Domain

- small processors
- resource constrained
- bare metal designs

Typical environment

- range of 64k flash
- 16k ram
- no RTTI no exceptions
- compile time not a problem
- unit testing very difficult
- real time in the sense of being on time rather than fast (http://www.voti.nl/blog/?p=44)

Tool chain

- GCC gaining market share (supports C++14)
- IAR/Keil etc. (supports most of C++03 with little template support)

Definition of terms

Efficiency

- work/second
- work/$
- work/watt
- work/hour
- work/hair loss and/or sleep deprivation

zero cost

- at least same amortized efficiency as hand coded equivalent
- equal functionality

The most important guidelines

- Scott Meyers: *Make interfaces easy to use correctly and hard to use incorrectly.*
- Bjarne Stroustrup: *Make simple things simple.*

Real code and common problems

From nxpUSBlib:

```
uint32_t DevCmdStat = LPC_USB->DEVCMDSTAT;      /* Device Status */

if(DevCmdStat & USB_DRESET_C){                  /* Reset */
     LPC_USB->DEVCMDSTAT |= USB_DRESET_C;
     HAL_Reset();
     USB_DeviceState = DEVICE_STATE_Default;
     Endpoint_ConfigureEndpointControl(USB_Device_ControlEndpointSize);
}
if(DevCmdStat & USB_DCON_C){                    /* Connect change */
     LPC_USB->DEVCMDSTAT |= USB_DCON_C;
}
if(DevCmdStat & USB_DSUS_C){                    /* Suspend/Resume */
     LPC_USB->DEVCMDSTAT |= USB_DSUS_C;
     if(DevCmdStat & USB_DSUS)                  /* Suspend */
     {  }
     else                                       /* Resume */
     {  }
}
```

Excerpt from table 233 of the LPC11U6x users manual

DRES_C

Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0).
The bit is reset by writing a one to it.

DCON_C

Device status - connect change. The Connect Change bit is set when the device's pull-up resistor is disconnected because VBus disappeared.
The bit is reset by writing a one to it.

DSUS

Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 milliseconds. It is reset to 0 on any activity.
When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up.
(Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.

same code from mbed:

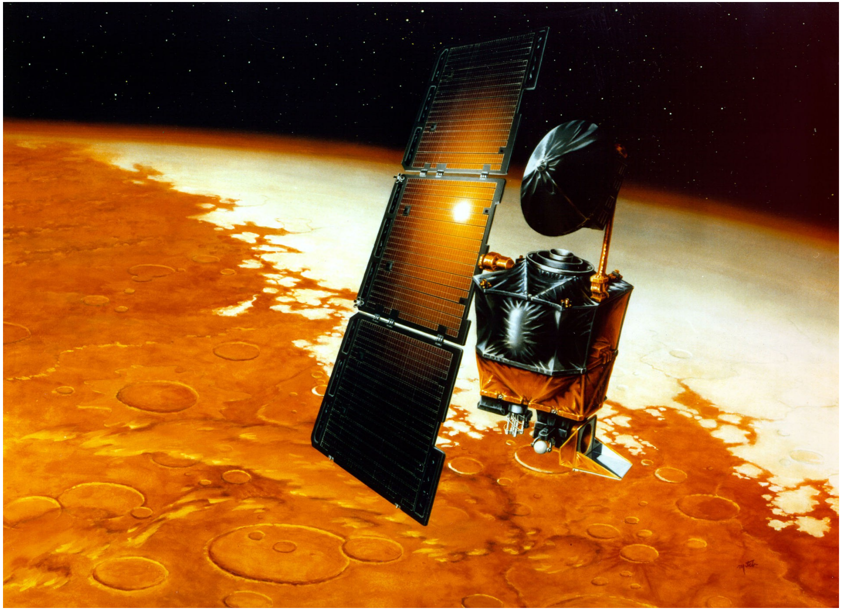
```
// Shadow DEVCMDSTAT register to avoid accidentally clearing flags or
// initiating a remote wakeup event.
static volatile uint32_t devCmdStat;
// Set device address 0, enable USB device, no remote wakeup
devCmdStat = DEV_ADDR(0) | DEV_EN | DSUS;
LPC_USB->DEVCMDSTAT = devCmdStat;

if(LPC_USB->DEVCMDSTAT & DSUS_C) {
     // Suspend status changed
     LPC_USB->DEVCMDSTAT = devCmdStat | DSUS_C;
     if(LPC_USB->DEVCMDSTAT & DSUS) {
          suspendStateChanged(1);
     } else {
          suspendStateChanged(0);
     }
}
```

