Notas sobre STM32

Ejemplos

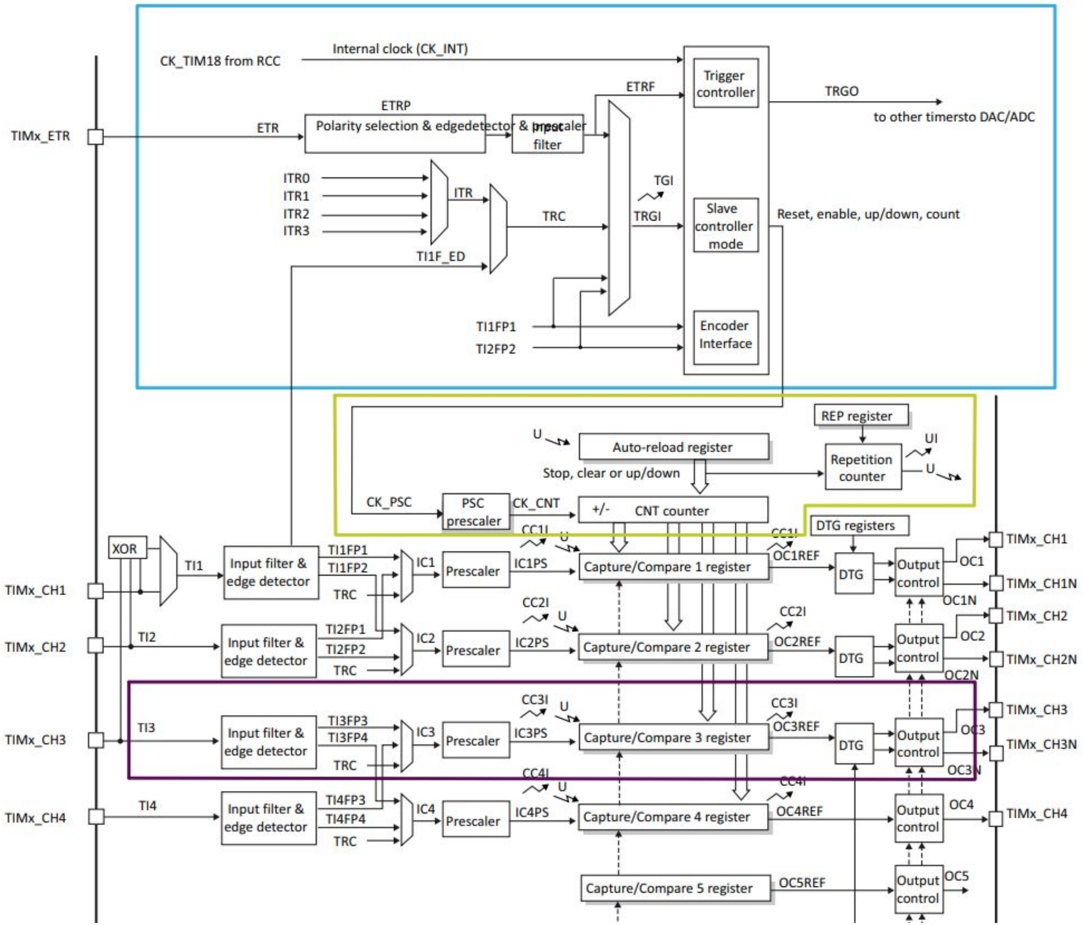
STM32 F1 HAL

https://www.st.com/resource/en/user_manual/dm00154093-description-of-stm32f1-hal-and-lowlayer-drivers-stmicroelectronics.pdf

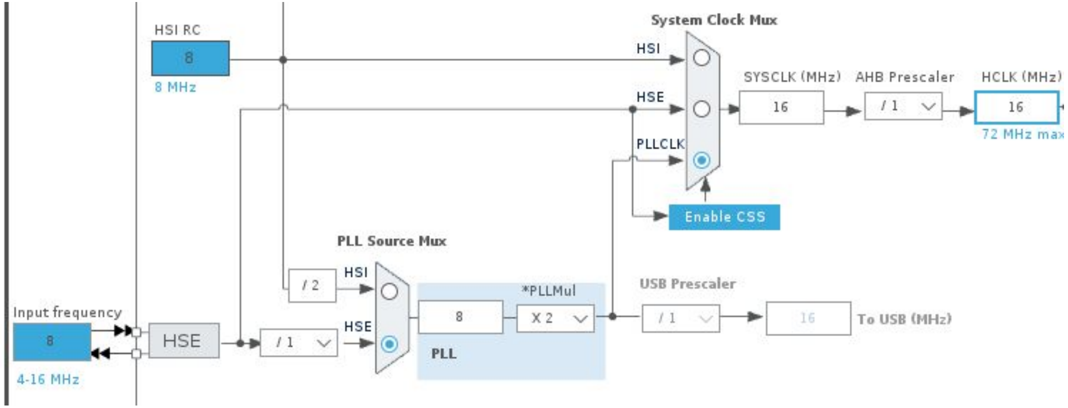
Contiene todas las referencias de las funciones de la HAL.

