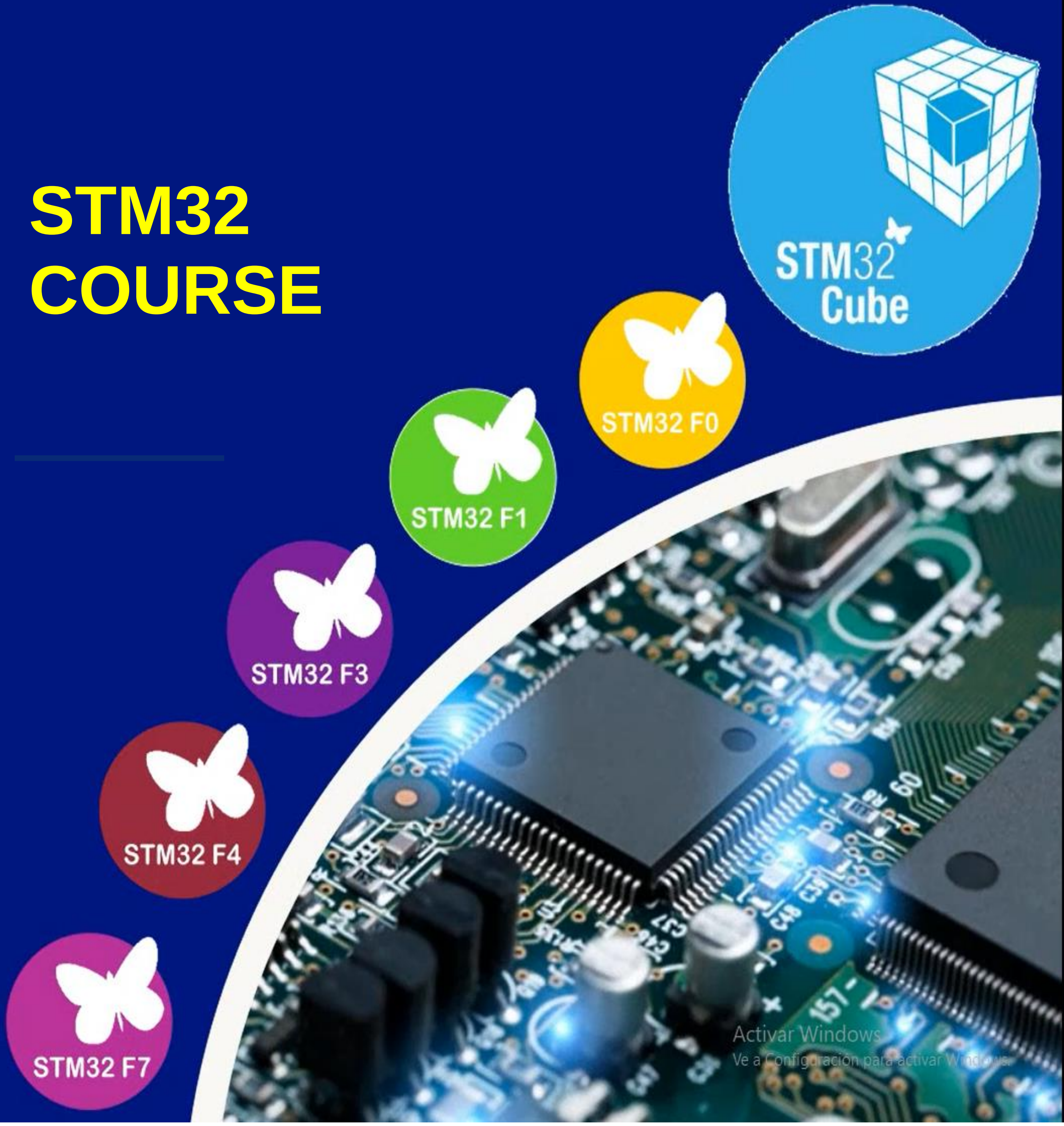
STM32 COURSE



FEBRUARY 23, 2021

004 Serial Communication Polling IT and DMA
Creado por: Ing. Christian Salazar

SECTION 4

004 Serial Communication Polling IT and DMA

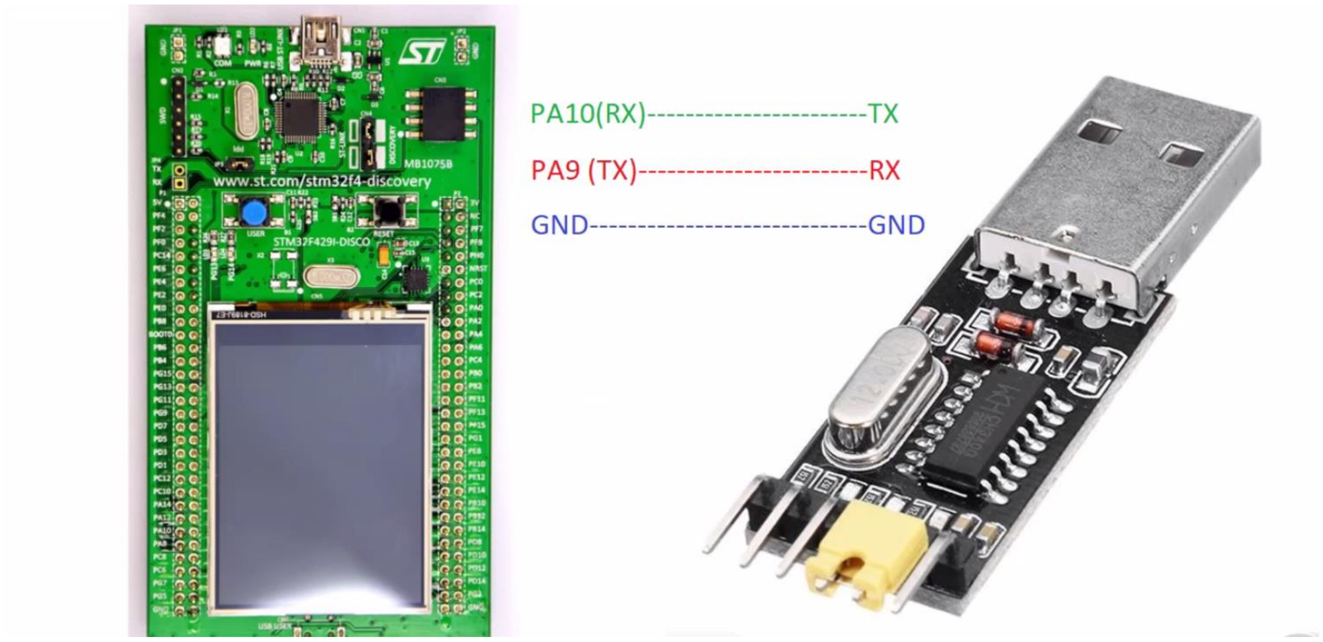
What will we learn?

In this video we are going to learn how to manipulate the USART peripheral (huart1) to transmit and receive data in polling mode, by interruption