ECED3901 Design Methods II

LECTURE #6: EMBEDDED PROGRAMMING #1

Colin O'Flynn

What are we covering?

- C Code Refresher
- Introduction to your microcontroller
- Basic embedded design method
  - How <u>not</u> to hack together things
  - Main processing loop
  - RTOS
- Selecting and using a coding standard
- Example framework for your robot code
- Advanced Topics: Static Analysis
- Advanced Topics: Using Doxygen for Code Documentation

C Code Refresher

C Code Refresher

WARNING: Learning to write C code **properly** easily takes a few years of constant work. This section is going to take 15-30 minutes. You can infer what level of quality I can teach you in that time-frame. Please prepare for self-study throughout this course.

Don’t worry, I watched a YouTube video earlier.

Relationships of Several Languages

C

C++

“Arduino”

Java

Python

MATLAB

Basic C Program

```c
#include <stdio.h>
#include "rangefinder.h"
#include "motordriver.h"

int main(void)
{
        int distance_cm;

        setup_world();

        while(1){
                //Read ranger
                distance_cm = get_range();

                motors_on();

                //Stop if object too close
                if (distance_cm < 200){
                        motors_off();
                }
        }

}
```

Simple C Tests (w/o micro)

http://cpp.sh/ (C is mostly valid C++)

C++ shell

```
// Example program
#include <stdio.h>

int main(void)
{
  printf("Hello Class\n");
  return 0;
}
```

Get URL

Run

options | compilation | execution

```
Hello Class
```

Exit code: 0 (normal program termination)

C++ Shell, 2014-2015

Compiler Outputs

options | compilation | execution

```
 In function 'int main()':
5:15: error: 'test_func' was not declared in this scope
 In function 'void test_func()':
10:17: error: 'y' was not declared in this scope
10:9: warning: unused variable 'x' [-Wunused-variable]
```

Compiler Outputs

main.c

main.h

main.o

main.lst

main.elf

Main.hex

C Files

```c
#include <stdio.h>
#include "rangefinder.h"
#include "motordriver.h"

int main(void)
{
    int distance_cm;

    setup_world();

    while(1){
        //Read ranger
        distance_cm = get_range();

        motors_on();

        //Stop if object too close
        if (distance_cm < 200){
            motors_off();
        }
    }

}
```

Header Files

- Normally have ".h" extension
- Entire file gets included into your program

```
#ifndef RANGEFINDER_H
#define RANGEFINDER_H
/*
This file includes defines for the range-finder
module
*/

/*
 Read the range of an object infront of robot.
 */
unsigned int get_range(void);

#endif
```

Preprocessor / Macros

command indicates being sent to “preprocessor”

Preprocessor happens *before* code is compiled… simple subs normally

i.e.:

#include “somefile.h” → Includes contents of somefile.h

#define L_GAIN 4.5 → Defines constant

#define LED_ON() PORTB |= (1<<4) → Macro with no arguments

Comments

Two comment types:

```
// End of line comment

/* Specific comment */
```

Don't comment "human version of the code":

```
//Multiply x1 by 5, then add two
x = x1 * 5 + 2;

//Apply gain and offset
x = x1 * 5 + 2;
```

Comments

Be careful nesting comments… i.e.:

```
/* Do something here */
x = Y + 2;

/* Do something else */
Y = x * 4;
```

Comments

Be careful nesting comments… i.e.:

```
/*
/* Do something here */
x = Y + 2;

/* Do something else */
Y = x * 4;
*/
```

Commenting Out

Using // comments has desired effect, but there is a better way too (see next slide):

```
/*
// Do something here
x = Y + 2;

// Do something else
Y = x * 4;
*/
```

Commenting Out

Using // comments has desired effect, but there is a better way too (see next slide):

```
#if 0
  // Do something here
  x = Y + 2;

  // Do something else
  Y = x * 4;
#endif
```

