Embedded Controller programming for Real Time Systems: ECE-40097

Lesson 6

Vijay Kumar

Main Topics

IDE File structure

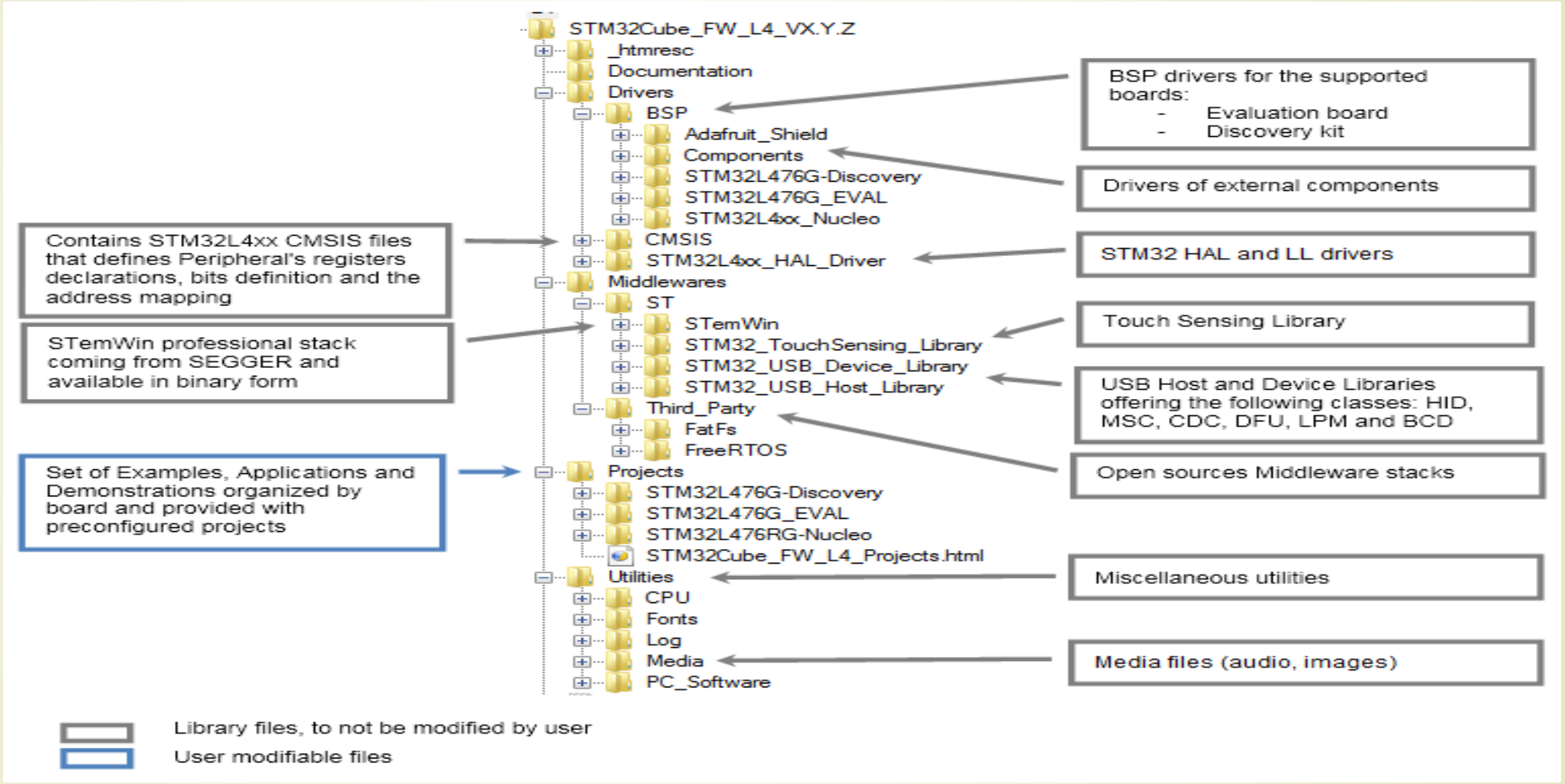
HAL API

CMSIS Layer

Interrupt controller registers

Code examples

File Structure



API

```
/* NVIC for USART */
  HAL_NVIC_SetPriority(UART4_IRQn, 0, 1);
  HAL_NVIC_EnableIRQ(UART4_IRQn);

#define NVIC_EnableIRQ          __NVIC_EnableIRQ


UART4_IRQn                = 52,     /*!< UART4 global Interrupt

void HAL_NVIC_EnableIRQ(IRQn_Type IRQn)
{
  /* Check the parameters */
  assert_param(IS_NVIC_DEVICE_IRQ(IRQn));

  /* Enable interrupt */
  NVIC_EnableIRQ(IRQn);
}
```

API

- in file stm32l4xx_hal.c

```
 /**
  * @brief This function provides minimum delay (in milliseconds) based
  *        on variable incremented.
  * @note In the default implementation , SysTick timer is the source of time base.
  *       It is used to generate interrupts at regular time intervals where uwTick
  *       is incremented.
  * @note This function is declared as __weak to be overwritten in case of other
  *       implementations in user file.
  * @param Delay  specifies the delay time length, in milliseconds.
  * @retval None
  */
        __weak void HAL_Delay(uint32_t Delay)
        {
        }

        // get tickcount
        __weak uint32_t HAL_GetTick(void)
        {
          return uwTick;
        }
```

Enable Interrupt

```
/**