and by DMA from a serial terminal (RealTerm free software) with the help of a USB to serial TTL converter (ch340 converter), we will also configure the USART to be able to use the printf () function from the stdio.h library.

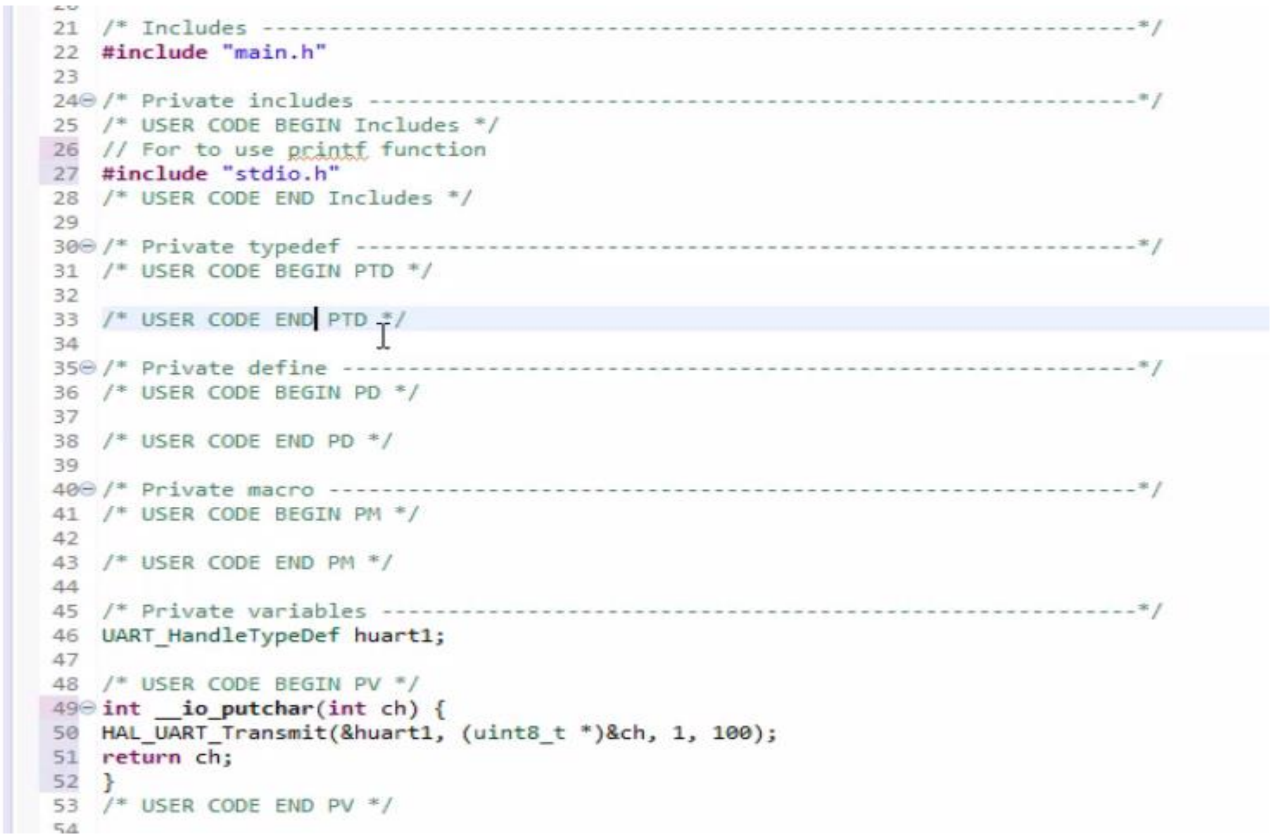
"We will use HAL Drivers, which will help us greatly to port and recycle code routines from one processor in one Family to another in another Family."

