done, the first read data becomes invalid, and data of the next address is read normally from the second read.* After read, the address is automatically incremented or decremented by one according to the entry mode. However, display shift is not performed regardless of entry mode types.

✹ FONT TABLE

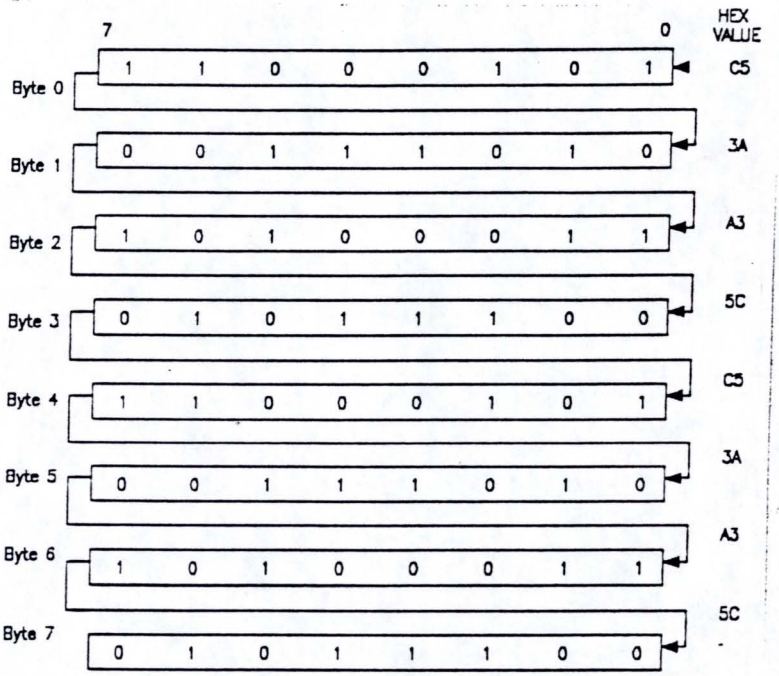
Bilage 4

UPPER 4 BIT HEXADECIMAL

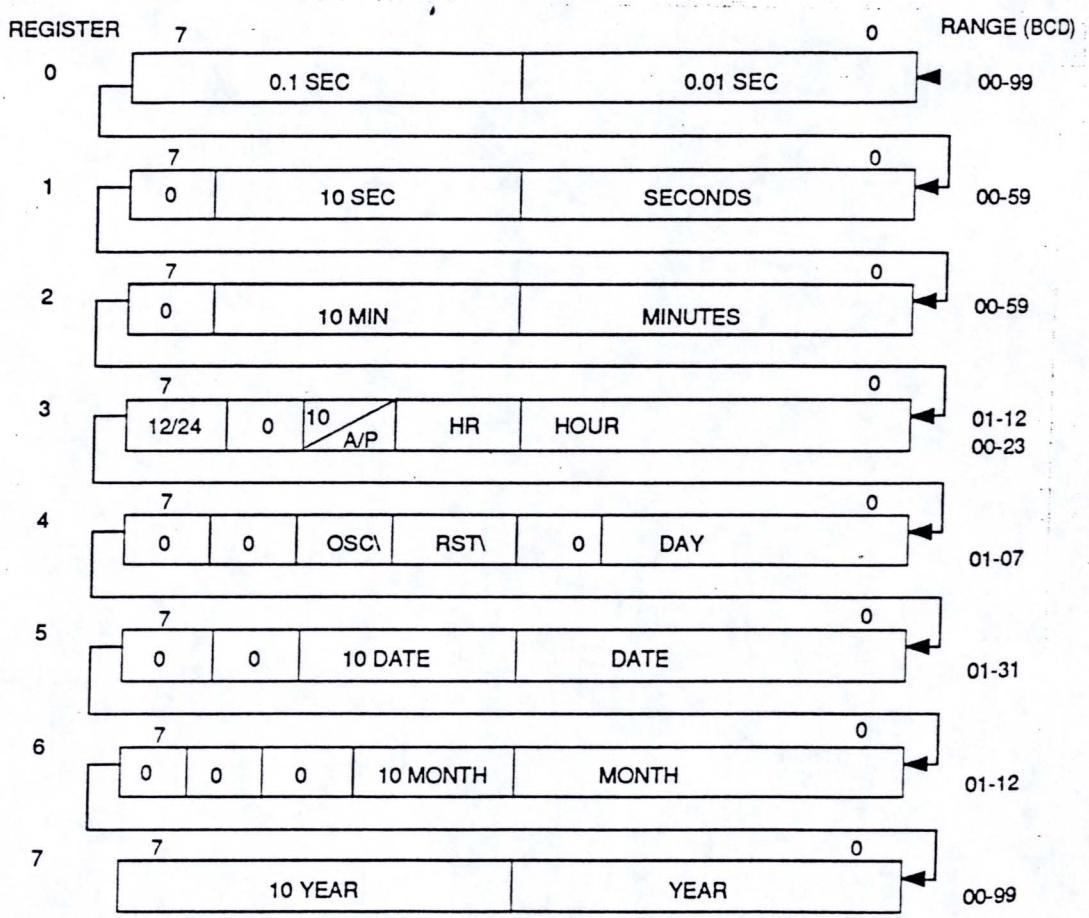
LOWER 4 BIT HEXADECIMAL

Lower 4 bit \ Higher 4 bit	0 (0000)	2 (0010)	3 (0011)	4 (0100)	5 (0101)	6 (0110)	7 (0111)	A (1010)	B (1011)	C (1100)	D (1101)	E (1110)	F (1111)
0 (xxxx0000)	CG RAM (1)												
1 (xxxx0001)	(2)												
2 (xxxx0010)	(3)												
3 (xxxx0011)	(4)												
4 (xxxx0100)	(5)												
5 (xxxx0101)	(6)												
6 (xxxx0110)	(7)												
7 (xxxx0111)	(8)												
8 (xxxx1000)	(1)												
9 (xxxx1001)	(2)												
A (xxxx1010)	(3)												
B (xxxx1011)	(4)												
C (xxxx1100)	(5)												
D (xxxx1101)	(6)												
E (xxxx1110)	(7)												
F (xxxx1111)	(8)												

Bÿlage 6


7
0
HEX VALUE
Byte 0
1 1 0 0 0 1 0 1
C5
Byte 1
0 0 1 1 1 0 1 0
3A
Byte 2
1 0 1 0 0 0 1 1
A3
Byte 3
0 1 0 1 1 1 0 0
5C
Byte 4
1 1 0 0 0 1 0 1
C5
Byte 5
0 0 1 1 1 0 1 0
3A
Byte 6
1 0 1 0 0 0 1 1
A3
Byte 7
0 1 0 1 1 1 0 0
5C

Bylage 7


REGISTER
7
0
RANGE (BCD)
0
0.1 SEC
0.01 SEC
00-99
1
0
10 SEC
SECONDS
00-59
2
0
10 MIN
MINUTES
00-59
3
12/24
0
10
A/P
HR
HOUR
01-12
00-23
4
0
0
OSC\
RST\
0
DAY
01-07
5
0
0
10 DATE
DATE
01-31
6
0
0
0
10 MONTH
MONTH
01-12
7
10 YEAR
YEAR
00-99