

```

1  /* This program counts from 0 to 99 on HEX1 and HEX0
2  * at a rate of 4Hz, pressing any key will stop/start the conter
3  */
4      .text
5      .global _start
6
7  _start:    LDR        R9, =0xFFFFEC600    // A9 private timer address
8            LDR        R4, =50000000      // 0.25 seconds on 200MHz clock
9            STR        R4, [R9]
10           MOV        R4, #0b011         // Start timer and set it to auto-reload
11           STR        R4, [R9, #0x8]
12           LDR        R6, =0xFF200020     // HEX3-HEX0 Address
13           LDR        R7, =0xFF200050     // KEY Address
14           MOV        R8, #BIT_CODES     // Address of BIT_CODES array
15           MOV        R2, #0             // R2 will be the counter
16           MOV        R3, #1             // R3 will determine whether to count or not
17  MAIN:     LDRB       R5, [R7, #0xC]     // Read Edgecapture register
18           CMP        R5, #0
19           BEQ        DELAY              // If Edgecapture is not 0 the a key has been
20           pressed
21  WAIT:     LDR        R5, [R7]           // Poll KEYs to see if the KEY has been released
22           CMP        R5, #0
23           BNE        WAIT              // Wait for KEY to be released
24           MOV        R5, #0xF          // Reset Edgecapture
25           STR        R5, [R7, #0xC]
26           MOV        R4, #1
27           SUB        R3, R4, R3         // Subtract R3 from 1 to invert it (1 <-> 0)
28  DELAY:    LDR        R4, [R9, #0xC]     // Load timer interrupt flag
29           CMP        R4, #0             // Keep on delaying until interrupt flag is 1
30           BEQ        DELAY
31           STR        R4, [R9, #0xC]     // Reset interrupt flag
32
33           CMP        R3, #1             // When R3 = 1, increment counter
34           BNE        DISPLAY
35           ADD        R2, #1
36           CMP        R2, #100           // Wrap around to 0 when R2 > 99
37           BNE        DISPLAY
38           MOV        R2, #0
39
40  DISPLAY:  MOV        R0, R2             // Separate R2 into its digits
41           BL         DIVIDE
42           LDRB       R0, [R8, +R0]      // Get pattern for ones digit
43           LDRB       R1, [R8, +R1]      // Get pattern for ones digit
44           LSL        R1, #8
45           ORR        R0, R1             // Put pattern in the same reg as the tens digit
46           STR        R0, [R6]          // Display counter
47           B         MAIN                // Program infinitely counts/loops
48
49
50  /* Subroutine to perform the integer division R0 / 10.
51  * Returns quotient in R1 and remainder in R0
52  */
53  DIVIDE:   PUSH       {R2,LR}
54           MOV        R2, #0
55  CONT:     CMP        R0, #10
56           BLT        DIV_END
57           SUB        R0, #10
58           ADD        R2, #1
59           B         CONT
60  DIV_END:  MOV        R1, R2             // quotient in R1 (remainder in R0)
61           POP        {R2,PC}
62
63  BIT_CODES: .byte     0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110
64            .byte     0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111
65            .skip     2                  // pad with 2 bytes to maintain word alignment
66

```