Lecture 1
Background

EE579 Advanced Microcontroller Applications

Dr James Irvine, EEE

DEPARTMENT OF ELECTRONIC & ELECTRICAL ENGINEERING

Background

Course Aims

- To provide advanced competence in the use of industry standard microcontrollers programmed in low and high level languages in real time applications

Background

Course Aims

- Understand trade offs between high level and low level design, and be able to partition design solutions accordingly, and identify appropriate problem/language partitioning in mixed level problems
- Demonstrate advanced competence in low level coding
- Work collaboratively on complex designs, including concepts of specification, modular design, change control and management, and systematic testing
- Interface peripheral hardware devices with real time critical specifications, and including an appreciation of Real Time Operating Systems

Topics

- Review of microcontrollers
- Hardware/software design issues
- Low level coding and assembly language
- Real time control - threading, concurrency, messaging
- Interfacing to complex systems
- Optimising power
- Collaborative group project

Logistics

Delivery

- One lecture each week from Wk 3 Monday 10-11 GH5.15
- One lab each week Wednesday 12-2 RC4.73
  - Depending on numbers we may run a second shift 2-4
- Assessed through labs and group project
- Microcontroller tutorial and reference guide same as EE312 Instrumentation & Microcontrollers
- Lab book/folder should be kept and will be checked as part of coursework

Logistics

Labs

- You will be given your own demo system (yours to keep)
  - Plus a wired pot and piezo in week 4
- Bring this to the lab
- (Book a lab or demo slot each week by selecting a group)
  - Only if the course numbers increase
- Sign the home working Risk Assessment in order to get the kit

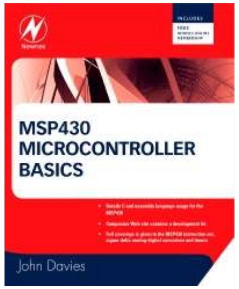
Logistics

Delivery

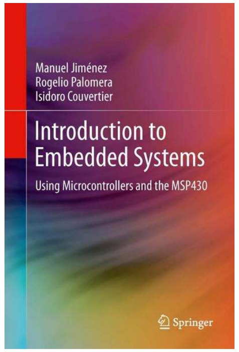
- It is expected that you work in your own time, not just the lab
  - Labs should be used to get support
- Code must be uploaded to myplace by the deadline
  - Request an extension if you are ill
- Code must be demoed in the lab
  - **DO NOT ATTEND THE LAB IF YOU ARE ILL OR SELF-ISOLATING**
  - Code may be demoed in a later week without penalty
  - An additional demo lab has been timetabled in January if needed

Logistics

Recommended Textbook



- MSP430 Microcontroller Basics, 2nd edition
  John H. Davies
  - Published by Newnes
  - ISBN-10: 0750682760; ISBN-13: 978-0750682763
  - Available online through the library


Manuel Jiménez
Rogelio Palomera
Isidoro Couvertier
Introduction to Embedded Systems
Using Microcontrollers and the MSP430
Springer

More advanced supplemental material

- Introduction to Embedded Systems: Using Microcontrollers and the MSP430
  Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier
  Published by Springer
  - ISBN-13: 978-1461431435
  - Available from www.springer.com or Google Play

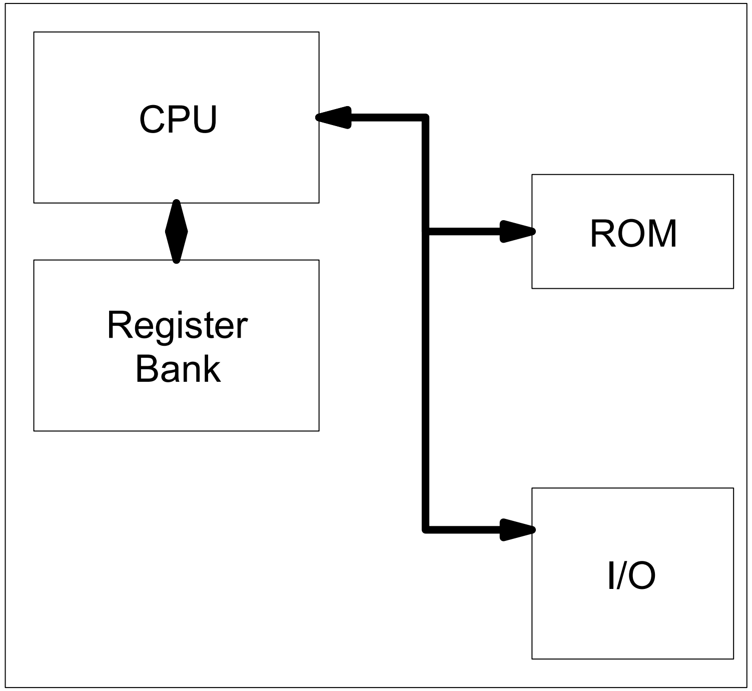
Logistics

Microcontrollers

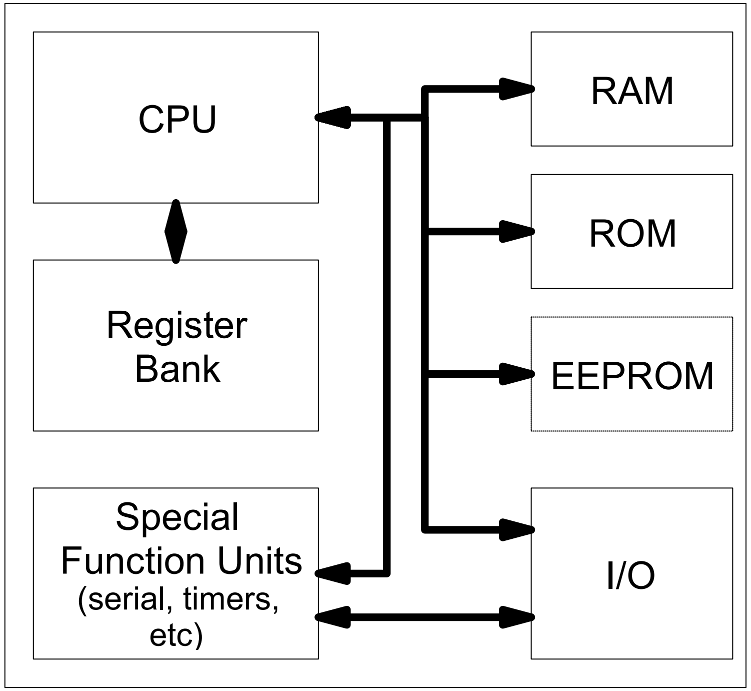
- MSP430 16-bit microcontroller
  - Many versions available
  - Complex version in development system
  - Much more limited (only one general purpose timer, shared interrupt lines, etc) in Launchpad
  - Ultra low power
- Uses IAR development environment
  - Remember to pick the processor when compiling code!
- You will be given an MSP-EXP430G2ET LaunchPad
  - this includes a DIL MSP430G2553

What is an embedded system?