Why is devCmdStat volatile?

```
void USBHAL::connect(void) {
    NVIC_EnableIRQ(USB_IRQ);
    devCmdStat |= DCON;
    LPC_USB->DEVCMDSTAT = devCmdStat;
}
void USBHAL::disconnect(void) {
    NVIC_DisableIRQ(USB_IRQ);
    devCmdStat &= ~DCON;
    LPC_USB->DEVCMDSTAT = devCmdStat;
}
void USBHAL::setAddress(uint8_t address) {
    devCmdStat &= ~DEV_ADDR_MASK;
    devCmdStat |= DEV_ADDR(address);
    LPC_USB->DEVCMDSTAT = devCmdStat;
}
USBHAL::~USBHAL(void) {
    // Ensure device disconnected (DCON not set)
    LPC_USB->DEVCMDSTAT = 0;
    // Disable USB interrupts
    NVIC_DisableIRQ(USB_IRQ);
}
```

Solution to similar problems in a different domain:

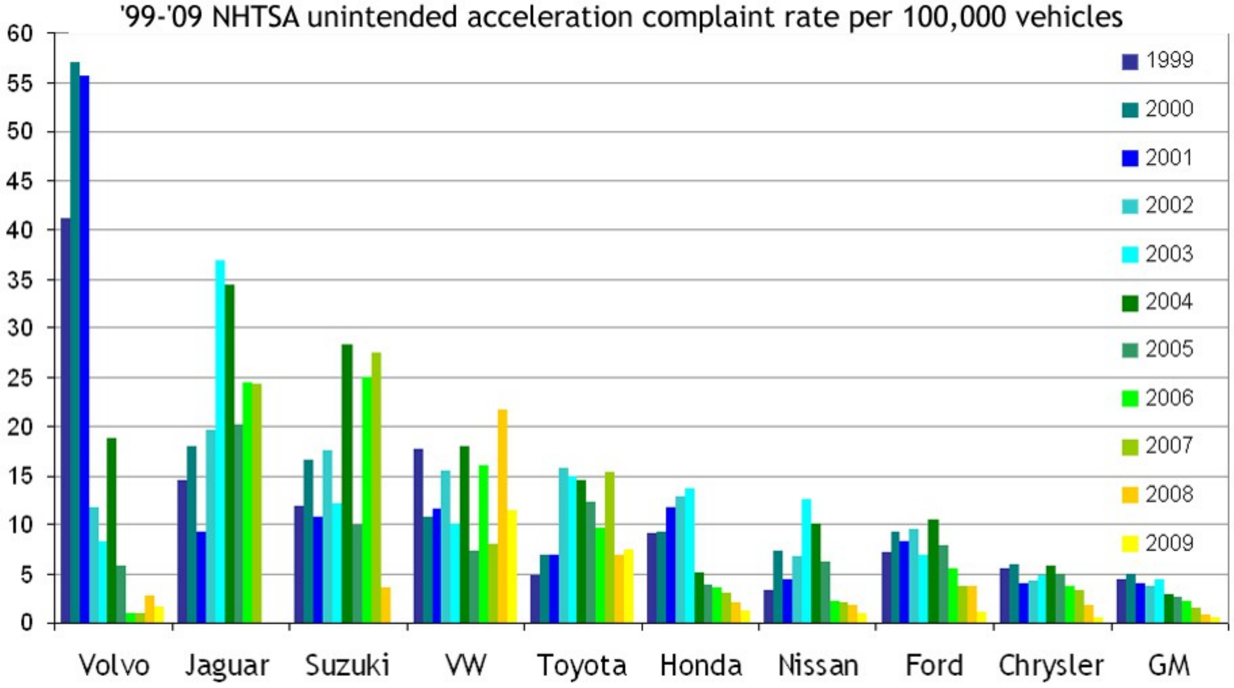


crashed because software used pound-seconds (lbf×s) instead of the metric units of newton-seconds (N×s)

boost.units

```
/// scalar
const double    s1 = 2;
const long                  x1 = 2;
const static_rational<4,3>  x2;
/// define some units
force       u1 = newton;
energy      u2 = joule;
/// define some quantities
quantity<force>      q1(1.0*u1);
quantity<energy>     q2(2.0*u2);
/// check scalar-unit algebra
std::cout //<< "U1+S1 : " << u1+s1 << std::endl    // illegal
     //<< "S1-U1 : " << s1-u1 << std::endl    // illegal
     << "U1*S1 : " << u1*s1 << std::endl
     << "U1/S1 : " << u1/s1 << std::endl
     //<< "U1+Q1 : " << u1+q1 << std::endl    // illegal
