Programmerbare styringssystemer ”Microcontrollers” er overalt: i husholdningsapparater, i underholdningselektronik, i måleinstrumenter og selv i ubemandede rumfartøjer. Overalt, udføres de kommandoer der er programmeret in i kontrolleren. Det er meget spændende at generere simple kontrolprogrammer dig selv så godt.

Første skridt er at vælge en microcontroller eller processor, der matcher din ønskede opgave så præcist som muligt. Du kan vælge mellem forskellige typer fra forskellige virksomheder. Du kan også vælge dit programmeringssprog. Assembler og C tilbydes oftest; i mange tilfælde, så er Basic eller et andet højniveau sprog. Normalt kræver programmering omfattende software og en betjeningsenhed. Ud over etableringsomkostninger, kan uddannelse også være en betragtelig omkostning.

Mikrocontrolleren som her anvendes, er helt anderledes. Du kan programmere den med kun to trykknapper. Knap - programmerbar styring (tastenprogrammierbare Steuerung, TPS) kender kun relativt få kommandoer, der kan læres nemt og som kan programmeres ind i controlleren med knapperne. Programmet kan ændres til enhver tid og uden særlige hjælpemidler.

Systemet er specielt velegnet til kompakte applikationer i de områder, der måler, styrer eller regulerer.

Mange opgaver kan let løses med dette system. Når microcontrolleren er programmeret, og programmet virker, kan microcontrolleren sættes ind i dit eget elektroniske kredsløb, Her er kendskab til elektronik påkrævet.

Systemet er velegnet til træning og de første skridt i microcontroller programmering. Man kan hurtigt opnå succes, det man lærer her kan let overføres til andre systemer. Strukturerne er meget lig andre programmeringssprog.

Indholdsfortegnelse

1 Introduktion

Princippet i TPS controlleren er enkelt. Du har fire digitale indgange E1 til E4 og fire digitale udgange A1 til A4. Der er også to analoge indgange AD1 og AD2 og en næsten analog PWM-udgang. Et reset input med en reset knap der nulstiller et program og starter forfra. Controlleren spændingfødes fra 3 AA-batterier med ca. 4,5 V og kan arbejde i et område fra 2,2 V til 5,5 V.

Technical Data:

Microcontroller: HT46F47

Clock frequency: 2 MHz

Intern EEPROM: 128 Bytes

Spændingsforsygning VCC: 2.2 V to 5.5 V

Strøm forbrug: 1 mA at 4.5 V

4 output ports: kan belastes op till 10 mA

1 PWM output: kan belastes til 10 mA

4 input ports: ikke forbundet logisk 1 tilstand

2 analoge inputs: 0 V ... VCC

2 key inputs: ikke forbundet logisk 1 tilstand

Undervisningssættet består af:

Breadboard

Batteriholder til 3 * AA

Ledning

HT46F47 med TPS-firmware

3 trykknapper

4 LEDs 5 mm, red

1 LED 5 mm, green

1 LDR

3 kondensatorer 100 nF

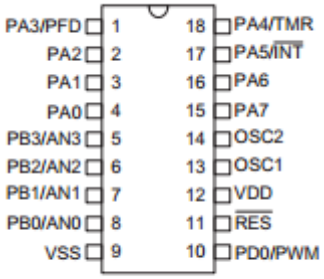
1 elektrolyt kondensator 47 µF

5 modstande 2.2 kO

1 modstand 10 kO

1 modstand 27 kO

2 modstande 100 kO



HT46F46E/HT46F47E
– 18 DIP-A/SOP-A

Til programmering, du har brug for de to knapper S1 og S2 og en enkel LED visning af fire lysdioder på udgangene A1 til A4. Der er i alt 14 simple kommandoer med tilknyttede data eller underkommandoer. Kommandoer og data hver indkodet som 4-bit binære tal 0000-1111 (decimal 0 til 15), og er direkte synlige på displayets lysdioder. Det respektive antal programmeres ved at trykke på S1. S2 skifter mellem kommando og data og øger adressen i kommandolinjen. Hele programmet struktur er så enkel, at du vil kunne det udenad med lidt øvelse.

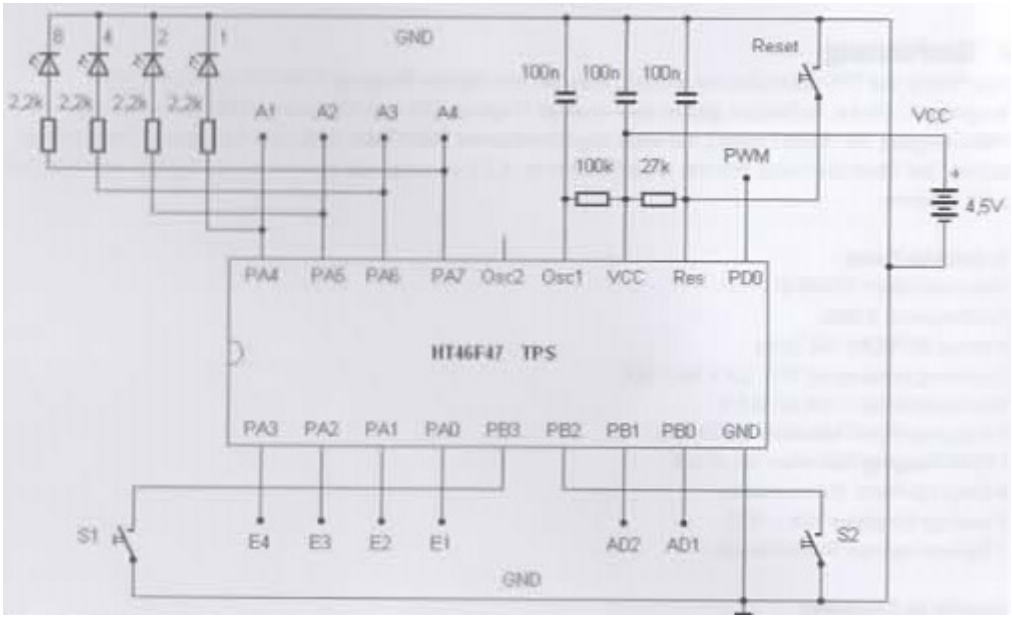


Fig. 1: Systemets basis kredsløb.

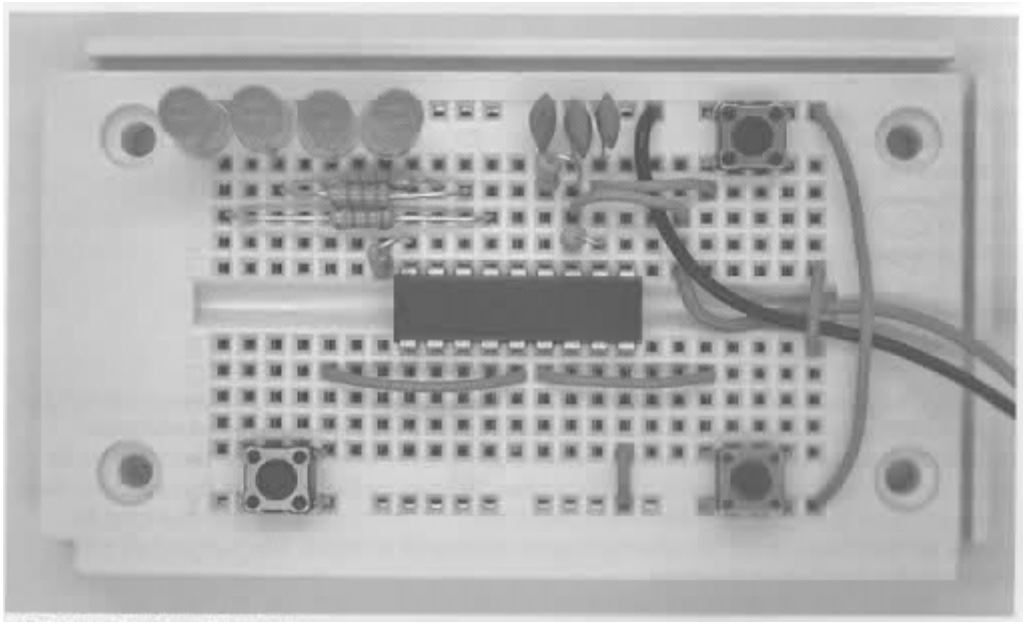


Fig. 2: Standard opsætning på breadbordet.

Nogle grundlæggende programmer er allerede programmeret ved leveringen af udviklingssystemet, disse kan startes direkte. Derfor er det muligt at tage microcontrolleren i drift trin for trin. Først sætter du dig ind i hardware-funktionerne og starte derefter dine egne programmer.

I den første test, vil du starte de små programmer, der allerede er programmeret ind i microcontrolleren. Disse programmer vil give et godt indtryk af mulighederne. De er kun kort forklaret. Den præcise forklaring af de enkelte kommandoer følger i næste kapitel.

Den første test, kræver kun den grundlæggende konfiguration med komponenterne på breadboardet,

Vær omhyggelig så komponenterne ikke ødelægges, de må ikke ramme hinanden, microcontrolleren skal vende rigtigt, lysdioderne skal også vende rigtigt .

Du mangler:

• Tilslutning af spændingsforsyningen på GND (minus)(sort) og VCC (plus)(rød)

• En kondensator 100 nF mellem VCC og GND

• Reset modstand 27 KO til VCC samt en kondensator 100 nF til GND

• Oscillator modstanden 100 kO til VCC og kondensator 100 nF til GND

Mikrocontrolleren HT46F47 drives med sin interne RC-oscillator. Modstanden på OSC1 indgang angiver clockfrekvensen. 100 kO giver en frekvens på ca.. 2 MHz. Hvis du ønsker det kan du arbejde med lavere eller højere hastigheder. Den tilsluttede kondensator tjener kun til at afkoble og har ingen indflydelse på clockfrekvensen. Forbindelsen OSC2 forbliver fri. Om ønskes, kan du tilslutte en ekstra modstand mod VCC her og derved opnå en fjerdedel af clockfrekvensen.

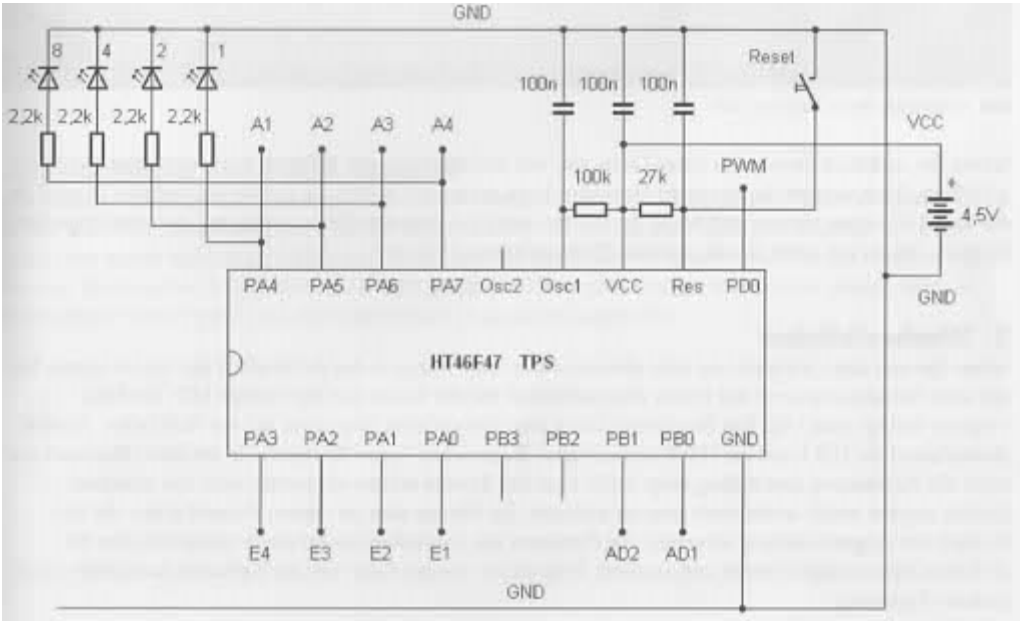


Fig. 3: Lysdioderne på output.