Microcontrollers



Very Basic Microcontroller (Clock, for example)



More Standard System

MSP430 processor

- 16 bit architecture
- 20 bit address (1Mbyte)
- 12 general purpose registers
- Ultra low power
- Low-cost DIL packaged variants available

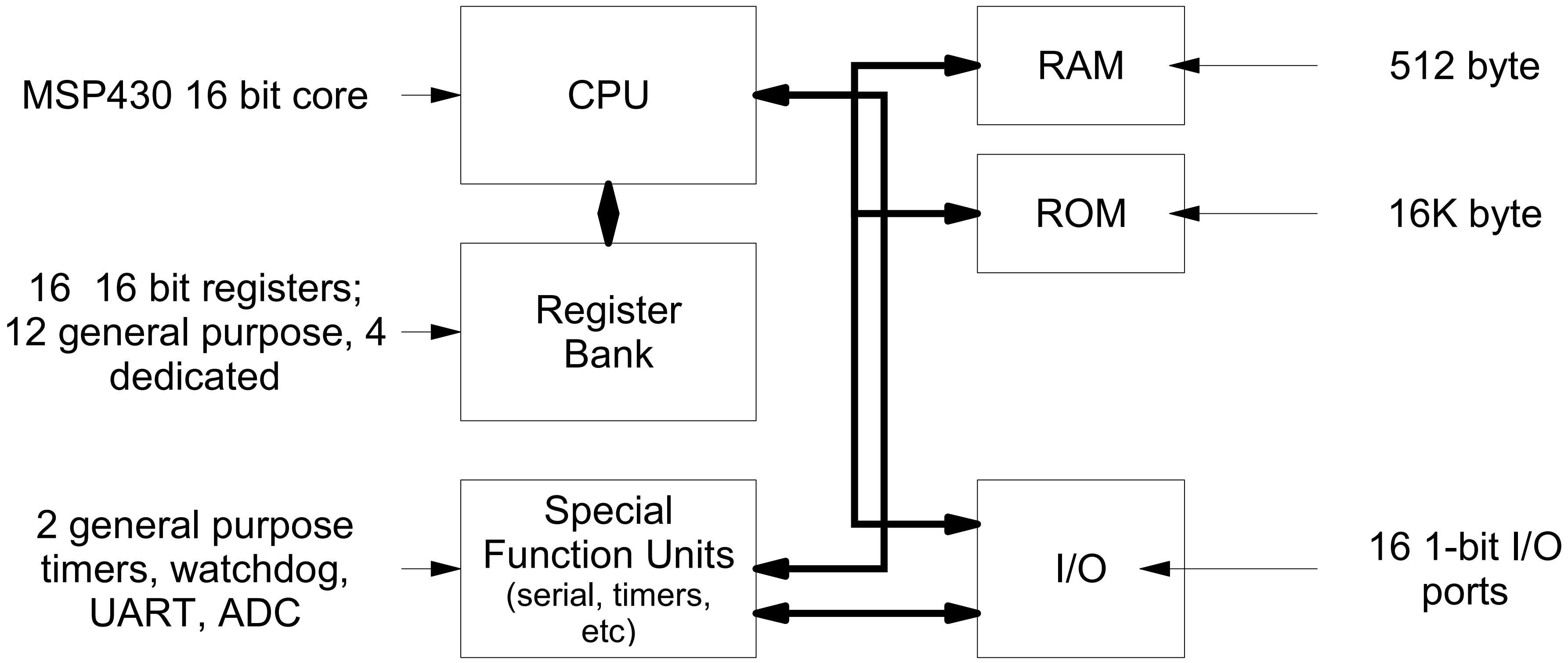
MSP430FR4133 Microcontroller

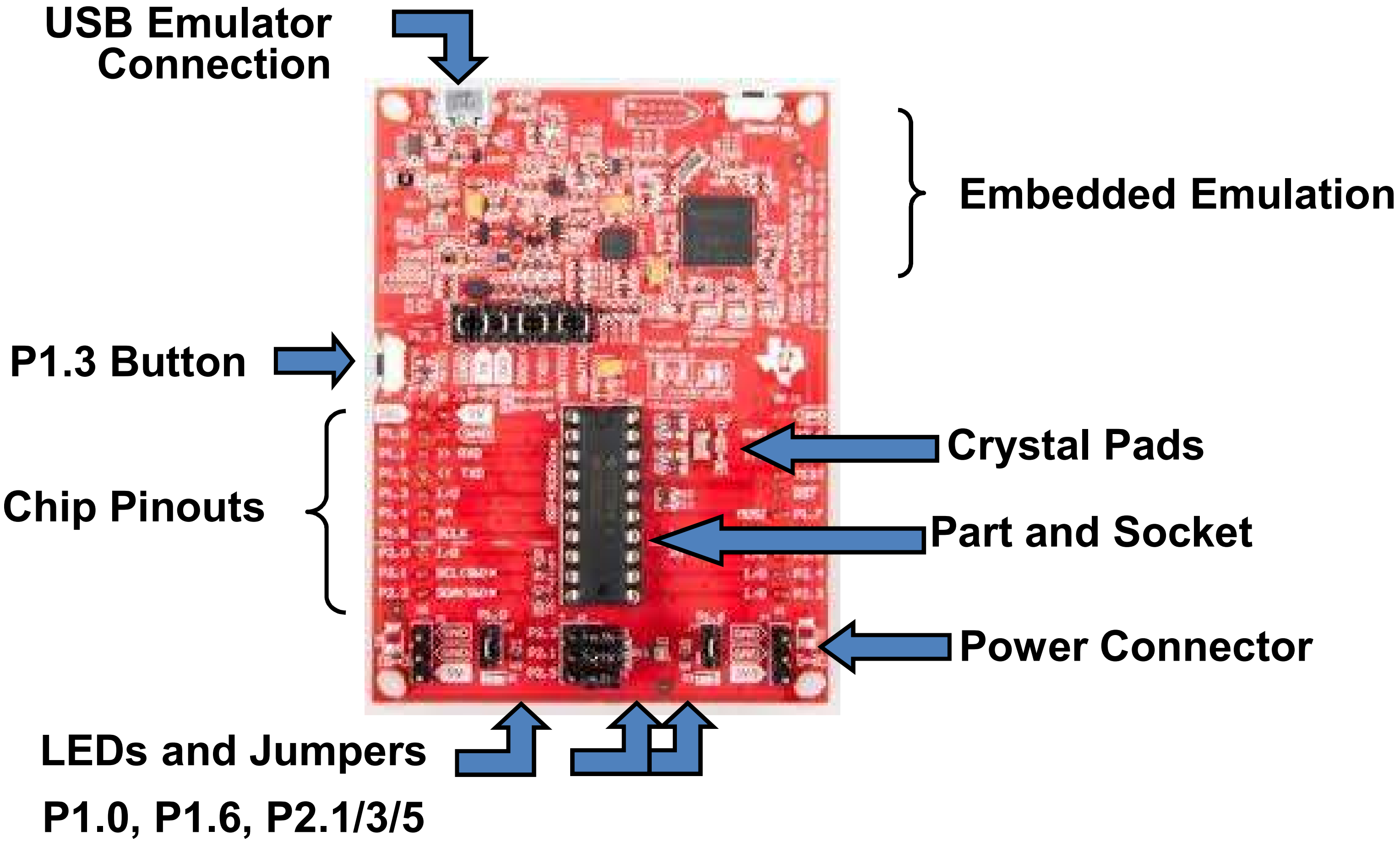
- 16MHz
- 16 KB FRAM
- 2 KB RAM
- 256-segment LCD controller
- 10-channel 10-bit ADC
- 3 16-bit timers
- 60 GPIO
- SPI, I2C and UART Support

MSP430G2553 Microcontroller

- 16MHz
- 16 KB Flash
- 512 bytes RAM
- ~~256-segment LCD controller~~
- 8-channel 10-bit ADC
- 2 16-bit timers
- 16 GPIO (24 in some surface mounted packages)
- SPI, I2C and UART Support

MSP430G2553 Microcontroller


MSP430 16 bit core
CPU
16 16 bit registers; 12 general purpose, 4 dedicated
Register Bank
2 general purpose timers, watchdog, UART, ADC
Special Function Units (serial, timers, etc)
RAM
512 byte
ROM
16K byte
I/O
16 1-bit I/O ports



