Speedcycle Program

I dette program benyttes der subroutiner til det hele. Det er sat op så der er et “main loop” som kalder hver funktion.

Det første der gøres er at sætte “stacken” op så der kan bruges subroutiner. Derefter sættes variabler. En af variablerne sættes i “.db” directivet med alle vores værdier for segmenterne. Denne liste af segmenter bliver gemt i program hukommelsen og det er derfor vigtigt at den gemmes langt nok ”væk” til at selve program koden ikke overskrider den og derfor ødelægger funktionen.
Hvis man ville være sikker kunne man også gemme det i SRAM eller EEPROM. Dette ville dog tage flere cycles da det både skulle gemmes og hentes.

Den første linje hopper til ”Main loop” for at undgå at ”RESET” køres til at starte med.

Den første subroutine i ”main loop” er ”READSWITH”. Denne function indlæser switch værdierne og sørger for at den ikke debouncer vha. et lille delay.
Til sidst hvis værdien er gyldig og ikke debounced så ”flippes” værdien og gemmes i register R17. Den ”flipper” værdien da switchene er aktiv lave og for at kunne bruge switch værdien til korrekt at styre hastigheden.

Den retunerer så til ”Main loop”. ”Main loop” tester derefter om switch er forskellige fra 0. Da hvis de er 0 så skal den branche og starte forefra. (Displayet skal blive på samme segment og ikke køre rundt)

Hvis switches ikke er 0 køres subroutine ”DELAYLOOP” som kører delayet.
Delayet virker ved at der er et ”1ms” delay som består af 2 loops. Uden om dette loop er der endnu et loop. Det yderste loop: ”DELAYLOOP” flytter switch værdien ind i et nyt register og kører så ”1ms” delayet. Når ”1ms” delayet er færdig så tæller den switchværdien ned en gang og derefter gentages ”1ms” delayet indtil switchværdien er 0, hvor den retunerer til ”main loop”.

Delayet er udregnet således:
(1+(4*250)+3)*(Switchværdi) = Clock cycles.

Ved blot 1 ms:
(1+(4*250)+2)*1 = 1003 Clock cycles ~ 1ms.

Ved 255 ms:
(1+(4*250)+3)*255 = 256 020 Clock cycles ~ 256ms.

”Main loop” er nu nået til at selve display segmentet skal skiftes. Derfor kalder den nu subroutinen: ”CYCLEDISPLAY”.

Denne subroutine virker ved først at loade vores ”segment” variable som indeholder alle segmenterne. Da den måde den gemmer dem på er to 8-bit per word, så startes der med at indlæse den lave og høje værdi af variablen til ”ZH” og ”ZL”.

Vi har også variablen ”COUNT” som blev sat til 0 under setup.

Det næste der sker er at ”COUNT” adderes til til ”ZL”. Der benyttes ”ZL” da vi skal bruge den værdi der viser segment a, som er den første segment. (Det segment som sættes til at være tændt i setup)

Der skrives så lige 0 ind i registeret: ”Temp1” for at behandle hvis der kommer carry. Næste kode tilføjer så carry til ”Temp1”.

Herefter skrives den værdi ”Z” peger på ind i ”Display” registeret. Hvorefter det skrives til porten.

Til sidst så skrives der ”5” ind i ”Temp1” registeret for at tjekke om ”COUNT” er lig ”5”. Hvis den er lig ”5” så branches til ”RESET” som nulstiller ”COUNT” og kører så til ”Main Loop”. Den tæller til 5 da der er 5 forskellige segmenter som skal lyser efter tur. Så når den har nået nummer 5 skal den starte forefra. (Dybeset bare et for-loop)

Er den ikke lig ”5” så inkrementeres ”COUNT” og der retuneres til ”Main Loop”.

Ideen i dette er at hver gang ”COUNT” bliver 1 højrer så når den ligges til ”ZL” så peger ”Z” på den næste i rækken af data gemt i ”Z”. Derved så skiftes der hele tiden segment som der skal.

Til sidst i ”Main Loop” så jumper den tilbage til ”Main loop” og processen starter forefra.

Frederik Mazur Andersen
Robtek 2. semester