Brug den øverste og nederste skinne på breadboardet som GND. Den sorte ledning fra batteriholderen, dvs. minus polen, tilsluttes GND. Den røde ledning forbindes til VCC. Forkert polaritet skal under alle omstændigheder undgås, da det kan ødelægge controlleren. Installer gerne en kort wire stykke som trækaflastning. Når batterierne er forbundet, skal de være tilsluttede hele tiden. Sluk for systemet ved at

fjerne et af batterierne.

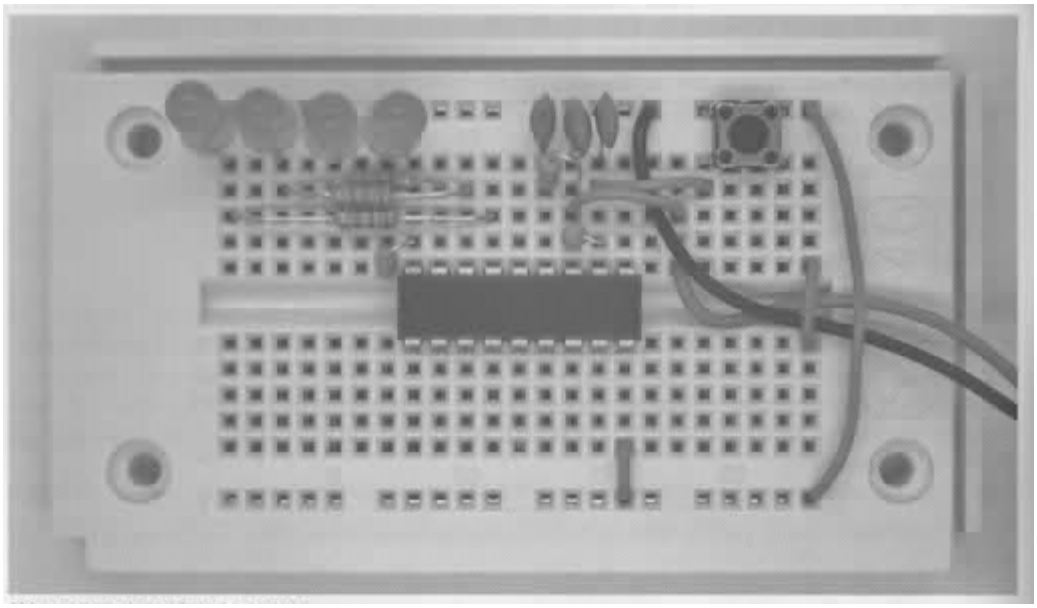


Fig. 4: Minimum application with LEDs

Den første øvelse kræver at reset knappen er monteret med tilhørende modstand og kondensator, samt at de fire lysdioder er forbundet til output pindene gennem 2,2 kO. Overhold rækkefølgen. A1 er forbundet til venstre LED og A4 til den højre. Den binære med den højeste værdi bit er til venstre. Dette er nyttigt især ved programmering.

2 Blinkende lysdioder

Nu indsættes de tre 1,5-V-batterier eller alternativt tre NiMh genopladelige batterier i batteriholderen. Nu starter det første program eksempel med blinkende lysdioder, vekslende mellem venstre og højre lysdiode. Lyset skifter ca. hvert sekund man siger at frekvensen er ca.. 1 Hz. Program listen viser et simpelt program med kun fem linjer. Skiftevis tændes LED 1 og LED 8. Mellem dem er der pause -kommandoer med en ventetid på 0,5 s. Sidste linie hopper(Jump) tilbage til begyndelsen, dette sikrer, at blinkende fortsættes evigt. De individuelle kommandoer er forklaret mere detaljeret nedenfor. Du kan se, hvor nem programmeringen er ved dette eksempel. Firmwaren af controlleren har en oversætter, der oversætter og udfører de simple kommandoer. Programmeringen er derfor er langt mere kompakt end i andre systemer.

Eksemplet indtager adressen fra 20h (20 hex) fremefter (decimal 32). Flere programmer i det øverste adresseområde kan startes senere fra dedikerede applikationer. Adresserne kan også overskrives med din egen programkode. Controlleren kan nulstilles således at det oprindelige program installeres.

Address	command	Data	Comment
20	1	1	LED1
21	2	8	Wait for 500 ms
22	1	8	LED8
23	2	8	Wait for 500 ms
24	3	4	Jump -4

Liste 1: Blinkende lysdioder

Hvis ikke det ønskede resultat opnås, check om lysdioderne vender rigtigt. Det er også en god ide at måle nogle spændinger. Brug altid, fx et digitalt multimeter i 20 V området og forbind minus ledningen på GND. Alle spændinger er således målt i forhold til GND:VCC: 4.5 V

Reset: 4.5 V

Osc1: 1.5 V

E1 to E4: 4.5 V

A1: vekslende

A2, A3: 0V

A3: vekslende

3 Binær tæller og PWM Output

Alle digitale indgange er trukket op (pulled up) med en intern modstand mod VCC og har en hvilespænding på mængden af driftsspændingen. Men du kan sætte hver af indgangene til GND med en ledning eller kontakt. Ved start, (standardprogrammet) læser alle inputs og evaluerer dem. Individuelle forbindelser kan sættes på GND.

Afhængigt af det samlede resultat, kaldes de forskellige programmer.

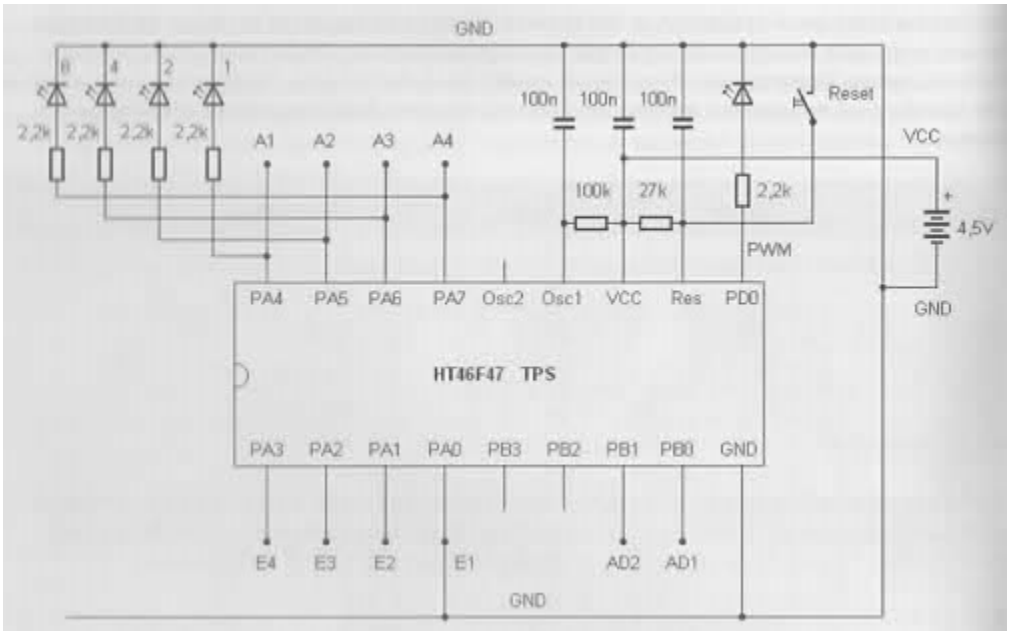


Fig. 5: Brug af PWM LED

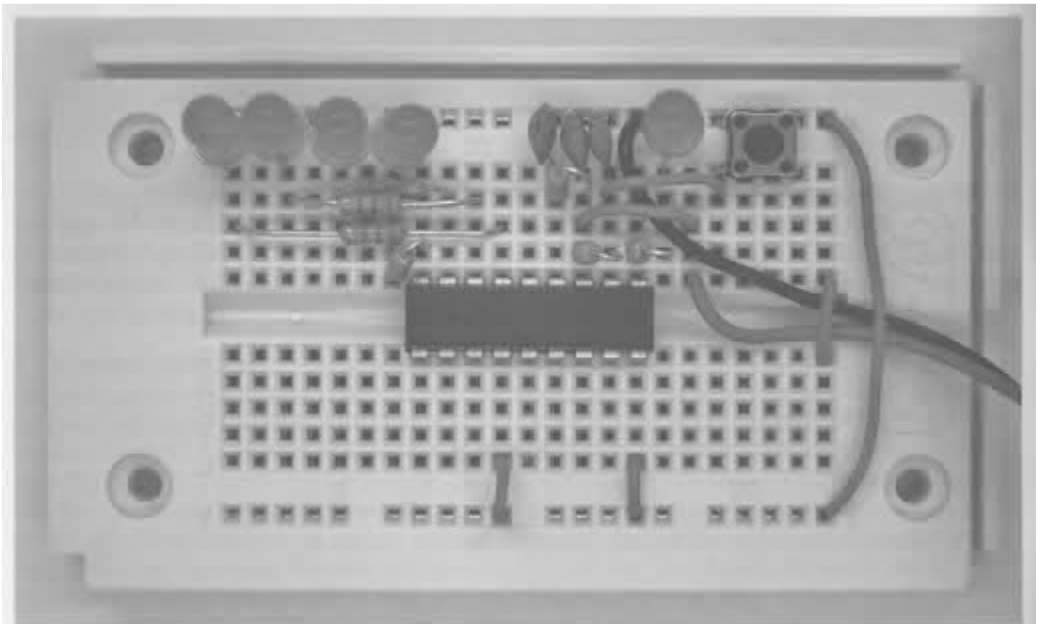


Fig. 6: Start den binære tæller

Sæt E1 til GND. Dette starter det andet eksempel program efter et tryk på reset knappen.

Det tæller output signalerne op fra 0000 (decimal 0) til 1111 (decimal 15). Programmet anvender variablen A til en simpel addition. A udlæses herefter til de fire output, PWM-udgangen. PWM (pulsbreddemodulation), denne udgang vil have en tilnærmet analog værdi, svarende til outputværdien. Kommandoerne 5 og 7 har sub-funktioner, der er skrevet som data. Herom senere.

Address	command	Data	Comment
25	7	1	A=A+1
26	5	4	Port = A
27	5	9	PWM = A
28	2	6	Wait for 100 ms
29	3	4	Jump -4

Liste 2: Binær tæller med LED and PWM udgange

Optællings programmet kan bruges som en øvelse i at læse binære tal, dette skal du kunne når du selv skal programmere. Hver af de fire lysdioder repræsenterer et bit. Alt i alt kan et 4-bit tal derfor blive vist. Lysdioderne kaldes 8, 4, 2 og 1 i henhold til deres værdi i kredsløbs diagrammet. Når vi tilskriver værdier i data feltet gøres dette ikke i decimaltal, men i hexadecimal her får værdierne fra 10 til 15 bogstaverne A til F.

Value				Decimal	Hex
8	4	2	1		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

Programmet kan bruges hvor LEDéne blinker med forskellige frekvenser.