USB Emulator Connection

Embedded Emulation

P1.3 Button

Crystal Pads

Chip Pinouts

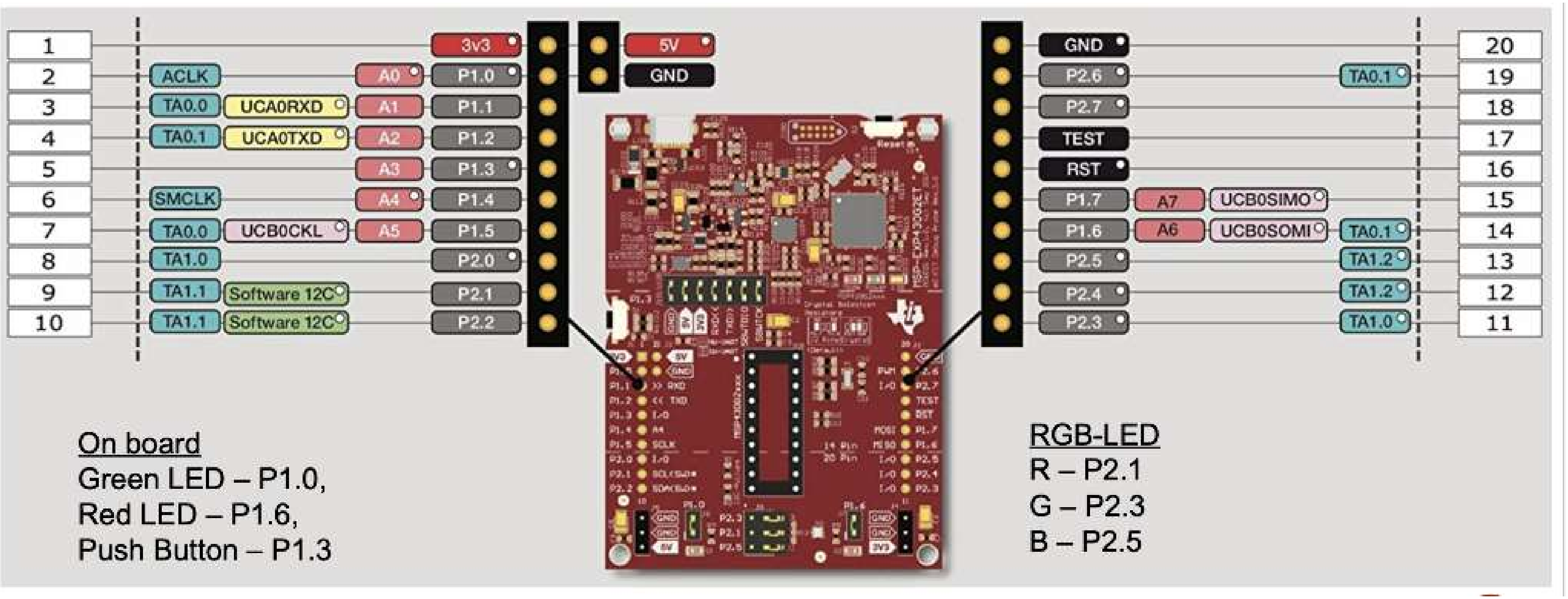
Part and Socket

Power Connector

LEDs and Jumpers

P1.0, P1.6, P2.1/3/5

LaunchPad Dev Board Pinout



1				3v3	
2	ACLK		A0	P1.0	
3	TA0.0	UCA0RXD	A1	P1.1	
4	TA0.1	UCA0TXD	A2	P1.2	
5			A3	P1.3	
6	SMCLK		A4	P1.4	
7	TA0.0	UCB0CKL	A5	P1.5	
8	TA1.0			P2.0	
9	TA1.1	Software 12C		P2.1	
10	TA1.1	Software 12C		P2.2	
				5V	
				GND	

GND				20
P2.6			TA0.1	19
P2.7				18
TEST				17
RST				16
P1.7	A7	UCB0SIMO		15
P1.6	A6	UCB0SOMI	TA0.1	14
P2.5			TA1.2	13
P2.4			TA1.2	12
P2.3			TA1.0	11

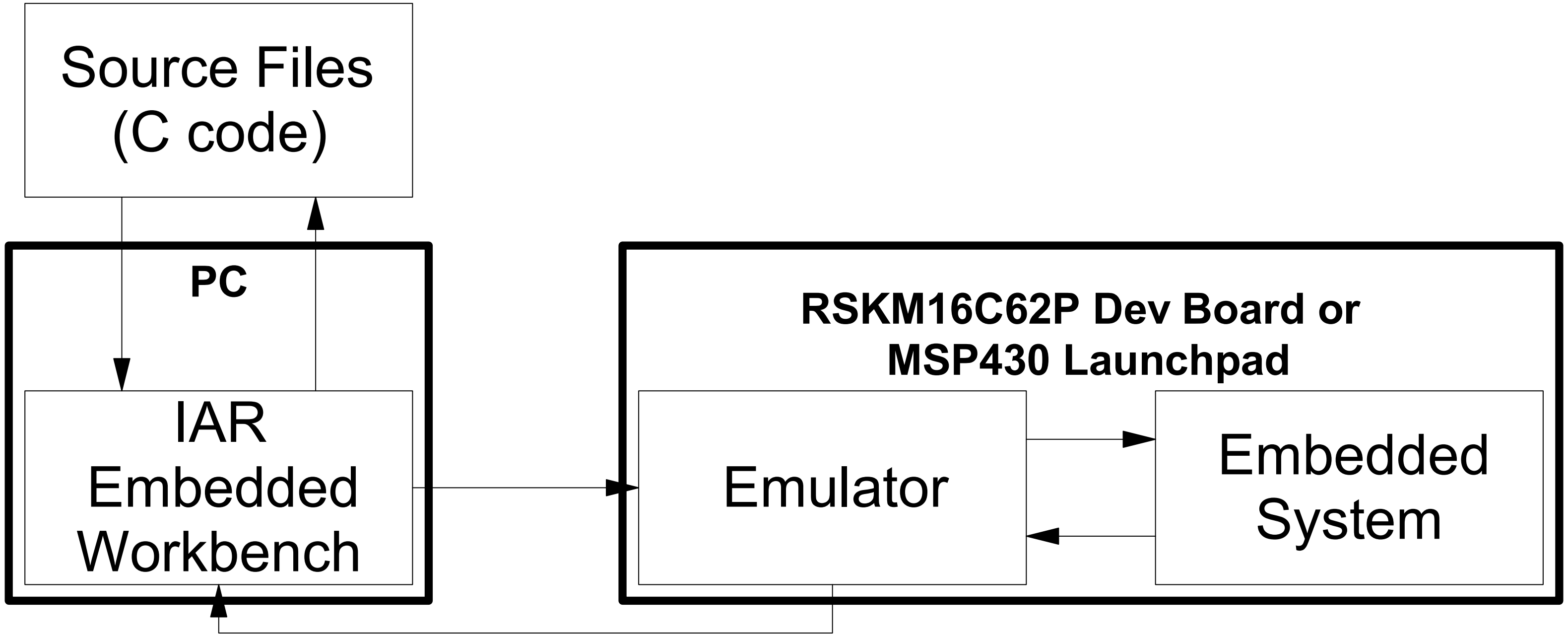
On board
Green LED – P1.0,
Red LED – P1.6,
Push Button – P1.3