C Headers for AVR-Libc

http://nongnu.org/avr-libc/user-manual/

Data Types

Examples:

int = integer

unsigned int = unsigned integer

<u>NOTE:</u>

- On AVR, int = 16-bit
- On larger devices, int = 32-bit or 64-bits
  - Depends on natural integer size the device handles!
  - But never smaller than 16-bits (i.e. AVR is 8-bit machine, but int is 16-bits)

Data Types – Specific Bit Width

Remove uncertainty for embedded code by *forcing* the bit width:

```
#include <stdint.h>
```

uint32_t = unsigned 32-bit int

int32_t = signed 32-bit int

uint16_t = unsigned 16-bit int

int16_t = signed 16-bit int

uint8_t = unsigned 8-bit int

int8_t = signed 8-bit int

Data Types – Specific Bit Width

Remove uncertainty for embedded code by *forcing* the bit width:

```
#include <stdint.h>
```

uint32_t = unsigned 32-bit int

int32_t = signed 32-bit int

uint16_t = unsigned 16-bit int

int16_t = signed 16-bit int

uint8_t = unsigned 8-bit int

int8_t = signed 8-bit int

C on your Micro

Accessing Registers

Example: Toggling a port pin

11.3.6 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
PORTA = 0xFF;
```

Setting Bits

```
PORTB = PORTB | (1<<4)
```

```
PORTB |= (1<<4)
```

```
PORTB |= (1<<4) | (1<<5)
```

Clearing Bits

```
PORTB = PORTB & ~(1<<4)
```

```
PORTB &= ~(1<<4)
```

```
PORTB &= ~((1<<3) | (1<<4))
```

What are all these macros?

```
#define sbi(port,bit)      (port) |= (1 << (bit))
```

sbi = Set Bit

```
#define cbi(port,bit)      (port) &= ~(1 << (bit))
```

cbi = Clear Bit

```
#define _BV(a)    (1<<a)
```

BV = Bit Value

What are all these macros?

- sbi/cbi are *obsolete*, as not required anymore. Suggested not to use.
- The _BV() macro is still used, and depends on your personal thoughts of which is clearer:

```
PORTB |= _BV(1) | _BV(5)

PORTB |= (1<<1) | (1<<5)
```

Bit Names

Bit	7	6	5	4	3	2	1	0	
(0xBC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
TWCR |= (1<<TWINT) | (1<<TWEN);
```

☺

```
TWCR |= (1<<7) | (1<<2);
```

☹

Basic Programming Rules

Rules to keep you sane

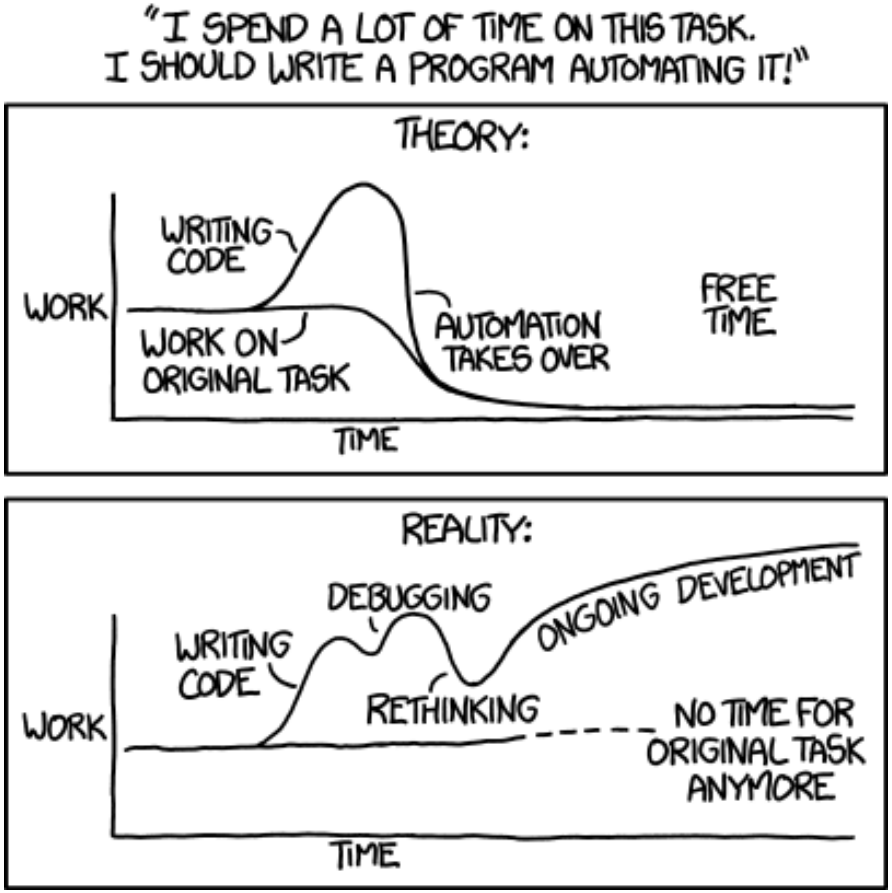
1. Write and test in **SMALL SECTIONS** at a time
   - If your integration of a small code sample stops working, **GO BACK TO THE SMALL SAMPLE**
   - Reduce all problems to the minimal set, otherwise **NOBODY WILL HELP YOU**
     - Not the TA's
     - Not the technicians
     - Not the internet
     - And *especially* not me