En høje forsinkelse giver en lavere frekvens. Halveres frekvensen bliver periodens varighed doblet:

A1: 200 ms

A2: 400 ms

A3: 800 ms

A4: 1,600 ms

Den numeriske værdi der er præsenteret på de fire output (LED 1 til8) sendes også til PWM output (impulsbreddemodulation). PWM-signalet er et rektangulært signal med en frekvens på ca.. 16 kHz. Pulslængden styres således, at impuls/pause-forholdet udtrykt i værdien, dermed lysstyrken af LED. Lysstyrken på den tilsluttede LED styres i 15 niveauer mellem nul og fuld lysstyrke.

Et PWM-signal kan udjævnes til en jævnspænding med et RC lavpasfilter. PWM-udgangen bliver således en analog udgang. Med dette program, vil du modtage en direkte spænding, der øges gradvist fra 0 V til 4,5 V. Se spændingen med et måleinstrument eller et oscilloskop.

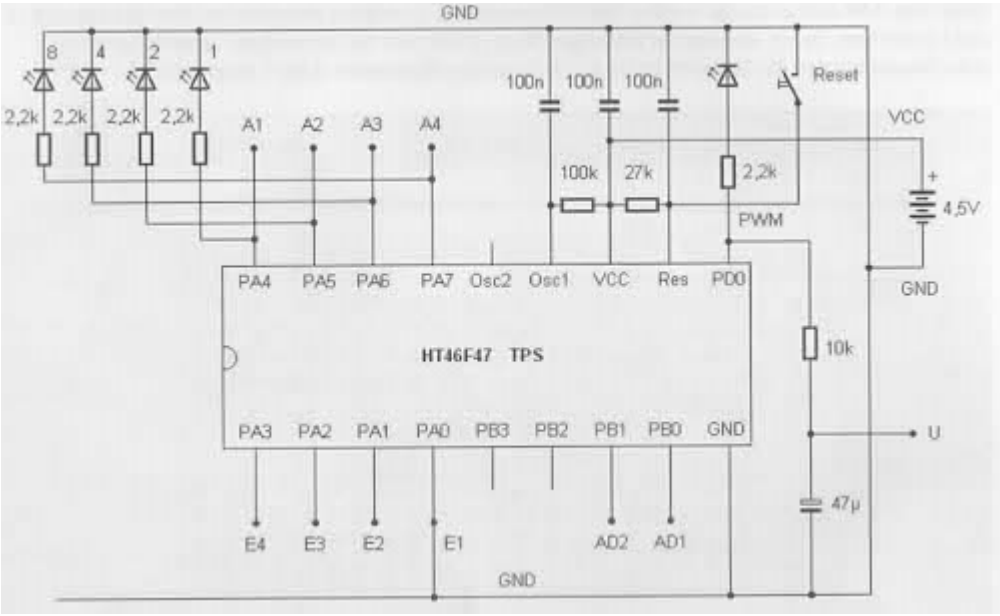


Fig. 7: Et lavpasfilter monteret på PWM udgangen

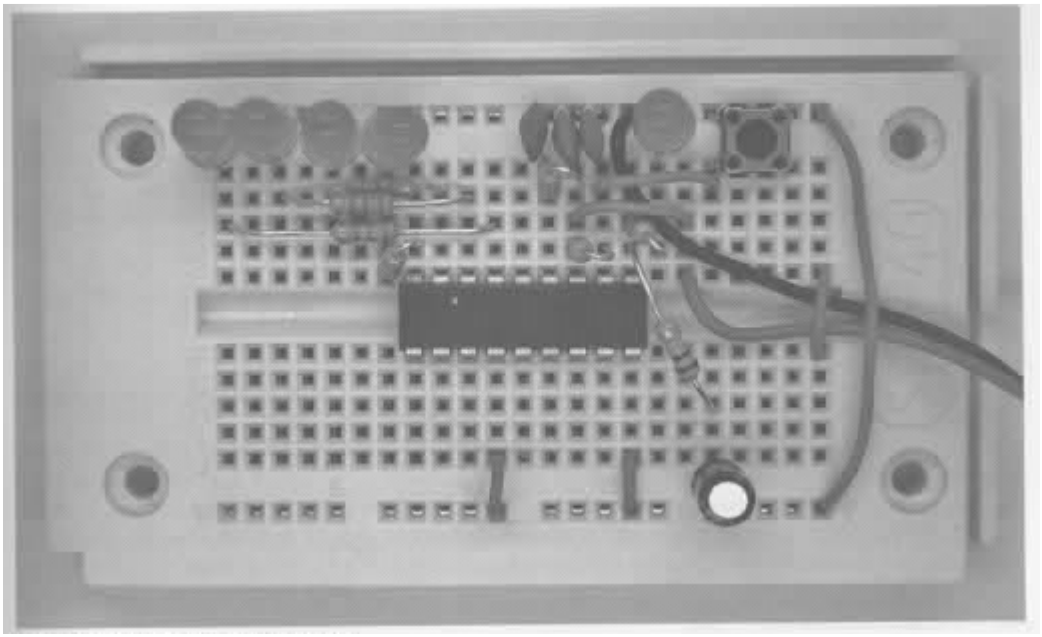


Fig. 8: PWM udgang med analog spænding, med indkobling af et lavpasfilter.

4 Analog til-Digital Konverter

Hvis E2 sluttes til GND og resetknappen aktiveres startes et lille eksempel program med analogtil digital konvertering (AD-konverter). Den analoge spænding på den analoge indgang AD1 måles og konverteres til en digitalt numerisk værdi. Fordi TPS controlleren fungerer med 4bit, vil resultatet af analog-digital konvertering være et tal fra 0 til 15. Resultatet 0 repræsenterer indgangsspændingen 0, resultatet 15 for en spænding, der svarer til driftsspændingen, dvs. fx 4,5 V. AD-værdien er output som et binært tal på de fire lysdioder samt også som PWM-output. Tilslut en spændingsdeler med en fast modstand og en lys-afhængig modstand (LDR) til den analoge indgang AD1.

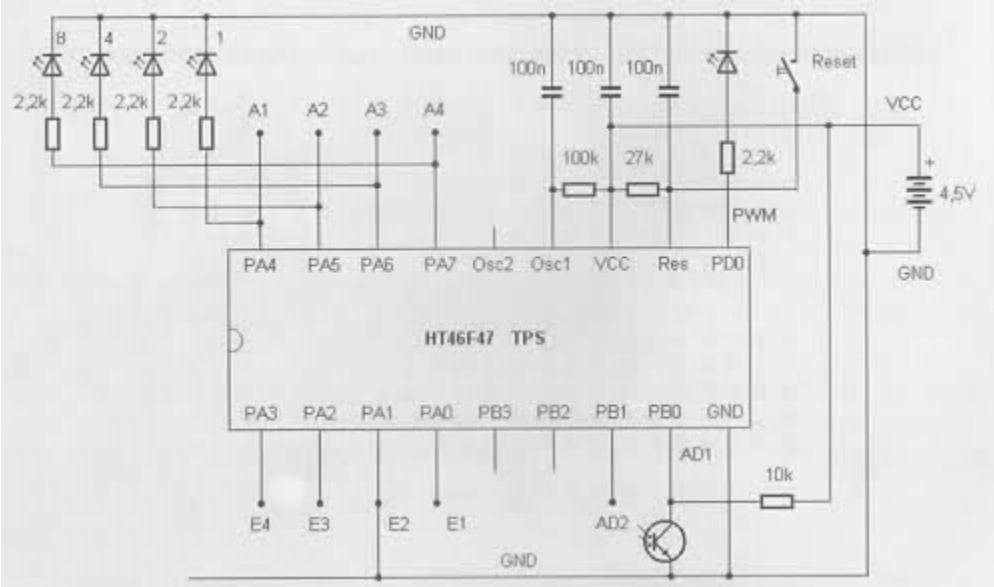


Fig. 9: Forbindelse af den lysafhængige modstand

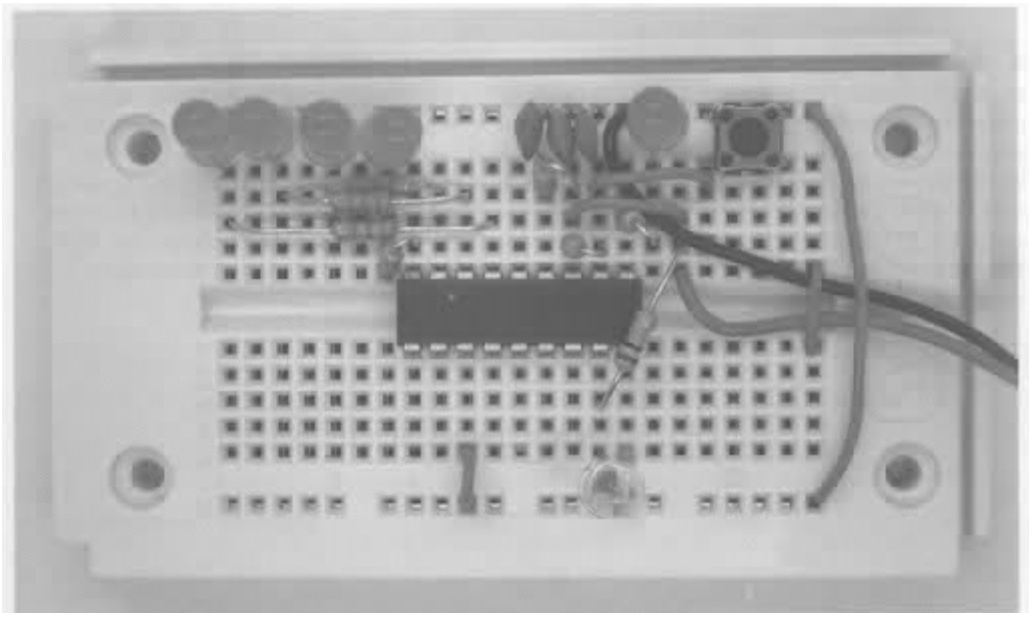


Fig. 10: Den lysafhængige modstand forbundet til AD1 indgangen

Eksempel programmet er meget lig programmet fra det sidste afsnit på grund af output til de digitale udgange og PWM-udgangen. Den første linje har kommandoen til at konvertere en analog værdi.

Address	command	Data	Comment
2A	6	9	A=AD1
2B	5	4	Port = A
2C	5	9	PWM = A
2D	2	6	Wait for 100 ms
2E	3	4	Jump -4

Liste 3: AD converter and PWM output

Prøv med forskellige lysstyrker mod lyssensoren. Jo mere lys på LDR, jo lavere er spændingen på AD1. Vice versa, mørke fører til maksimal AD-værdier og dermed maksimal lysstyrke på LED på PWM output. Aflæs den binære værdi fra LED-display og forsøg at justere lysstyrken, f.eks til præcist halvdelen af området. Den digitale værdi,vil så være 0111 eller 1000.

5 Tilfældigheds Generator

Sæt en jumper E3 til GND for at starte et grundlæggende program for en tilfældigheds generator. Knappen S1 bruges her. Den tilhørende indgangs har en intern pull-up modstand, der trækker spændingen til VCC-niveau. Knappen er forbundet til GND. Trykkes knappen ned går input S1 til nul.