     //<< "U1-Q1 : " << u1-q1 << std::endl    // illegal
     << "U1*Q1 : " << u1*q1 << std::endl
     << "U1/Q1 : " << u1/q1 << std::endl;
```

see Bjarne Stroustrup talk minute 20 youtube.com/watch?v=3xMc-rYPdsM



This embedded programming problem is not solved.

Questions?

A new library:

Definition of requirements

- zero cost
- intuitive interface
- static checking for unexpected register behavior
- atomic actions and thread safe support
- well packaged meta programming
- C++11 support
- Header only and easily configurable
- Few macros
- Static analysis tool friendly (no indirect calls, no recursion)

Meta programming tools

- constexpr
- auto and decltype
- templates
- using
- variadic templates
- data storage
- containers
- loops
- value and template wrappers
- if and switch
- Curiously Recurring Template pattern (CRTP)

constexpr

```
void f(int i);
void f2(const int& i);

int i = 4;
f(i);

constexpr int ci = 4;
f(ci);
f2(ci);

constexpr int cf(int i){   return i<20?i+2:i-15;      }
auto test = cf(i); //works, done at runtime
int a[cf(i)]; //error
int a2[cf(ci)];
```

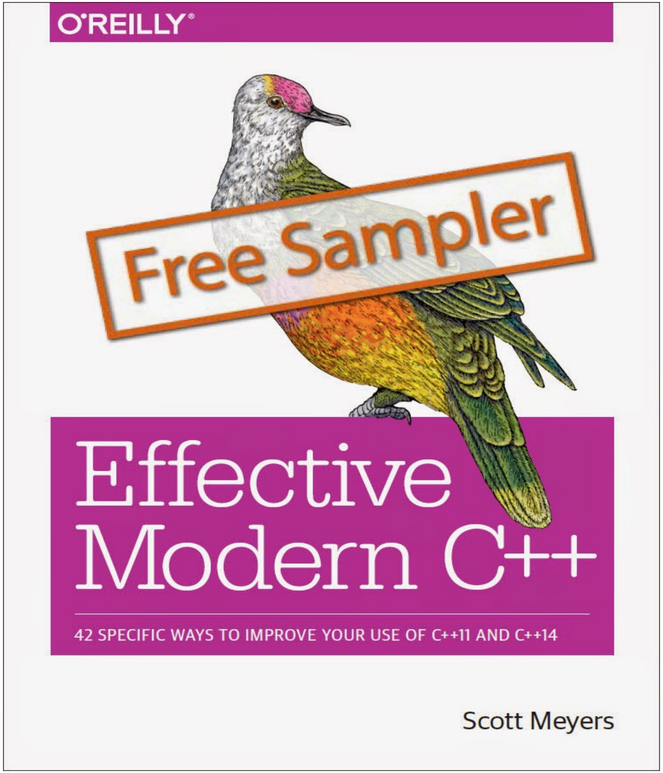
auto and decltype

```
int s = std::string(“aaaaaaaahhh”); //error std::string is not an int

auto s2 = std::string(“aaaaaaahhh”);

constexpr auto soundTheAlarm = set(alarmPin);

std::vector<decltype(s.begin())> v;
```