2. Do not make assumptions – test things (more on this in the debugging lecture).

3. Embrace that you will make dumb/frustrating mistakes

Embedded Design

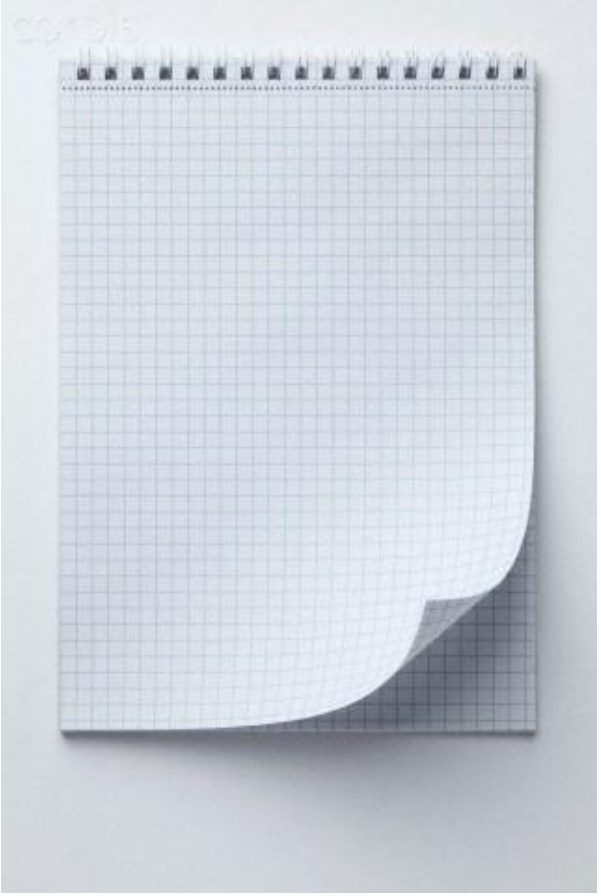
How NOT to Program



https://xkcd.com/1319/

Design Languages?

Design Languages



Colin O’Flynn

Event Loop Structure

```
#include <stdio.h>
#include <avr/io.h>
#include "sensors.h"
#include "motors.h"
#include "navigation.h"

int main(void)
{
            init_system();
            init_sensors();
            init_motors();

            //Ensure Left and Right motor off
            motorL(0);
            motorR(0);

            printf("System Booted. Built %s %s\n", __TIME__, __DATA__);

            while(1){
                        read_sensors();
                        process_navigation();
                        process_movement();
            }
}
```

RTOS?