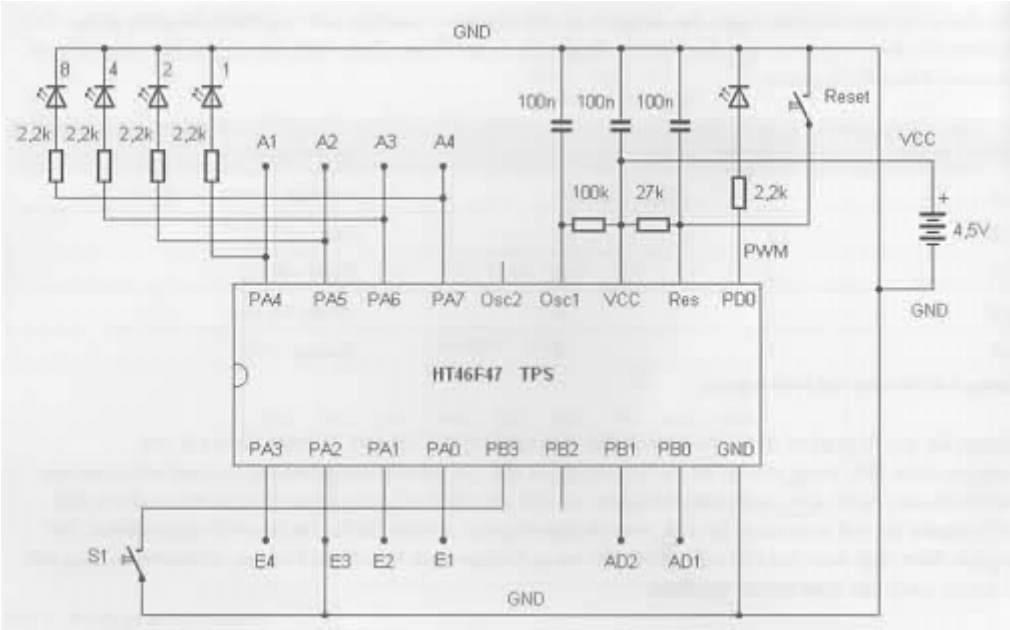


Fig. 11: Start of the random switch

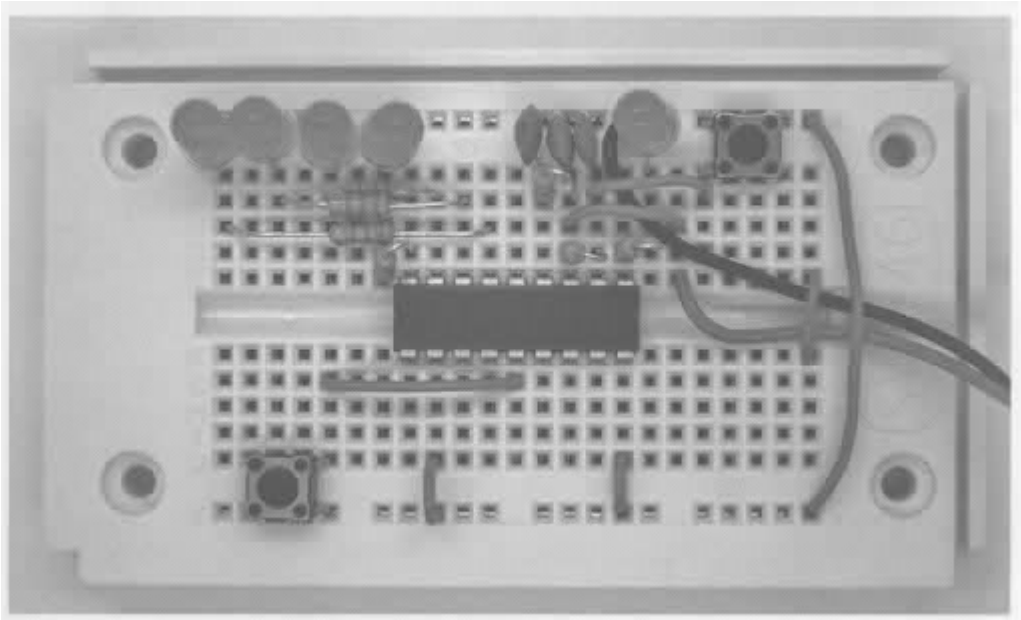


Fig. 12: Jumper between E3 and GND

Programmet bruger en betinget hop-kommando. Når S1 input tilstanden er en(høj), er følgende kommando sprunget over. Trykkes der på knappen bliver tilstanden nul, og øger dermed variablen A. Dette fører til en

hurtig optælling af start- tilstanden. Når knappen slippes, er det den sidste tæller læsning der bevares. På grund af den høje optællings hastighed, har ingen indflydelse på resultatet, som derfor bliver tilfældigt.

Address	command	Data	Comment
30	5	4	Dout = A (Random)
31	C	E	Skip if S1 = 1
32	7	1	A = A+1
33	3	3	Jmp -3

Liste 4: Tilfældigheds generatoren

Et kort tryk på knappen vil give et nyt tilfældigt resultat. Test tilfældigheds funktionen ved at generere en statistik af resultaterne. Efter et tilstrækkeligt antal kørsler, bør det være klart, at alle resultater er omtrent lige hyppige. Programmet er også velegnet som et spil, hvor du er nødt til at forsøge at få fx 1111.

Samtidig er programmet en tæller med den maksimalt mulige arbejdshastighed fordi der ikke anvendes pause (wait) kommandoer. Derfor kan du bruge dette eksempel til at undersøge arbejdshastigheden på TPS controlleren. Mens knappen er trykket, viser output A1 en rektangulær signal med en frekvens på ca.. 133 Hz og en periode varighed på 7,5 ms. Udgangsportene ændrer tilstand efter 3,75 ms. Programmet går gennem fire kommandoer i optællingens sløjfen. Omkring et millisekund er påkrævet per kommando. Det sidste output A4 viser en frekvens på 16,6 Hz, hvilket stadig er synlige som en flimren.

Hvis kritiske opgaver kræver højere arbejds- hastigheder, kan du øge Klokfrekvensen på controlleren ved at reducere modstanden ved OSC1. 100 kO giver en cyklus på 2 MHz. Udskift modstand med en 27 kO en. Dette vil føre til en næsten fire gange højere cyklus sats og en kommandotid på ca.. 0,25 ms. I det normale tilfælde, bør controlleren køre med 100 kO på OSC1. Således sikres et lavt strømforbrug og driftsikkerhed ved en lav driftsspænding helt ned til 2,2 V.

6 Impuls længde måling

Sættes en jumper fra E4 til GND startes et eksempelprogram til at måle impulslængde efter et reset. Igen er det input på s1 der beregnes.

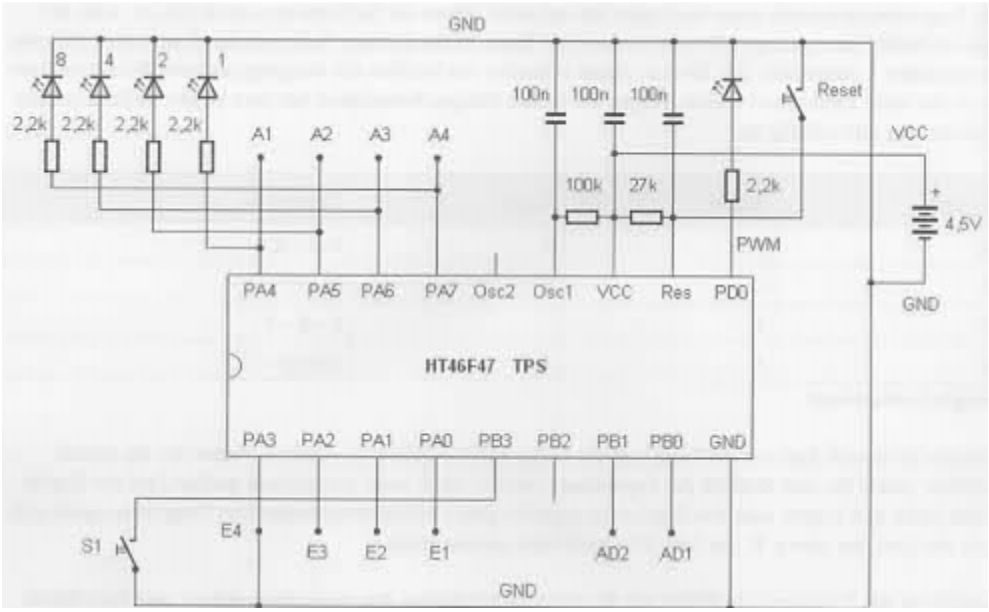


Fig. 13: E4 til GND

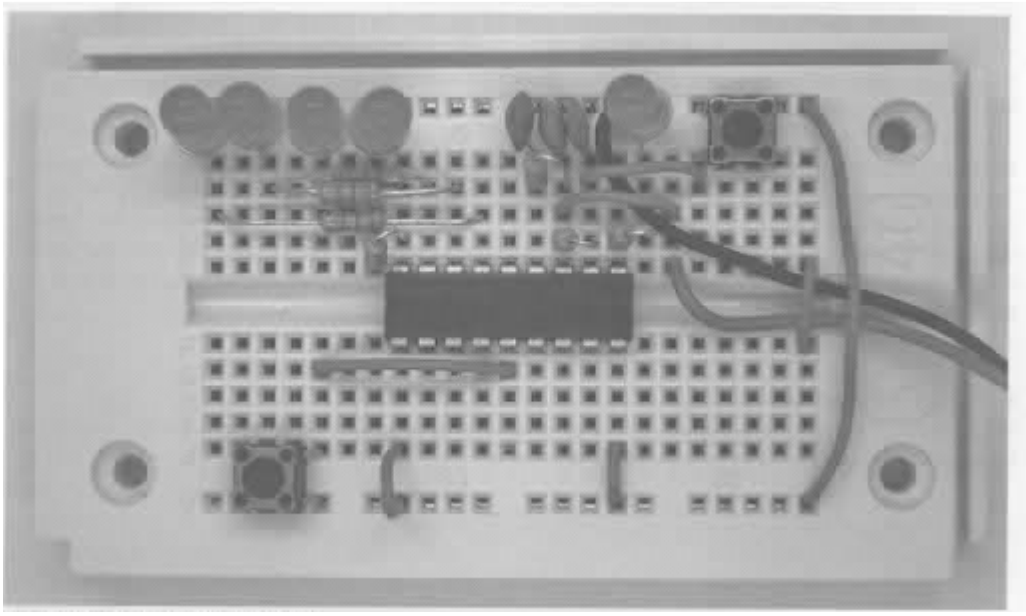


Fig. 14: Start af impuls længde måleren.

En tidsmåling kører når S1 = 0, dvs. med knappen trykket. Der tilsættes ca. 5 ms medens der ventes på at knappen slippes tilsættes yderligere 5 ms til udførelsen af i alt fem kommandoer i optællings sløjfen. Derfor er tidsenheden af målingen 10 ms.

Address	command	Data	Comment
34	2	2	Wait for 5 ms
35	C	C	Skip if S1 = 0
36	3	2	Jmp -2
37	4	0	A = 0
38	2	2	Wait for 5 ms
39	7	1	A = A + 1
3A	5	4	Port = A
3B	C	E	Skip if S1 = 1
3C	3	4	Jmp -4
3D	3	9	Jmp -9

Liste 5: Tids måling

Tryk på knappen S1 så kort som muligt. Du vil f.eks se resultatet 1010, dvs. decimal 10. Eftersom tidsenhed af programmet er 10 ms, vil pulslængden være 10*10ms hvilket svarer til 100 ms. Med lidt træning, kan du måske nå helt ned på 50 ms.

