



DesignNews

Leveraging AI to Accelerate Embedded Software Development

DAY 4 : Abstracting Your Hardware with an AI-Generated HAL

Sponsored by

DigiKey



©2023 Beningo Embedded Group, LLC. All Rights Reserved.

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

THE SPEAKER



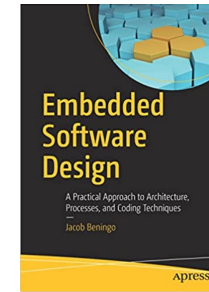
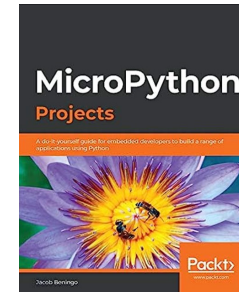
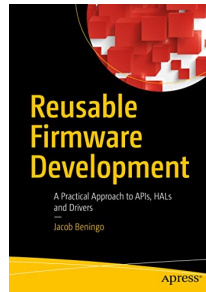
Jacob Beningo

Jacob@beningo.com

Beningo Embedded Group - President

Focus: Embedded Software Consulting and Training

Specializes in *creating and promoting* embedded software excellence in businesses around the world.



Blogs for:

- DesignNews.com
- Embedded.com
- EmbeddedRelated.com
- MLRelated.com

Visit www.beningo.com to learn more



Hardware Abstraction Layers

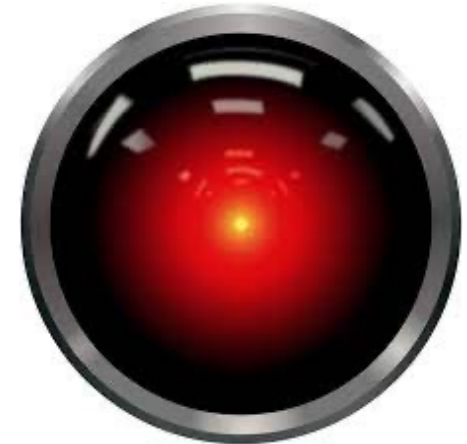
01

Hardware Abstraction Layers (HALs)

Definitions

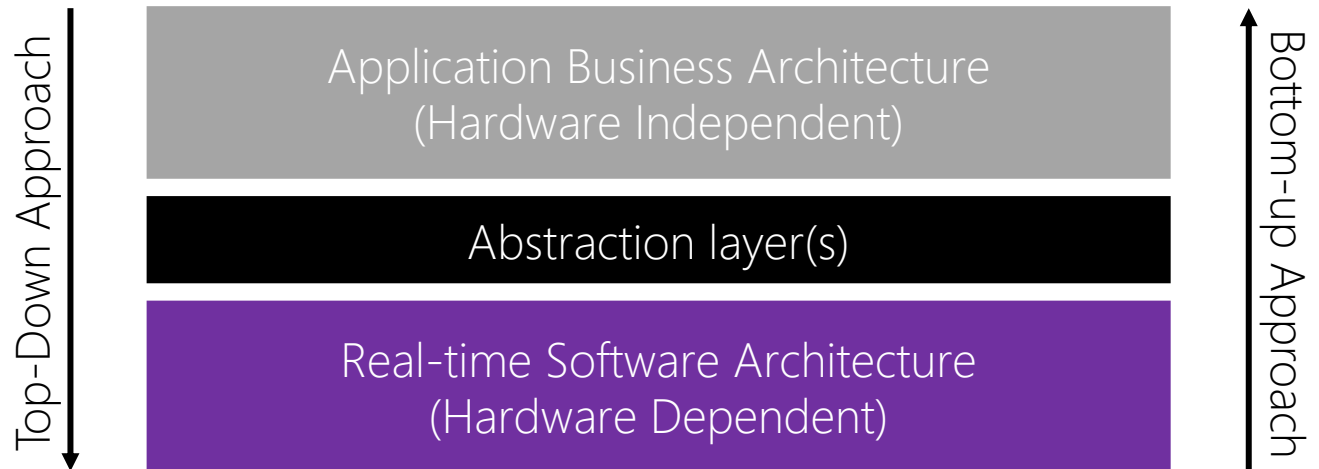
A **HAL** is a layer of firmware that replaces hardware-level accesses with higher-level function calls.

An **API** is used to simplify application programming by abstracting the application into black boxes.