RTOS = Real Time Operating System

Basic idea:

- Write your code in *tasks*
- The OS switches tasks for you
- Simplifies ensuring events run on certain times, or after certain events

Example: RTOS for a Car

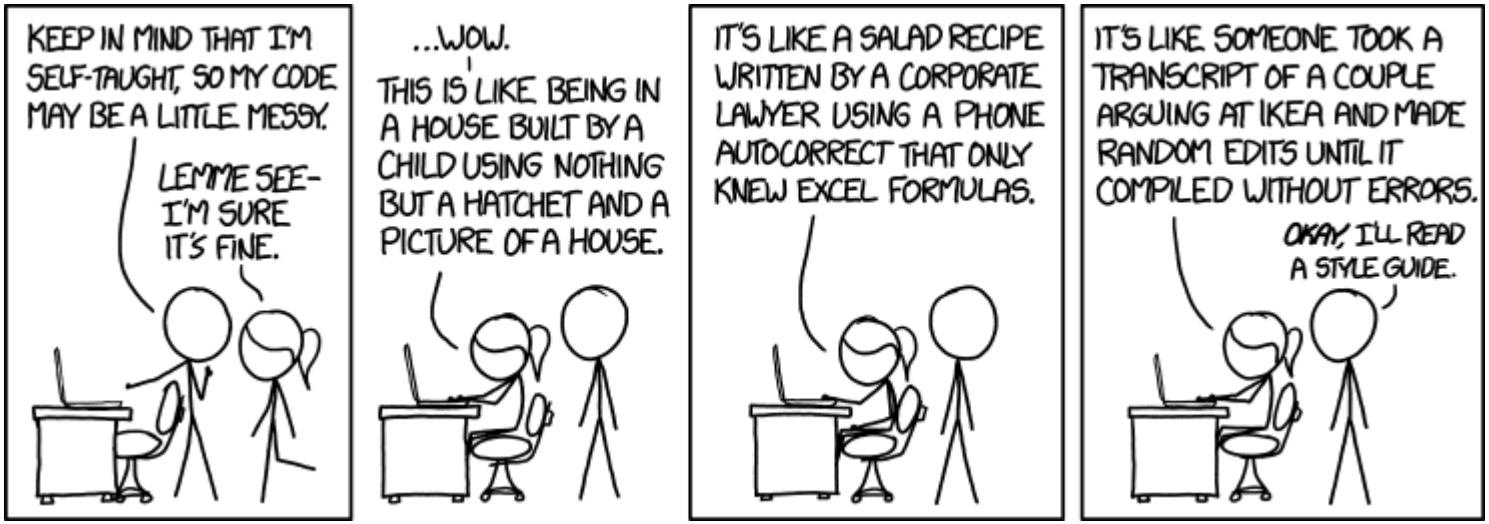
Coding Standard

Example C Coding Standard

http://users.ece.cmu.edu/~eno/coding/CCodingStandard.html

Don’t Fight

- There may be personal preferences about standards, comments, etc.
- Worthwhile to adapt to suit your own preferences (where such preferences are *reasonable)*
- More important you remain consistent, as helps other people looking at code know your intentions



http://xkcd.com/1513/

Static Analysis

The Problem

Specification

Design

C Implementation

Executable

The Problem

- Compiler does not know what your *intentions* were
- Only flags things as warnings that are almost certain to result in errors
- Happily compiles syntax-correct but design-incorrect programs
- Require a tool which not only detects possible problems, but maintains some information about variable values to detect out of bound errors

```c
#include <stdio.h>

unsigned int sum20(unsigned char * input);

int main(void)
{
    unsigned char testdata[16];

    //load some stuff
    for(int i = 0; i < 16; i++){
        testdata[i] = i;
    }

    printf("Sum = %d\n", sum20(testdata));

}


unsigned int sum20(unsigned char * input)
{

    int sum = 0;
    for(int i = 0; i < 20; i++){
        sum += input[i];
    }

    return sum;
}
```

It Works! Ship it!

options | compilation | execution

```
Sum = 120
```

Wait… why is it failing in the field!?

PC-Lint to the Rescue!