  \brief   Enable Interrupt
  \details Enables a device specific interrupt in the NVIC interrupt controller.
  \param [in]      IRQn  Device specific interrupt number.
  \note    IRQn must not be negative.
 */
__STATIC_INLINE void __NVIC_EnableIRQ(IRQn_Type IRQn)
{
        if ((int32_t)(IRQn) >= 0)
        {
          NVIC->ISER[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
        }
}
```

Interrupt Priority

```
void HAL_NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
  /* Check the parameters */
  assert_param(IS_NVIC_PRIORITY_GROUP(PriorityGroup));

  /* Set the PRIGROUP[10:8] bits according to the PriorityGroup
parameter value */
  NVIC_SetPriorityGrouping(PriorityGroup);
}
```

#define NVIC_PRIORITYGROUP_3
((uint32_t)0x00000004) /*!< 3 bits for pre-emption priority,

- 3 bits are used for priority
  - Max of 8 values (0 to 7)
- 1 bits are use for sub priority
  - Max of 2 values (0 or 1)

CMSIS function

```
* ##################################   Core Function Access
################################## */
/** \ingroup  CMSIS_Core_FunctionInterface
    \defgroup CMSIS_Core_RegAccFunctions CMSIS Core
Register Access Functions
  @{
 */

/**
  \brief   Enable IRQ Interrupts
  \details Enables IRQ interrupts by clearing the I-bit in the CPSR.
           Can only be executed in Privileged modes.
 */
__STATIC_FORCEINLINE void __enable_irq(void)
{
  __ASM volatile ("cpsie i" : : : "memory");
}
```