Timers

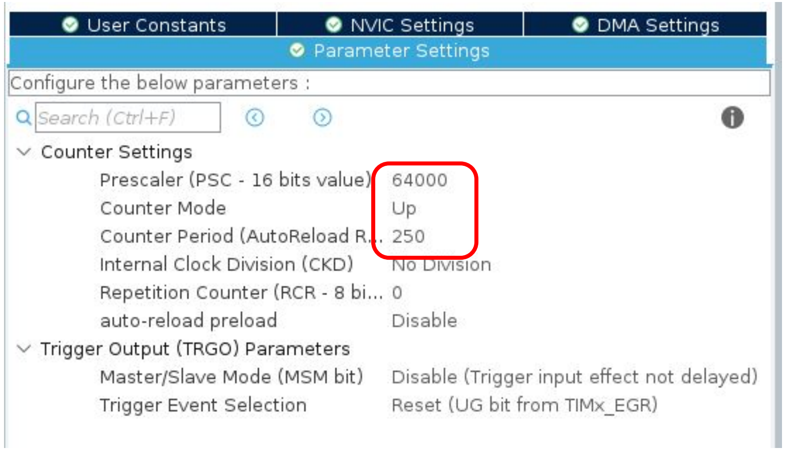
https://www.st.com/resource/en/application_note/dm00236305-generalpurpose-timer-cookbook-for-stm32-microcontrollers-stmicroelectronics.pdf

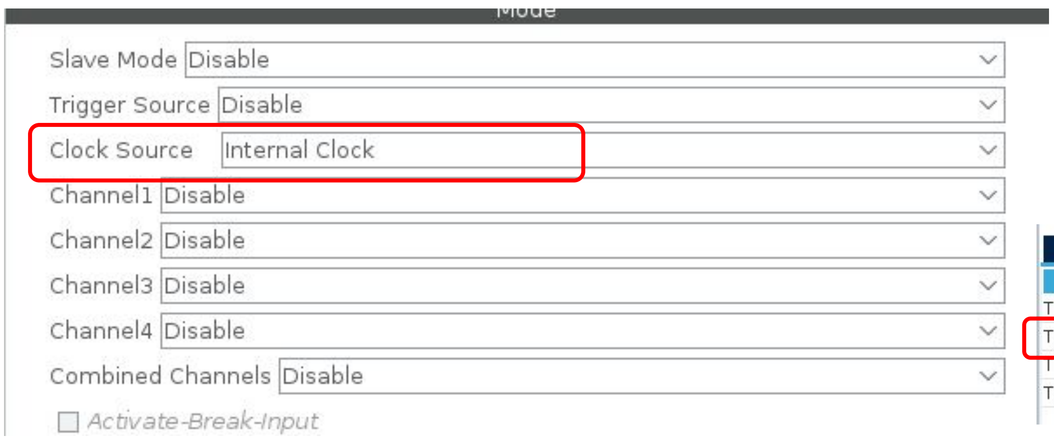


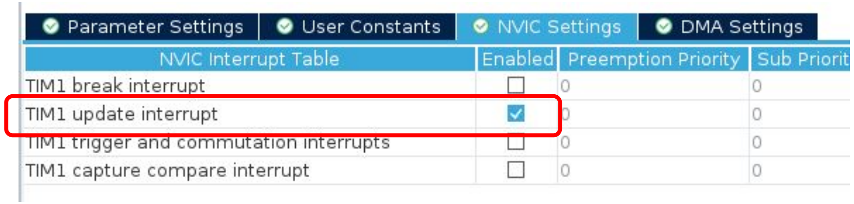
Interrupción de timer (mal ejemplo)



Oscilador	16000000	Hz
	16	MHz
	64000	Prescaler
	250	Freq Salida Div







Interrupción de timer (mal ejemplo)

```c
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 64000;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 250;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */
  HAL_TIM_Base_Start_IT(&htim1);
  /* USER CODE END TIM1_Init 2 */

}
```

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim){
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
}

