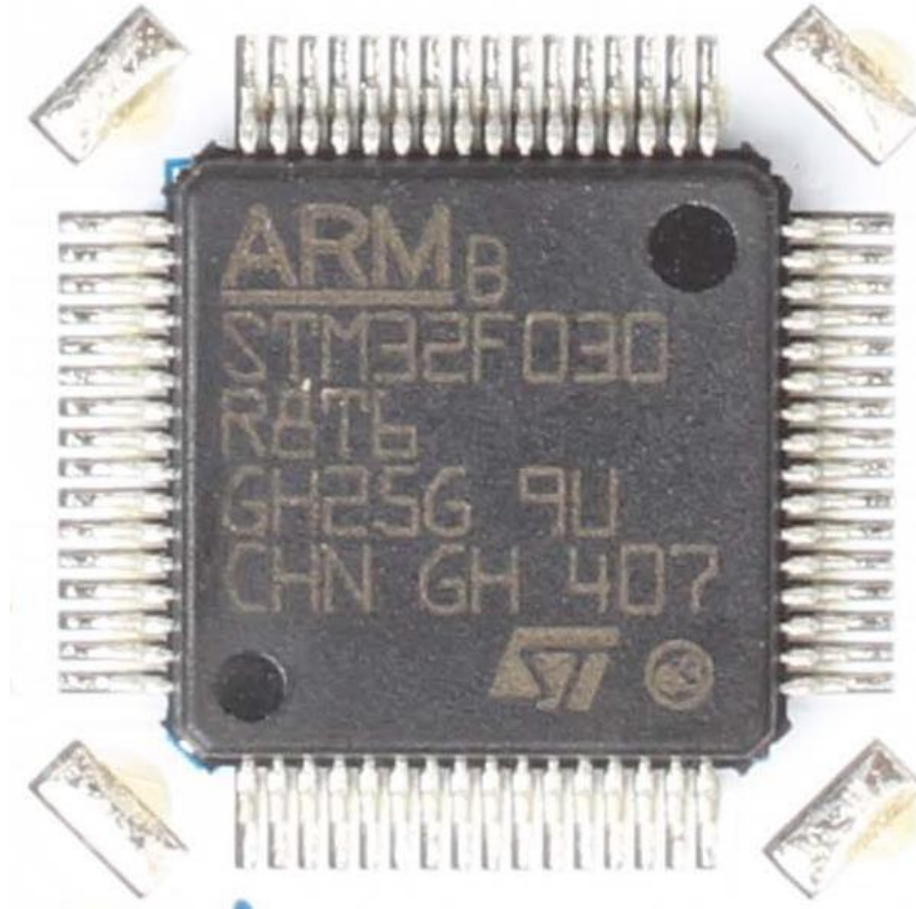


Essential Coding Techniques for Hardware Engineers



Pouring a Logical Foundation

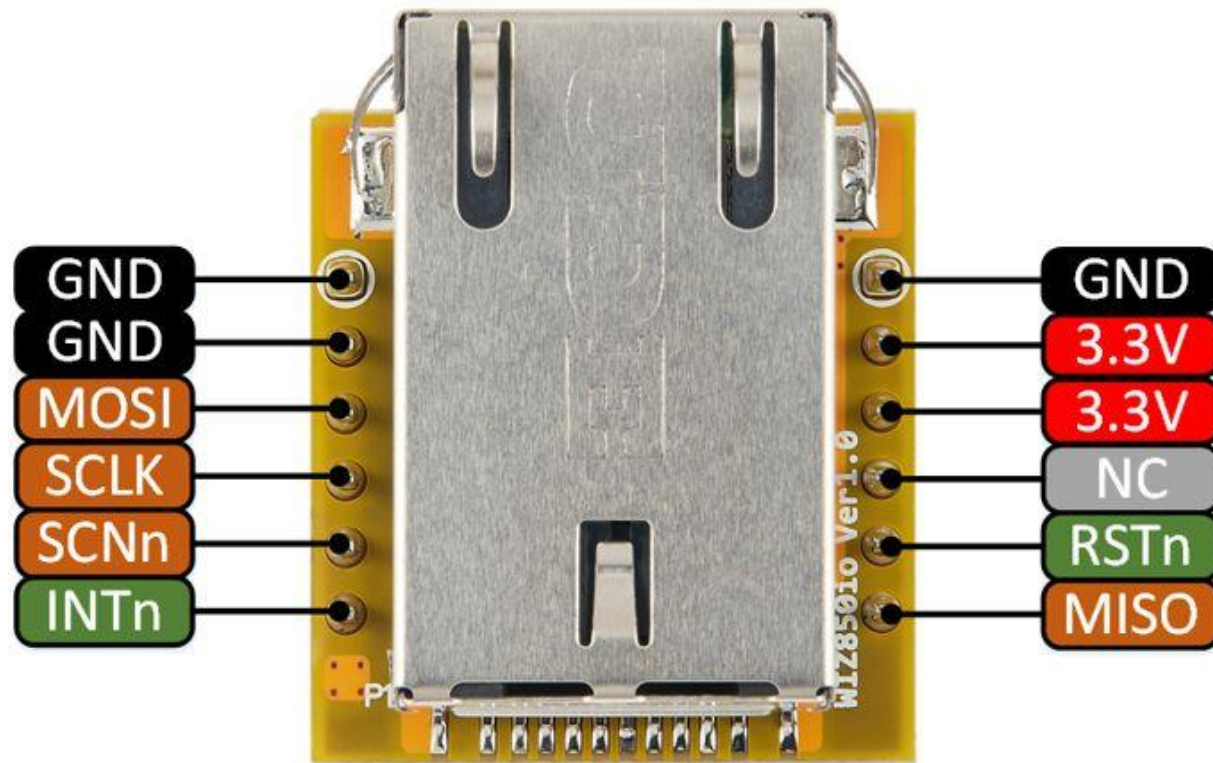
January 28, 2019

Fred Eady

Essential Coding Techniques for Hardware Engineers

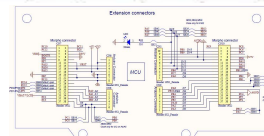
