Embedded System Design Techniques™

Transitioning from C to C++

Class 5: Getting into the Bits and the Bytes

October 13th, 2017

Jacob Beningo

DesignNews CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS ROHDE&SCHWARZ

Course Overview

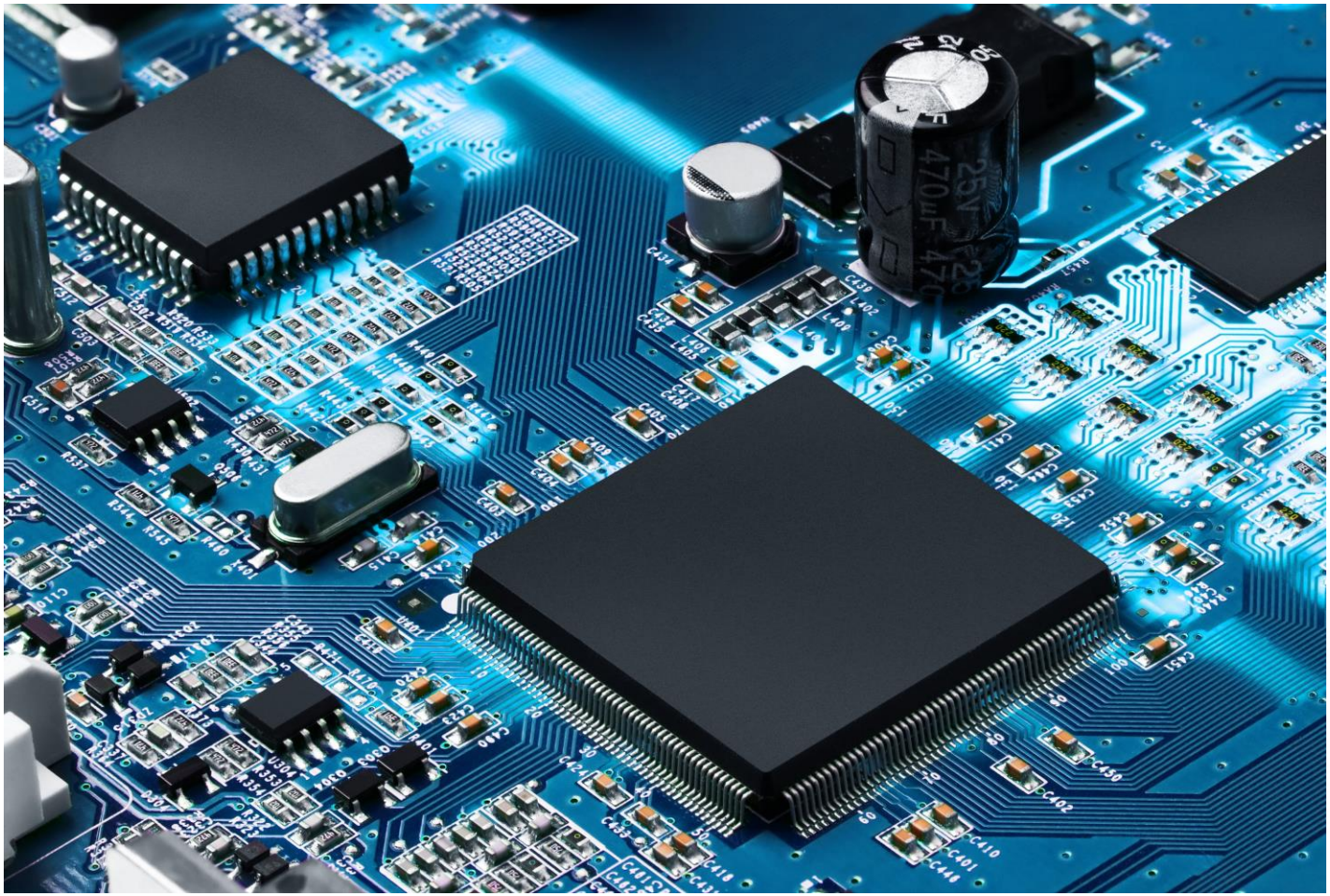
Topics:

- C++ Fundamentals
- Designing a C++ Application
- Beginning the Transition
- Real-Time C++
- **Getting into the Bits and the Bytes**

Session Overview

- Memory Mapping
- Registers
- Cast Operators
- Bit Mapped Structures
- Course Review

Introduction



Traditional Memory Mapping

```
uint32_t * volatile const Gpio_PortA = (uint32_t *) 0x1000U;

// Set the 0 bit high on PortA
*Gpio_PortA |= 0x01;

-------------------------------------------------------------------------

#define GPIO_PORTA              0x1000U
*((volatile uint32_t*) GPIO_PORTA) = 0;
```