RGB-LED
R – P2.1
G – P2.3
B – P2.5

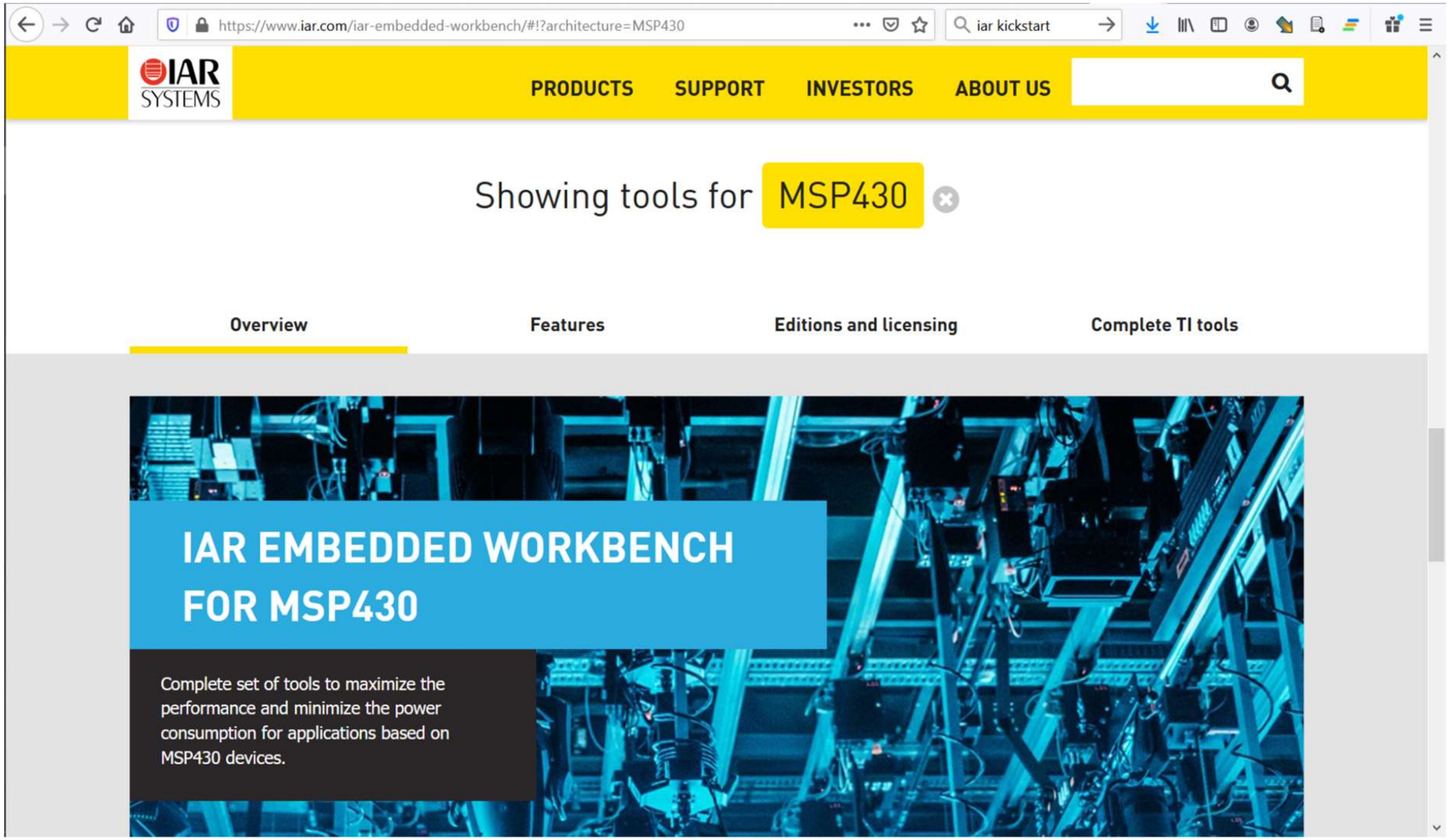
IoT Temperature Sensor

University of Strathclyde Engineering

Development System



IAR Embedded Workbench



https://www.iar.com/iar-embedded-workbench/#!?architecture=MSP430

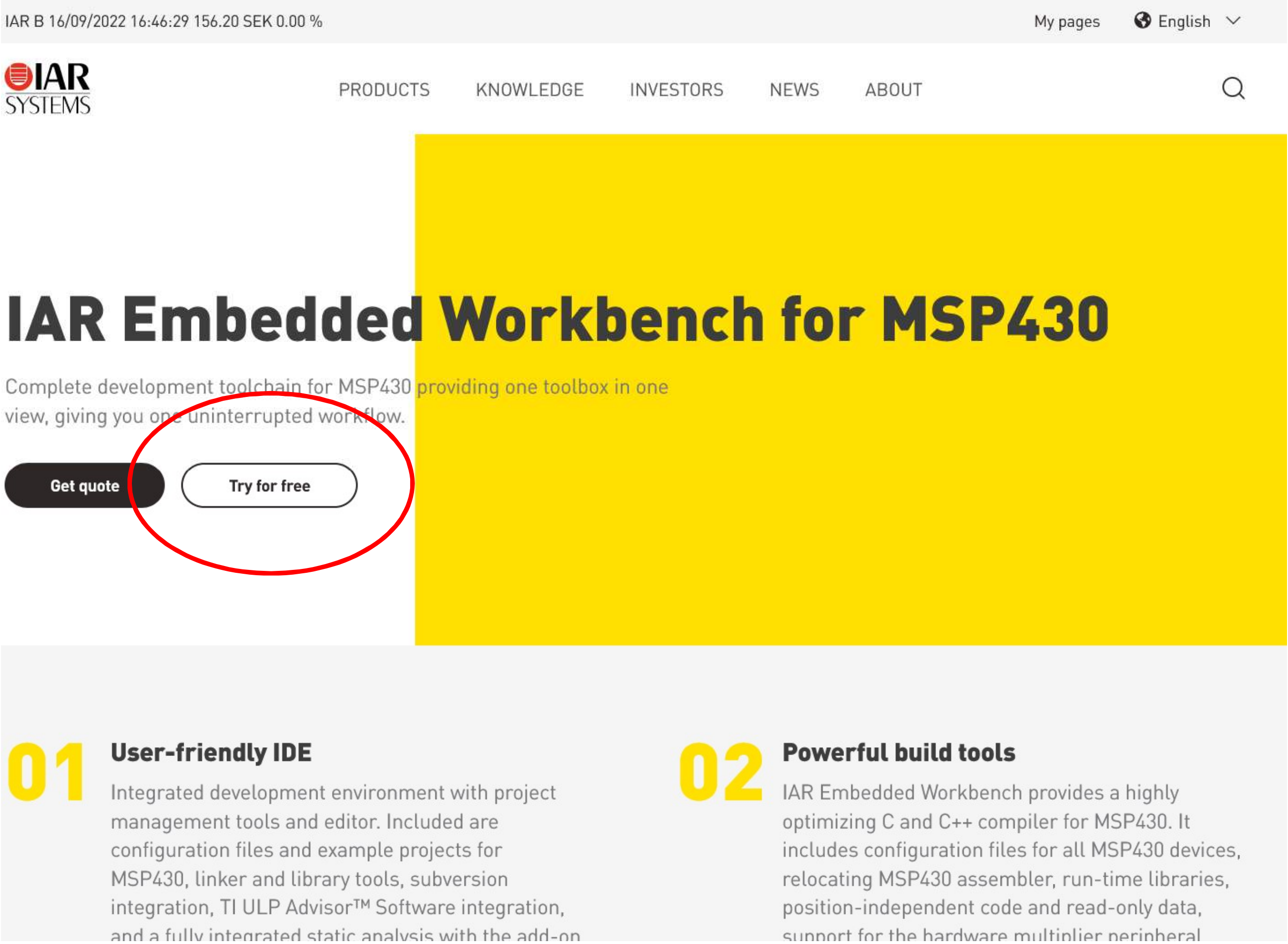
Showing tools for MSP430

IAR EMBEDDED WORKBENCH FOR MSP430

Complete set of tools to maximize the performance and minimize the power consumption for applications based on MSP430 devices.