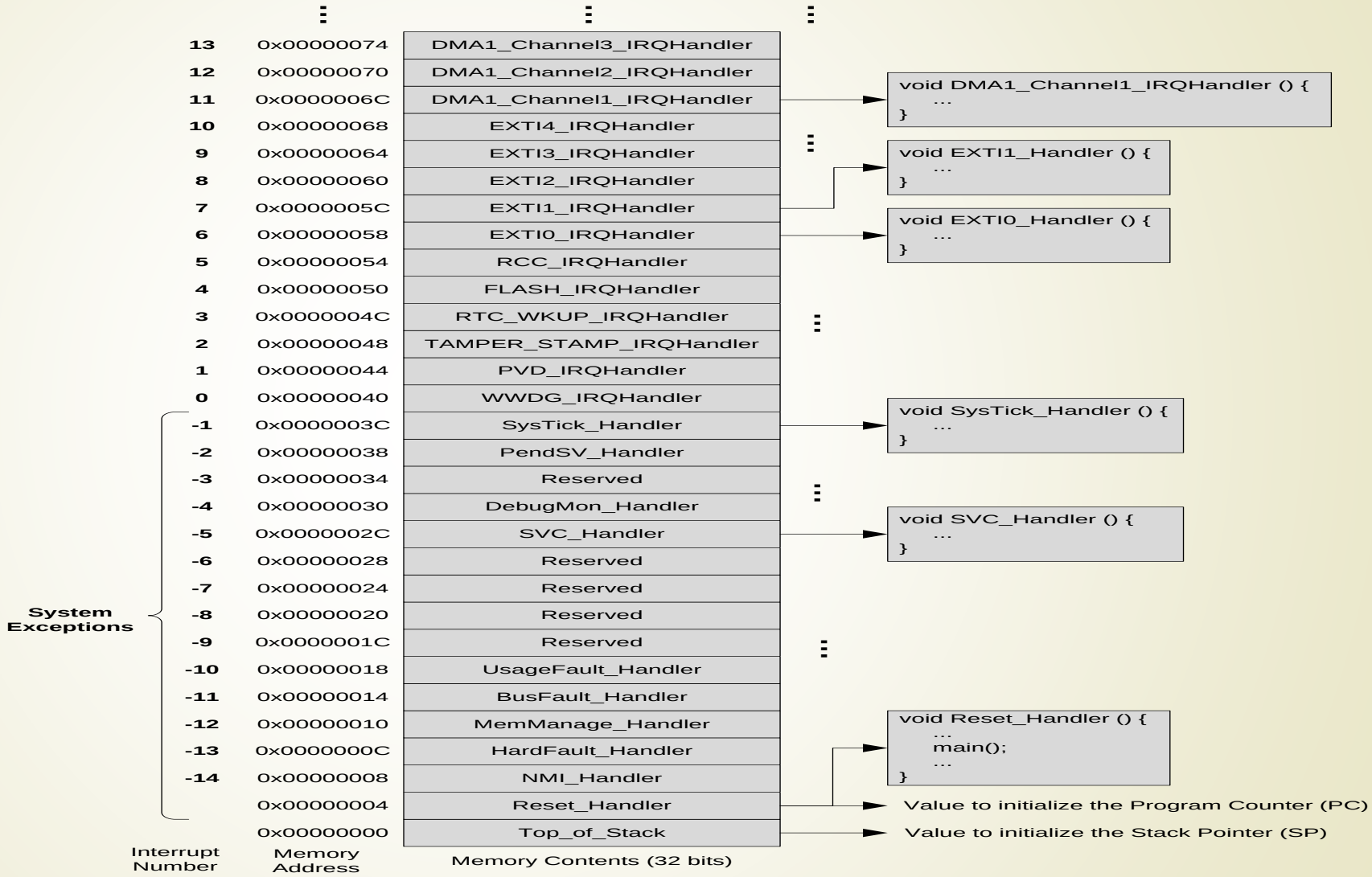
startup_stm32l475xx.s File

- Located under startup file
- First instruction executed after board reset/power on

```
 .section.text.Reset_Handler
.weakReset_Handler
.typeReset_Handler, %function
Reset_Handler:
  ldr   sp, =_estack    /* Atollic update: set stack
pointer */

/* Copy the data segment initializers from flash to
SRAM */
  movsr1, #0
  bLoopCopyDataIn
}
```

Vector Table



Vector Table

```
* ############################  Core Function Access  ############################ */
/** \ingroup  CMSIS_Core_FunctionInterface
    \defgroup CMSIS_Core_RegAccFunctions CMSIS Core Register Access Functions
  @{
 .section.isr_vector,"a",%progbits
.typeg_pfnVectors, %object
.sizeg_pfnVectors, .-g_pfnVectors


g_pfnVectors:
.word_estack
.word           Reset_Handler
.word           NMI_Handler
.word           HardFault_Handler
.word           MemManage_Handler
.word           BusFault_Handler
.word           UsageFault_Handler

.word           SPI2_IRQHandler
.word           USART1_IRQHandler
.word           USART2_IRQHandler
.word           USART3_IRQHandler
.word           EXTI15_10_IRQHandler
 */
```

