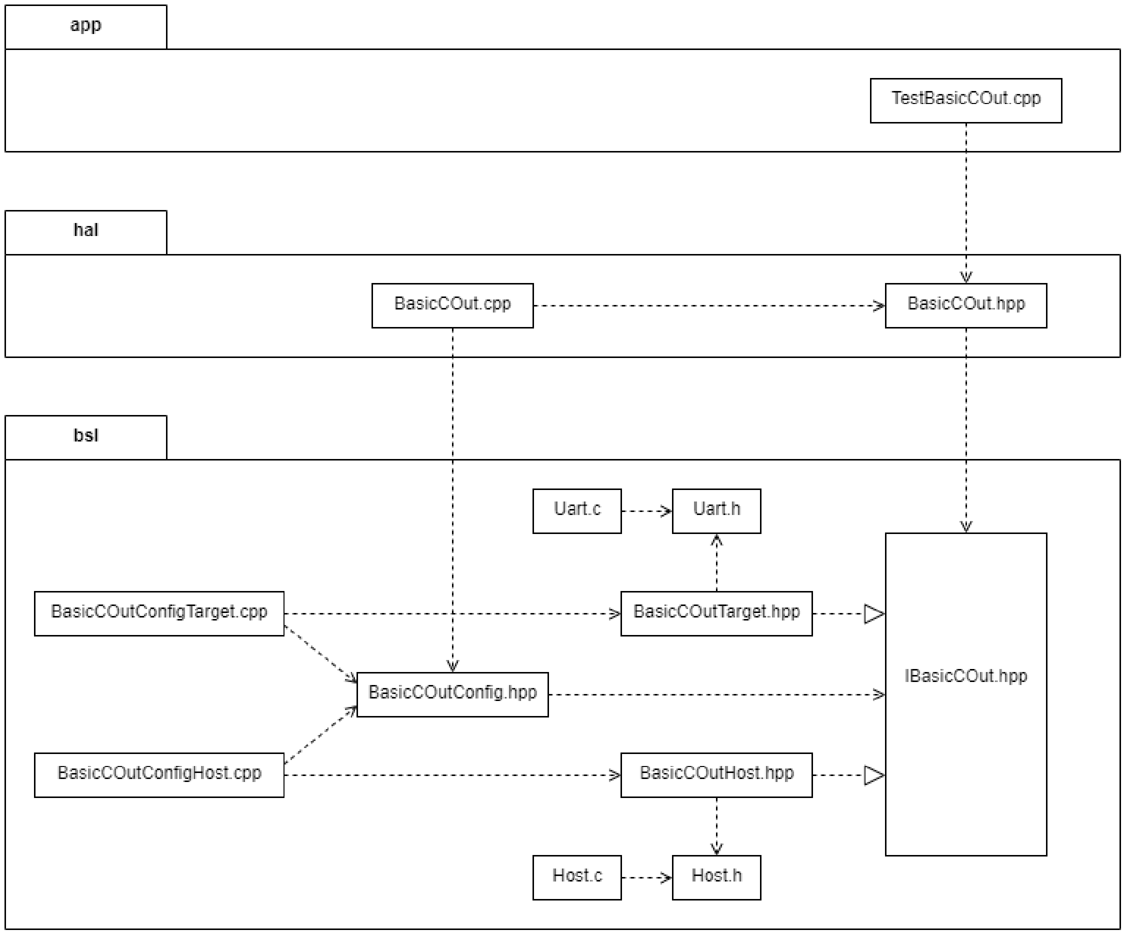
Dependencies Between Layers for a Basic Console Output Component in C/C++

Dr. Michel de Champlain
SOEN 422 — Embedded Systems and Software
Fall 2021

November 17, 2021

Contents

1 Dependencies Between Layers in C++



Test in the Application layer in C++

1.1 hal_TestBasicCOut.cpp

```
// hal_TestBasicCOut.cpp - Generic Basic Console Output in C++.

#include "hal_BasicCOut.hpp"

int main (void) {
    hal::BasicCOut::Init();  // Hiding the factory and all dependencies with the host and targets.

    hal::BasicCOut::PutS("Test HAL Basic Console Output in C++\n");
    hal::BasicCOut::PutS("[t]\n");  // Expected output.

    hal::BasicCOut::PutC('[');      // Current output.
    hal::BasicCOut::PutC('t');
    hal::BasicCOut::PutC(']');
    hal::BasicCOut::PutN();
}
```