/* USER CODE END 0 */
```

Es un mal ejemplo ya que limita el contador del TIM1 a 250 pulsos… y luego pasa a 0 de vuelta. El problema es que este timer queda muy limitado en el uso de los OutputCompare, InputCapture, PWM, Encoders, etc.

Si bien es una opción en el caso que nos sobren timers ya que su uso es muy simple, desperdicia los canales del timer.

Interrupción de timer (usando Output Compare)

Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	Output Compare No Output
Channel2	Disable
Channel3	Disable
Channel4	Disable
Combined Channels	Disable

☐ *Activate-Break-Input*

☐ Use ETR as Clearing Source

User Constants | NVIC Settings | DMA Settings | Parameter Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM1 break interrupt	☐	0	0
TIM1 update interrupt	☐	0	0
TIM1 trigger and commutation interrupts	☐	0	0
TIM1 capture compare interrupt	☑	0	0

Oscilador	16000000	Hz
	16	MHz
	64000	Prescaler
	250	Freq Salida Div

User Constants | NVIC Settings | DMA Settings | Parameter Settings

Configure the below parameters :

Search (Ctrl+F)

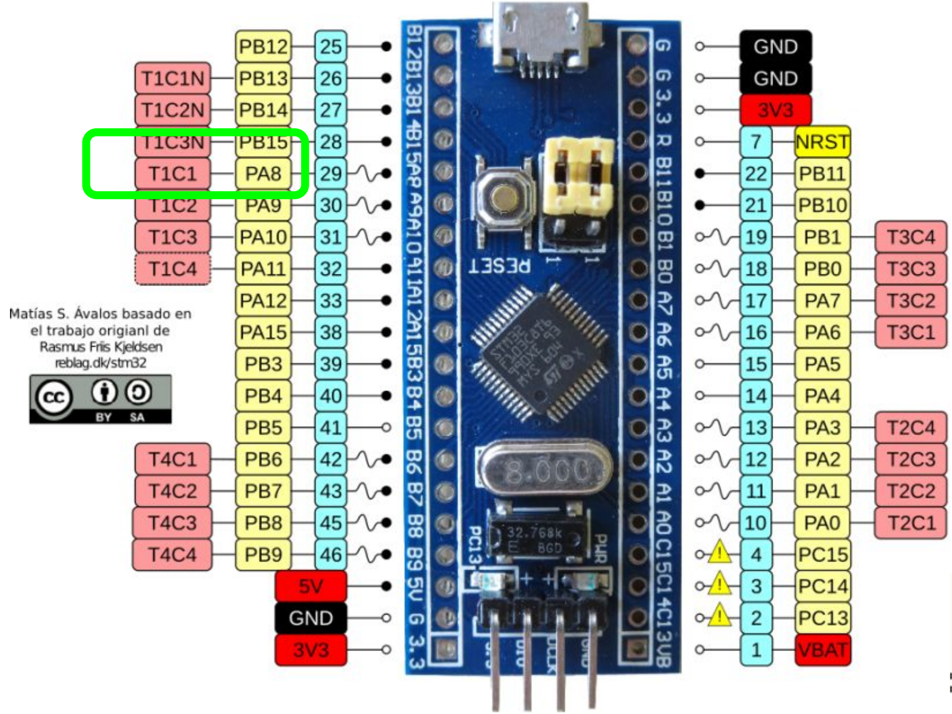
Counter Settings	
Prescaler (PSC - 16 bits value)	64000
Counter Mode	Up
Counter Period (AutoReload R...	65535
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bi...	0
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Break And Dead Time management ...	
BRK State	Disable
BRK Polarity	High
Break And Dead Time management ...	
Automatic Output State	Disable
Off State Selection for Run Mo...	Disable
Off State Selection for Idle Mo...	Disable
Lock Configuration	Off
Output Compare No Output Channel 1	
Mode	Frozen (used for Timing base)
Pulse (16 bits value)	250
Output compare preload	Disable
CH Polarity	High
CH Idle State	Reset

Interrupción de timer (usando Output Compare)

```c
/* USER CODE BEGIN TIM1_Init 2 */
HAL_TIM_OC_Start_IT(&htim1, TIM_CHANNEL_1);
/* USER CODE END TIM1_Init 2 */
```

```c
/* Private user code ---------------------------------------------------------
/* USER CODE BEGIN 0 */
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim) {
    uint32_t pulse;
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        pulse = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);

        __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, (pulse + 250));

    }
}
/* USER CODE END 0 */
```

Interrupción de timer (usando Output Compare y T1C1)



Lock Configuration	Off
Output Compare Channel 1	
Mode	Toggle on match
Pulse (16 bits value)	250
Output compare preload	Disable

```
/* USER CODE BEGIN 0 */
void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef *htim) {
    uint32_t pulse;
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
        //HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        pulse = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);