NVIC Registers

Location: core_cm4.h

```
// brief  Structure type to access the Nested Vectored Interrupt Controller (NVIC).
typedef struct
{
  __IOM uint32_t ISER[8U];               /*!< Offset: 0x000 (R/W)  Interrupt Set Enable
Register */
       uint32_t RESERVED0[24U];
  __IOM uint32_t ICER[8U];               /*!< Offset: 0x080 (R/W)  Interrupt Clear Enable
Register */
       uint32_t RSERVED1[24U];
  __IOM uint32_t ISPR[8U];               /*!< Offset: 0x100 (R/W)  Interrupt Set Pending
Register */
       uint32_t RESERVED2[24U];
  __IOM uint32_t ICPR[8U];               /*!< Offset: 0x180 (R/W)  Interrupt Clear Pending
Register */
       uint32_t RESERVED3[24U];
  __IOM uint32_t IABR[8U];               /*!< Offset: 0x200 (R/W)  Interrupt Active bit Register
*/
       uint32_t RESERVED4[56U];
  __IOM uint8_t  IP[240U];              /*!< Offset: 0x300 (R/W)  Interrupt Priority Register
(8Bit wide) */
       uint32_t RESERVED5[644U];
  __OM  uint32_t STIR;                  /*!< Offset: 0xE00 ( /W)  Software Trigger Interrupt
Register */
}  NVIC_Type
```

Peripherals Address

Location: stm32l475xx.h

```
/** @addtogroup Peripheral_memory_map
  * @{
  */
#define FLASH_BASE            (0x08000000UL) /*!< FLASH(up to 1 MB) base address */
#define SRAM1_BASE            (0x20000000UL) /*!< SRAM1(up to 96 KB) base address */
#define SRAM2_BASE            (0x10000000UL) /*!< SRAM2(32 KB) base address */
#define PERIPH_BASE           (0x40000000UL) /*!< Peripheral base address */
#define FMC_BASE              (0x60000000UL) /*!< FMC base address */
#define QSPI_BASE             (0x90000000UL) /*!< QUADSPI memories accessible over AHB base address */

#define FMC_R_BASE            (0xA0000000UL) /*!< FMC  control registers base address */
#define QSPI_R_BASE           (0xA0001000UL) /*!< QUADSPI control registers base address */
#define SRAM1_BB_BASE         (0x22000000UL) /*!< SRAM1(96 KB) base address in the bit-band region */
#define PERIPH_BB_BASE        (0x42000000UL) /*!< Peripheral base address in the bit-band region */

/* Legacy defines */
#define SRAM_BASE             SRAM1_BASE
#define SRAM_BB_BASE          SRAM1_BB_BASE
```

Peripherals Address

Location: stm32l475xx.h

```
/* Memory mapping of Core Hardware */
#define SCS_BASE            (0xE000E000UL)                    /*!< System Control Space Base Address */
#define ITM_BASE            (0xE0000000UL)                    /*!< ITM Base Address */
#define DWT_BASE            (0xE0001000UL)                    /*!< DWT Base Address */
#define TPI_BASE            (0xE0040000UL)                    /*!< TPI Base Address */
#define CoreDebug_BASE      (0xE000EDF0UL)                    /*!< Core Debug Base Address */
#define SysTick_BASE        (SCS_BASE +  0x0010UL)            /*!< SysTick Base Address */
#define NVIC_BASE           (SCS_BASE +  0x0100UL)            /*!< NVIC Base Address */
#define SCB_BASE            (SCS_BASE +  0x0D00UL)            /*!< System Control Block Base Address */

#define SCnSCB              ((SCnSCB_Type    *)    SCS_BASE     )   /*!< System control Register not in SCB */
#define SCB                 ((SCB_Type       *)    SCB_BASE     )   /*!< SCB configuration struct */
#define SysTick             ((SysTick_Type   *)    SysTick_BASE )   /*!< SysTick configuration struct */
#define NVIC                ((NVIC_Type      *)    NVIC_BASE    )   /*!< NVIC configuration struct */
#define ITM                 ((ITM_Type       *)    ITM_BASE     )   /*!< ITM configuration struct */
#define DWT                 ((DWT_Type       *)    DWT_BASE     )   /*!< DWT configuration struct */
#define TPI                 ((TPI_Type       *)    TPI_BASE     )   /*!< TPI configuration struct */
#define CoreDebug           ((CoreDebug_Type *)    CoreDebug_BASE)   /*!< Core Debug configuration struct */
```