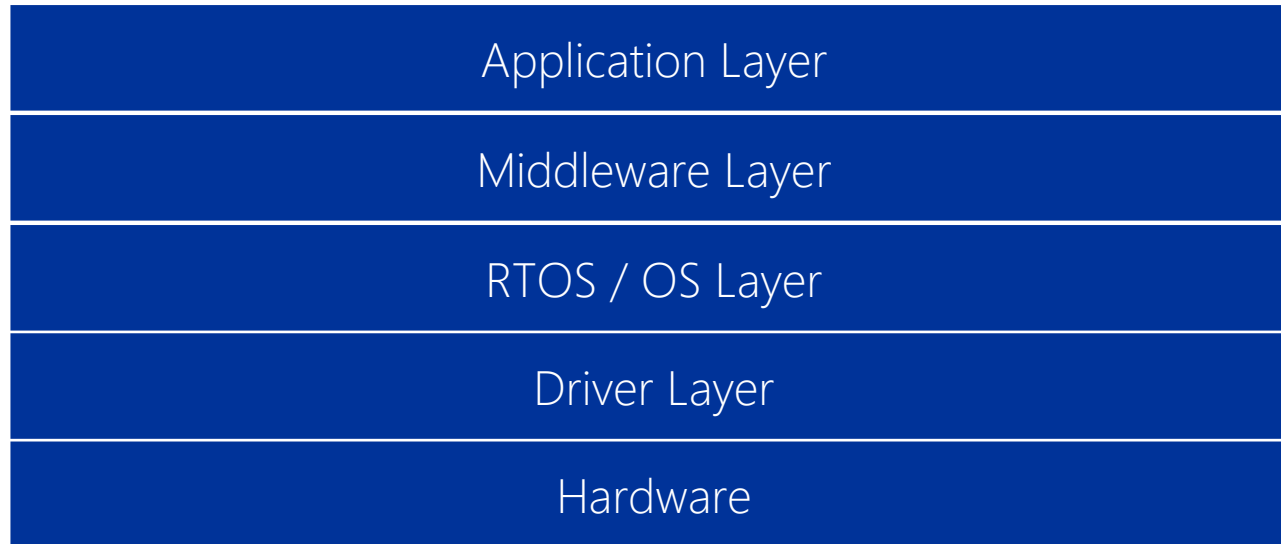
Hardware Abstraction Layers (HALs)

A tale of two architectures



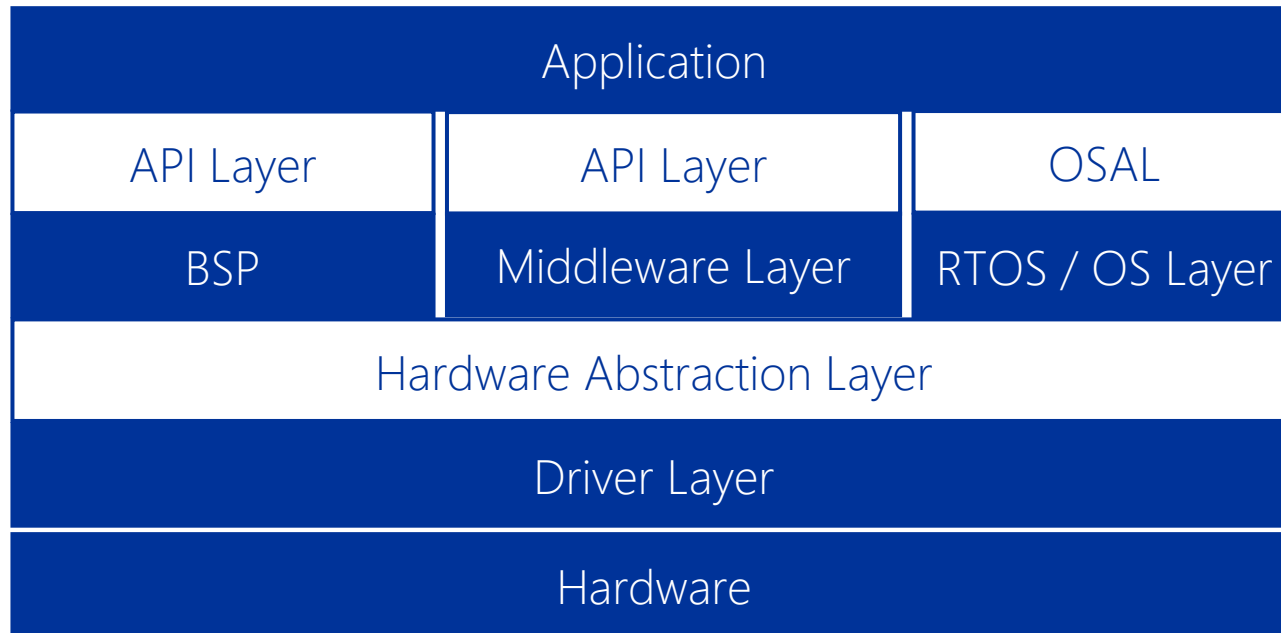
Hardware Abstraction Layers (HALs)