        __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, (pulse + 250));

    }
}
```

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	Output Compare CH1

UART

Starware

EN STOCK

MÁS VENDIDO 10° en Programadores

$ 1.849

en 6x $ 308[17] sin interés

Ver los medios de pago

Llega mañana por $ 292[49] ~~$ 449[99]~~

Beneficio Mercado Puntos

Ver más formas de entrega

Devolución gratis

Tenés 30 días desde que lo recibís.

Conocer más

Stock disponible

Cantidad: **1 unidad** (25 disponibles)

Nuevo | 115 vendidos

Progrmador Ftdi Usb Ttl Ft232 Ft232rl Arduino Esp32cam

MÁS VENDIDO 9° en Arduino Arduino

$ 790

en 6x $ 131[67] sin interés

Ver los medios de pago

Llega **el martes** por $ 789[99]

Ver más formas de entrega

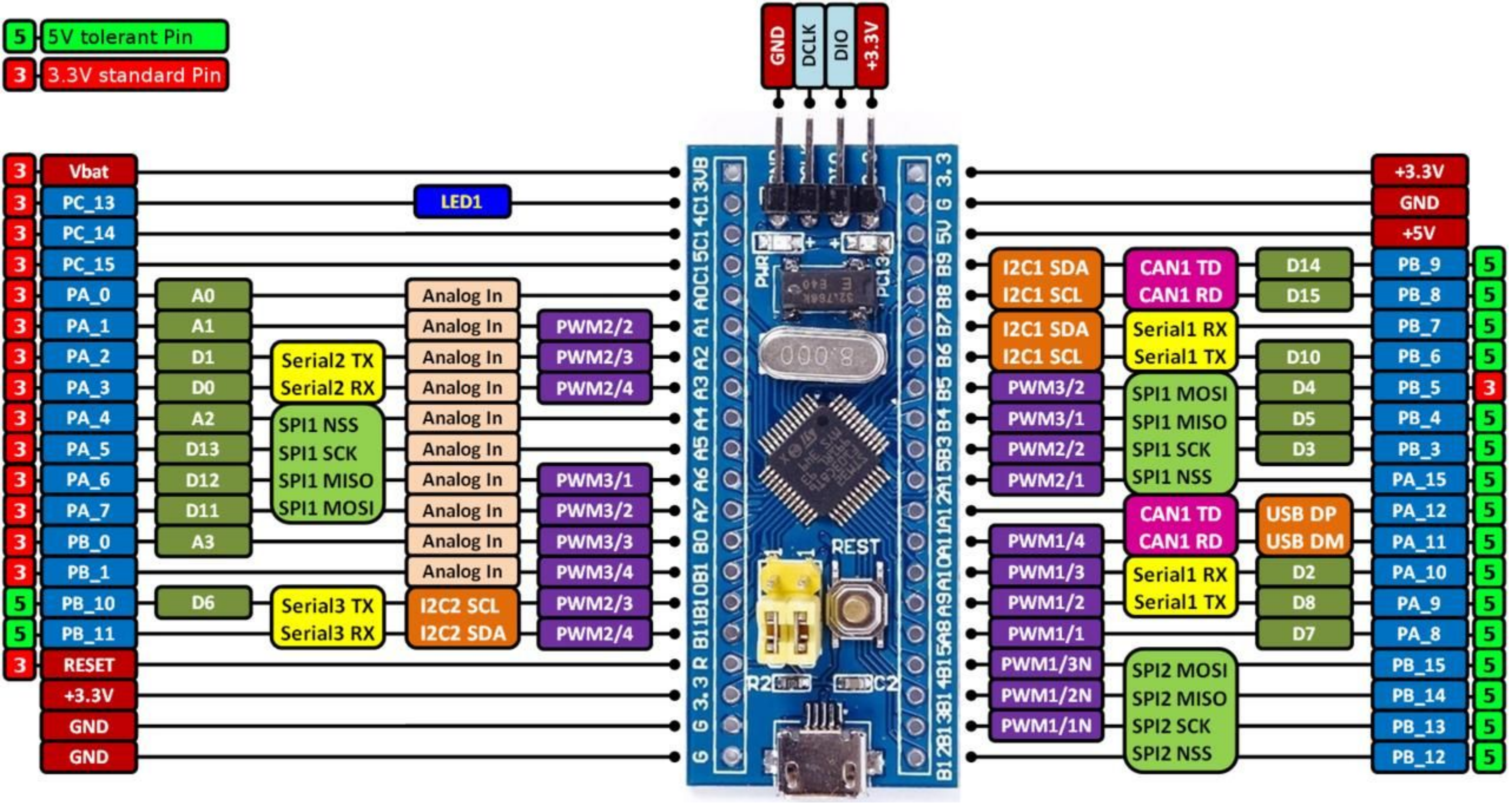
Devolución gratis

Tenés 30 días desde que lo recibís.

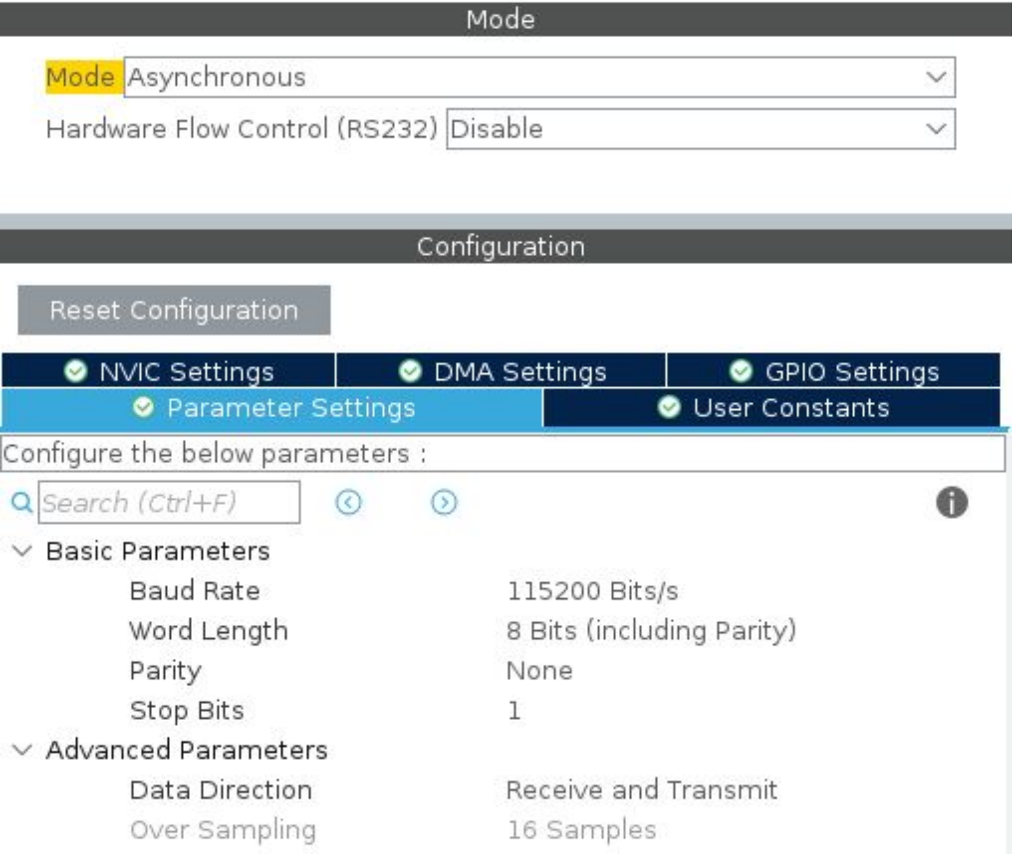
Conocer más

Stock disponible

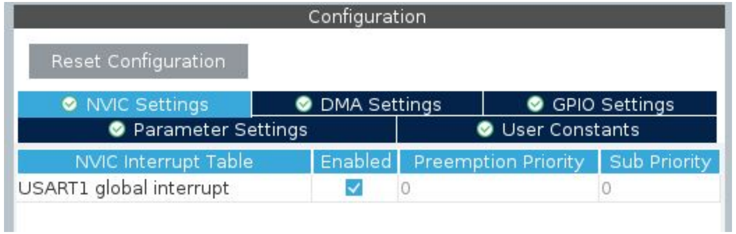
UART


5 5V tolerant Pin
3 3.3V standard Pin
GND
DCLK
DIO
+3.3V
3 Vbat
3 PC_13 LED1
3 PC_14
3 PC_15
3 PA_0 A0 Analog In
3 PA_1 A1 Analog In PWM2/2
3 PA_2 D1 Serial2 TX Analog In PWM2/3
3 PA_3 D0 Serial2 RX Analog In PWM2/4
3 PA_4 A2 SPI1 NSS Analog In
3 PA_5 D13 SPI1 SCK Analog In