IRQ Number

Location: stm32l475xx.h

```
typedef enum
{

/******  STM32 specific Interrupt Numbers
*****************************************************************/
  WWDG_IRQn                 = 0,      /*!< Window WatchDog Interrupt
*/
  PVD_PVM_IRQn              = 1,      /*!< PVD/PVM1/PVM2/PVM3/PVM4 through EXTI
Line detection Interrupts    */
  TAMP_STAMP_IRQn           = 2,      /*!< Tamper and TimeStamp interrupts through the
EXTI line            */

  SPI1_IRQn                 = 35,     /*!< SPI1 global Interrupt                    */
  SPI2_IRQn                 = 36,     /*!< SPI2 global Interrupt                    */
  USART1_IRQn               = 37,     /*!< USART1 global Interrupt
*/
  USART2_IRQn               = 38,     /*!< USART2 global Interrupt
*/
  USART3_IRQn               = 39,     /*!< USART3 global Interrupt
*/
  EXTI15_10_IRQn            = 40,     /*!< External Line[15:10] Interrupts
*/

} IRQn_Type;
```

ISR

- In File – stm32l4xx_it.c

```
void SysTick_Handler(void)
{
        /* USER CODE BEGIN SysTick_IRQn 0 */

        /* USER CODE END SysTick_IRQn 0 */
        HAL_IncTick();
        /* USER CODE BEGIN SysTick_IRQn 1 */

        /* USER CODE END SysTick_IRQn 1 */
}
```

- in file stm32l4xx_hal.c

```
 /**
  * @brief This function is called to increment a global variable "uwTick"
  *        used as application time base.
  * @note In the default implementation, this variable is incremented each 1ms
  *       in SysTick ISR.
 * @note This function is declared as __weak to be overwritten in case of other
  *      implementations in user file.
  * @retval None
  */
        _weak void HAL_IncTick(void)
        {
                uwTick += uwTickFreq;
        }
```

Callback method

- in file stm32l4xx_hal_uart.c

```
  /**
  * @brief Handle UART interrupt request.
  * @param huart UART handle.
  * @retval None
  */
void HAL_UART_IRQHandler(UART_HandleTypeDef *huart)
{

   UART_EndTransmit_IT(huart);

}
static void UART_EndTransmit_IT(UART_HandleTypeDef *huart)
{

        HAL_UART_TxCpltCallback()
}

_weak void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(huart);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_UART_TxCpltCallback can be implemented in the user file.
   */
}
```

Calling method

- in file stm32l4xx_hal_uart.c

```
/**
  * @brief Send an amount of data in interrupt mode.
  * @param huart UART handle.
  * @param pData Pointer to data buffer.
  * @param Size  Amount of data to be sent.
  * @retval HAL status
  */
HAL_StatusTypeDef
HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t
*pData, uint16_t Size)
{

}
```

WHAT WE NEED TO IMPLEMENT INTERRUPT

- We need an interrupt number
  - May be priority
- Need to enable the interrupt and set priority
- We need a vector table
  - In the startup file - startup_stm32l475xx.s
- Need Interrupt Service Routine (ISR)
- Need way to communicate back to main program
  - Will use Callback method
- Need method to call from main

Next Lesson Topic

- Timers
- SysTick (System Timer)
- Watchdog Timer