During Specific Walk:
bug573.cpp 14 sum20([16]) #1
bug573.cpp 24 Warning 662: Possible creation of out-of-bounds pointer (4 beyond end of data) by operator '[' [Reference: file bug573.cpp: lines 14, 23, 24]

```
14      printf("Sum = %d\n", sum20(testdata));
15
16  }
17
18
19  unsigned int sum20(unsigned char * input)
20  {
21
22      int sum = 0;
23      for(int i = 0; i < 20; i++){
24          sum += input[i];
25      }
26
```

Info Flags

```
 9      //load some stuff
10      for(int i = 0; i < 16; i++){

11          testdata[i] = i;
bug573.cpp  11   Info 734:  Loss of precision (assignment) (31 bits to 8 bits)
12      }
```

```
27      return sum;
bug573.cpp  27   Info 732:  Loss of sign (return) (int to unsigned int)
```

A Trickier Example

Following program calculates the average temperature for a month given all the daily temperature readings.

```c
#include <stdio.h>

const int march[31] = {
    8, 5, 7, 2, -4, -14,  -7, -4, -2,  0,
    0, 2, 5, 7,  2,  -4, -14, -7, -4, -2,
    1, 7, 2, 2, -2,  -3,  -4,  6, -4,  3, 9 };

int main()
    {
    unsigned i, count = 31;
    int sum = 0;

    for( i = 0; i < count; i++ )
        {
        sum += march[ i ];
        }
    printf( "The average low temperature in March was"
            " %d degrees\n", sum / count );
    return 0;
    }
```

Running this example…

options | compilation | execution

```
The average low temperature in March was 138547331 degrees
```

…that can’t be right. Let’s check the warnings…

options | compilation | execution

```
Compilation successful
```

!!!!?????

PC-Lint to the Rescue

Gimpel Software

Sample Program

On-Line Demonstrations of FlexeLint and PC-lint (aka FlexeLint for Windows)

Bug of the Month for Mar 2007

Bug of the Month Samples

Temperature readings from Point Barrow, Alaska, give an unusually high value for the average temperature in March. Try as they might they could not attribute this to global warming. Could the bits and bytes be frozen? What's going on?

Modify this example, if desired, and then hit the Analyse Code button

```
const int march[31] = {
    8, 5, 7, 2, -4, -14,  -7, -4, -2,  0,
    0, 2, 5, 7,  2,  -4, -14, -7, -4, -2,
    1, 7, 2, 2, -2,  -3,  -4,  6, -4,  3, 9 };

int main()
    {
    unsigned i, count = 31;
    int sum = 0;

    for( i = 0; i < count; i++ )
        {
        sum += march[ i ];
        }
    printf( "The average low temperature in March was"
            " %d degrees\n", sum / count );
    return 0;
    }
```

Analyse Code

http://gimpel-online.com/cgi-bin/genPage.py?srcFile=bug573.cpp

FlexeLint for C/C++ (Unix) Vers. 9.00L, Copyright Gimpel Software 1985-2014

--- Module: bug573.cpp (C++)

```
     1  #include <stdio.h>
     2
     3  const int march[31] = {
     4      8, 5, 7, 2, -4, -14,  -7, -4, -2,  0,
     5      0, 2, 5, 7,  2,  -4, -14, -7, -4, -2,
     6      1, 7, 2, 2, -2,  -3,  -4,  6, -4,  3, 9 };
     7
     8  int main()
     9      {
    10      unsigned i, count = 31;
    11      int sum = 0;
    12
    13      for( i = 0; i < count; i++ )
    14          {
    15          sum += march[ i ];
    16          }
    17      printf( "The average low temperature in March was"

    18              " %d degrees\n", sum / count );
bug573.cpp  18   Warning 573:  Signed-unsigned mix with divide
bug573.cpp  18   Info 737:  Loss of sign in promotion from int to unsigned int
bug573.cpp  18   Warning 573:  Signed-unsigned mix with divide
bug573.cpp  18   Info 737:  Loss of sign in promotion from int to unsigned int
bug573.cpp  18   Info 705:  Argument no. 2 nominally inconsistent with format (int vs. unsigned int)
    19      return 0;
    20      }
    21
```

```c
#include <stdio.h>

const int march[31] = {
    8, 5, 7, 2, -4, -14,  -7, -4, -2,  0,
    0, 2, 5, 7,  2,  -4, -14, -7, -4, -2,
    1, 7, 2, 2, -2,  -3,  -4,  6, -4,  3, 9 };

int main()
    {
    unsigned i, count = 31;
    int sum = 0;

    for( i = 0; i < count; i++ )
        {
        sum += march[ i ];
        }
    printf( "The average low temperature in March was"
            " %d degrees\n", sum / (int)count );
    return 0;
    }
```