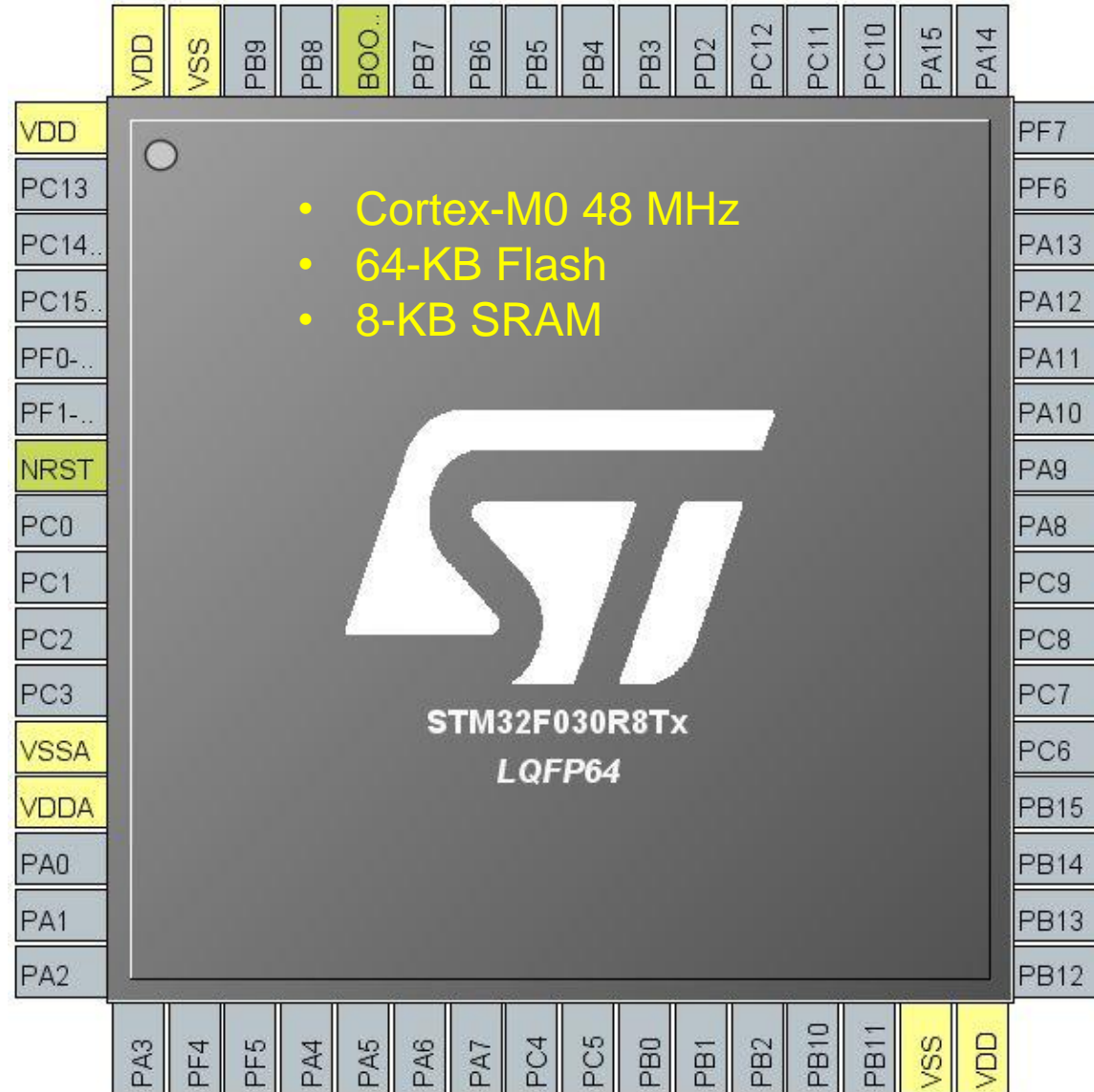
AGENDA

- Hardware – **The Graphical View**
- Hardware – **The Physical View**
- Hardware – **The Logical View**
- Day 1 Summary



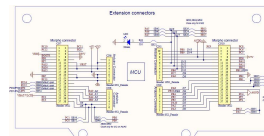
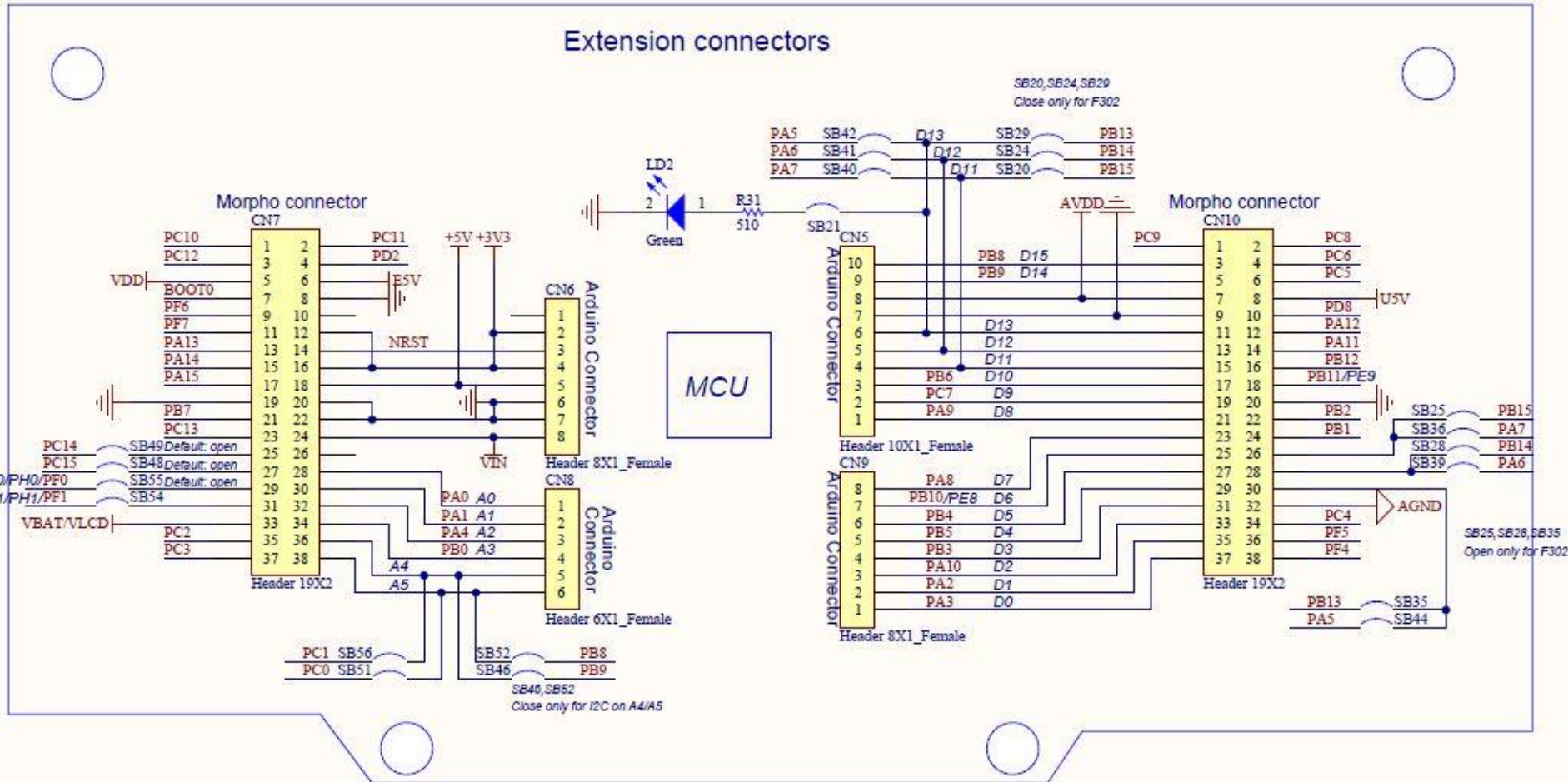
Essential Coding Techniques for Hardware Engineers