IAR Embedded Workbench

https://www.iar.com/products/architectures/iar-embedded-workbench-for-msp430/



IAR B 16/09/2022 16:46:29 156.20 SEK 0.00 %

IAR Embedded Workbench for MSP430

Complete development toolchain for MSP430 providing one toolbox in one view, giving you one uninterrupted workflow.

01 **User-friendly IDE**

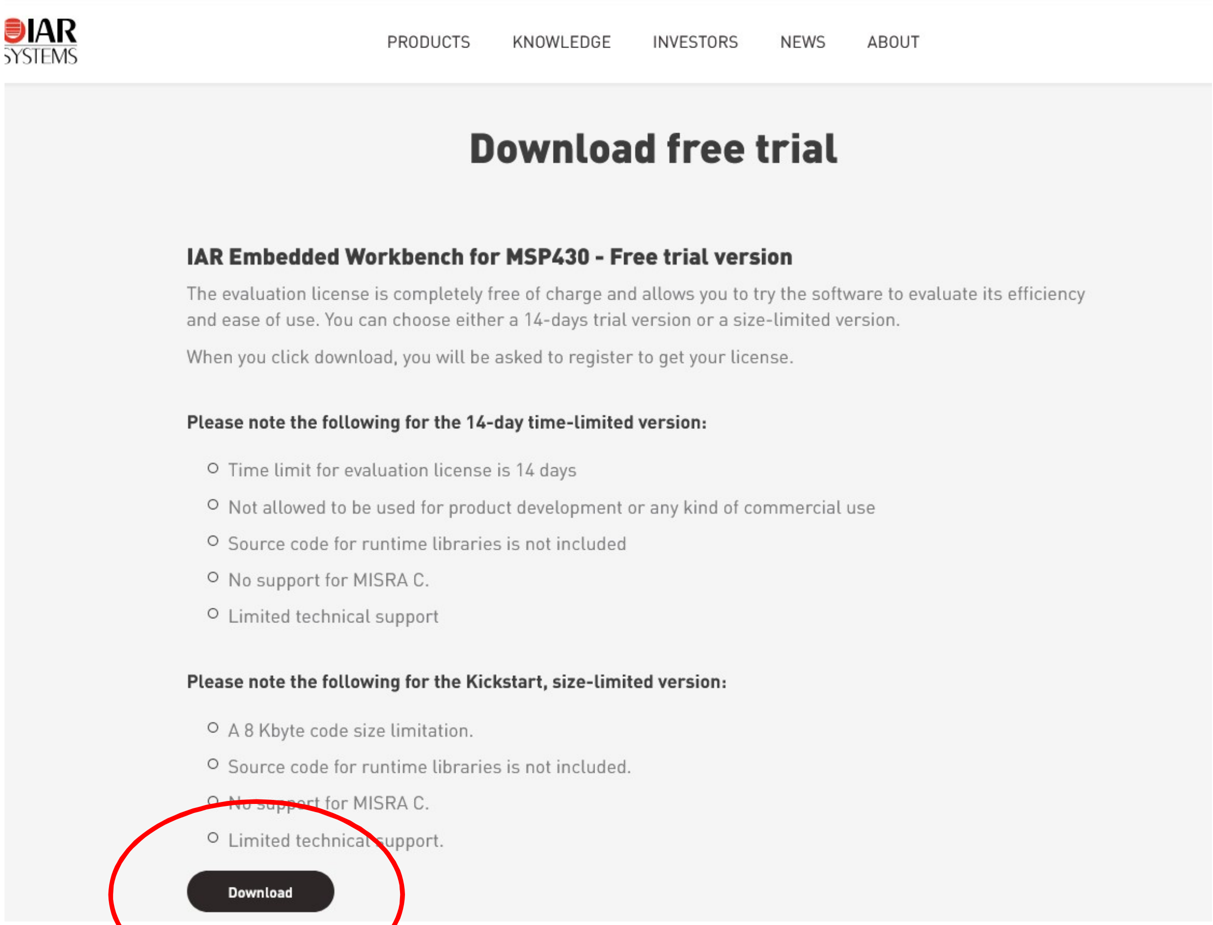
Integrated development environment with project management tools and editor. Included are configuration files and example projects for MSP430, linker and library tools, subversion integration, TI ULP Advisor™ Software integration, and a fully integrated static analysis with the add-on

02 **Powerful build tools**

IAR Embedded Workbench provides a highly optimizing C and C++ compiler for MSP430. It includes configuration files for all MSP430 devices, relocating MSP430 assembler, run-time libraries, position-independent code and read-only data, support for the hardware multiplier peripheral

IAR Embedded Workbench

https://www.iar.com/products/architectures/iar-embedded-workbench-for-msp430/



Download free trial

IAR Embedded Workbench for MSP430 - Free trial version

The evaluation license is completely free of charge and allows you to try the software to evaluate its efficiency and ease of use. You can choose either a 14-days trial version or a size-limited version.

When you click download, you will be asked to register to get your license.

Please note the following for the 14-day time-limited version:

- Time limit for evaluation license is 14 days
- Not allowed to be used for product development or any kind of commercial use
- Source code for runtime libraries is not included
- No support for MISRA C.
- Limited technical support

Please note the following for the Kickstart, size-limited version:

- A 8 Kbyte code size limitation.
- Source code for runtime libraries is not included.
- No support for MISRA C.
- Limited technical support.

IAR Programming Options

- The Embedded Workbench allows the use of C, assembler or ‘Embedded C++’ modules
- Mixed modules (i.e. C with in-line assembler) is not allowed (bad practice anyway)
- If you must, asm(“<opcode>”);
- The IAR C compiler is iccm16c
- The assembler is am16c
- Documentation for both on myplace

Programming the MSP430 in C

- C provides a considerably more ‘user friendly’ interface to the microcontroller’s functions
- Register addresses are defined in include file
- Note
  - Some C functions are available in different sizes to reduce memory (notably printf and scanf)
  - Some additional functions are added in order to make use of the features of the device

Programming the MSP430 in C

- Two run time libraries are provided - one with 32 bit doubles and a much larger one with 64 bit doubles
- With 32 bit doubles:
  - bit is a single bit
  - char has 8 bits
  - short int and int have 16 bits
  - longs have 32 bits
  - pointers have 16 bits (far pointers have 32 bits)
  - floats, doubles and long doubles all have 32 bits

Lab Assignments

- Implement an AOCL
- **A**utomatic
- **O**pen
- **C**rossing
- **L**ocally monitored

Lab Assignments

- Lab 1: Yellow and red light sequence
- Lab 2: Accurate timing
- Lab 3: Klaxon sound and tuning
- Lab 4: As above in assembler
- Lab 5: Complete sequence with driver lights