Running fixed version

```
The average low temperature in March was 0 degrees
```

…still issues due to use of integer math, but at least not *drastically* wrong!

Doxygen

Example of Doxygen Page

http://flexibleipfip.sourceforge.net/modules.html

Main Page | Related Pages | Modules | Data Structures | Files

Modules

Here is a list of all modules:

- Flexible IP
  - FIP Architecture
    - Data Types
    - Compiler-specific header
  - Framework
    - Packet Buffer
      - IPv6 Specific Packet Handling
      - ICMPv6 Specific Packet Handling
    - Host Interface for FIP
    - Timer
    - Generic Data Structures
    - Random Number Generation
  - IP Core Functions
    - Interface Module and Data Structure Access
    - Generic Link-Layer Example
    - ICMPv6 Handling
      - ND Data Structure Access
      - Neighbour Discovery (RFC4861)
    - IPv6 Routing
  - 6LoWPAN Implementation
    - 6LoWPAN-ND
  - Debug
    - Trace
      - Trace Levels

Document Everything

```
void fip_icmp6_generateEchoRequest ( fip_ifnum_t      ifnum,
                                     fip_ip6addr_t *  destAddr,
                                     unsigned int     length
                                   )
```

Generate an ICMPv6 Echo Request (ping) to a remote host.

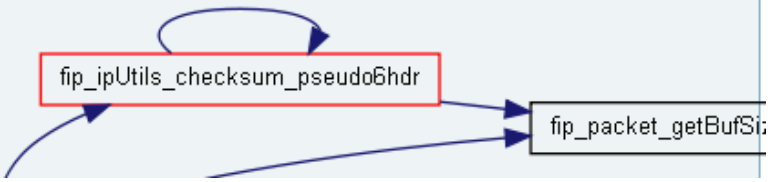
Parameters:

ifnum	Interface to send ping on
destAddr	Destination IPv6 address to send to
length	Length of data to send in ping

References **fip_ip6addr_t::addr**, **fip_icmp6_echoRequest_t::checksum**, **fip_icmp6_echoRequest_t::code**, **fip_icmp6_echoRequest_t::data**, **fip_hostInterface_getMiliSeconds32()**, **fip_icmp6_finalize()**, **fip_if6_selectSrcAddress()**, **fip_packet_activeNew()**, **fip_packet_copyToIP6Hdr_destAddr()**, **fip_packet_getBufPtr**, **fip_packet_setBufSize**, **fip_packet_setToIP6Hdr_hopLimit**, **fip_packet_setToIP6Hdr_nextHeader**, **fip_icmp6_echoRequest_t::identifier**, **RV_OK**, **fip_icmp6_echoRequest_t::sequenceNumber**, and **fip_icmp6_echoRequest_t::type**.

Referenced by **fip_console_ping6()**.