Hardware - The Graphical View



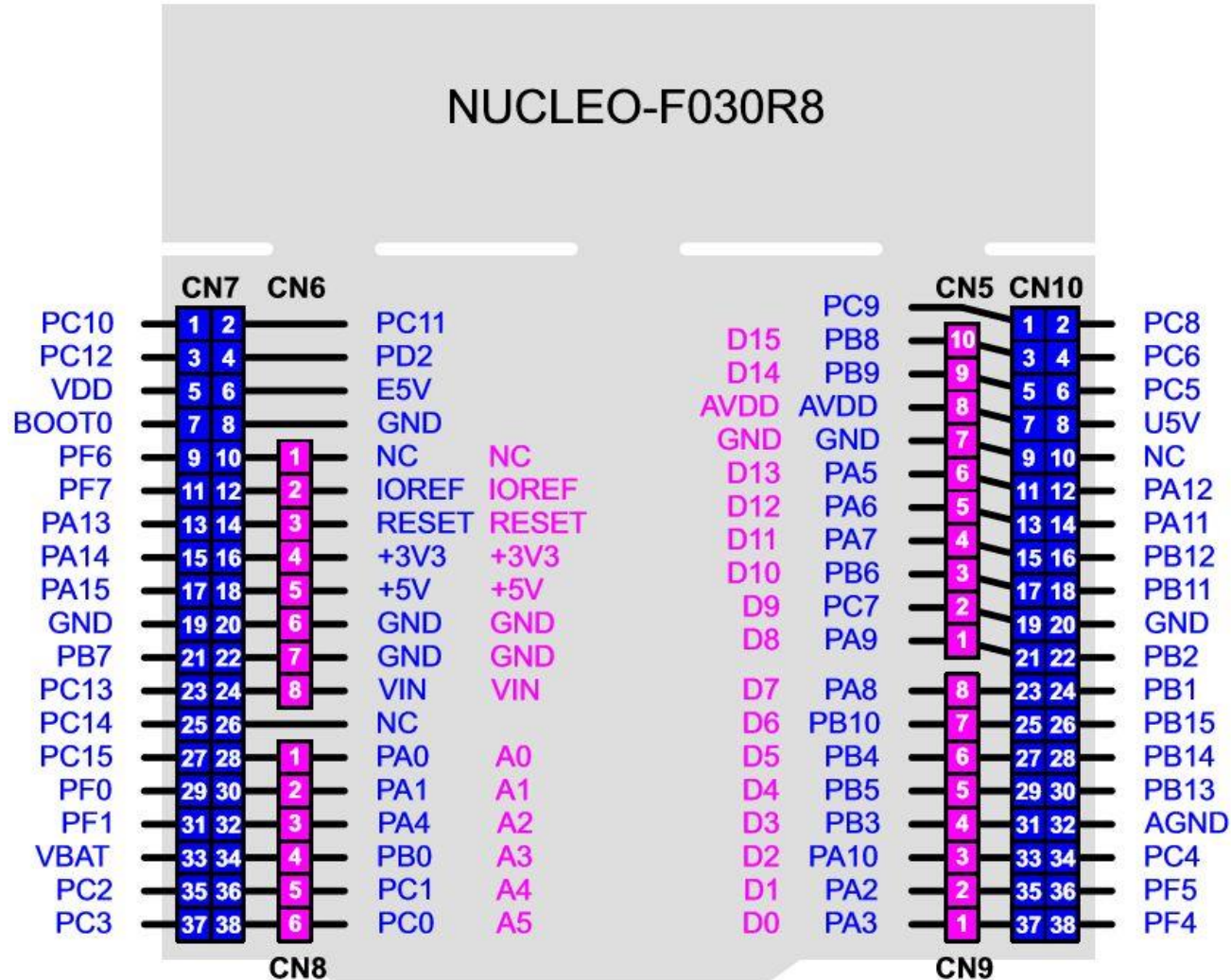
Essential Coding Techniques for Hardware Engineers