HAL in C++

1.2 hal_BasicCOut.hpp

```
/* hal_BasicCOut.hpp - HAL Basic Console Output C++ Interface
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __hal_BasicCOut_hpp
#define __hal_BasicCOut_hpp

#include "bsl_IBasicCOut.hpp"  // Need the interface declaration to declare a qualified opaque type.

class ::bsl::IBasicCOut;       // Forward reference to the opaque type.

namespace hal {
    class BasicCOut {
    public:
        static void Init();
        static void PutC(char c);
        static void PutS(const char* s);
        static void PutN(void);
    private:
        static bsl::IBasicCOut* cout;  // Opaque type for internal (BSL) console output.
        BasicCOut() = delete;          // To never generate default constructor (C++11).
       ~BasicCOut() = delete;          // To never generate default destructor  (C++11).
    };
}

#endif
```

1.3 hal_BasicCOut.cpp

```
/* hal_BasicCOut.cpp - HAL Basic Console Output C++ Interface
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#include "hal_BasicCOut.hpp"
#include "bsl_BasicCOutConfig.hpp"

void hal::BasicCOut::Init()              { if (cout == nullptr) cout = bsl::BasicCOutConfig::Get(); }

void hal::BasicCOut::PutC(char c)        { cout->PutC(c); }
void hal::BasicCOut::PutS(const char* s) { cout->PutS(s); }
void hal::BasicCOut::PutN(void)          { cout->PutN();  }

bsl::IBasicCOut* hal::BasicCOut::cout = nullptr;
```

BSL in C++

1.4 bsl_IBasicCOut.hpp

```
/* bsl_IBasicCOut.hpp - BSL Basic Console Output C++ Interface
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __bsl_IBasicCOut_hpp
```

```
#define __bsl_IBasicCOut_hpp

namespace bsl {
    class IBasicCOut {
    public:
//      void virtual Init();  // No Ctor() or Init() in an (emulated) interface in C++.
        void virtual PutC(char c) = 0;
        void virtual PutS(const char* s) = 0;
        void virtual PutN(void) = 0;
    };
}
#endif
```

1.5 bsl_BasicCOutConfig.hpp

```
/* bsl_BasicCOutConfig.hpp - BSL Basic Console Output Config (Static Factory Method) C++ Interface
//                           This is a unique and uniform interface for all BSL BasicCOut configurations.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __bsl_BasicCOutConfig_hpp
#define __bsl_BasicCOutConfig_hpp

#include "bsl_IBasicCOut.hpp"

namespace bsl {
    class BasicCOutConfig {
    public:
        static IBasicCOut* Get(); // Static Factory method
    };
}
#endif
```

1.6 bsl_BasicCOutHost.hpp

```
/* bsl_BasicCOutHost.hpp - BSL Basic Console Output Host C++ Interface (inlining the class)
//
// Note: In this header file (.hpp), I implicitly declare inline functions in the class declaration by
// simply defining (implementing) functions inside the class declaration. No inline keyword is necessary.
// It is a way to give hints to the C++ compiler to consider optimizing these functions as macros.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __bsl_BasicCOutHost_hpp
#define __bsl_BasicCOutHost_hpp

extern "C" {
#include "bsl_Host.h"
}

#include "bsl_IBasicCOut.hpp"

namespace bsl {
    class BasicCOutHost : public IBasicCOut {
    public:
                     BasicCOutHost()     { bsl_Host_Init(); }
        void virtual PutC(char c)        { bsl_Host_PutC(c); }
        void virtual PutS(const char* s) { while (*s) bsl_Host_PutC(*s++); }
        void virtual PutN()              { bsl_Host_PutC('\n'); }
    };
}
#endif
```