3 PA_6 D12 SPI1 MISO Analog In PWM3/1
3 PA_7 D11 SPI1 MOSI Analog In PWM3/2
3 PB_0 A3 Analog In PWM3/3
3 PB_1 Analog In PWM3/4
5 PB_10 D6 Serial3 TX I2C2 SCL PWM2/3
5 PB_11 Serial3 RX I2C2 SDA PWM2/4
3 RESET
+3.3V
GND
GND
+3.3V
GND
+5V
I2C1 SDA CAN1 TD D14 PB_9 5
I2C1 SCL CAN1 RD D15 PB_8 5
I2C1 SDA Serial1 RX PB_7 5
I2C1 SCL Serial1 TX D10 PB_6 5
PWM3/2 SPI1 MOSI D4 PB_5 3
PWM3/1 SPI1 MISO D5 PB_4 5
PWM2/2 SPI1 SCK D3 PB_3 5
PWM2/1 SPI1 NSS PA_15 5
CAN1 TD USB DP PA_12 5
PWM1/4 CAN1 RD USB DM PA_11 5
PWM1/3 Serial1 RX D2 PA_10 5
PWM1/2 Serial1 TX D8 PA_9 5
PWM1/1 D7 PA_8 5
PWM1/3N SPI2 MOSI PB_15 5
PWM1/2N SPI2 MISO PB_14 5
PWM1/1N SPI2 SCK PB_13 5
SPI2 NSS PB_12 5

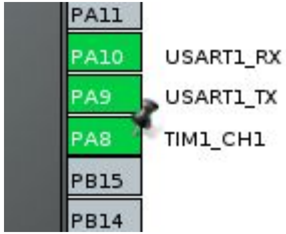
UART



Mode
Mode Asynchronous
Hardware Flow Control (RS232) Disable
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate 115200 Bits/s
Word Length 8 Bits (including Parity)
Parity None
Stop Bits 1
Advanced Parameters
Data Direction Receive and Transmit
Over Sampling 16 Samples



Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
NVIC Interrupt Table
Enabled
Preemption Priority
Sub Priority
USART1 global interrupt
0
0



PA11
PA10
USART1_RX
PA9
USART1_TX
PA8
TIM1_CH1
PB15
PB14

UART (TX modo bloqueante)