7 Udlæsning af et program

Til programmering, har du brug knapperne S1 (datainput, venstre) og S2 (programmering, højre). Reset knappen er også påkrævet. Disse knapper alene kan bruges til at læse programmer og indtaste programmer. Med lidt øvelse, kan du selv indtaste nye programmer meget hurtigt og ændre nuværende.

For at indtaste programmer, udføres en nulstilling ved tryk på knappen S2, slip S2 igen omkring et halvt sekund efter nulstillingen. Nu kan du bruge knappen S2 at rulle gennem programmet, og se kommandoer og data. Hver adresse kræver to tryk på S2. Dette skifter mellem visning af kommando og data. Den nuværende adresse vises også kortvarigt.

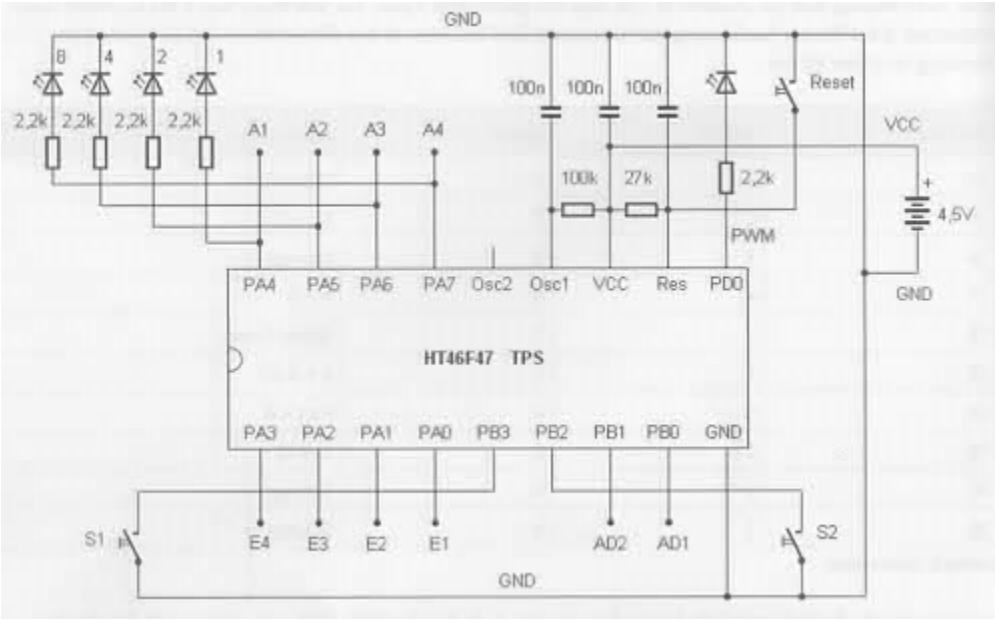


Fig. 15: S1 og S2 I programmerings mode

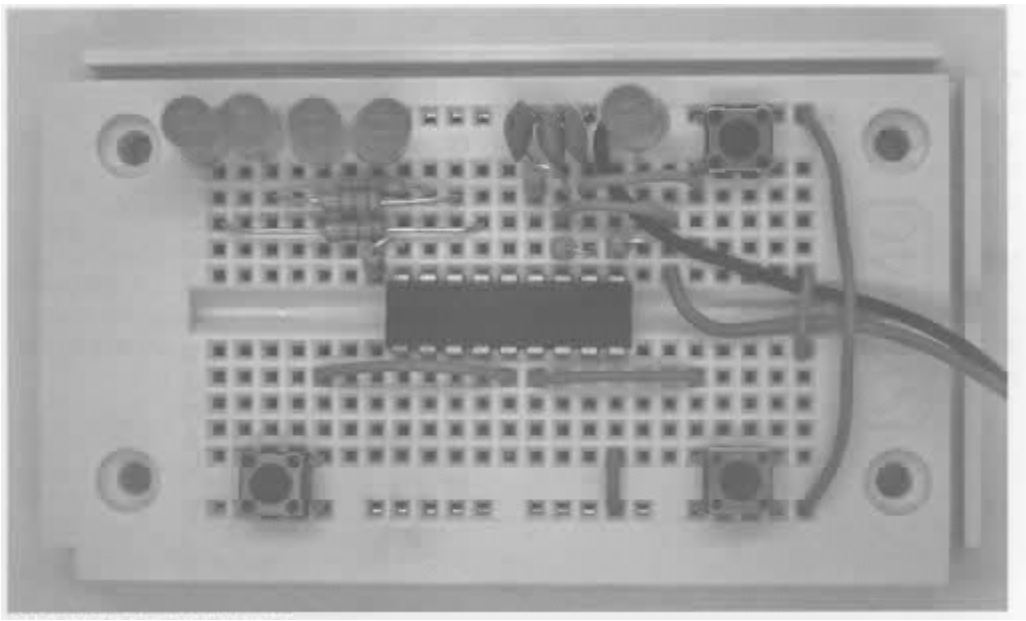


Fig. 16: Tre knapper og LED display.

• Tryk på S2

• Displayet viser adressens laveste fire bit I 300 ms

• Displayet slukket I 300 ms

• Displayet viser kommandoen

• Tryk igen på S2

• Displayet viser data

• Tryk igen S2

• Displayet viser den næste adresse i 300 ms

• og så videre.

Hvis du ønsker at se det nuværende program bestående af fem trin, uden at ændre det, vil 10 tryk på S2 føre dig gennem til enden af programmet. Fordi den nuværende adresse vises kort, er det nemt at orienterer sig. Du vil altid vide, om displayet i øjeblikket viser en kommando eller data. I udlæsningstilstanden, er følgende kommandoer og data i de fem første adresser. Dette er starten på programmet der vælger imellem og starter de individuelle eksempel programmer.

Address	command	Data	Comment
0	6	4	A = Din
1	5	1	B = A
2	4	E	a = 14
3	B	0	Addr Hi = 0
4	C	3	Skip if A = B

Liste 6: Program koden I standard opstarten

En 4-bit kommando og det tilhørende 4-bit data danner tilsammen en byte, dvs. et 8-bit tal. En halv byte kaldes også en "Nibble". Den høje nippe danner kommandoen, den nederste nibble de tilknyttede data. Controllerens EEPROM omfatter i alt 128 bytes. Derfor kan et program indeholde op til 128 kommandoer med tilhørende data. Dette er nok til de fleste applikationer, da programkoden er ekstremt kompakt. Mange nyttige programmer kan programmeres med færre end ti kommandoer.

Efter at have udlæst kommandoer og data og programmet er som du ønsker det, tryk derefter på reset knappen. Det gamle program starter herefter uændret.

8 Indlæsning af et program

Knappen S1 bruges kun, hvis en kommando eller dens data skal ændres eller indtastes på ny. Generelt kan der indtastes tal mellem 0 og 15. Med det første tryk på S1, nul indstilles data på adressen. Hvert følgende tryk på en knap øger antallet med 1. Den aktuelle status vises binært med de fire lysdioder. Hvis du ønsker at indtaste 4, trykkes S1 fem gange: 0, 1, 2, 3, 4. Det binære display viser så 0100.

S2 skifter hele tiden mellem kommando og data, efter data øges adressen med 1.

Når du har indtastet kommando og data. Tryk på S2 så gemmes denne byte i EEPROM. For at tydeliggøre dette er LED-displayet slukket i 600 ms, før den næste adresse og den næste kommando vises.

Du kan ændre det nuværende program i en enkelt byte eller, Bruge S2 til at rulle til den ønskede placering og ændre kommando eller data med S1, for igen at gemme byten i EEPROM med S2.

Her er vist et eksempel kun to kommandoer. Det tænder tre lysdioder og ender i en endeløs løkke.

Address	command	Data	Comment
0	1	7	A1 - 4 = 0111
1	3	0	Jmp 0

Liste 7: Switching on LEDs

Hex 17 30

I stedet for en detaljeret liste, kan du vælge en forkortet form. De to bytes er sammenfattet i hex tal: 17h og 30h. Herunder ser du hvordan du bruger den hexadecimale form.

Du skal skrive følgende som input:

S2 + Reset

2 x S1

S2

8 x S1

S2

4 x S1

S2

1 x S1

S2

Hvis du fik trykket knappen S1 en gang for meget ved et uheld, kan du stadig nå det rigtige tal. Når du har tastet tallet 15.efterfølges tallet af værdien 0, så taster du bare forfra indtil den ønskede værdi vises på LED displayet..

Når du er færdig med at taste programmet ind, startes dette ved et tryk på reset knappen. Du kan se, at tre lysdioder tændes. Der sker ikke andet. Controlleren reagerer ikke længere på indgangene E1 til E4, da det oprindelige program er blevet delvist overskrevet. Program eksemplerne kan ikke startes længere.

Da du kun har ændret de første to bytes, kan du nemt starte det oprindelige program igen.

Du skal blot taste de oprindelige første to kommandoer (64 51) igen. På samme måde som du lige har gjort.

Test nu den oprindelige funktion af program eksemplerne.

Du ved nu hvordan du programmerer den lille computer.

9 Genindlæs program eksemplerne

Hvis du vil gendanne controllerens oprindelige tilstand (factory settings), kan dette gøres ved at indtaste to bytes FF.

Det svarer til at EEPROM er tom. Firmwaren af TPS Regulatoren indeholder en start-funktion, der gennemgår de første to adresser til at genkende en tom hukommelse. Hvis to FF-bytes læses her antager controlleren, at der ikke er indtastet et program. I dette tilfælde fyldes EEPROM automatisk med det oprindelige program med program eksemplerne.

Denne funktion kan bruges til enhver tid, til at gendanne det oprindelige program.

Address	command	Data	Comment
0	F	F	
1	F	F	

Liste 8: Retur til factory settings

Start programmerings tilstanden ved at trykke knappen S2. Indtast værdien F (decimal 15) fire gange så, hvor alle LED'er A1 til A4 er på.

Tryk til sidst F2 for at afslutte programmeringen.

Controlleren har brug et øjeblik længere end normalt til at omprogrammere alle bytes.

Dette gendanner den oprindelige tilstand. Test, fx den alternerende flash fra side 8 uden jumper ved indgangene.

10 TPS Kommandoer

TPS kender alt 14 kommandoer (1-14). Mange af disse kommandoer har en parameter (data) i form af et 4-bit tal 0000 til 1111 (0-F), dvs. med en nummerserie 0 til 15 (decimal).

For eksempel kommando 7 betyder "Compute A = ...". Parameteren indikerer hvilken beregning funktion skal udføres.

Nedenfor er kommandoer og data skrevet i hexadecimal kode som en byte. Kommando 1 sammen med parameter 4 bliver således kommandoen 14h.

De første tre kommandoer er:

10–1F: Direkte Port output på A1–A4, 0–15, binær 0000 til 1111

20–2F: Timer for pause (wait) tid 0–15 (1, 2, 5, 10, 20, 50, 100, 200, 500, 1,000, 2,000, 5,000, 10,000, 20,000, 30,000, 60,000 ms)

30–3F: Hop tilbage 0–15

Kommando 1 bruges til at tildele portene A1 til A4 en værdi. Dette tillader at sætte hvilken som helst værdi på lysdioderne.

Pause kommandoen 2 bruger en parameter, som indeholder tiden i millisekunder.Selv om der kun anvendes 4 bit som data, tillader dette udførelse af pauser mellem et millisekund og et minut. Hvis du ønsker længere ventetid, kan du udføre en løkke.

Hop tilbage kommandoen 3 bruges hvor en programdel skal gentages, der hoppes det antal adresser tilbage som data angiver, hvis data er 0 (nul) dannes en uendelig løkke og programmet stopper her.