Modern Software Stack



Hardware Abstraction Layers (HALs)

Modern Software Stack with Abstractions



Audience POLL Question

Do you leverage hardware abstractions in your software?

- a) Yes
- b) No
- c) Working on it



02

Generating a USART HAL using ChatGPT

Generating a USART HAL using ChatGPT

JA We are going to write an embedded software interface in C for a usart peripheral. We are going to use the Google C++ coding standard. Don't use built-in types such as char, but use fixed width integer types from stdint.h. Please create an initial draft of the usart interface

Analysis:

- I like the configuration structure for initialization.
- The interface is a bit verbose and repetitive.
 - SendByte, SendBytes, etc
- Interface support only a single USART device
- Not a fan of variables with _. Would rather use camel case
- The use of const in parameters is a good practice
 - USART_Init should really have a const config
- There's no considerations for interrupts or DMA through callback functions

Note: You may get different results. I've done this exercise several times.

```
#ifndef USART_H
#define USART_H

#include <stdint.h>

// USART configuration struct
typedef struct {
    uint32_t baud_rate; // Baud rate for USART communication
    uint8_t data_bits; // Number of data bits (5-9)
    uint8_t stop_bits; // Number of stop bits (1 or 2)
    uint8_t parity; // Parity mode (0: none, 1: odd, 2: even)
} USART_Config;

// Initialize USART with the given configuration
void USART_Init(USART_Config config);

// Send a single byte over USART
void USART_SendByte(uint8_t byte);

// Receive a single byte from USART
uint8_t USART_ReceiveByte(void);

// Send a byte array over USART
void USART_SendBytes(const uint8_t* data, uint32_t size);

// Receive a byte array from USART
void USART_ReceiveBytes(uint8_t* buffer, uint32_t size);

// Send a null-terminated string over USART
void USART_SendString(const uint8_t* str);

// Receive a null-terminated string from USART
void USART_ReceiveString(uint8_t* buffer, uint32_t buffer_size);

#endif // USART_H
```

🔄 Regenerate response

Generating a USART HAL using ChatGPT

JA

This is a good start. Can you update the usart interface based on the following changes 1) Can you simplify the interface so that there are not so many repetitive functions 2) The interface only supports one device. Can you add support for multiple usart devices 3) Instead of using _ in parameter variable names, please use camel. For example, buffer_size would become bufferSize

Analysis:

- The interface still has the string functions which I would not include in a low-level driver. This is a library function.
- The device has been added but coupled with USART_Config.
- Each function does have a device pointer, but these should be const
- There are a lot of useless comments in the code
- There's no considerations for interrupts or DMA through callback functions

Note: You may get different results. I've done this exercise several times.

©2023 Beningo Embedded Group, LLC. All Rights Reserved

```
#ifndef USART_H
#define USART_H

#include <stdint.h>

// USART configuration struct
typedef struct {
    uint32_t baudRate;    // Baud rate for USART communication
    uint8_t dataBits;    // Number of data bits (5-9)
    uint8_t stopBits;    // Number of stop bits (1 or 2)
    uint8_t parity;      // Parity mode (0: none, 1: odd, 2: even)
} USART_Config;

// USART device struct
typedef struct {
    uint32_t usartId;    // Identifier for the USART device
    USART_Config config; // Configuration for USART communication
} USART_Device;

// Initialize USART device with the given configuration
void USART_Init(USART_Device* device);

// Send data over USART
void USART_Send(USART_Device* device, const uint8_t* data, uint32_t size);

// Receive data from USART
void USART_Receive(USART_Device* device, uint8_t* buffer, uint32_t bufferSize);

// Send a null-terminated string over USART
void USART_SendString(USART_Device* device, const uint8_t* str);

// Receive a null-terminated string from USART
void USART_ReceiveString(USART_Device* device, uint8_t* buffer, uint32_t bufferSize);

#endif // USART_H
```

Regenerate response

Generating a USART HAL using ChatGPT

JA Update the latest version of the interface with the following: 1) remove the string functions from the interface. 2) Remove the comments above each function and structure. 3) USART_Device is not a useful structure. Remove it. 4) Update USART_Device to be an enum of usartDevice_t that lists the available usart interfaces.