1.7 bsl_BasicCOutTarget.hpp

```
/* bsl_BasicCOutTarget.hpp - BSL Basic Console Output Target C++ (inlining the class)
//
// Note: In this header file (.hpp), I implicitly declare inline functions in the class declaration by
// simply defining (implementing) functions inside the class declaration. No inline keyword is necessary.
// It is a way to give hints to the C++ compiler to consider optimizing these functions as macros.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __bsl_BasicCOutTarget_hpp
#define __bsl_BasicCOutTarget_hpp

extern "C" {
#include "bsl_Uart.h"
}

#include "bsl_IBasicCOut.hpp"

namespace bsl {
    class BasicCOutTarget : public IBasicCOut {
    public:
                     BasicCOutTarget()   { bsl_Uart_Init(); }
        void virtual PutC(char c)        { bsl_Uart_TxChar(c); }
        void virtual PutS(const char* s) { while (*s) bsl_Uart_TxChar(*s++); }
        void virtual PutN()              { bsl_Uart_TxChar('\n'); }
    };
}
#endif
```

1.8 bsl_BasicCOutConfigHost.cpp

```
/* bsl_BasicCOutConfigHost.cpp - BSL Basic Console Output Config (Factory Method) C++ Implementation
//                                Note: Can be compiled separately, but link only one at a time.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#include "bsl_BasicCOutConfig.hpp"
#include "bsl_BasicCOutHost.hpp"

bsl::IBasicCOut* bsl::BasicCOutConfig::Get() { return new bsl::BasicCOutHost(); }
```

1.9 bsl_BasicCOutConfigTarget.cpp

```
/* bsl_BasicCOutConfigTarget.cpp - BSL Basic Console Output Config for Target (Factory Method) C++ Implementation
//                                  Note: Can be compiled separately, but link only one at a time.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#include "bsl_BasicCOutConfig.hpp"
#include "bsl_BasicCOutTarget.hpp"

bsl::IBasicCOut* bsl::BasicCOutConfig::Get() { return new bsl::BasicCOutTarget(); }
```

BSL in C

1.10 bsl_Host.h

```c
/* bsl_Host.h - Host interface for Console Output
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#ifndef __bsl_Host_h
#define __bsl_Host_h

void bsl_Host_Init(void);
void bsl_Host_PutC(char c);
char bsl_Host_GetC(void);

#endif
```