Here is the call graph for this function:



Document Source

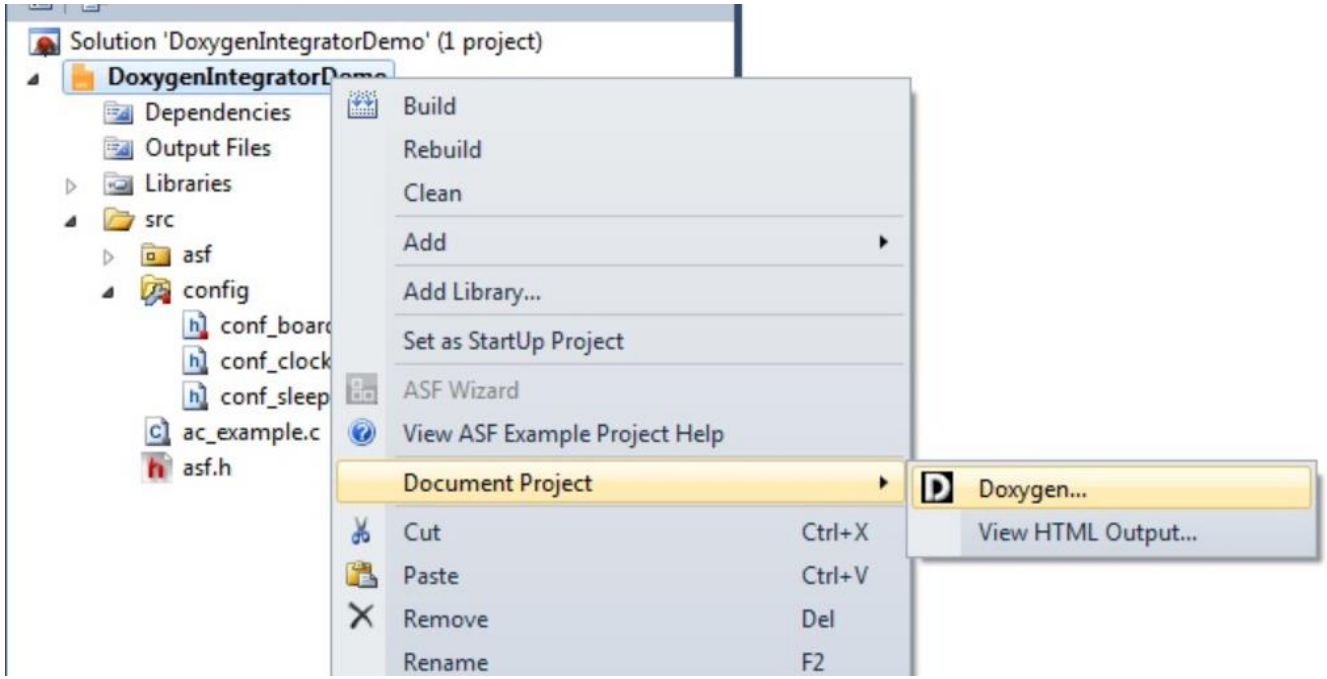
```
/**
 * Generate an ICMPv6 Echo Request (ping) to a remote host
 * @param ifnum Interface to send ping on
 * @param destAddr Destination IPv6 address to send to
 * @param length Length of data to send in ping
 */
void fip_icmp6_generateEchoRequest(fip_ifnum_t ifnum, fip_ip6addr_t * destAddr,
{
    /* Generate the packet */
    if(fip_packet_activeNew(FIP_TLBUF, length + 8, ifnum) != RV_OK) return;

    /* Copy destination over */
    fip_packet_copyToIP6Hdr_destAddr(destAddr->addr);

    /* Select source address */
    fip_if6_selectSrcAddress();
```

Setting up in Atmel Studio

http://www.atmel.com/webdoc/doxygenIntegrator/doxygenIntegrator.Configuration.html



Source Code Managment

The Problem…

Report.docx
Report_Draft.docx
Report_Draft_May5.docx
Report_Final.docx
Report_Final_FINAL2.docx
Report_Final_FINAL2_withedits.docx

Repository

- Central “codebase”
- Lets you see what changes between versions
- Allows ”branches” which don’t affect main codebase

Tutorial example: https://www.youtube.com/watch?v=cFbCusX9bKs

GITHub Demo

GITHub → Very popular host of GIT repositories

Conclusions

- C is complex… lots to learn!
- Always break down problems to verify your code
  - We'll talk on Monday about debugging
- Use resources such as PC-Lint to test code (in your case probably small chunks due to lack of full setup)