Blinkende LED programmet bruger kun disse tre kommandoer, det er skrevet i adresseområdet fra 0 (nul), her i en lidt ændret form.

Address	command	Data	Comment
0	1	1	A1 - 4 = 0001 (LED1)
1	2	7	Wait for 200 ms
2	1	4	A1 - 4 = 0100 (LED4)
3	2	7	Wait for 200 ms
4	3	4	Jump - 4

Liste 9: Programmet - Blinkende lysdioder

I kort hexadecimal form ser det således ud:

11 27 14 27 34

Med disse tre første kommandoer, kan man fremstille mange simple programmer. Analyser og test de tre følgende programmer. Målet er at intuitivt kunne bruge disse kommandoer. Simple programmer som disse bør endda kunne programmeres udenad. Et eksempel på dette er den en løbende lys med fire output mønstre:

Address	command	Data	Comment
0	1	1	A1 = 0001 (LED1)
1	2	8	Wait for 500 ms
2	1	2	A1 = 0010 (LED2)
3	2	8	Wait for 500 ms
4	1	4	A1 = 0100 (LED3)
5	2	8	Wait for 500 ms
6	1	8	A1 = 1000 (LED4)
7	2	8	Wait for 500 ms
8	3	8	Jmp - 8

11 28 12 28 14 28 18 28 38

Liste 10: Løbe lys 1

Udvid programmet med to output mønstre og tilhørende pauser, så programmer kan få lysdioderne løbe frem og tilbage

Address	command	Data	Comment
0	1	1	A1 = 0001 (LED1)
1	2	8	Wait for 500 ms
2	1	2	A1 = 0010 (LED2)
3	2	8	Wait for 500 ms
4	1	4	A1 = 0100 (LED3)
5	2	8	Wait for 500 ms
6	1	8	A1 = 1000 (LED4)
7	2	8	Wait for 500 ms
8	1	4	A1 = 0100 (LED3)
9	2	8	Wait for 500 ms
A	1	2	A1 = 0010 (LED2)
B	2	8	Wait for 500 ms
C	3	C	Jmp - 12

11 28 12 28 14 28 18 28 14 28 12 28 3C

Liste 11: Løbe lys 2, frem og tilbage

En pause funktion kan indeholde en forsinkelse på op til ét minut. For enden er der et hop tilbage med værdien 0 (nul), dvs. en endeløs løkke den tjener som afslutning af programmet. Programmet startes igen

med reset-knappen. Prøv at udvide programmet til en tre-minutters køkken timer. Du kan få vist den resterende tid med antallet af tændte lysdioder som et lys

Address	command	Data	Comment
0	1	F	A1 = 1111 (LED1+2+4+8)
1	2	F	Wait for1000 ms
2	1	0	A1 = 0000 (All LEDs off)
3	3	0	End (Jump 0)

1F 2F 10 30

Liste 12: Timer på et minut

11 Beregning ved hjælp af variabler

Hidtil har vi kun anvendt konstante talværdier som parametre for de enkelte kommandoer. Dette er fornuftigt, når et program skal køre på samme måde hver gang. Mere komplekse programmer, arbejder med variable data. F.eks. kan der udføres en beregning som A = A + B. A vil ende op med en værdi afhængigt af indholdet af variablerne A og B. Resultatet kan styre lysdioderne ved udgangene som følger, f.eks.

Styringen har de fire variable A, B, C og D. Den vigtigste variabel er A. Det kaldes også akkumulatoren eller kortvarigt "Accu". A er involveret i alle beregnings operationer og resultatet gemmes i A. A anvendes også til transport af data. B er mest anvendt til beregnings operationer. C og D kan anvendes som midlertidige hukommelser og kan senere bruges som tællere for optællings løkker.

Der er to analoge indgange (AD1 og AD2) og en PWM-output. De behandlede data er begrænset til fire bit og kun tilgængelige via variabel A (A = AD1, PWM = A). A kan også indlæses direkte med en række (kommandoer 40-4F). For at fylde B, C eller D, Værdien skal først ind i A, derefter kan man tildele indholdet til den anden variabel (kommandoer 51-53). A og B kan anvendes i visse beregninger (kommandoer 71-7A).

Kommandoerne 40-4F tildeler således A en ny værdi. Kommandoen gruppe 51-5A overfører indholdet af A til en variabel, en output port eller PWM output.

Skal vi således læse data ind i A bruges kommando gruppen 61-6A, hvor data fra en kilde læses ind i A. Kommando gruppen 71-7A udfører beregninger, hvor resultatet lagres i A. output port D-out består af fire udgange A1 til A4, disse kan sættes enten sammen eller som individuelle bits D-out.0 til D-out.3. Tilsvarende kan Indgangene E1 til E4 læses samlet, eller som individuelle bits. AD1 og AD2 indlæses med kommandoerne 69 og 6A.

40–4F: A = 0–15

51–5A: Target 1–9 = A

51: B = A

52: C = A

53: D = A

54: D-out = A

55: D-out.0 = A.0

56: D-out.1 = A.0

57: D-out.2 = A.0

58: D-out.3 = A.0

59: PWM = A

61–6A: A = Source 1–10

61: A = B

62: A = C

63: A = D

64: A = Din

65: A = Din.0

66: A = Din.1

67: A = Din.2

68: A = Din.3

69: A = AD1

6A: A = AD2

71 –7A: A = Expression 1–10

71: A = A + 1

72: A = A – 1

73: A = A + B

74: A = A – B

75: A = A * B

76: A = A / B

77: A = A And B

78: A = A Or B

79: A = A Xor B

7A: A = Not A

Et eksempel på brugen af variabel A findes i program eksemplet i kapitel 3. Programmet blev skrevet på adressen 0 (nul). Her er det lidt udvidet. Desuden er det defineret med start værdien 0 i variablen A. Adresse 01 indeholder en beregnings kommando, indholdet i A stiger med én, og herefter sættes PWM-udgangen og udgangsporten til værdien af A.

Address	command	Data	Comment
0	4	0	A = 0
1	7	1	A = A + 1
2	5	4	Port = A
3	5	9	PWM = A
4	2	6	Wait for 100 ms
5	3	4	Jmp -4

40 71 54 59 26 34

Liste 13: A tælles op med een

Et andet eksempel er allerede blevet vist i kapitel 4. Dataene kommer fra den analoge indgang AD1 og overføres til output port og PWM output. Det ændrede program indeholder et yderligere beregnings trin, dvs. indholdet af variablen A inverteres. På denne måde får værdien 0000 den nye værdi 1111, dvs. 0 bliver 15 og vice versa. Den stigende indgangsspænding fører dermed til et mindre PWM-output.

Address	command	Data	Comment
0	6	9	A = AD1
1	5	4	Port = A
2	7	A	A = Not A
3	5	9	PWM = A
4	2	6	Wait for 100 ms
5	3	5	Jmp - 5

69 54 7A 59 26 35

Liste 14: Invertering af data

12 Jumps and Branches

Indtil nu, har vi kun arbejdet med et simpelt hop tilbage (kommando 3), der hopper tilbage op til 15 adresser. Nu tilføjer vi et absolut hop. Da selve hoppet kun kan angives med 4 bit, er der en yderligere kommando, som angiver de øverste 4 bit adressen. Dette giver et adresseområde på 0-255. Dette er mere end du har brug for, da EEPROM controlleren indeholder kun 128 bytes, dvs. området 00 til 7F (decimal 0-127). Hukommelsen er således delt ind i 8 sider(page 0 til 7). Pagen skal så angives før det absolutte hop.

To tælle løkker med variablerne C og D kan også udføre absolutte spring, men siden (page) skal angives først.

De betingede hop virker således. Når den respektive betingelse er sand, springes en adresse over. Betingelserne kan være sammenligninger mellem A og B eller direkte bit forespørgsler på indgangs porten.

Der kan også kaldes et underprogram (sub program), underprogrammet udføres, og der hoppes til bage til den næste adresse i hovedprogrammet. Et underprogram må ikke kalde et andet underprogram, da controlleren kun kan huske en retur adresse ad gangen.

80–8F: Adr-high = 0–15

90–0F: Direct jump to Adr-high, Adr-low (0–15)

A0–AF: Counting loop C-times Adr-high, Adr-low (0–15)

B0–BF: Counting loop D-times Adr-high, Adr-low (0–15)

C1–CF: Conditional jump: if (condition 1–15) then skip

C1: if A > B then Adr = Adr + 1

C2: if A > B then Adr = Adr + 1

C3: if A = B then Adr = Adr + 1

C4: if Din.0 = 1 then Adr = Adr + 1

C5: if Din.1 = 1 then Adr = Adr + 1

C6: if Din.2 = 1 then Adr = Adr + 1

C7: if Din.3 = 1 then Adr = Adr + 1

C8: if Din.0 = 0 then Adr = Adr + 1

C9: if Din.1 = 0 then Adr = Adr + 1

CA: if Din.2 = 0 then Adr = Adr + 1

CB: if Din.3 = 0 then Adr = Adr + 1

CC: if S1 = 0 then Adr = Adr + 1

CD: if S2 = 0 then Adr = Adr + 1

CE: if S1 = 1 then Adr = Adr + 1

CF: if S2 = 1 then Adr = Adr + 1

D0–DF: Subprogram call Adr-high, Adr-low (0-15)

E0–EF: Return from subprogram

One example of using conditional jump commands is found in the example program in chapter 6. Here, it has been slightly modified and put in address 0. Since the upper part of the address (Adr-hi) is in the quiescent condition 0, and the controller starts on page 0, the command 80 does not need to be used here. The length of a push of a button is measured and displayed again. All waiting commands have been removed from the program so that it now works with a higher resolution.

Et eksempel på anvendelse af betinget hop findes i program eksemplet i kapitel 6. Her er det igen blevet ændret en smule og sat ind i adressen 0. Da den øvre del af adressen (Adr-hi) er 0, og programmet starter på side 0 (page = 0), betyder det at kommandoen 80 ikke behøver at blive brugt her. Længden af et tryk på en knap måles og vises igen. Alle vente (pause) kommandoer er blevet fjernet fra programmet, så det nu fungerer med en højere opløsning.

Address	command	Data	Comment
0	C	C	Skip if S1 = 0
1	3	1	Jmp -1
2	4	0	A = 0
3	7	1	A = A + 1
4	5	4	Port = A
5	C	E	Skip if S1 = 1
6	3	3	Jmp - 3
7	3	7	Jmp - 7

CC 31 40 71 54 CE 33 37

Liste 15: Reaktioner på knappen S1

Hop kommandoen CC i adresse 00 ser på tilstanden af knapper S1. I inaktiv tilstand, er S1 = 1. Betingelsen er derfor ikke sandt, og adresse 01 bliver ikke sprunget over. Der er en relativt hop kommando til start. Programmet gentager kommandoerne i adresserne 00 og 01 indtil knappen bliver trykket. Betingelsen bliver sand og adresse 01 springes over. Dette starter den egentlige måleprocess. A nulstilles, og tælles én op, skrives herefter som output til LED'erne. En anden betinget hop kommando CE placeres i adresse 05. Her er betingelsen for at springe at S1 = 1. Da knappen stadig er trykket ned er betingelsen ikke er sandt. Kommandoen i adresse 06 udføres og forårsager programmet kommer tilbage til adresse 03. Når knappen slippes vil programmet komme til adresse 07 og dermed vende tilbage til starten.

Tast programmet ind og test det. Reaktionstiden er ikke meget hurtigere, tidsenheden er ca. 5 ms.