Hardware - The Graphical View



Essential Coding Techniques for Hardware Engineers

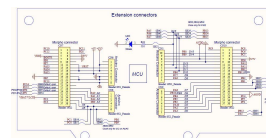
Hardware - The Graphical View



Arduino

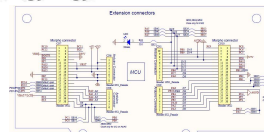
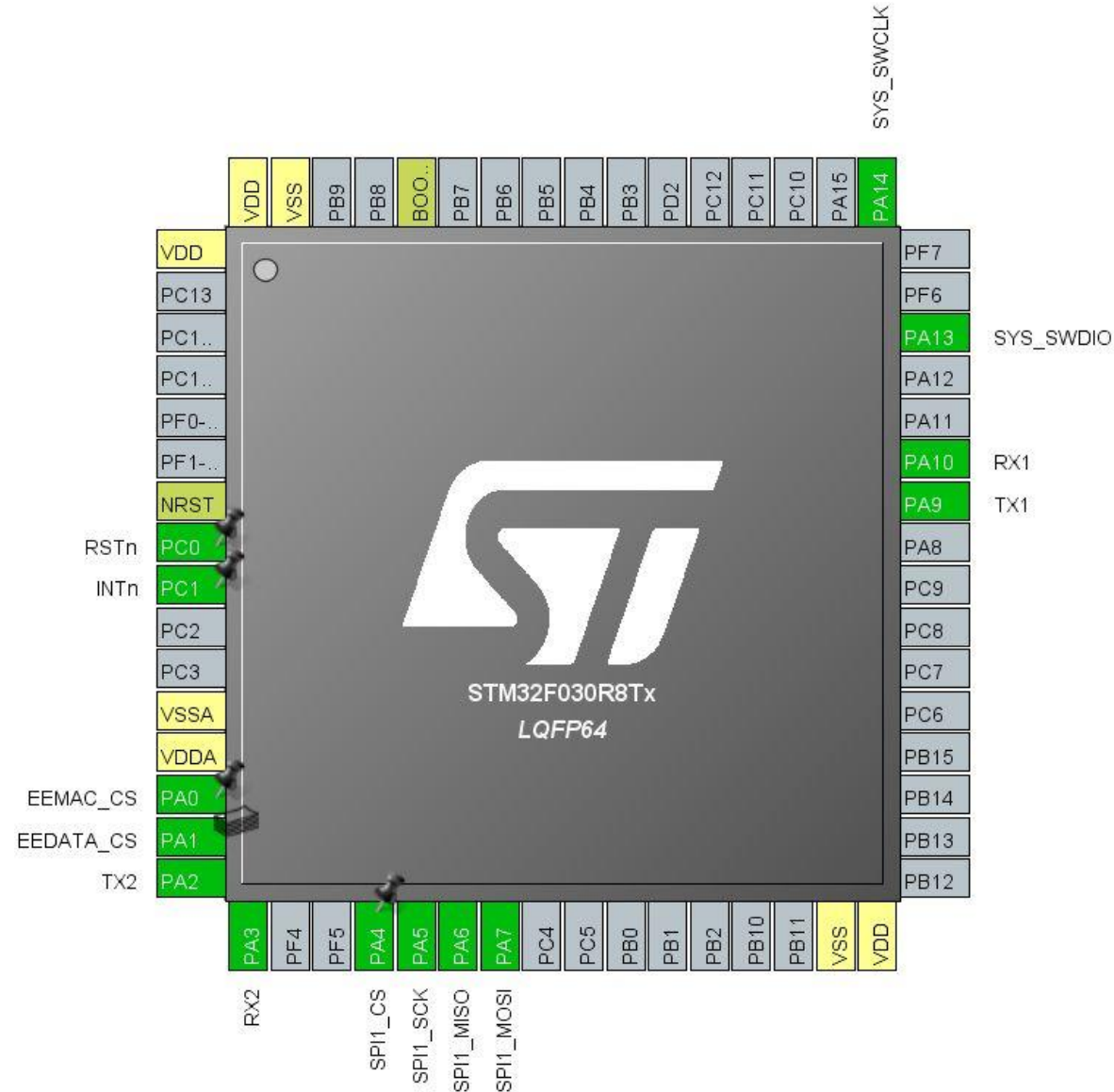


Morpho



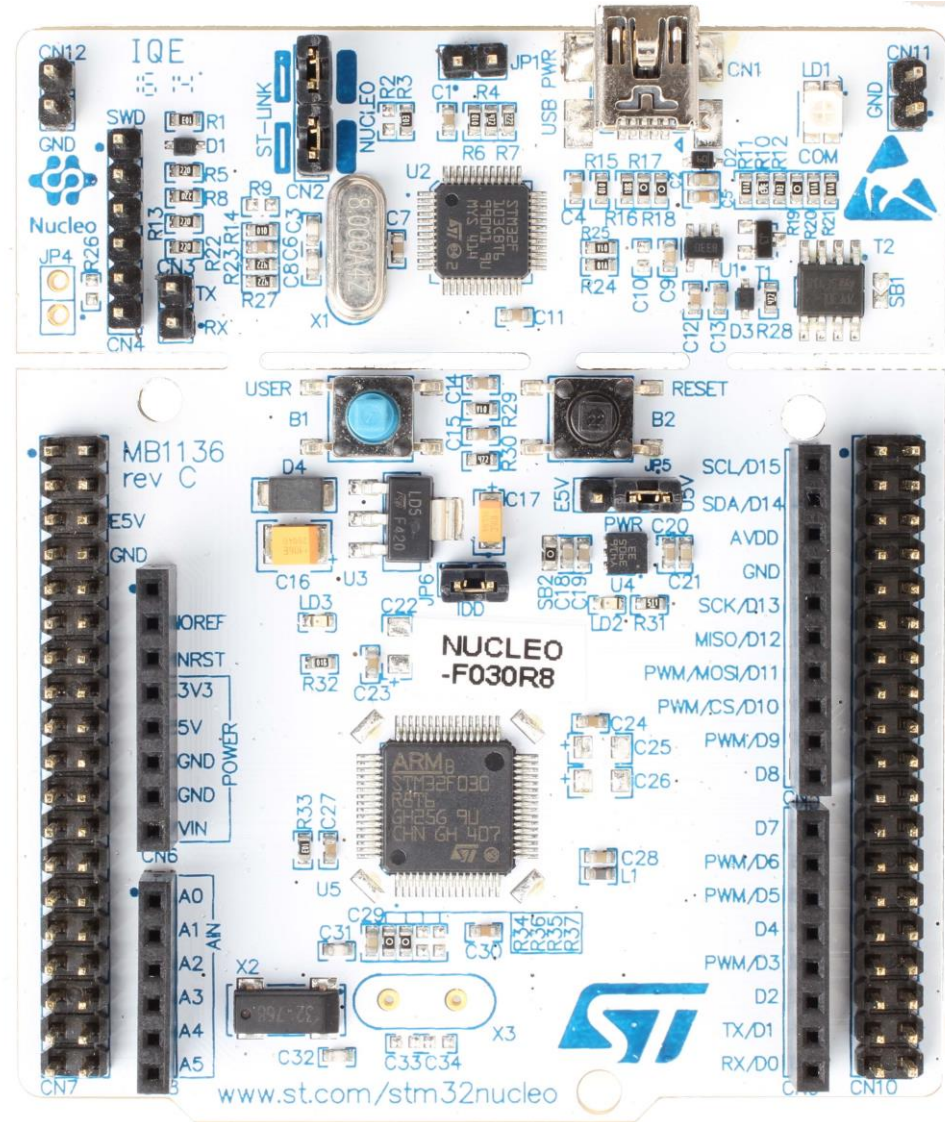
Essential Coding Techniques for Hardware Engineers