Using namespaces for registers

```
namespace mcu
{
  namespace gpio
  {
    constexpr std::uint32_t porta = 0x1000U;
    constexpr std::uint32_t portb = 0x1100U;
    …
  }
}
```

C++ Template Cast Operators

reinterpret_cast – convert unrelated casts

reinterpret_cast<volatile uint32_t>(GPIO_PORTA) = 0

static_cast – cast between types

int data = static_cast<int>(floatVar);

const_cast – adds or removes const from an object
dynamic_cast – converts pointers and references

Accessing Registers

```
*reinterpret_cast<volatile std::uint32_t*>
                                (mcu::gpio::porta) = 0x10;
```

Equilalent to:

```
*((volatile uint32_t*) 0x1000) = 0x10;
```

Templates for Register Access

```
template<typename addr_type,
         typename reg_type,
         const addr_type addr,
         const reg_type val>
class reg_access
{
public:
  static void reg_set()
  {
    *reinterpret_cast<volatile reg_type*>(addr) = val;
  }
}
```

Templates for Register Access

```
reg_access<std::uint32_t,
           std::uint32_t,
           mcu::gpio::porta,
           0x00>::reg_set();

reg_access<std::uint8_t,
           std::uint8_t,
           mcu::gpio::portc,
           0x10>::reg_set();
```

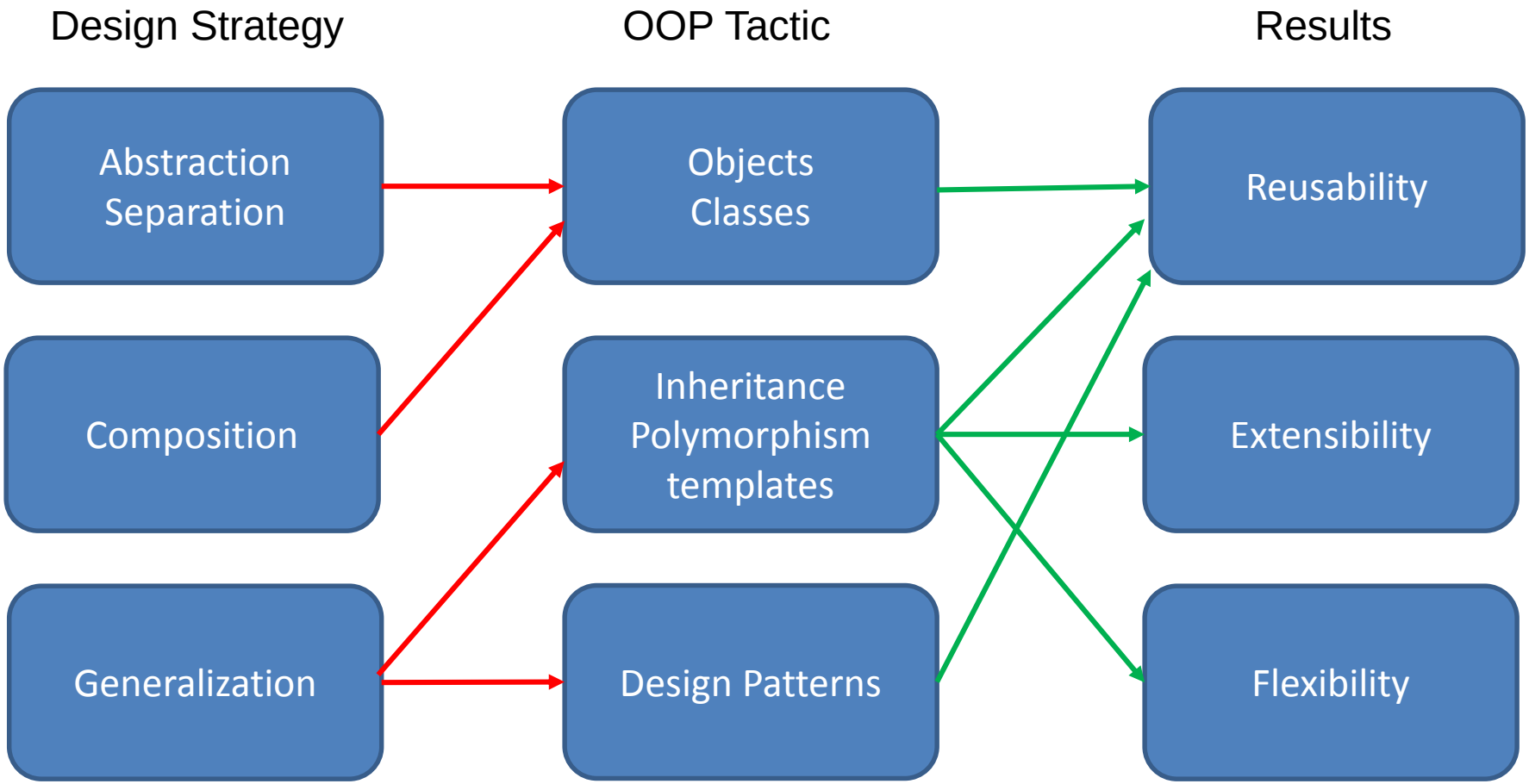
Templates for Register Access

```
template<typename addr_type,
         typename reg_type,
         const addr_type addr,
         const reg_type val>
class reg_access
{
public:
   static void reg_set(){}
   static void reg_get(){}
   static void reg_and(){}
   …
}
```