Lab 1, for demo in week 3

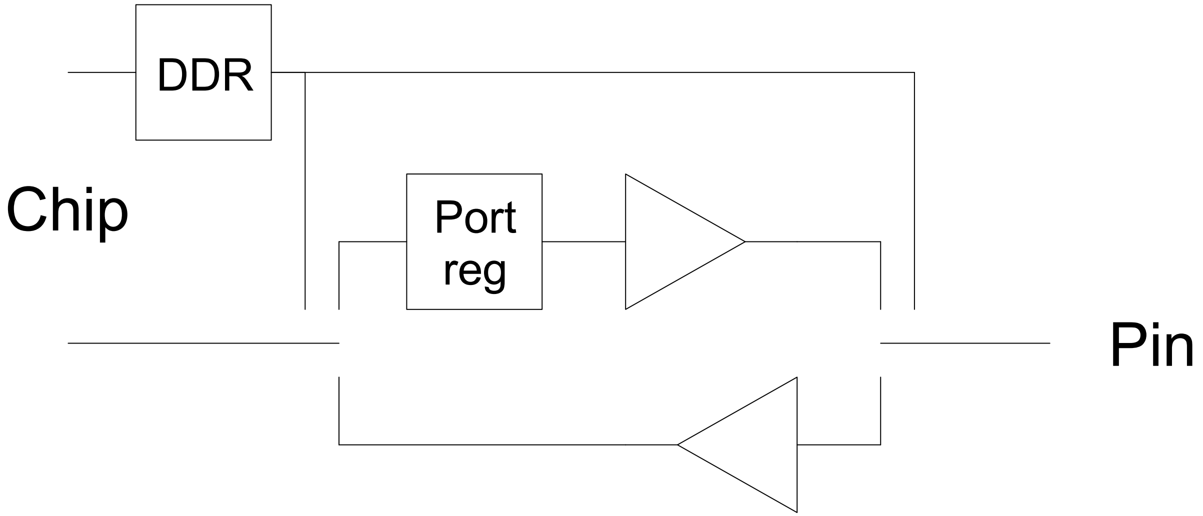
- Implement the traffic lights for an AOCL level crossing. Use the red LED and the multi-colour LED to show a yellow light and then two flashing red lights.
- The sequence should trigger when switch P1.3 is pressed, and continue for as long as the switch is depressed
- A yellow light should show for 3 seconds from the time the switch is depressed
- Two red LEDs should then flash at a rate of 80 flashes/minute
- Include a design in your log book (design pages from log book should be uploaded by end of week 2)

I/O Ports

- I/O ports can be used as general purpose pins (GPIO) **or** through special functions
- Special functions include address, data and control lines for external memory, external interrupts, analogue I/O, serial I/O, clocks, etc
- Default is *usually* GPIO

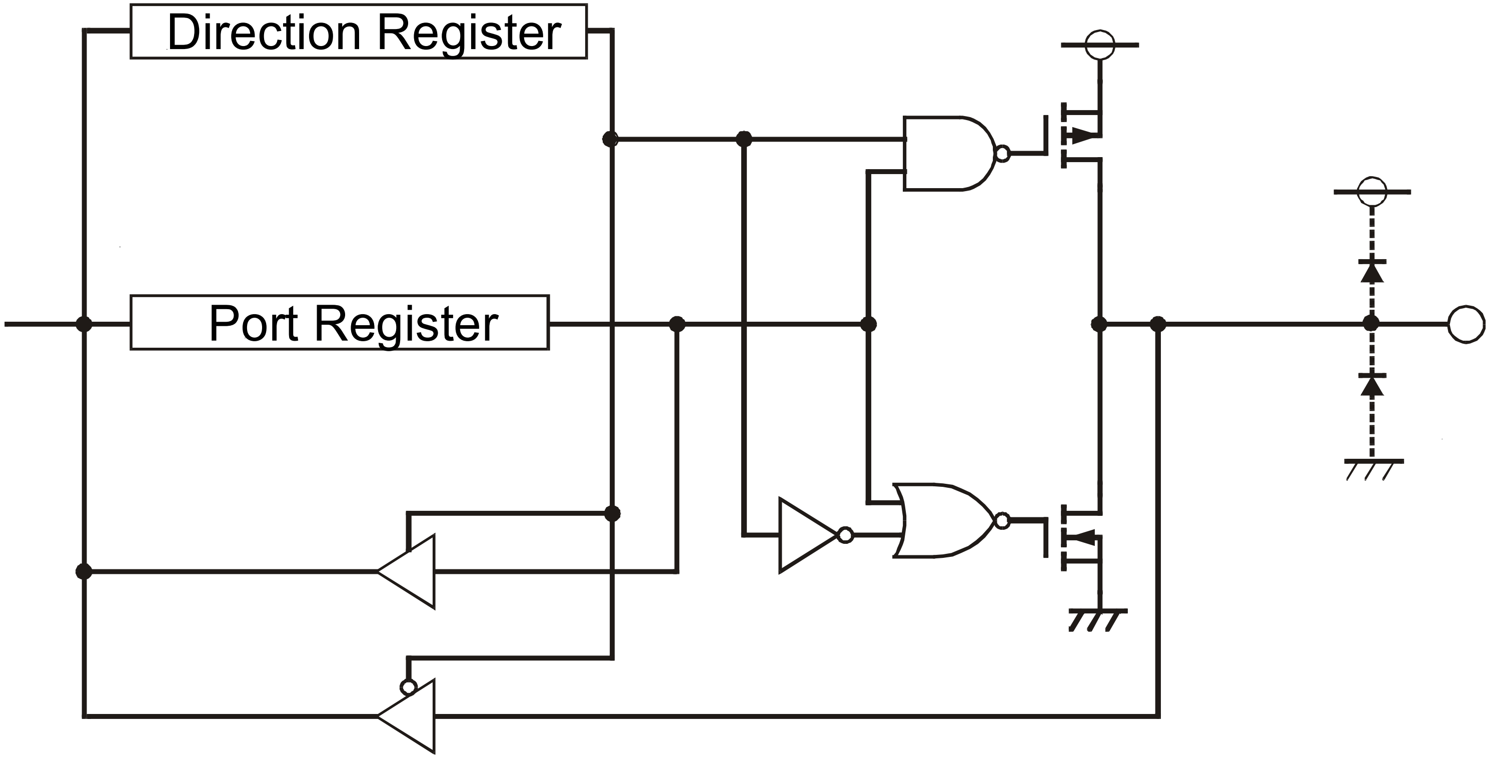
Basic I/O port

- Diagram of a ‘basic’ I/O port



- Usually a Data Direction Register (DDR) value of 0 activates the input buffer (so pins are inputs on reset)

Practical Implementation



Typical characteristics

- Most pins behave as standard MOS (low <0.5V, high > 2V, sink or source 10s of μA
- Some I/O pins may have open collector/drain outputs with or without pullups instead
- Often some of these pins can **sink** several milliamps, and drive LEDs directly
- Watch total chip dissipation limits - this **includes** current sunk from peripherals by the device

Basic I/O

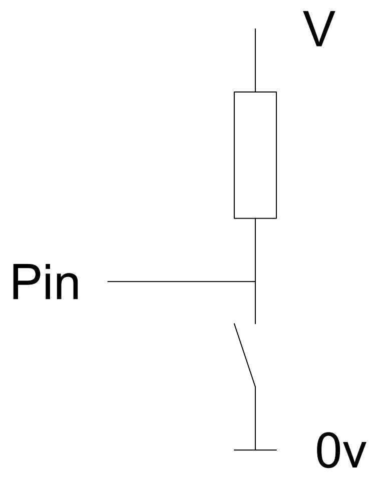
MSP430 I/O Ports

- Grouped in bytes as ports (P1.0, P1.1, etc)
- Multiple control registers, default to 0

Register	Function	Setting
PxDIR	Direction register	0 for input, 1 for output
PxIN	Input register	Value read at pin
PxOUT	Output register	Output register for output; Pull up/down for input (only is PxREN set)
PxSEL	Select register	Selects special function for pin, 0 for GPIO
PxREN	Resistor enable	Enables a weak (~35K) pull up/down when 1