Det originale program er stadig i memorien, fra adresse 34, det kun er den lave del af memorien der er overskrevet. Skriv derfor et lille program der hopper absolute til adresse 34.

Address	command	Data	Comment
0	8	3	Page 3
1	9	4	Address = 34

83 94

Liste 16: Absolut hop til tids måle programmet

Det originale program eksempel starter igen. Se også om du kan starte andre ekeempel programmer.

En oversigt over mulige programmer er listet i Appendix.

13 Kommando oversigt

Alle kommandoerne på én gang, dette skulle gøre det enkelt at arbejde med controlleren, følgende tabel indeholder kommandoerne i en kompakt form.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
	Port=	Wait	Jump-	A=	=A	A=	A=	Page	Jump	C*	D*	Skip if	Call	Ret
0	0	1 ms	0	0				0	0	0	0		0	
1	1	2 ms	1	1	B = A	A = B	A = A+1	1	1	1	1	A>B	1	
2	2	5 ms	2	2	C = A	A = C	A = A-1	2	2	2	2	A<B	2	
3	3	10 ms	3	3	D = A	A = D	A = A+B	3	3	3	3	A=B	3	
4	4	20 ms	4	4	Dout = A	A = Din	A = A-B	4	4	4	4	Din.0=1	4	
5	5	50 ms	5	5	Dout.0 = A.0	A = Din.0	A = A*B	5	5	5	5	Din.1=1	5	
6	6	100 ms	6	6	Dout.1 = A.0	A = Din.1	A = A/B	6	6	6	6	Din.2=1	6	
7	7	200 ms	7	7	Dout.2 = A.0	A = Din.2	A = A AND B	7	7	7	7	Din.3=1	7	
8	8	500 ms	8	8	Dout.3 = A.0	A = Din.3	A = A OR B		8	8	8	Din.0=0	8	
9	9	1 sec	9	9	PWM = A	A = AD1	A = A XOR B		9	9	9	Din.1=0	9	
A	10	2 sec	10	10		A = AD2	A = NOT A		A	A	A	Din.2=0	A	
B	11	5 sec	11	11					B	B	B	Din.3=0	B	
C	12	10 sec	12	12					C	C	C	S1=0	C	
D	13	20 sec	13	13					D	D	D	S2=0	D	
E	14	30 sec	14	14					E	E	E	S1=1	E	
F	15	60 sec	15	15					F	F	F	S2=1	F	

14 Løkker

En process skal udføres f.eks 5 gange, til dette udføres en løkke, en hop kommando udføres 5 gange.

Tællevariablen er i dette tilfælde C, først sættes A=5, så sættes C=A. Kommandoen A2 udfører et absolut hop til adresse 02, samtidig reduceres variablen C med 1, når C får værdien 0 udføres hop kommandoen ikke mere

Address	command	Data	Comment
0	4	5	A = 5
1	5	2	C = A
2	1	5	Port = 0101
3	2	8	Wait for 500 ms
4	1	A	Port = 1010
5	2	8	Wait for 500 ms
6	8	0	Page 0
7	A	2	C = C*2
8	3	0	End (Jmp 0)

45 52 15 28 1A 28 80 A2 30

Liste 17: En tælle løkke

Test programmet, det vil vise 0101 og 1010 i hver passage, programmet har ikke blot 5 passager, men 6. Hop kommandoen i adresse 07 udføres 5 gange, men for at komme der første gang bliver lysdiode processen vist en gang, derfor ses lysdiode skiftende seks gange. Skift tællevariablen til 4, og se om blinkende nu kommer 5 gange.

Du kan også bruge tælle løkken så der hopes forlængs, nu vil processen blive udført 5 gange. C loades denne gang med værdien 5. Hop kommandoen I adresse 04 hopper til sig selv, og udgør således slutningen på programmet.

Address	command	Data	Comment
0	4	5	A = 5
1	5	2	C = A
2	8	0	Addr high = 0
3	A	5	C = C*5
4	3	0	Ende
5	1	5	Port = 0101
6	2	8	Wait for 500 ms
7	1	A	Port = 1010
8	2	8	Wait for 500 ms
9	3	6	Jmp -6

45 52 80 A5 30 15 28 1A 28 36

Liste 18: Fem blink

15 Sammenligninger

To talværdier kan sammenlignes. Afhængig af udfaldet af sammenligningen bliver udført et hop. De to værdier der sammenlignes skal være i A og B. I det følgende eksempel sættes A=5 herefter B=A. A henter herefter værdien fra AD1 her er lyssensoren forbundet som i kapitel 4 programmet køres nu kontinuerligt

If AD1 > 5

then: all LEDs on

else: all LEDs off

Alt I alt vil du have en lyssensor der kan tænde og slukke LED's afhængig af lysindfaldet på lyssensoren.

Address	command	Data	Comment
0	4	5	A=5
1	5	1	B = A
2	8	0	Addr high = 0
3	6	9	A = AD1
4	C	1	Skip if A>B
5	9	8	Addr = 8
6	1	F	LEDs = 1111
7	3	4	Addr 3
8	1	0	LEDs=0000
9	3	6	Addr 3

45 51 80 69 C1 98 1F 34 10 36

Liste 19: Simpel skumringsdetektor

Test programmet ved at skygge for sensoren med hånden, du vil opdage at basis funktionen er ok, men du vil også opdage at omkring tænd sluk punktet vil lysdioderne blinke underligt. Kapitel 18 vil vise en forbedret udgave.

16 AND, OR og XOR

To binære værdier kan blive til en tredie. Et eksempel er den AND funktion: Når bit 1 har betingelse 1 og bit 2 har betingelse 1, vil output også blive 1. binære tal med flere bits kan også AND'es f.eks "10 AND 3 = 2" Det bliver forståeligt, når du skriver det i binære tal:

1010 AND 0011 = 0010

Følgende program AND'er input Din til det konstante nummer 3 praktisk dette at de to nedre bit filtreres ud. Indgangsporten har betingelse 1111. AND med 0011 bliver resultatet 0011 på LED. Hvis du slutter en af indgangene E1 eller E2 til GND, vil du se en ændring på LED'erne, men ændringer på E3 og E4 har ingen effekt.

0	6	4	A = Din
1	5	1	B = A
2	4	3	A = 3
3	7	7	A = A AND B
4	5	4	Port = A
5	3	5	Jmp -5

64 51 43 77 54 35

Liste 20:Program der anvender en AND funktion

Skriv programmet om og test også en OR (78) funktion

Change the program and test other logical functions as well. The OR function (78).

Hereafter vil programmet se således ud: 64 51 43 78 54 35

1010 OR 0011 = 1011

Brug hereafter XOR funktionen (exclusive-or, 79) til at inverterer individuelle bits: 64 51 43 79 54 35

1010 XOR 0011 = 1001

17 Underprogrammer

When parts of a program are to be reused, write them into a subprogram. This often saves memory space, and sometimes also a lot of typing. The following example shows use of a subprogram that is called in two places in the main program. The subprogram only contains one instruction (A = A-1) and the return jump command here. This does not save memory, but the example is only meant to demonstrate the CALL and RET commands.

Hvis du skriver dele i dit program der skal bruges flere gange kan du skrive dem i et underprogram, det sparer hukommelsplads, samt at du kun behøver at skrive programmet en gang. Det følgende eksempel viser brugen af et underprogram, som kaldes to steder i hovedprogrammet. Underprogrammet indeholder kun en instruktion (A = A-1) og retur hop kommandoen her. I dette tilfælde sparer vi ikke hukommelse, men eksemplet er kun beregnet til at demonstrere CALL og RET kommandoerne.

Main program:

Address	command	Data	Comment
0	8	0	Addr high = 0
1	D	8	Call 08
2	5	4	Output
3	2	9	Wait for 1 sec
4	D	8	Call 08
5	5	4	Output
6	2	8	Wait for 500 ms
7	3	7	Jump -7

Subprogram:

Address	command	Data	Comment
8	7	2	A = A - 1
9	E	0	ret

80 D8 54 29 D8 54 28 37 72 E0

Liste 21: Kald af et underprogram

Resultatet af dette program er binær tæller der tæller baglængs.

Der er flere brugbare subprogrammer som kan anvendes generelt, de er listet under Appendix

50: Subprogram: Long sound

52: Subprogram: Short sound

53: Subprogram: Any sound, length in A

60: Subprogram: Wait for pushed button S1

68: Subprogram: Wait for pushed button S2

70: Subprogram: Number input with S1 and S2

Underprogrammet fra adresse 60 er kun brugt til danne en tæller der kontrolleres med knappen S1.

Hovedprogrammet er nu relativt kort, da mange af kommandoerne er i underprogrammet.

Address	command	Data	Comment
0	4	0	A = 0
1	5	4	Output
2	7	1	A = A + 1
3	8	6	page 6
4	D	0	Call 60 (wait for push S1)
5	3	4	Jump -4

40 54 71 86 D0 34 Liste 22: Counter controlled via S1

Test the program. If you push S1 ten times, the result should be 1010. Change the program so that the subprogram is used from address 68. Now the counter reacts to S2.

18 Skumningsrelæ forbedret udgave

Et skumringsrelæ skal tænde for lampen, når den omgivende lysstyrke falder under en vis grænse. Når lyset bliver lysere, skal lampen gå ud igen. Det bør sikres, at lyset ikke flimrer på tærsklen mellem lys og mørke. Dette er muligt med en hysterese, dvs. en vis afstand mellem aktivering og deaktivering. Programmet her arbejder med følgende regler: • Hvis spændingen ved AD1 er ikke over 5 er lampen slukket. • Hvis spændingen på AD1 ikke er under 9, er lampen tændt.

Dette tilvejebringer et midter område, hvor output tilstanden ikke ændres. Denne forskel forhindrer LED'erne i at flimrer.

0–5: LEDs off

6–8: LEDs unchanged

9–15: LEDs on

Address	command	Data	Comment
0	1	0	LEDs = 0000
1	4	5	A = 5
2	5	1	B = A
3	6	9	A = AD1
4	C	1	Skip if a>b
5	1	0	LEDs = 0000
6	4	9	A = 9
7	5	1	B = A
8	6	9	A = AD1
9	C	2	skip if A<B
A	1	F	LEDs=1111
B	3	A	Jump -10

10 45 51 69 C1 10 49 51 69 C2 1F 3A

Liste 23: skumringsrelæ med hysterese

19 LED-Dæmpning

Målet med dette program eksempel er en styrbar LED-lampe. Lysstyrken af en LED på PWM output skal kunne indstilles på en knap. Du kan kort trykke på en knap for at komme til næste lysstyrke, eller du kan holde knappen nede for en kontinuerlig ændring af lysstyrken.

I programmet er skip kommandoer, som du allerede kender anvendt. Hvis den pågældende knap ikke er trykket, springes over kommandoen der øger eller reducerer A indholdet. Problemet med dette er at vi skal kontrollerer at der ikke forekommer en overskridelse 15-0 eller fra 0 til 15. Dette kræver en indsats. Du er du nødt til at spørge, om den nedre ende (0) eller den øvre ende (15) er nået. Da A altid bruges i

beregninger, skal dets indhold sættes i midlertidig hukommelse. Dette bruges ved at sætte værdien variablen C.