```c
void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef *htim) {
    uint32_t pulse;
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
        flag=1;
        pulse = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
        __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, (pulse + 250));
    }
}
```

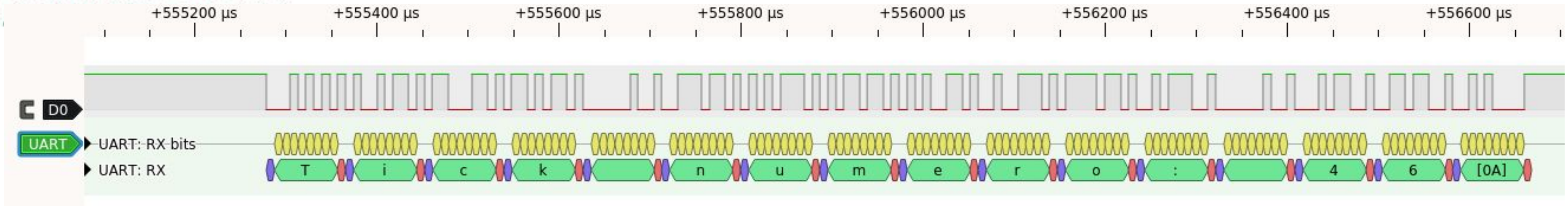
```c
/* USER CODE BEGIN WHILE */
  while (1) {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
      if (flag){
          flag=0;
          HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
          sprintf(buffer,"Tick numero: %d\n",tick++);
          HAL_UART_Transmit(&huart1, buffer,strlen(buffer),500);
      }
  }
```

```c
/* USER CODE BEGIN PV */
uint8_t flag=0;

/* USER CODE BEGIN 1 */
  uint8_t buffer[20];
  uint8_t tick=0;