Switches

- Simple switch connection is as follows

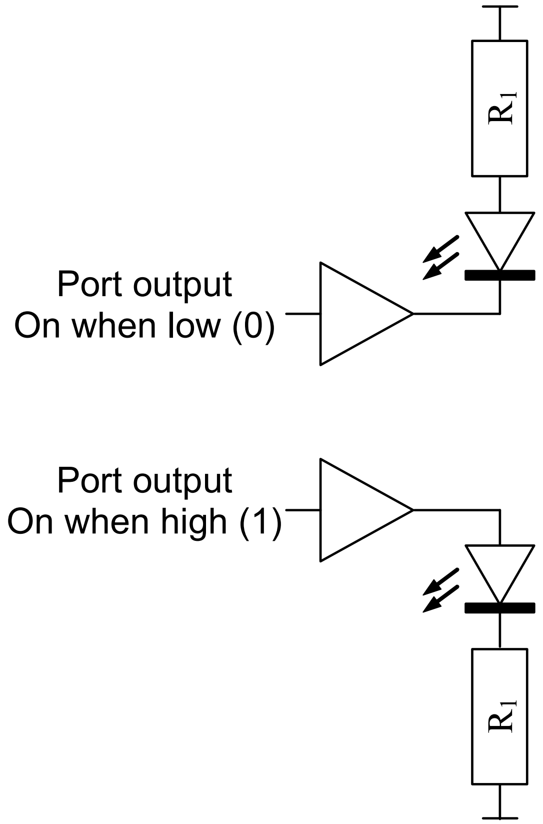


- Depressing the switch results in a ‘0’ at Pin
- Development system has one such switch connected to P1.3

LED Connections

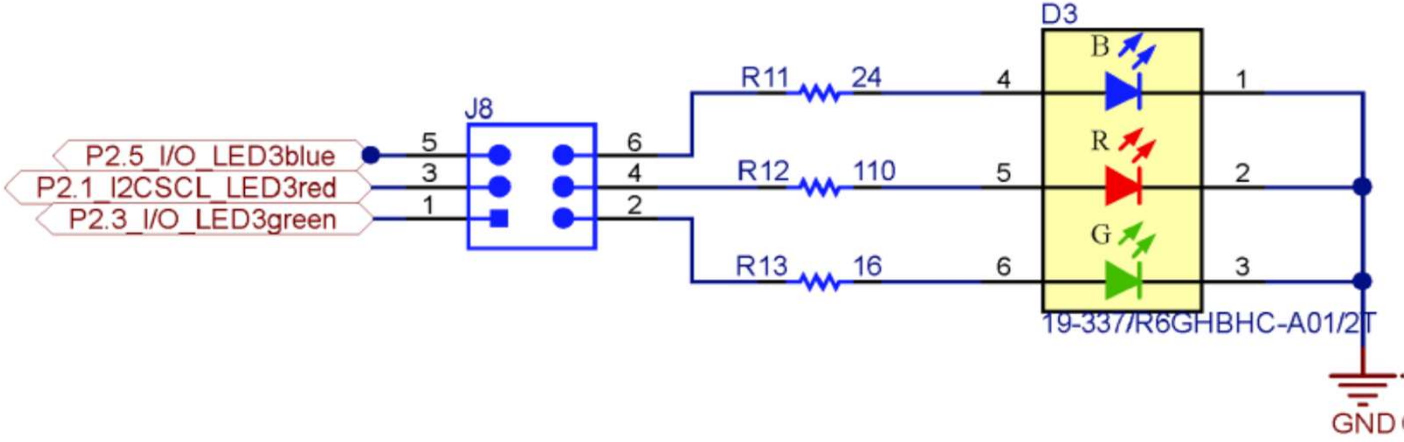
- Most micros can directly drive an LED
- Most common arrangement
  - Micros can often sink more current than they can source
  - Negative logic (0 for on)

- Positive logic
  - 0 for off
  - More intuitive, but less common



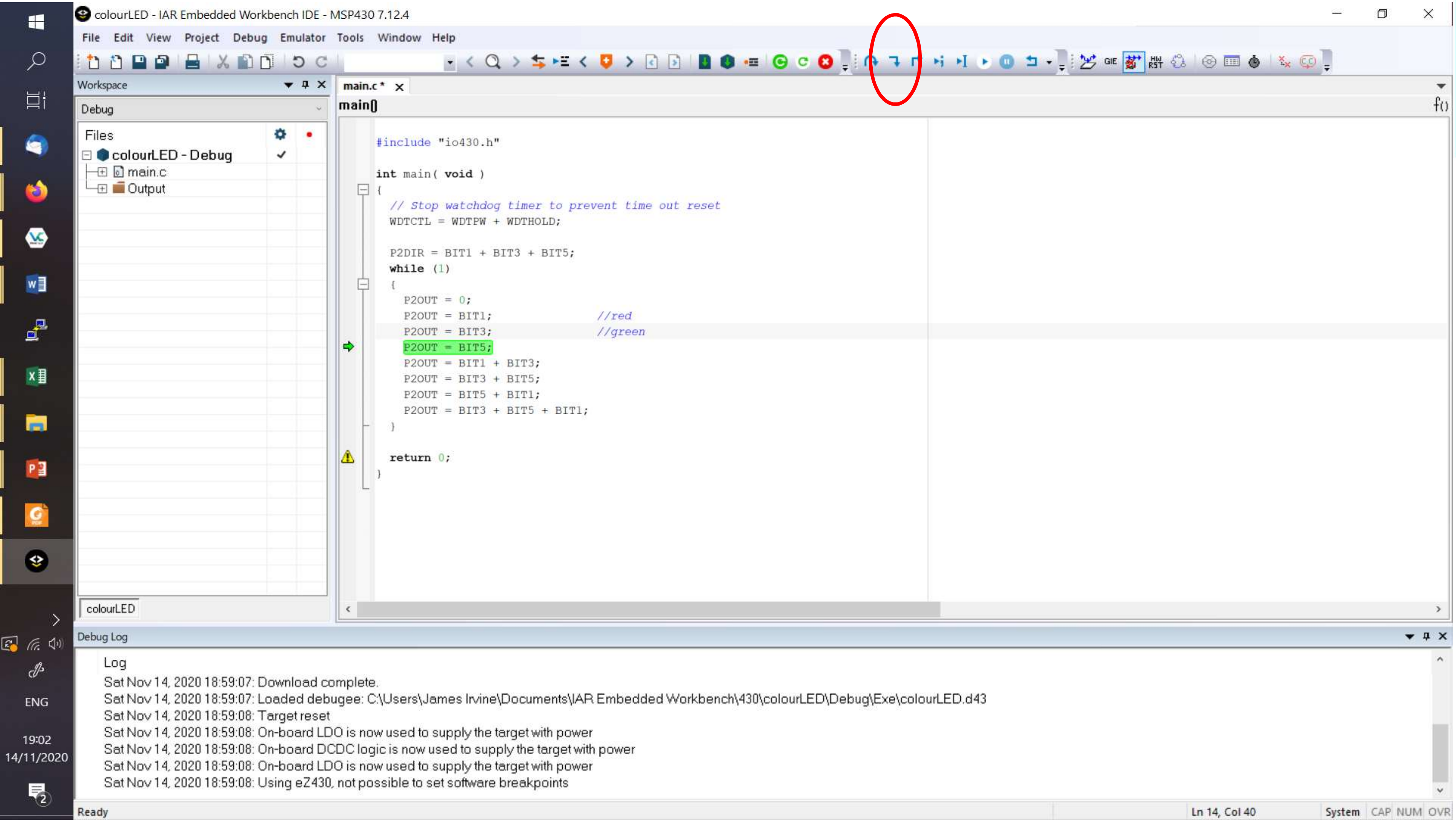
EXP430G2ET Multicolour LED

- Switched from Port 2, lines P2.1, P2.3 & P2.5
- Positive logic (high switches on)
- P2.1 – red
- P2.3 – green
- P2.5 – blue
- All three give white



What colours do I have?

