arm Education

Embedded Systems Essentials with Arm: Getting Started

Module 3

TP (3): Thinking point

We've learned about microcontrollers, embedded systems, and the Mbed environment. Now we want to start actually designing and implementing an embedded system. We're aiming to make some pretty smart embedded systems, but we'd better start with a simple one. So, what's the simplest possible embedded system you can think of? Remember, it has to have input and output, and some computing power.

OK, let's have one input, and one output – can anything be simpler than a microcontroller connected to one switch and one light emitting diode? Both switch and LED are classic logic devices, because both can be either on or off. Put another way, the switch can generate one bit of data, and the LED can display one bit of data.

We need to connect our simple switch and LED combination to a microcontroller, with all its cleverness and high speed. This is a non-trivial ask, and that ask is the core of this module. Once we've managed to make these single connections, to these simple devices, then we can extend the techniques we've learned. A switch can become a keypad, an LED can become a display, and interface to a wide range of sensors and actuators can be made.

So that's our task for this module: getting digital data into and out of the microcontroller. It seems simple enough to say that we'll make the connection to the microcontroller through its external pins, but there's never enough of these to suit all the possible forms of data transfer that may be needed. So instead, we're going to make each pin flexible and multi-purpose. Furthermore, the CPU, under the control of our program, is going to need to locate each or any pin and exchange data with it. Remembering that Cortex CPUs are 32-bit, and hence designed to work with 32-bit numbers, it seems unfair to ask them to deal with single bits of information, but sometimes that's necessary. The outcome of all of this is that behind every pin there's going to be quite a complex circuit that we need to be able to understand. Because of their flexible, general-purpose nature, we'll begin to call these microcontroller pins General Purpose Input/Output, or GPIO.

We briefly spoke about programming in the first two modules, and in this module we're going to start practicing our programming skills, with the Mbed API and the C and C++ programming languages.

We'll start with 1s and 0s as our input/output data, and then we'll start connecting these single digits together to form binary numbers. Then, because binary numbers are really awkward for a human to look at and write down, we often switch to hexadecimal.

Write down three columns of numbers, in decimal, binary and hexadecimal, from 0 to 15 (or better to 31). Make sure you can transition between the different number systems, and be ready to use them!