Hardware - The Graphical View



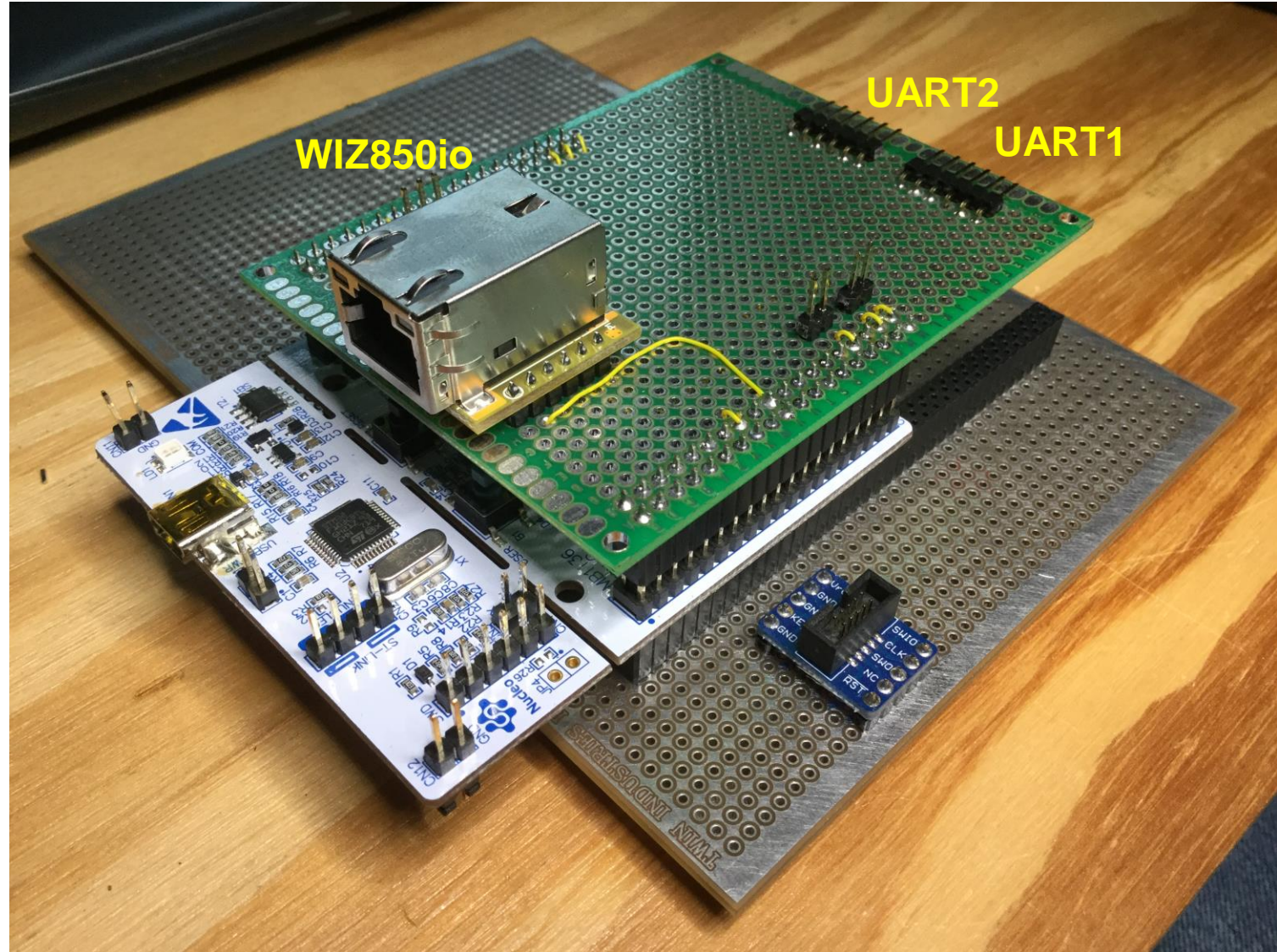
Essential Coding Techniques for Hardware Engineers