Key points

CONECTION


PA10(RX)----------------------TX
PA9 (TX)-----------------------RX
GND----------------------------GND

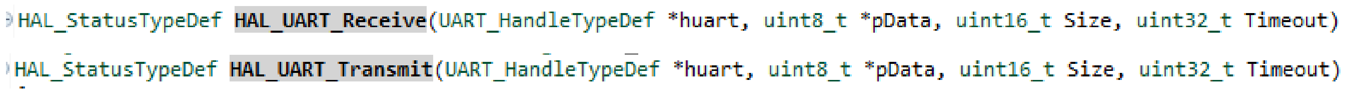
LIBRARY STDIO Y FUNCTION PRINTF



```
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
// For to use printf function
#include "stdio.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
int __io_putchar(int ch) {
HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 100);
return ch;
}
/* USER CODE END PV */
```

As we should know, the printf function of the stdio.h library is very useful for transmitting ASCII data from any part of our code, this function is usually widely used to debug code where ASCII frames are handled, for this we will include line 27 which will compile the available ANCI C functions for that library, we will also put the function of printing a data through USART 1 (line 50) which will be used when calling printf ().

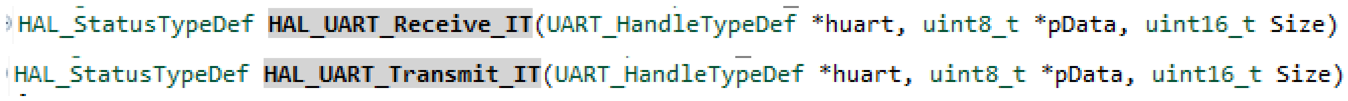
POLLING MODE



```
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
```

To transmit or receive data in Polling Mode we have two functions, which need the USART handler that will be used to communicate in the case of video is the huart1, also the pointer to the information to be sent or in turn the pointer of the variable where the data frame is going to be received, the size of the frame to be transmitted or received and finally a timeout in milliseconds, which is the maximum time to wait for the peripheral to finish its work when the IDLE indicates that it has started a process the USART, normally this way of transmitting is used in the infinite while, for the reception it is used for fixed frames.

INTERRUPT MODE



```
HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
```

In interrupt mode we have two basic functions, which are similar to the previous two only without the timeout since now we have the option of not consulting the state of the peripheral (IF it is necessary to receive or transmit frames) in the infinite while but that with these functions, when there is something to receive, the USART1 interrupt will be lifted in our case and it is then that, as in the EXTI interruptions previously studied, we have a Callback function that is executed immediately after attending to the peripheral interruption, this function is called for all USART peripherals that have the interrupt enabled, for which it receives as an argument the USART Instance that activated or called it and we can differentiate them basically in the same way with an if () loop, then these functions configure and order the USART peripheral so that once it is the case of receiving a data and this data is in the buffer register of the USART1 it will inform us that there is information Available action, it is then at this moment that we can download the information in our own buffer and then use that information received, if the information is not downloaded upon receipt, it will be overwritten by the next information frame that may arrive, also for the case of the transmission we can call the function at any moment in our program and an interrupt will be raised as soon as the last byte of the frame that it has been ordered to send has finished transmitting.

DMA MODE

```
HAL_StatusTypeDef HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
HAL_StatusTypeDef HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
```

The DMA (Direct Memory Access) peripheral, like the CPU, is a Master type peripheral, that is, it can read and write information from a peripheral (TIMER, USART, I2C, etc.) to memory, from memory to a peripheral, from memory to memory and from peripheral to peripheral, in the case of the USART we use DMA transmission to send information from memory to the peripheral and to receive it is done from peripheral to memory, the syntax is similar to the two previous functions, it only changes the name of the function, these functions also use the same Callback function as the previous two in addition to the fact that they can raise the interruption of the DMA peripheral, that is, when we finish sending or receiving data we can make decisions at the end of said processes, the use of the DMA relieves the main CPU of load or processing since this process of transporting the information whatever the mode will be done by the DMA peripheral.

RECOMMENDATION

It is recommended to use for the reception, the interruption of one Byte at a time and the transmission by DMA of the complete transmission frame, it is very practical for the ModBus protocol for example, you can use everything by DMA and the reception can only be byte in Byte.