Address	command	Data	Comment
0	8	0	Addr high = 0
1	5	9	PWM = A
2	2	7	Wait for 200 ms
3	5	2	C = A
4	4	F	A = 15
5	5	1	B = A
6	6	2	A = C
7	C	2	Skip if A<B
8	9	B	Jump 11
9	C	F	Skip if S2=1
A	7	1	A = A + 1
B	5	2	C = A
C	4	0	A = 0
D	5	1	B = A
E	6	2	A = C
F	C	1	Skip if a>b
10	9	0	jmp 0
11	C	E	Skip if S1=1
12	7	2	A = A-1
13	9	0	Jmp 0

80 59 27 52 4F 51 62 C2 9B CF 71 52 40 51 62 C1 90 CE 72 90

Liste 24: Lysstyrke control via PWM

20 Kode lås

Her vises en simpel kode lås LED på PWM udgang tænder, hvis brugeren indtaster den korrekte nummer rækkefølge. Antallet input skal være præcist via knapper S1 og S2. Følgende program viser input af et enkelt nummer via knappen S1. Som ved programmering, vil det første tryk føre til værdien 0000. Enhver efterfølgende tryk på S1 øger værdien med 1. ved tryk på S2 afsluttes input. I dette tilfælde vil programmet ende i en endeløs løkke.

Address	command	Data	Comment
0	C	C	Skip if S1 = 0
1	3	1	Jmp -1
2	4	0	A=0
3	5	4	Dout = A
4	2	3	Wait for 10 ms
5	C	E	Skip if S1 = 1
6	3	2	Addr = 4
7	C	F	Skip if S2 = 1
8	3	0	End (jmp 0)
9	C	C	Skip if S1 = 0
A	3	3	Addr = 7
B	7	1	A = A+1
C	2	3	Wait for 10 ms
D	C	C	Skip if S1 = 1
E	3	1	Addr = D
F	3	C	Addr = 3

CC 31 40 54 23 CE 32 CF 30 CC 33 71 23 CC 31 3C

Liste 25: Indtast et nummer

Input af et tal er også tilgængelig som et færdigt underprogram fra adressen 70 og fremefter. I stedet for den endeløse sløjfe på linje 08, der er en RET kommando. Underprogrammet efterlades med resultatet af det tal der er tastet ind i A.

Følgende kode lås kalder input et tal tre gange og sammenligner resultaterne til at pre-definerede numre. I dette eksempel er det korrekte input 3, 5, 2. Derefter PWM output er fuldt aktiveret med værdien 15. Ethvert forkert input, fører imidlertid til en endeløs løkke, der kun kan efterfølges af en reset.

PWM output er behandlet som en normal digital port i dette eksempel. Dette er nødvendigt, fordi alle fire udgange A1 til A4 er nødvendige for at taste et nummer ind. Efter hvert input, er de fire lysdioder slettet for at give betragteren så lidt information om den hemmelige kombination som muligt.

Address	command	Data	Comment
0	8	7	Page 7
1	4	3	A = 3
2	5	1	B = A
3	D	0	Call 70
4	C	3	Skip if A = B
5	3	0	End (Jmp 0)
6	1	0	LEDs = 0000
7	4	5	A = 5
8	5	1	B = A
9	D	0	Call 70
A	C	3	Skip if A = B
B	3	0	End (Jmp 0)
C	1	0	LEDs = 0000
D	4	2	A = 2
E	5	1	B = A
F	D	0	Call 70
10	C	3	Skip if A = B
11	3	0	End (Jmp 0)
12	1	0	LEDs = 0000
13	4	F	A = 15
14	5	9	PWM = A
15	3	0	End (Jmp 0)

87 43 51 D0 C3 30 10 45 51 D0 C3 30 10 42 51 D0 C3 30 10 4F 59 30

Liste 26: The number lock

21 Appendix

Liste af eksempel programmerne

Address	command	Data	Comment
0	6	4	a = Din
1	5	1	B = A
2	4	E	A = 1110
3	8	0	Page 0
4	C	3	Skip if A = B
5	9	8	Jmp 8
6	8	2	Page 2
7	9	5	Jmp 5 (25)
8	4	D	A = 1101
9	8	0	Page 0
A	C	3	Skip if A = B
B	9	E	Jmp 14
C	8	2	Page 2
D	9	A	Jmp A (2A)
E	4	B	A = 1011
F	8	1	Page 1

64 51 4E 80 C3 98 82 95 4D 80 C3 9E 82 9A 4B 81

Page 0: Her skiftes mellem eksempelprogrammerne

Address	command	Data	Comment
10	C	3	Skip if A = B
11	9	4	Jmp 4 (14)
12	8	3	Page 3
13	9	0	Jmp 0 (30)
14	4	7	A = 0111
15	8	1	Page 1
16	C	3	Skip if A = B
17	9	A	Jmp a (1A)
18	8	3	Page 3
19	9	4	Jmp 4 (34 Stopwatch S1)
1A	4	3	A = 0011
1B	8	2	Page 2
1C	C	3	Skip if A = B
1D	9	0	Jmp 0 (20 Alternating flash)
1E	8	4	Page 4
1F	9	0	Jmp 0 (40 Stopwatch S1/S2)

C3 94 83 90 47 81 C3 9A 83 94 43 82 C3 90 84 90

Page 1: fortsat – skift mellem eksempel programmerne

Address	command	Data	Comment
20	1	1	Port = 1 (Alternating flash)
21	2	8	Wait for 500ms
22	1	8	Port = 8 (1000)
23	2	8	Wait for 500ms
24	3	4	Jmp -4
25	7	1	A = A+1 (Count up)
26	5	4	Dout = A
27	5	9	PWM = A
28	2	6	Wait for 100 ms
29	3	4	Jmp -4
2A	6	9	A = AD1 (AD/PWM)
2B	5	4	Dout = A
2C	5	9	PWM = A
2D	2	6	Wait for 100 ms
2E	3	4	Jmp -4
2F	F	F	

11 28 18 28 34 71 54 59 26 34 69 54 59 26 34 FF

Page 2: Program eksemplerne: Skiftende lysdioder, optælling af AD/PWM

Address	command	Data	Comment
30	5	4	Dout = A (Random)
31	C	E	Skip if S1 = 1
32	7	1	A = A+1
33	3	3	Jmp -3
34	2	2	Wait for 2 ms (Stop watch S1)
35	C	C	Skip if S1=0
36	3	2	Jmp -2
37	4	0	A = 0
38	2	2	Wait for 2 ms
39	7	1	A = A+1
3A	5	4	Dout = A
3B	C	E	Skip if S1=1
3C	3	4	Jmp -4
3D	3	9	Jmp -9
3E	F	F	
3F	F	F	

54 CE 71 33 22 CC 32 40 22 71 54 CE 34 39 FF FF

Page 3: Program eksemplerne: tilfældighedsgenerator og stopur S1

Address	command	Data	Comment
40	8	6	Page 6 (Stopwatch start/stop)
41	D	0	Call 0 (60 Wait for S1)
42	4	0	A = 0
43	7	1	A = A+1
44	5	4	Dout = A
45	2	3	Wait for 10 ms
46	C	D	Skip if S2=0
47	3	4	Jmp -4
48	D	8	Call 8 (68 Wait for S2)
49	4	0	A = 0
4A	5	4	Dout = A
4B	3	B	Jmp -11
4C	F	F	
4D	F	F	
4E	F	F	
4F	F	F	

86 D0 40 71 54 23 CD 34 D8 40 54 3B FF FF FF FF

Page 4: Program eksempel: stopur start/stop

Address	command	Data	Comment
50	4	F	a = 15 (Long tone)
51	9	3	Jmp 3
52	4	5	A = 5 (Short tone)
53	5	3	D = A
54	1	9	A4 = 1 (Port = 1001)
55	1	1	A4 = 0 (Port = 0001)
56	2	1	Wait for 2 ms
57	1	9	A4 = 1 (Port = 1001)
58	1	1	A4 = 0 (Port = 0001)
59	2	1	Wait for 2 ms
5A	1	9	A4 = 1 (Port = 1001)
5B	1	1	A4 = 0 (Port = 0001)
5C	2	0	Wait for 1 ms
5D	B	4	D = D*4
5E	1	0	Dout = 0
5F	E	0	Return

4F 93 45 53 19 11 21 19 11 21 19 11 20 B4 10 E0

Page 5: Underprogram Lyd output

Address	command	Data	Comment
60	2	3	Wait for 10 ms (Wait for S1)
61	C	E	Skip if S1=1
62	3	2	Jmp -2
63	2	3	Wait for 10 ms
64	C	C	Skip if S1=0
65	3	1	Jmp -1
66	E	0	Return
67	F	F	
68	2	3	Wait for 10 ms (Wait for S2)
69	C	F	Skip if S2=1
6A	3	2	Jmp -2
6B	2	3	Wait for 10 ms
6C	C	D	Skip if S2=0
6D	3	1	Jmp -1
6E	E	0	Return
6F	F	F	

23 CE 32 23 CC 31 E0 FF 23 CF 32 23 CD 31 E0 FF

Page 6: Underprogram vent på S1 og vent på S2

Address	command	Data	Comment
70	C	C	Skip if S1=0 (Button input)
71	3	1	Jmp -1
72	4	0	A = 0
73	5	4	Dout = A
74	2	3	Wait 10 ms
75	C	E	Skip if S1=1
76	3	2	Jmp -2
77	C	F	Skip if S2=1
78	E	0	Return
79	C	C	Skip if S1=0
7A	3	3	Jmp -3
7B	7	1	A = A+1
7C	2	3	Wait for 10 ms
7D	C	C	Skip if S1=1
7E	3	1	Jmp -1
7F	3	C	Jmp -12

CC 31 40 54 23 CE 32 CF 30 CC 33 71 23 CC 31 3C

Page 7: Underprogram knap input

Command table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
	Port=	Wait	Jump-	A=	=A	A=	A=	Page	Jump	C*	D*	Skip if	Call	Ret
0	0	1 ms	0	0				0	0	0	0		0	
1	1	2 ms	1	1	B = A	A = B	A = A+1	1	1	1	1	A>B	1	
2	2	5 ms	2	2	C = A	A = C	A = A-1	2	2	2	2	A<B	2	
3	3	10 ms	3	3	D = A	A = D	A = A+B	3	3	3	3	A=B	3	
4	4	20 ms	4	4	Dout = A	A = Din	A = A-B	4	4	4	4	Din.0=1	4	
5	5	50 ms	5	5	Dout.0 = A.0	A = Din.0	A = A*B	5	5	5	5	Din.1=1	5	
6	6	100 ms	6	6	Dout.1 = A.0	A = Din.1	A = A/B	6	6	6	6	Din.2=1	6	
7	7	200 ms	7	7	Dout.2 = A.0	A = Din.2	A = A AND B	7	7	7	7	Din.3=1	7	
8	8	500 ms	8	8	Dout.3 = A.0	A = Din.3	A = A OR B		8	8	8	Din.0=0	8	
9	9	1 sec	9	9	PWM = A	A = AD1	A = A XOR B		9	9	9	Din.1=0	9	
A	10	2 sec	10	10		A = AD2	A = NOT A		A	A	A	Din.2=0	A	
B	11	5 sec	11	11					B	B	B	Din.3=0	B	
C	12	10 sec	12	12					C	C	C	S1=0	C	
D	13	20 sec	13	13					D	D	D	S2=0	D	
E	14	30 sec	14	14					E	E	E	S1=1	E	
F	15	60 sec	15	15					F	F	F	S2=1	F	

Electronic devices must not be disposed of in the domestic waste! At the end of its service life, dispose of the product according to the relevant statutory provisions. Collection points have been set up for return. You may dispose of your electrical devices there free of charge. Your community will inform you of where such collection points are located.

This product complies with the relevant CE directives if used according to the included instructions. The description is part of the product and must be included when you pass on the product.