Hardware - The Physical View



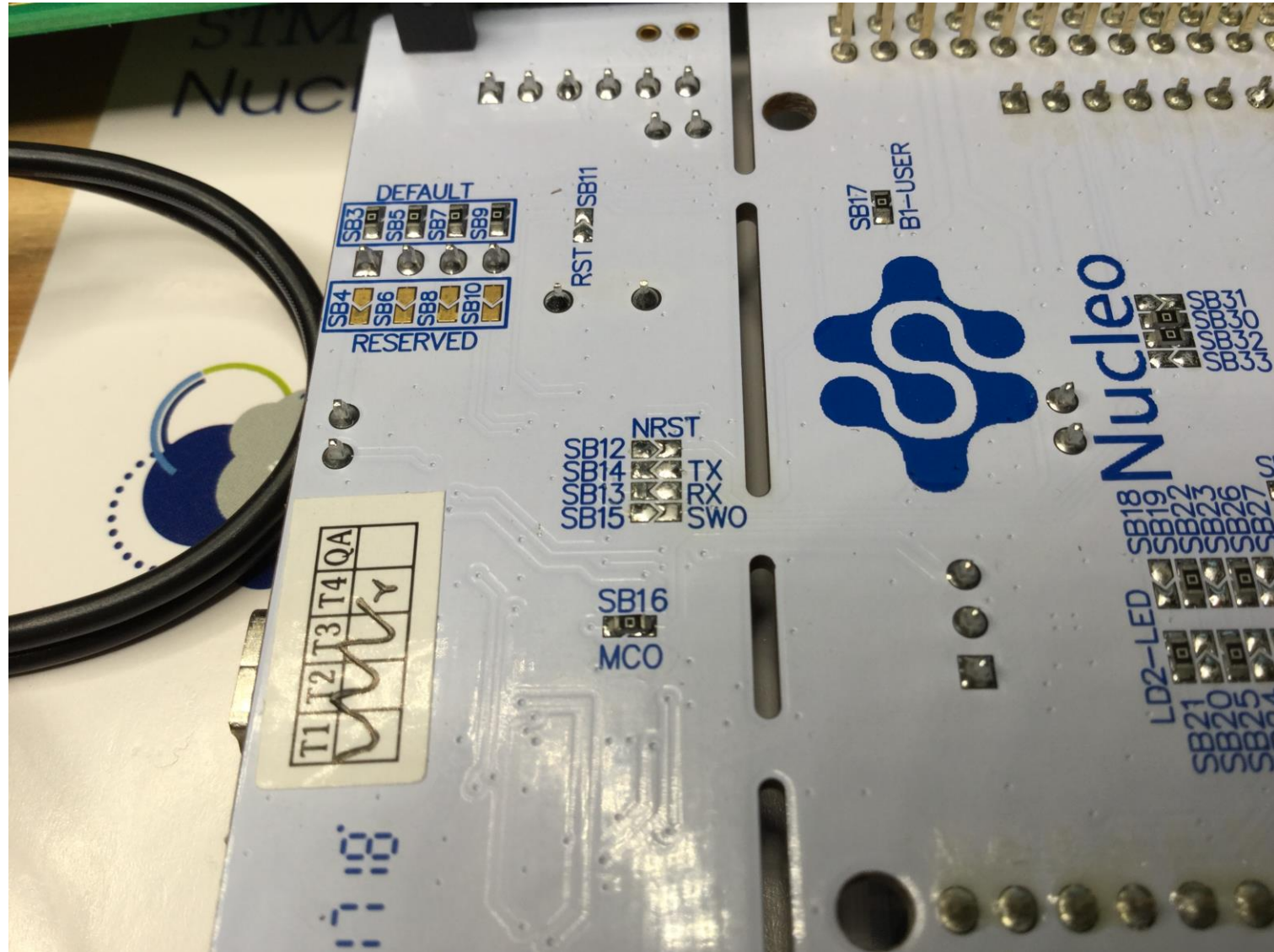
Essential Coding Techniques for Hardware Engineers

Hardware - The Physical View



Essential Coding Techniques for Hardware Engineers

Hardware - The Physical View



Essential Coding Techniques for Hardware Engineers

Hardware - The Physical View

6 pin header
0.1in pitch

1	GND	BLACK
2	CTS#	BROWN
3	VCC	RED
4	TXD	ORANGE
5	RXD	YELLOW
6	RTS#	GREEN

CABLE 1.8m

LAN USB
j-link Pro
SEGGER
www.segger.com
Target

Uref SWIO CN7 Pin 13 - PA13
GND CLK CN7 Pin 15 - PA14
GND SWO CN10 Pin 31 - PB3
KEY NC
GND RST CN7 Pin 14 - NRST

CN7 Pin 12 - 3.3V
CN7 Pins 19-20-22 = GND



Essential Coding Techniques for Hardware Engineers

Hardware - The Logical View: UART Initialization

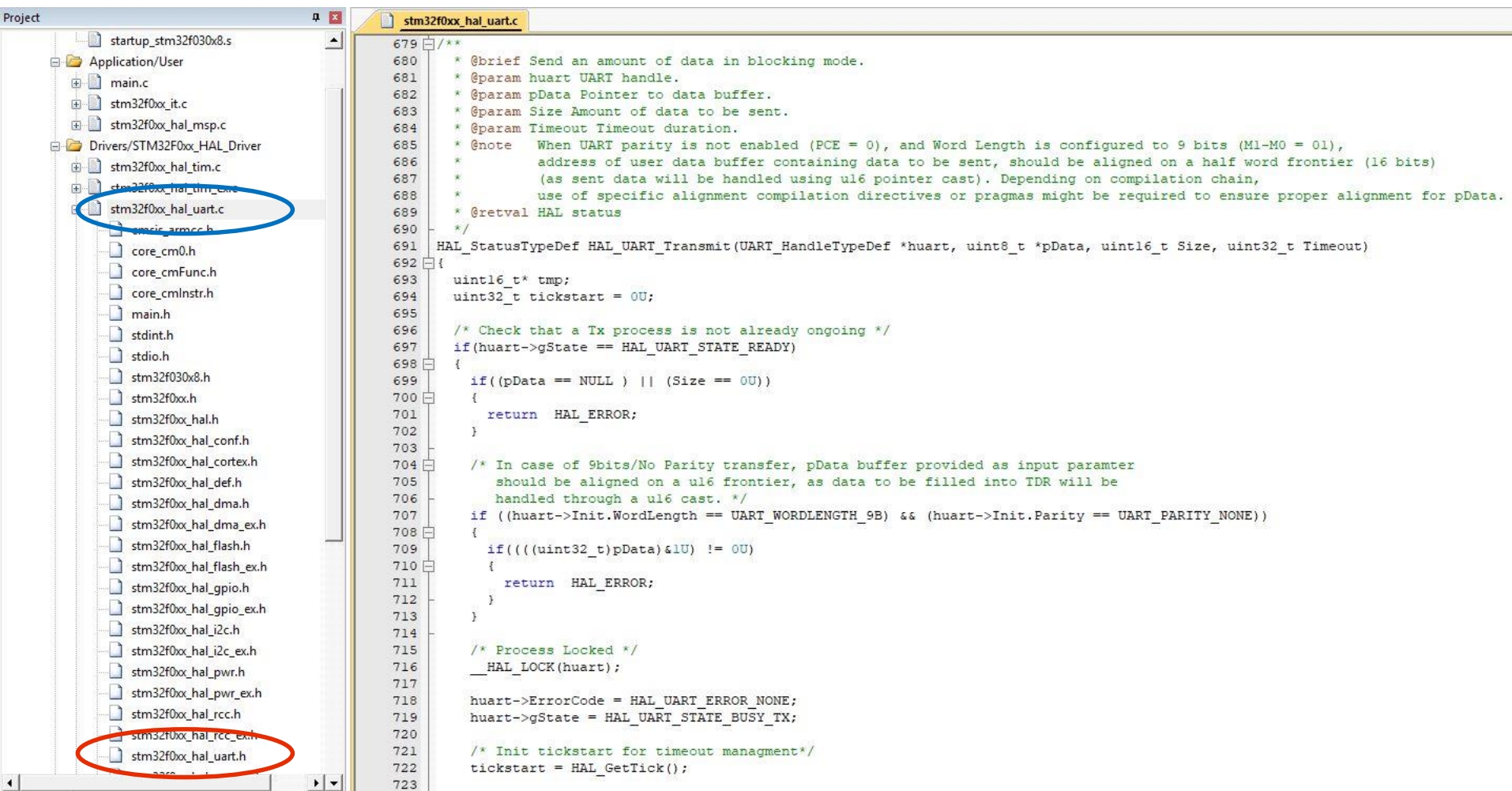
```
80 //*****
81 /** TYPEDEFS
82 //*****
83 SPI_HandleTypeDef hspil;
84
85 UART_HandleTypeDef huart1;
86 UART_HandleTypeDef huart2;
```

```
897 static void MX_USART2_UART_Init(void)
898 {
899     huart2.Instance = USART2;
900     huart2.Init.BaudRate = 115200;
901     huart2.Init.WordLength = UART_WORDLENGTH_8B;
902     huart2.Init.StopBits = UART_STOPBITS_1;
903     huart2.Init.Parity = UART_PARITY_NONE;
904     huart2.Init.Mode = UART_MODE_TX_RX;
905     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
906     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
907     huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
908     huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
909     huart1.Mask = 0x00FF;
910     if (HAL_UART_Init(&huart2) != HAL_OK)
911     {
912         Error_Handler();
913     }
914 }
915
```



