Department of Mechanical, Materials and Manufacturing Engineering

Computer Engineering and Mechatronics MMME/3085

Exercise Sheet 2: Digital input and output

1. A student who is an keen Arduino enthusiast (but has no knowledge of the underlying AVR microcontroller or register-level programming) has written some code for an Arduino Mega 2560 to write a byte of output to eight pins (pins 22-29), noting that any non-zero quantity will be interpreted as a Boolean TRUE value:

```
pinMode(22, OUTPUT); pinMode(23,OUTPUT); pinMode(24, OUTPUT);
pinMode(25, OUTPUT); pinMode(26,OUTPUT); pinMode(27, OUTPUT);
pinMode(28, OUTPUT); pinMode(29,OUTPUT);
digitalWrite(22, outByte & 0x01); digitalWrite(23, outByte & 0x02);
digitalWrite(24, outByte & 0x04); digitalWrite(25, outByte & 0x08);
digitalWrite(26, outByte & 0x10); digitalWrite(27, outByte & 0x20);
digitalWrite(28, outByte & 0x40); digitalWrite(29, outByte & 0x80);
```

   Re-write this code using register-level port access so it takes up less space and runs much, much faster. Note: you will need to look on the Arduino Mega pin mapping sheet to identify what pins 22-29 have in common, and to exploit this. The answer is very, very simple!.

2. The student has realised that only four of the Arduino pins (22-25) should be overwritten, and the other four need to be left alone. Write some register-level code which outputs bits 0-3 of outByte to pins 22-25 respectively without affecting the state of pins 26-29

3. Using the Atmega2560 data sheet (specifically, section 13.4) and the Arduino Mega pinout, work out what these lines of code would do.

```
PORTA=0x20;

PORTC= PORTC | 0x10;

bool pinStatus = PINC  & 0x08;
```