Bit Mapped Structures

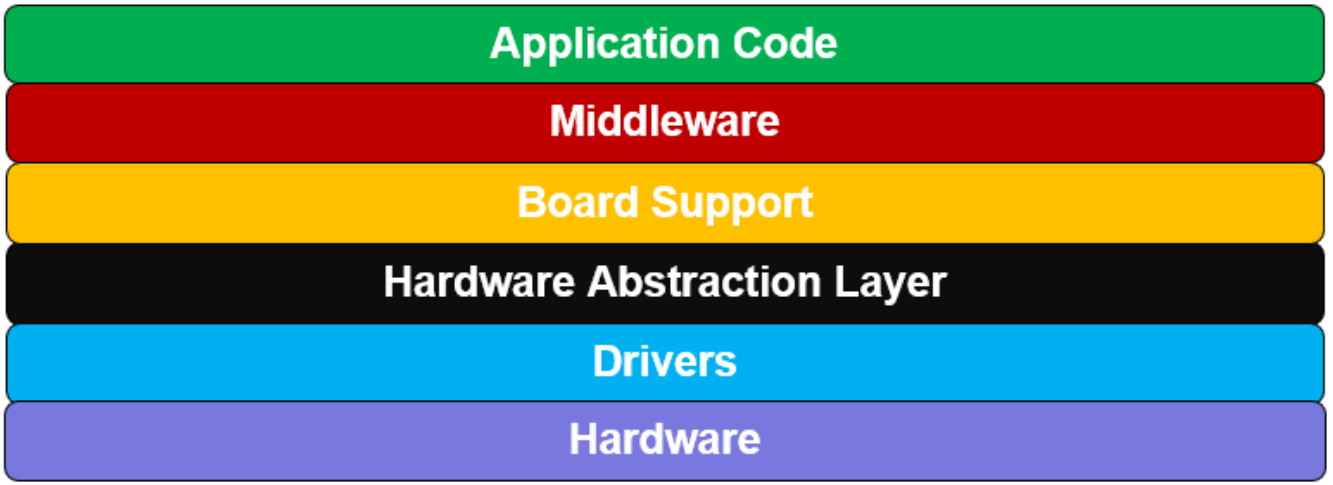
```
typedef struct struct_bit8_t
{
  std::uint8_t b0 : 1;
  std::uint8_t b1 : 1;
  …
  std::uint8_t b7 : 1;
}bit8_t;

reinterpret_cast<volatile bit8_t*>
                              (mcu::gpio::porta)->b1 = 0x10;
```