Essential Coding Techniques for Hardware Engineers

Hardware - The Logical View: Transmit a Character Using HAL



```
679 /**
680  * @brief Send an amount of data in blocking mode.
681  * @param huart UART handle.
682  * @param pData Pointer to data buffer.
683  * @param Size Amount of data to be sent.
684  * @param Timeout Timeout duration.
685  * @note When UART parity is not enabled (PCE = 0), and Word Length is configured to 9 bits (M1-M0 = 01),
686  *       address of user data buffer containing data to be sent, should be aligned on a half word frontier (16 bits)
687  *       (as sent data will be handled using ul6 pointer cast). Depending on compilation chain,
688  *       use of specific alignment compilation directives or pragmas might be required to ensure proper alignment for pData.
689  * @retval HAL status
690  */
691 HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
692 {
693     uint16_t* tmp;
694     uint32_t tickstart = 0U;
695
696     /* Check that a Tx process is not already ongoing */
697     if(huart->gState == HAL_UART_STATE_READY)
698     {
699         if((pData == NULL) || (Size == 0U))
700         {
701             return HAL_ERROR;
702         }
703
704         /* In case of 9bits/No Parity transfer, pData buffer provided as input parameter
705          should be aligned on a ul6 frontier, as data to be filled into TDR will be
706          handled through a ul6 cast. */
707         if ((huart->Init.WordLength == UART_WORDLENGTH_9B) && (huart->Init.Parity == UART_PARITY_NONE))
708         {
709             if((((uint32_t)pData)&1U) != 0U)
710             {
711                 return HAL_ERROR;
712             }
713         }
714
715         /* Process Locked */
716         __HAL_LOCK(huart);
717
718         huart->ErrorCode = HAL_UART_ERROR_NONE;
719         huart->gState = HAL_UART_STATE_BUSY_TX;
720
721         /* Init tickstart for timeout management*/
722         tickstart = HAL_GetTick();
723     }
```



Essential Coding Techniques for Hardware Engineers

Hardware - The Logical View: Transmit a Character Using HAL

The screenshot shows the Hercules SETUP utility interface. The main window is titled "Hercules SETUP utility by HW-group.com". It has several tabs: "UDP Setup", "Serial", "TCP Client", "TCP Server", "UDP", "Test Mode", and "About". The "Serial" tab is selected. The interface is divided into several sections:

- Received/Sent data:** A large text area showing a stream of 'S' characters. A red box highlights a code snippet within this area:

```
115     uint8_t txBuf = 0x35;
116     while (1)
117     {
118         HAL_UART_Transmit (&huart2, &txBuf, 1, 0xFFFF);
119     }
```

- Serial Settings:** A panel on the right with the following settings:
 - Name: COM23
 - Baud: 115200
 - Data size: 8
 - Parity: None
 - Stop bits: 1
 - Flow control: None
- Modem lines:** A row of checkboxes for CD, RI, DSR, CTS, DTR, and RTS. CD, RI, and DSR are checked.
- Send:** Three input fields for sending data, each with a "Send" button and a "HEX" checkbox.
- Buttons:** "Close" (with a red X), "HW/g FW update", and "HWgroup" logo.