1.11 bsl_Host.c

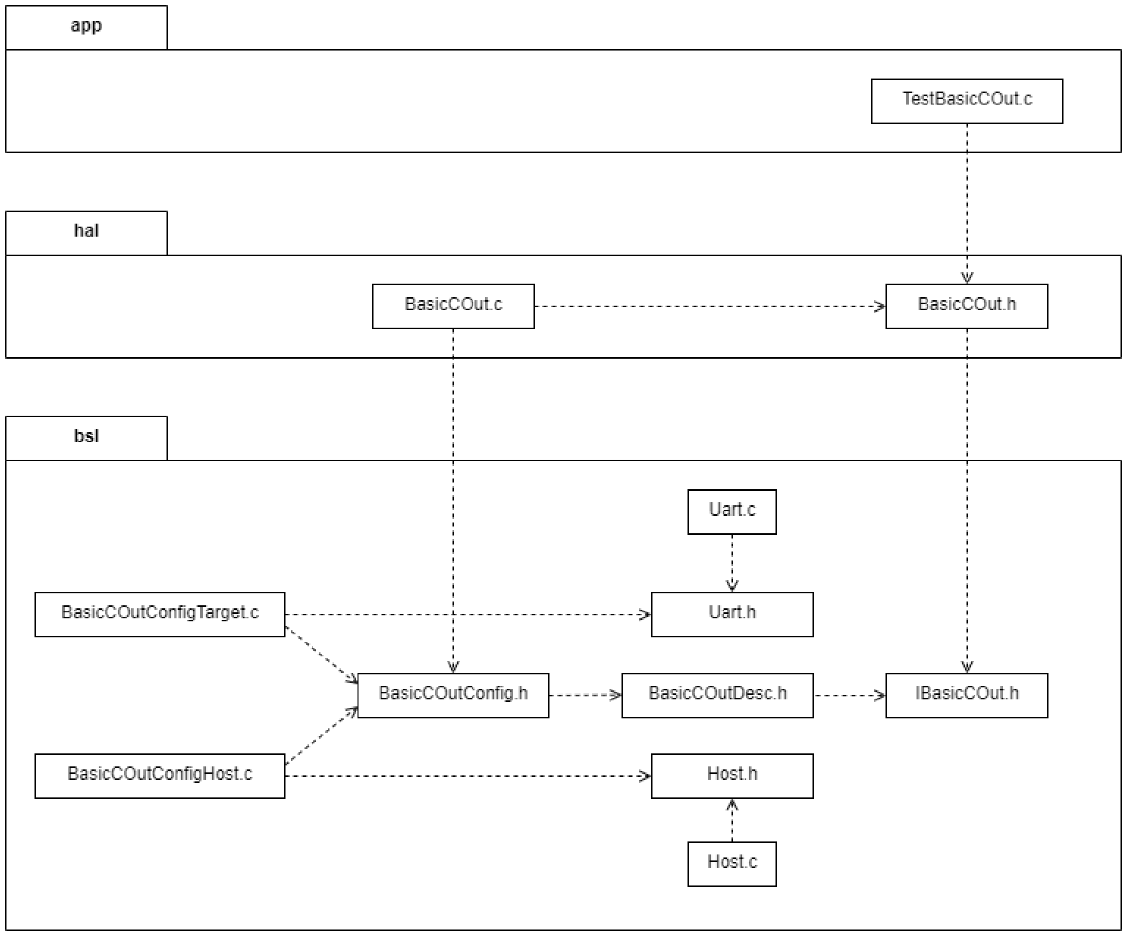
```c
/* bsl_Host.c - Host implementation for Console Output
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#include <stdio.h>

void bsl_Host_Init(void)   { }
void bsl_Host_PutC(char c) { putchar(c); }
char bsl_Host_GetC(void)   { return getchar(); }
```

1.12 bsl_Uart.h

```c
/* bsl_Uart.h - Uart header for Arduino Nano
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#ifndef __bsl_Uart_h
#define __bsl_Uart_h

void bsl_Uart_Init(void);
void bsl_Uart_TxChar(char c);
char bsl_Uart_RxChar(void);

#endif
```

1.13 bsl_Uart.c

2 Dependencies Between Layers in C



Test in the Application layer in C

2.1 hal_TestBasicCOut.c

```c
// hal_TestBasicCOut.c - Generic Basic Console Output in C

#include "hal_BasicCOut.h"

int main (void) {
    hal_COut_Init();         // Hiding the factory and all dependencies with the host and targets.

    hal_COut_PutS("Test HAL Basic Console Output in C\n");
    hal_COut_PutS("[t]\n");  // Expected output.

    hal_COut_PutC('[');      // Current output.
    hal_COut_PutC('t');
    hal_COut_PutC(']');
    hal_COut_PutN();
}
```

HAL in C

2.2 hal_BasicCOut.h

```c
/* hal_BasicCOut.h - HAL Basic Console Output Interface
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __hal_BasicCOut_h
#define __hal_BasicCOut_h

void hal_COut_Init(void);
void hal_COut_PutC(char c);
void hal_COut_PutS(const char* s);
void hal_COut_PutN(void);

#endif
```

2.3 hal_BasicCOut.c

```c
/* hal_BasicCOut.c - HAL Basic Console Output Implementation
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#include "hal_BasicCOut.h"
//#include "bsl_BasicCOutDesc.h"     // Basic Output Descriptor.
#include "bsl_BasicCOutConfig.h"   // bsl_ICOut opaque type and the factory method.

static bsl_IBasicCOut cout;

void hal_COut_Init(void)          { cout = bsl_BasicCOut_Get(); }
void hal_COut_PutC(char c)        { cout->pc(c); }
void hal_COut_PutS(const char* s) { cout->ps(s); }
void hal_COut_PutN(void)          { cout->pn();  }
```