/* USER CODE END 1 */
```

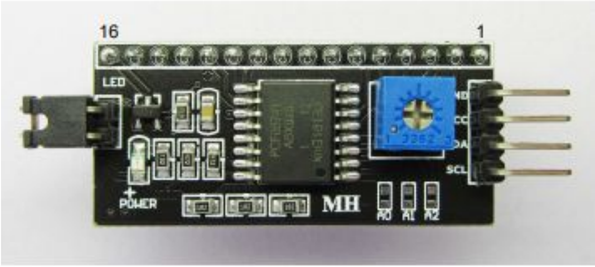
```
[19:51:11:177] Tick numero: 123
[19:51:12:184] Tick numero: 124
[19:51:13:175] Tick numero: 125
[19:51:14:182] Tick numero: 126
[19:51:15:174] Tick numero: 127
[19:51:16:181] Tick numero: 128
[19:51:17:172] Tick numero: 129
[19:51:18:179] Tick numero: 130
[19:51:19:186] Tick numero: 131
[19:51:20:177] Tick numero: 132
[19:51:21:184] Tick numero: 133
```

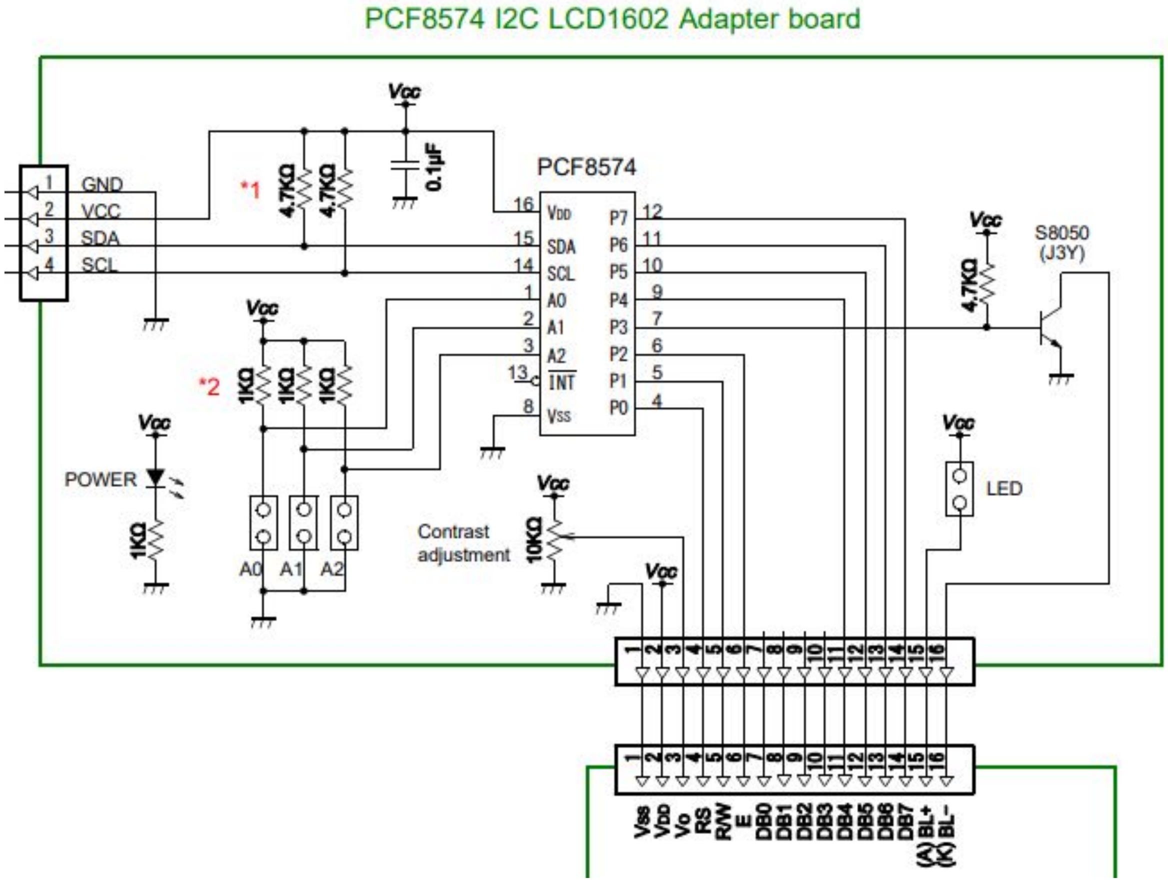
Clear   Hex output   Logging to: /home/edgardog/cutecom.log

Device: FTDI FT232R USB UART @ttyUSB0 Connection: 115200 @ 8-N-1



I^2C LCD





https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