Essential Coding Techniques for Hardware Engineers

Hardware - The Logical View: Send a Character Using *fputc*

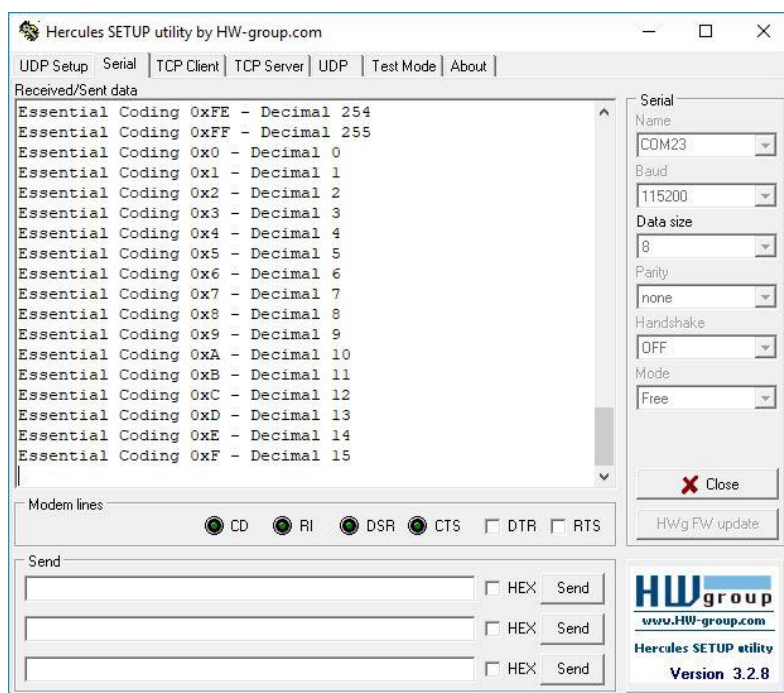
```
39  /* Includes -----  
40  #include "main.h"  
41  #include "stm32f0xx_hal.h"  
42  #include <stdio.h>
```

```
stdio.h  
673 extern _ARMABI int fputc(int /*c*/, FILE * /*stream*/) __attribute__((__nonnull__(2)));  
674 /*  
675  * writes the character specified by c (converted to an unsigned char) to  
676  * the output stream pointed to by stream, at the position indicated by the  
677  * associated file position indicator (if defined), and advances the  
678  * indicator appropriately. If the file position indicator is not defined,  
679  * the character is appended to the output stream.  
680  * Returns: the character written. If a write error occurs, the error  
681  * indicator is set and fputc returns EOF.  
682  */
```



Essential Coding Techniques for Hardware Engineers

Hardware - The Logical View: *printf* Implementation



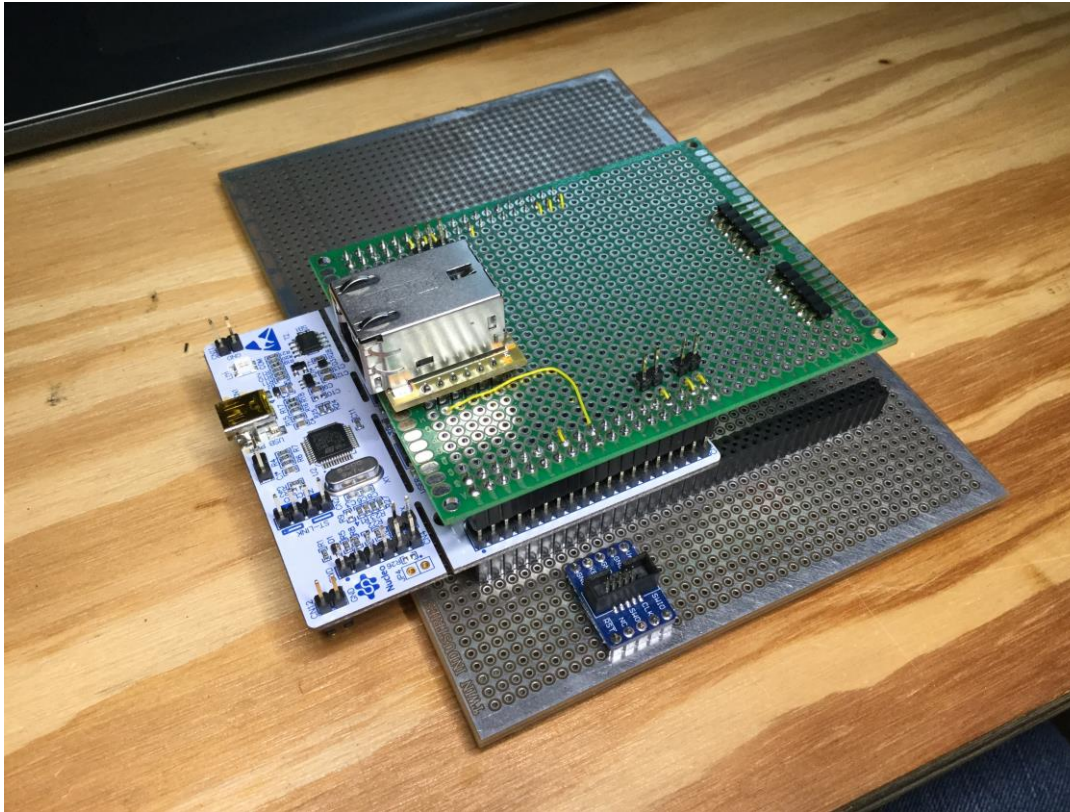
```
168 //*****
169 /* PRINTF
170 //*****
171 int fputc(int ch, FILE *f)
172 {
173     HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
174     return ch;
175 }
```

```
731     scratch8 = 0;
732     do
733     {
734         printf("Essential Coding 0x%0X - Decimal %u\r\n",scratch8,scratch8);
735         ++scratch8;
736         HAL_Delay(10);
737     }while(1);
```

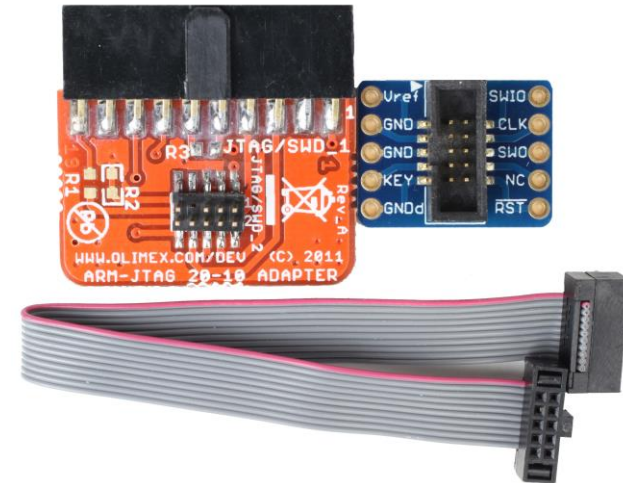


Essential Coding Techniques for Hardware Engineers

Day 1 Summary



```
168 //*****
169 /** PRINTF
170 //*****
171 int fputc(int ch, FILE *f)
172 {
173     HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
174     return ch;
175 }
```



Presented by:

Essential Coding Techniques for Hardware Engineers

A Peek At What's To Come