Analysis:

- The interface now looks slim and cleaner!
- We have a typedef list of USARTs to interact with.
- Unfortunately, USART_Config comments were removed.
- Not a big fan of USART_COUNT. I prefer USART_DEVICE_ID_MAX.
- Would prefer usartDevice_t be usartDeviceID_t.
- There's no considerations for interrupts or DMA through callback functions

Note: You may get different results. I've done this exercise several times.

```
c Copy code
#ifndef USART_H
#define USART_H

#include <stdint.h>

typedef enum {
    USART_1,
    USART_2,
    USART_3,
    // Add more USART interfaces as needed
    USART_COUNT
} usartDevice_t;

typedef struct {
    uint32_t baudRate;
    uint8_t dataBits;
    uint8_t stopBits;
    uint8_t parity;
} USART_Config;

void USART_Init(usartDevice_t device, USART_Config config);
void USART_Send(usartDevice_t device, const uint8_t* data, uint32_t size);
void USART_Receive(usartDevice_t device, uint8_t* buffer, uint32_t bufferSize);

#endif // USART_H
```

Generating a USART HAL using ChatGPT

JA Update the latest interface version with the following changes: 1) change usartDevice_t to usartDeviceID_t 2) Change USART_COUNT to USART_MAX 3) Change USART_Cong to usartConfig_t 4) When passing the usartConfig_t to USART_Init, the parameter should be const

Analysis:

- We're zeroing in on a reasonable interface!
- Our changes have been implemented.

What's missing?:

- The functions don't return error codes
- We are ignoring the dependency inversion rule
- There's no considerations for interrupts or DMA through callback functions

Note: You may get different results. I've done this exercise several times.

```
c Copy code
#ifndef USART_H
#define USART_H

#include <stdint.h>

typedef enum {
    USART_1,
    USART_2,
    USART_3,
    // Add more USART interfaces as needed
    USART_MAX
} usartDeviceID_t;

typedef struct {
    uint32_t baudRate;
    uint8_t dataBits;
    uint8_t stopBits;
    uint8_t parity;
} usartConfig_t;

void USART_Init(usartDeviceID_t device, const usartConfig_t config);
void USART_Send(usartDeviceID_t device, const uint8_t* data, uint32_t size);
void USART_Receive(usartDeviceID_t device, uint8_t* buffer, uint32_t bufferSize);

#endif // USART_H
```

Generating a USART HAL using ChatGPT

JA Update the latest interface with the following changes: 1) Each function in the interface should return an error code of type `usartError_t` 2) The functions should be wrapped in a typedef struct where each function of the interface is a function pointer. 3) Add a callback function to the `usartConfig_t` structure.

Analysis:

- At this point, we have a pretty well-defined interface. There are still minor changes that should be fixed by hand:
 - The callback function probably would have void parameters
 - The USART_Receive function pointer has a bufferSize parameter which should really just be size
 - The header file is USART_H when it should be HAL_USART or USART_INTERFACE
 - The AI decided to define an extern `usartInterface` variable that can be removed.

Note: You may get different results. I've done this exercise several times.

```
#ifndef USART_H
#define USART_H

#include <stdint.h>

typedef enum {
    USART_1,
    USART_2,
    USART_3,
    // Add more USART interfaces as needed
    USART_MAX
} usartDeviceID_t;

typedef enum {
    USART_OK,
    USART_ERROR
} usartError_t;

typedef struct {
    uint32_t baudRate;
    uint8_t dataBits;
    uint8_t stopBits;
    uint8_t parity;
    void (*callback)(usartError_t error); // Callback function for error handling
} usartConfig_t;

typedef struct {
    usartError_t (*USART_Init)(usartDeviceID_t device, const usartConfig_t* config);
    usartError_t (*USART_Send)(usartDeviceID_t device, const uint8_t* data, uint8_t size);
    usartError_t (*USART_Receive)(usartDeviceID_t device, uint8_t* buffer, uint8_t bufferSize);
} usartInterface_t;

extern const usartInterface_t USART_Interface;

#endif // USART_H
```

Regenerate response

Audience POLL Question

How do you rate the quality of the USART HAL?