Review – Design Strategies

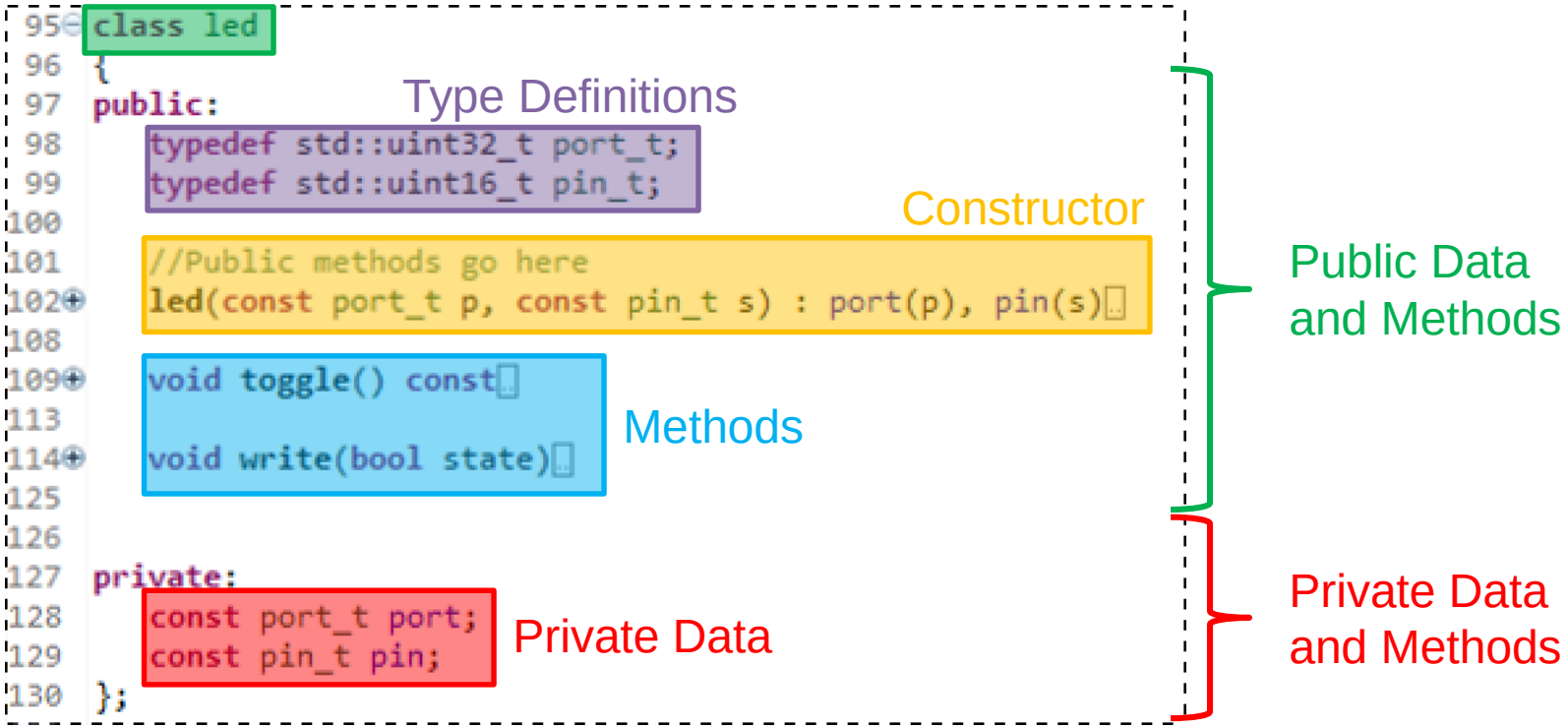


Review - Abstraction

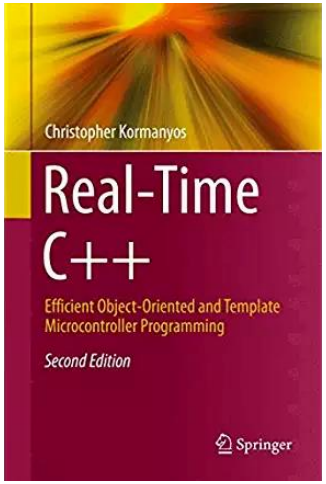
Abstraction – is the act of representing essential features without including the background details or explanations

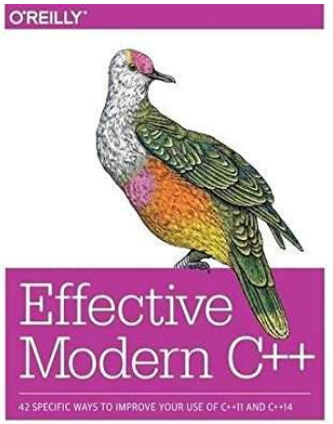


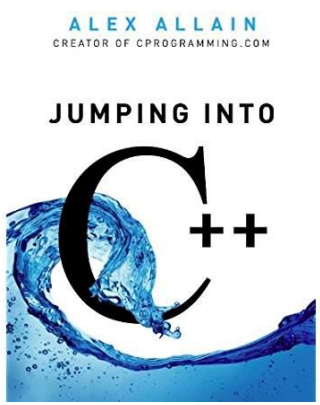
Review - Classes

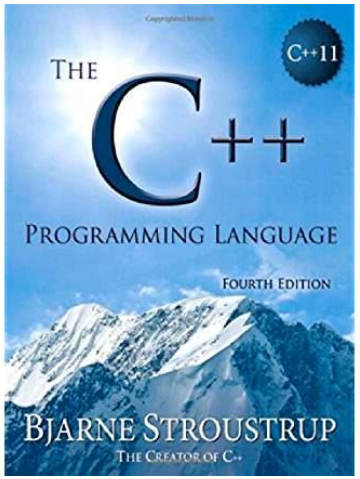


Review – Useful Resources









Presented by:

Additional Resources

- Download Course Material for
  - C/C++ Doxygen Templates
  - Example source code
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Transitioning from C to C++

The Lecturer – Jacob Beningo

Jacob Beningo

Principal Consultant

Social Media / Contact

- E : jacob@beningo.com
- T : 810-844-1522
- : Jacob_Beningo
- : Beningo Engineering
- : JacobBeningo
- EDN : Embedded Basics
- ARM Connected Community

Consulting

- Advising
- Coaching
- Content
- Consulting
- Training

BENINGO EMBEDDED GROUP

www.beningo.com