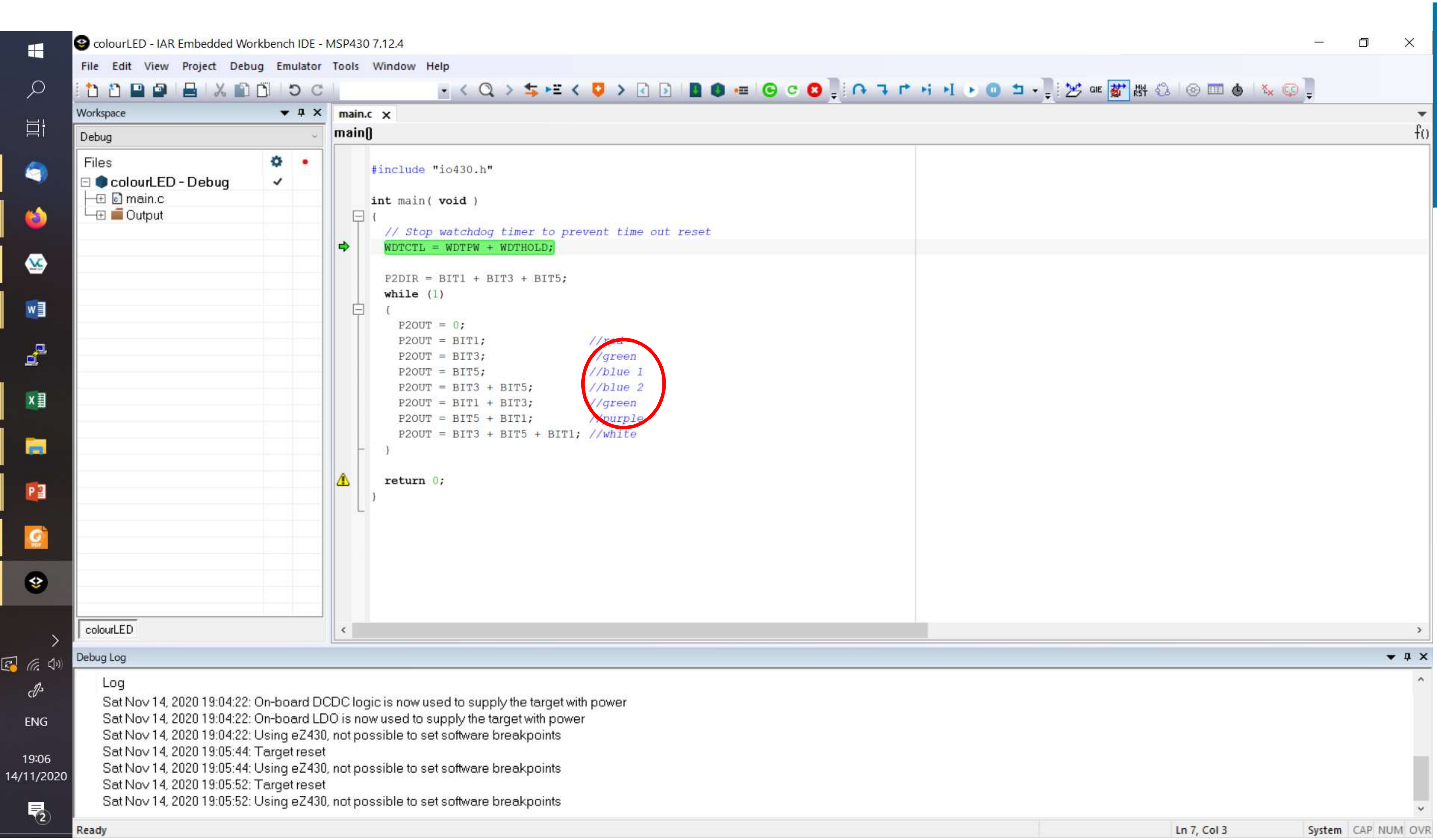
```
#include "io430.h"

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  P2DIR = BIT1 + BIT3 + BIT5;
  while (1)
  {
    P2OUT = 0;
    P2OUT = BIT1;
    P2OUT = BIT3;
    P2OUT = BIT5;
    P2OUT = BIT1 + BIT3;
    P2OUT = BIT3 + BIT5;
    P2OUT = BIT5 + BIT1;
    P2OUT = BIT3 + BIT5 + BIT1;
  }

  return 0;
}
```



```c
#include "io430.h"

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  P2DIR = BIT1 + BIT3 + BIT5;
  while (1)
  {
    P2OUT = 0;
    P2OUT = BIT1;                //red
    P2OUT = BIT3;                //green
    P2OUT = BIT5;
    P2OUT = BIT1 + BIT3;
    P2OUT = BIT3 + BIT5;
    P2OUT = BIT5 + BIT1;
    P2OUT = BIT3 + BIT5 + BIT1;
  }

  return 0;
}
```

Log

```
Sat Nov 14, 2020 18:59:07: Download complete.
Sat Nov 14, 2020 18:59:07: Loaded debugee: C:\Users\James Irvine\Documents\IAR Embedded Workbench\430\colourLED\Debug\Exe\colourLED.d43
Sat Nov 14, 2020 18:59:08: Target reset
Sat Nov 14, 2020 18:59:08: On-board LDO is now used to supply the target with power
Sat Nov 14, 2020 18:59:08: On-board DCDC logic is now used to supply the target with power
Sat Nov 14, 2020 18:59:08: On-board LDO is now used to supply the target with power
Sat Nov 14, 2020 18:59:08: Using eZ430, not possible to set software breakpoints
```



```
#include "io430.h"

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  P2DIR = BIT1 + BIT3 + BIT5;
  while (1)
  {
    P2OUT = 0;
    P2OUT = BIT1;             //red
    P2OUT = BIT3;             //green
    P2OUT = BIT5;             //blue 1
    P2OUT = BIT3 + BIT5;      //blue 2
    P2OUT = BIT1 + BIT3;      //green
    P2OUT = BIT5 + BIT1;      //purple
    P2OUT = BIT3 + BIT5 + BIT1; //white
  }

  return 0;
}
```

```
Log
Sat Nov 14, 2020 19:04:22: On-board DCDC logic is now used to supply the target with power
Sat Nov 14, 2020 19:04:22: On-board LDO is now used to supply the target with power
Sat Nov 14, 2020 19:04:22: Using eZ430, not possible to set software breakpoints
Sat Nov 14, 2020 19:05:44: Target reset
Sat Nov 14, 2020 19:05:44: Using eZ430, not possible to set software breakpoints
Sat Nov 14, 2020 19:05:52: Target reset
Sat Nov 14, 2020 19:05:52: Using eZ430, not possible to set software breakpoints
```

University of Strathclyde Engineering

Change Default Settings

- Remember to set the chip to 430G2553 (General)
- Set the debugger to FET Debugger
    - If left at Simulator (the default), nothing will show on the board

University of Strathclyde Glasgow

DEPARTMENT OF ELECTRONIC & ELECTRICAL ENGINEERING