- a) High quality
- b) Medium quality
- c) Low quality
- d) Sorry, what's a HAL?



Generating SPI and I2C HALs using ChatGPT

03

Generating SPI and I2C HALs using ChatGPT

Best Practices for generating code:

- 1) Be very specific one what you want. The more information you feed it, the closer it will get on the first try.
- 2) Use iterative feedback to get to the results you want
- 3) Feed in examples to more closely get the results that you want.

Generating SPI and I2C HALs using ChatGPT

JA Based on the usart interface design and coding style, can you create an interface for a SPI peripheral?

Analysis:

- With an example, it generated a pretty decent Spi interface!
- There may be a few tweaks needed, (I'd just simplify the interface to SPI_Transfer), but it could be used as is!

```
#ifndef SPI_INTERFACE_H
#define SPI_INTERFACE_H

#include <stdint.h>

typedef enum {
    SPI_1,
    SPI_2,
    SPI_3,
    // Add more SPI interfaces as needed
    SPI_MAX
} spiDeviceID_t;

typedef enum {
    SPI_OK,
    SPI_ERROR
} spiError_t;

typedef struct {
    uint32_t clockSpeed;
    uint8_t dataMode;
    uint8_t bitOrder;
    void (*callback)(void);
} spiConfig_t;

typedef struct {
    spiError_t (*SPI_Init)(spiDeviceID_t device, const spiConfig_t* config);
    spiError_t (*SPI_Transmit)(spiDeviceID_t device, const uint8_t* data, uint
    spiError_t (*SPI_Receive)(spiDeviceID_t device, uint8_t* buffer, uint32_t
    spiError_t (*SPI_TransmitReceive)(spiDeviceID_t device, const uint8_t* txD
} spiInterface_t;

#endif // SPI_INTERFACE_H
```

Regenerate response

Generating SPI and I2C HALs using ChatGPT

JA based on the spi interface design and coding style, can you create an interface for a I2C peripheral?

Analysis:

- With an example, it generated a pretty decent I2C interface!
- There may be some small adjustments needed, but once I fed in an example, it took ~5 seconds to get the results.

```
#ifndef I2C_INTERFACE_H
#define I2C_INTERFACE_H

#include <stdint.h>

typedef enum {
    I2C_1,
    I2C_2,
    I2C_3,
    // Add more I2C interfaces as needed
    I2C_MAX
} i2cDeviceID_t;

typedef enum {
    I2C_OK,
    I2C_ERROR
} i2cError_t;

typedef struct {
    uint32_t clockSpeed;
    uint8_t address;
    void (*callback)(void);
} i2cConfig_t;

typedef struct {
    i2cError_t (*I2C_Init)(i2cDeviceID_t device, const i2cConfig_t* config);
    i2cError_t (*I2C_Write)(i2cDeviceID_t device, const uint8_t* data, uint32_t size);
    i2cError_t (*I2C_Read)(i2cDeviceID_t device, uint8_t* buffer, uint32_t size);
    i2cError_t (*I2C_WriteRead)(i2cDeviceID_t device, const uint8_t* txData, uint8_t* rxData, uint32_t size);
} i2cInterface_t;

#endif // I2C_INTERFACE_H
```

Audience POLL Question

How do you view AI after this session:

- a) An excellent tool to help you accelerate development
- b) An interesting tool that requires more time to mature
- c) A threat to your job
- d) A threat to all mankind



Next Steps

04

Additional Resources

Please consider the resources below:

- [Jacob's AI Blogs](#)
- [Jacob's CEC courses](#)
- [Jacob's ML Blogs](#)
- Embedded Bytes Newsletter
 - <http://bit.ly/1BAHYXm>

www.beningo.com



Consulting

Coaching

Training

Next Steps

- ✓ The Rise of AI in Embedded Software
- ✓ Writing Better Prompts for Code Generation
- ✓ Optimizing your Build System with AI
- ✓ Abstracting your Hardware with an AI-Generated HAL
- Managing AI and ML Code



DesignNews

Thank You

Sponsored by

DigiKey

