OBJECTIVE

This example has the following objectives:

- Present a subroutine that can implement a long delay between stepper motor steps
- Introduce the concept of how to achieve double precision delays (4-hex) using single precision (2-hex) variables

PROGRAM EX_LONG_DELAY

This program illustrates the use of subroutine DLAY_SR. This subroutine performs a long delay action using TOC1. Although the delay is in double precision (4-hex), the controlling variable, DELAY, is simple precision. This is one of the special features of this subroutine. DELAY controls the higher 2-hex digits. This is how this works:

- First, clear the OC1F flag, which is bit 7 in TFLG1. When doing this, we use the OR operation in preserve the other bits in TFLG1.
- We load DELAY in accA, which is the higher part of accD. Then, we fill accB, i.e., the lower part of accD, with $00. If DELAY=$XX, then the value in accD is $XX00.
- Next, add TCNT to accD and store the result in TOC1. This is the new time
- Step in place until OC1F is set. When the number of cycles $XX00 have elapsed, you are done. The subroutine return control to the main program.

A stepper motor is controlled by sending binary patterns (step sequences) to its controller board. Eight distinct binary patterns (S0 – S7) are recognized by the stepper motor controller board in your lab. When these patterns are hit in increasing order, one after the other, the motion is called ‘half speed forward’. If only the odd sequences are hit (i.e., every second pattern), the motion is ‘full speed forward’. If the patterns are hit in decreasing order, the motion is, respectively, ‘half-speed backward’ and ‘full speed backward’. The corresponding step sizes are, respectively, +1, +2, -1, -2 (Table 1). Using 2’s complement 8-bit signed numbers convention, these values correspond to the variable STEP taking hex values $01, $02, $ff, $fe.

<table>
<thead>
<tr>
<th>Table 1 Stepper motor speed definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step size</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Half speed forward</td>
</tr>
<tr>
<td>Full speed forward</td>
</tr>
<tr>
<td>Half-speed backward</td>
</tr>
<tr>
<td>Full speed backward</td>
</tr>
</tbody>
</table>

The program is intended for stepper motor control. The stepper-motor speed is controlled through the delay inserted between steps. A short delay ensures a high speed, while a long delay produces a slow speed. The shortest delay that can be physically accommodated by the stepper motor dynamics is 5 ms. For a 2 MHz processor (0.5 μs/cycle), this corresponds to 10,000 ($2710) cycles. The maximum delay is, obviously, $ffff. Since we use a single precision variable, DELAY, we need to round the numbers $2710 and $ffff to 2 significant hex numbers. We get $2800 and $ff00 (To maintain delay integrity, the first number was rounded up, the
second number was rounded down.) Hence, the single precision variable, DELAY, can take values between $28 and $ff.

FLOWCHART AND CODE
The program flowchart is show below. The essential code for this program is shown to the right of the flowchart. The essential code was incorporated into the standard template asm to generate the code file Ex_Long_Delay.asm. Please refer to this file for more coding details, and coding style.

EXECUTION
Open THRSim11. Close the Commands window. Open and assemble Ex_Long_Delay.asm. View CPU registers, timer registers, memory list for variable DELAY. Arrange windows for maximum benefit. Reset registers. Set standard labels (Label/Set Standard Labels). Press the RESET button. Set breakpoints at SWI and RTS. Put $28 in DELAY. Your screen should look like this:
a) Step through the program. The program should initialize X=$1000, and then turn control to the subroutine DLAY_SR.

b) Step through the subroutine. Observe how, first, the flag OC1F is reset.

c) Then note that the value of DELAY is loaded in accA (i.e., in higher part of accD), then $00 is loaded in accB (i.e., the lower part of accD), and finally the value of TCNT is added to accD. The result is stored in TOC1. Your screen should look like this:

Note that the value of TOC1 is close to $2800.

d) Let the program run. It should loop continuously on LABEL5 while checking for OC1F. When TCNT goes over the value of TOC1, the flag OC1F (bit 7 in TFLG1) gets set and the code is ready to return to the main program. Your screen looks like this:
You are done.

e) Repeat the above procedure for other values in \(\text{DELAY}: \$,30, \$,40, \$,80\) and \$ff. Make sure that you understand the way this subroutine works.

WHAT YOU HAVE LEARNED

In this example, you have learned:

- A subroutine that can implement a long delay between stepper motor steps
- How to achieve double precision delays (4-hex) using single precision (2-hex) variables. The single precision variable \(\text{DELAY}\) was used. The variable was loaded into the higher part of double accD. The higher part of accD is accA. Hence, the variable was loaded in accA using single precision arithmetic. The lower part of the double precision accumulator accD, which is the single precision accumulator accB, was loaded with $00. Hence, if \(\text{DELAY}=$XX, at the end of the process accD=$XX00.