http://www.oreilly.com/free/effective-modern-c++.html

templates

```
template<typename T, int I>
struct Array{
      T data_[I];
// implementation here
};

template<typename T>
T square(T in){
      return in*in;
}
```

using

```
typedef std::vector<int> intVec;

using intVec = std::vector<int>;

template<unsigned I>
using intArray = std::array<int, I>;

intArray<4> ia;
```

variadic templates

```
template<typename... Ts>
void myPrintf(std::string s, Ts...args){
     printf(s.c_str(), args... );
}

template<typename... Ts>
struct S : Ts... {};
```

data storage

```
struct S{
      friend int getI(const S& s){ return s.i_; }
      friend bool getB(const S& s){ return s.b_; }
      S(int i, bool b):i_{i},b_{b}{}
private:
      int i_;
      bool b_;
};

S myS(4,false);

auto i = getI(myS);
```

data storage 2

```cpp
template<int I, bool B>
struct S{};

template<typename T>
struct GetI;
template<int I, bool B>
struct GetI<S<I,B>>{ static constexpr int value = I; }

using MyS = S<4,false>;

auto i = GetI<MyS>::value;
```

containers / loops

```cpp
template<typename...Ts>
struct List{};

template<int I, typename T>
struct At;
template<int I, typename T, typename...Ts>
struct At<int I, List<T,Ts...> : At<I-1, List<Ts...>>{};

template<typename T, typename... Ts>
struct At<0,List<T,Ts...> {
     using Type = T;
}

using L = List<int, bool, float, bool>;

typename At<2,L>::Type f = 1.4;
```

value and type wrappers

A parameter can be a:

- type
- value of an integral type
- template

There is no polymorphism or function overloading. The solution is wrapping values and templates in type wrappers.

```
template<typename T, T V>
struct Integral{
	static constexpr T value = V;
};
using T = Integral<int,4>;
auto i = T::value;
```

```
std::vector<std::any> v;
void f(const std::any& a);
```

If and Switch

```
int f(int i, bool b){
        if(b){
                switch(i){
                default:
                        return 99;
                case 42:
                        return 1
                };
        }
        return 22;
}
```

If and Switch 2

```
template<int I, bool B>
struct F : Integral<int,22>{};

template<int I>
struct F<I,true> : Integral<int, 99>{};

template<>
struct F<42,true> : Integral<int, 1>{};
```