BSL in C

2.4 bsl_IBasicCOut.h

```c
/* bsl_IBasicCOut.h - BSL for Console Output Interface to hide the internal descriptor.
//
// Copyright (C) 1999-2020 by Michel de Champlain
//
*/

#ifndef __bsl_IBasicCOut_h
#define __bsl_IBasicCOut_h

        struct bsl_BasicCOutDesc;
typedef struct bsl_BasicCOutDesc* bsl_IBasicCOut;

#endif
```

2.5 bsl_BasicCOutDesc.h

```c
/* bsl_BasicCOutDesc.h - Interface for Basic Output Descriptor
//
// Copyright (c) 1999-2021 by Michel de Champlain
```

```
//
*/

#ifndef __bsl_BasicCOutDesc_h
#define __bsl_BasicCOutDesc_h

#include "bsl_IBasicCOut.h"

// Private Basic Output Function Pointer Types:
typedef void (*bsl_PutC)(char);
typedef void (*bsl_PutS)(const char*);
typedef void (*bsl_PutN)(void);

// Private Interface Basic Output Descriptor
typedef struct bsl_BasicCOutDesc {
    bsl_PutC pc;
    bsl_PutS ps;
    bsl_PutN pn;
} bsl_BasicCOutDesc;

#endif
```

2.6 bsl_BasicCOutConfig.h

```
/* bsl_BasicCOutConfig.h - BSL for Basic Console Output Config (Static Factory Method) C Interface
//                         This is a unique and uniform interface for all BSL BasicCOut configurations.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#ifndef __bsl_BasicCOutConfig_h
#define __bsl_BasicCOutConfig_h

#include "bsl_BasicCOutDesc.h"          // Basic Output Descriptor which in turn depends on IBasicCOut.h

bsl_IBasicCOut bsl_BasicCOut_Get(void); // Static Factory Method (Function)

#endif
```

2.7 bsl_BasicCOutConfigHost.c

```
/* bsl_BasicCOutConfigHost.c - Basic Console Output Configuration implementation for Host.
//
// Copyright (C) 1999-2021 by Michel de Champlain
//
*/

#include "bsl_BasicCOutConfig.h"  // Depends of BasicCOutDesc.h
#include "bsl_Host.h"

static void COut_PutC(char c)        { bsl_Host_PutC(c); }
static void COut_PutS(const char* s) { while (*s) bsl_Host_PutC(*s++); }
static void COut_PutN(void)          { bsl_Host_PutC('\n'); }

static bsl_BasicCOutDesc cout = {
    COut_PutC,
    COut_PutS,
    COut_PutN
};

bsl_IBasicCOut bsl_BasicCOut_Get(void) { return &cout; }
```

2.8 bsl_BasicCOutConfigTarget.c

2.9 bsl_Host.h

```
/* bsl_Host.h - Host interface for Console Output
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#ifndef __bsl_Host_h
#define __bsl_Host_h

void bsl_Host_Init(void);
void bsl_Host_PutC(char c);
char bsl_Host_GetC(void);

#endif
```

2.10 bsl_Host.c

```
/* bsl_Host.c - Host implementation for Console Output
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#include <stdio.h>

void bsl_Host_Init(void)   { }
void bsl_Host_PutC(char c) { putchar(c); }
char bsl_Host_GetC(void)   { return getchar(); }
```

2.11 bsl_Uart.h

```
/* bsl_Uart.h - Uart header for Arduino Nano
//
// Copyright (C) 2020-2021 by Michel de Champlain
//
*/

#ifndef __bsl_Uart_h
#define __bsl_Uart_h

void bsl_Uart_Init(void);
void bsl_Uart_TxChar(char c);
char bsl_Uart_RxChar(void);

#endif
```

2.12 bsl_Uart.c