Embedded Systems. 6EJ505.

C Tutorial 4: Practicing Simple Embedded C Programming

rev 6.10.16 tjw

Embedded C programming is about taking the general-purpose C language, and applying and adapting it to control a hardware environment. This requires a good knowledge of that environment. The good news is that many embedded programmes can be completed using rather simple elements of C. When writing such programmes, it is essential to be able to declare variables, set up Port Bits, do simple arithmetic and logical operations, construct programs with simple branching and looping, manipulate bytes and bits, set up features of the Configuration Word, and use the simulator. The programming tasks below will develop your skills in these areas, before you start downloading to the hardware.

Write and simulate a C programme for a PIC 18F2420 which:

1. Continuously reads the value of an 8-bit digital word connected to Port B, and outputs it to Port C.
2. Continuously reads the value of a switch connected to bit 1 of Port B, and outputs it to bit 7 of Port C.
3. Continuously increases by 1 the value of an 8-bit internal variable called **counter**, and outputs it to Port B. When the value reaches 255, it overflows to zero.
4. Continuously increases by 2 the value of an 8-bit internal variable called **counter**, and outputs it to Port B. Similar overflow pattern to 3.
5. Continuously increases by 1 the value of an 8-bit internal variable called **counter**, and outputs it to Port B. After the value reaches 25, it returns to zero.
6. The same as 5, but use an alternative method to determine when 25 has been reached.
7. Adds the 4-bit number present on the lower 4 bits of Port B to the 4-bit number present on the lower 4 bits of Port C, and places the result in an internal variable called **result.**
8. Outputs 10 digital pulses, of no special duration, to Port B bit 1, then does nothing.
9. Outputs 10 digital pulses, of no special duration, to Port B bit 1, then outputs 10 similar pulses to Port C bit 1, then repeats from the beginning.
10. Counts the number of times a digital input is pulsed on bit 2 of Port C, and outputs the value of the count as an 8-bit binary number to Port B.
11. Reads the 4-bit number present on the lower 4 bits of Port B and the 4-bit number present on the lower 4 bits of Port C. If they are equal, sets bit 7 of Port B high, otherwise set it low.
12. Reads the 4-bit number present on the lower 4 bits of Port B and the 4-bit number present on the lower 4 bits of Port C. If B > C, sets bit 7 of Port B high, otherwise set it low.