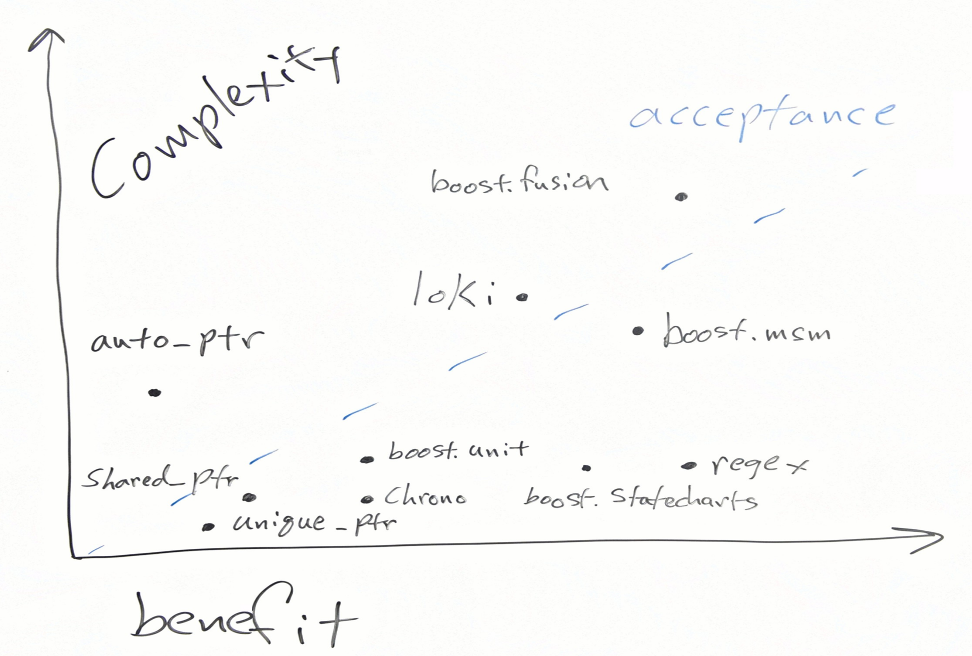
CRTP

```
template<typename TDerived>
struct Base{
    void f(){
        static_cast<TDerived*>(this)->g();
    }
    void g(){} //default
}

struct S : Base<S> {
//no g
};

struct S2 : Base<S2> {
    void g(){}
};
```

Public interface



Constxpr metafunction

```
template<typename T>  //somewhere in the Kvasir library
struct Mutate;
template<int I, int J, type T>
struct Mutate<Structure<I,J,T>> : Other<(1<<I),someFunc<T>(J)> {}
//user code
using Data = Structure<3,0x40081000,bool>;
using MutatedData = typename Mutate<Data>::Type;
Kvasir::Register::apply<MutatedData>(); //execute

template<typename T>      //add layer of sexyness to library
typename Mutate<T>::Type mutate(T){    return {}; }

constexpr Full::Namespace::Structure<3,0x40081000,bool> data;
constexpr auto mutatedData = mutate(data);
apply(mutatedData);
```

What does this code do?

```
using Kvasir::Io;
constexpr auto statusLed = makePinLocation(port0,pin13);

apply(
     makeOpenDrain(statusLed),
     makeOutput(statusLed),
     set(statusLed));

if(something){
     apply(toggle(statusLed));
}
```

Is there a race condition?

```
//main thread
apply(atomic(set(Can::txPacketSent)));
//ISR
apply(atomic(set(Can::rxPacketReceived)));
```

bonus question, how about this?

```
//main thread
apply(atomic(set(Can::txPacketSend)));
//ISR
apply(set(Can::rxPacketReceived));
```

BitLocation

```
template<
    typename TAddress,
    unsigned Mask,
    typename Access = ReadWriteAccess,
    typename TFieldType = unsigned>
struct BitLocation{
    using Type = BitLocation<TAddress, Mask, Access,
TFieldType>;
};
```

Access

```
template<
    bool Readable,
    bool Writable,
    bool ClearOnRead = false,
    bool Popable = false,
    bool SetToClear = false>
struct Access {
    using Type =
Access<Readable,Writable,ClearOnRead,Popable,SetToClear>;
};
```

Address

```
template<
    unsigned A,
    unsigned WriteIgnoredIfZeroMask,
    unsigned WriteIgnoredIfOneMask = 0,
    typename TRegType = unsigned,
    typename TMode = NormalMode>
struct Address{
    using Type = Address<A, WriteIgnoredIfZeroMask,
WriteIgnoredIfOneMask,               TRegType,TMode>;
    static constexpr unsigned value = A;
};
```

Questions?