Better than hand coded C?

- use bit-banding
- use blind writes
- local temporaries encouraged by public interface*
- reorder access
- merge access

local temporaries encouraged by public interface

```
auto res = apply(read(thing1,thing2));
if(get<0>(res)){/*...*/}
if(get<1>(res)){/*...*/}
```

reorder access

```
LDR       rn, [pc, #offset to literal pool]
```

LDR and STR instructions can have a hard coded offset
LDM and STM still untapped potential

```
unsigned volatile &reg1 = *(unsigned*)0x40013004;
unsigned volatile &reg2 = *(unsigned*)0x40024000;
unsigned volatile &reg3 = *(unsigned*)0x40013008;

reg1 += 4;
reg2 += 5;
reg3 += 6;

//becomes
auto a = reg1;
auto b = reg3;
a += 4;
b += 5;
reg1 = a;
reg3 = b;
reg2 += 6;
```

merge access

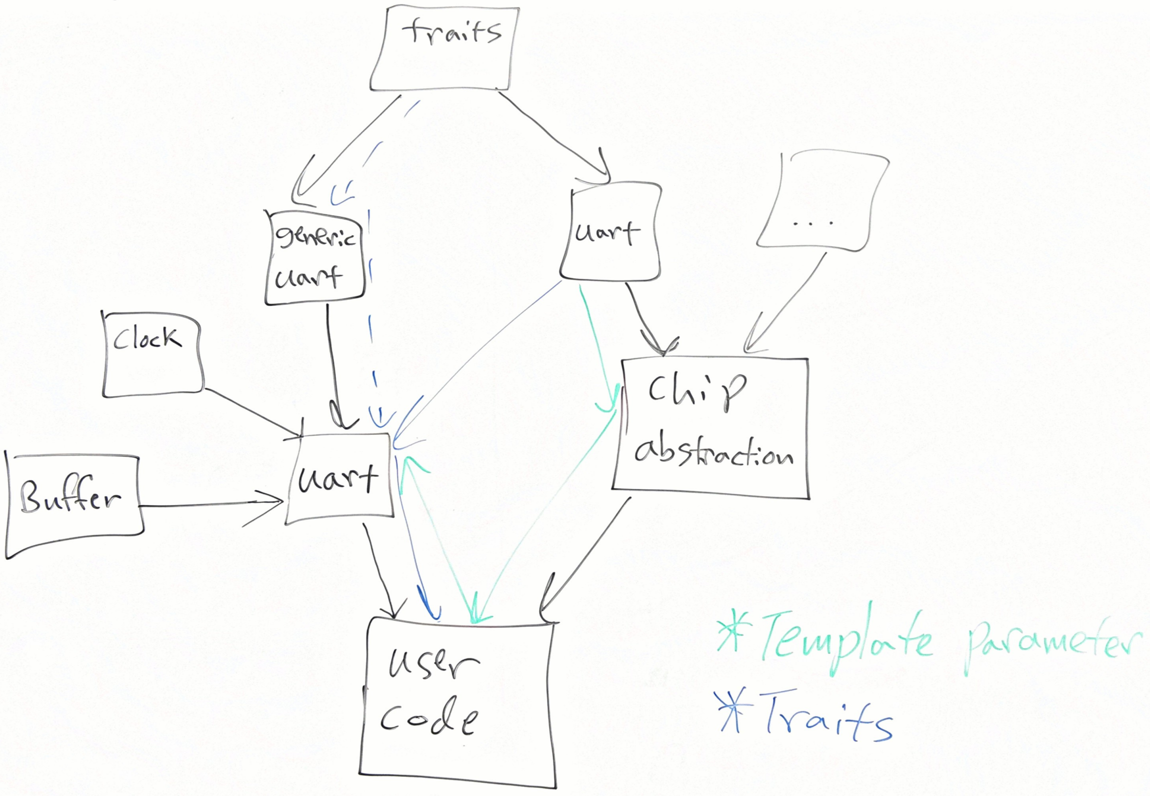
```
auto i = reg;
i &= ~0x10;
i |= 0x100;
reg = i;

auto i = reg;
i &= ~0x03;
i |= 0x08;
reg = i;

//becomes

auto i = reg;
i &= ~0x13;
i |= 0x108;
reg = i;
```

Kvasir::StartUp

- Inject start up code into the projects source files
- Facilitate merged initialization
- Hook up ISRs
- Configure the system clock
- Provide hooks for power users to inject other init steps

A quick overview of Kvasir macro architecture:

- generic code calls chip specific code via traits specialization
- start up code is injected via macro


traits
generic uart
uart
...
Clock
Chip abstraction
Buffer
uart
user code
*Template parameter
*Traits

Questions?