I²C LCD

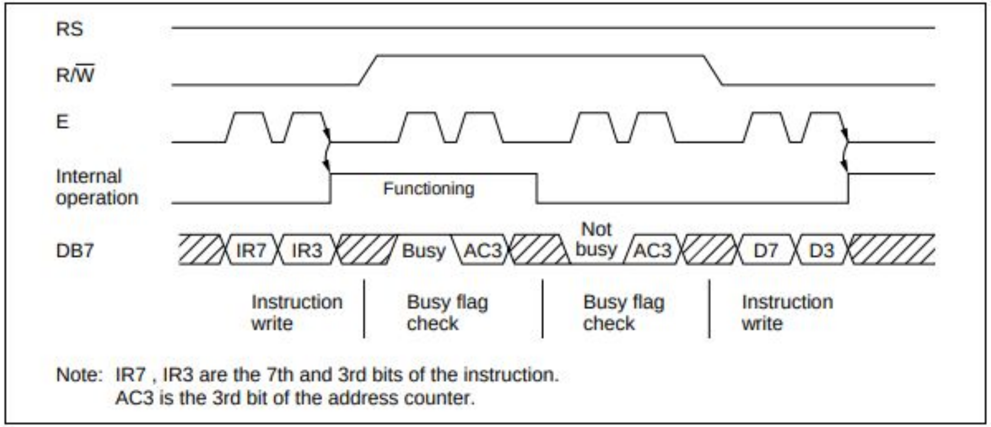


Figure 17 Example of 4-Bit Data Transfer Timing Sequence

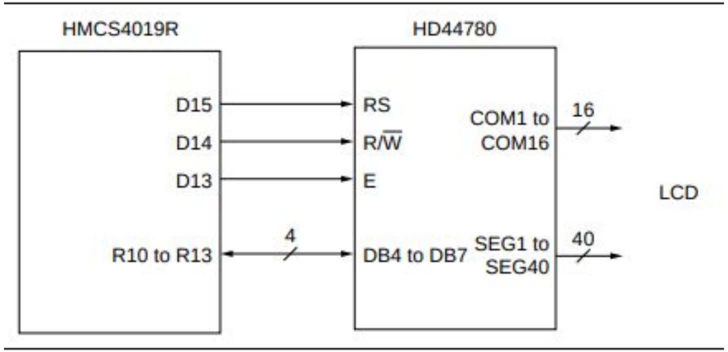


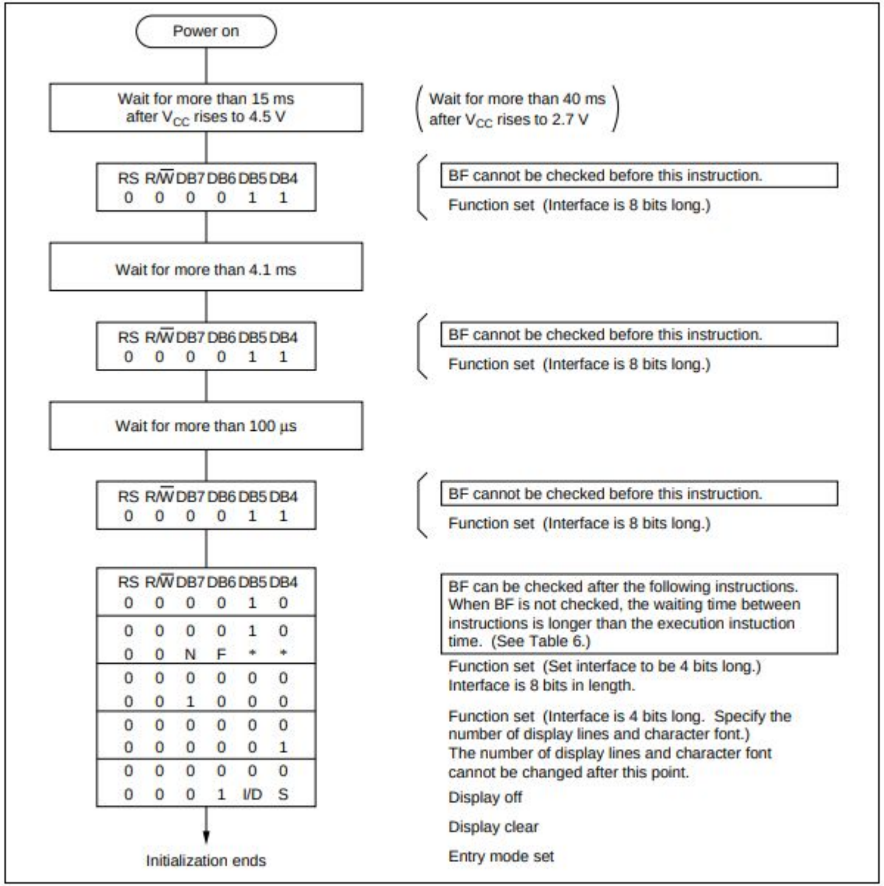
Figure 18 Example of Interface to HMCS4019R

P0	P1	P2	P3	P4	P5	P6	P7
RS	RW	E	BL	DB4	DB5	DB6	DB7

Pin Functions

Signal	No. of Lines	I/O	Device Interfaced with	Function
RS	1	I	MPU	Selects registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
R/W	1	I	MPU	Selects read or write. 0: Write 1: Read
E	1	I	MPU	Starts data read/write.

I^{2}C LCD



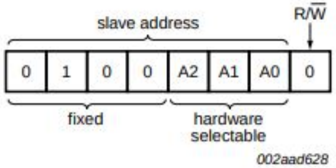

```
{
0x03,
0x03,
0x03 ,
0x02,
0x02 ,
0x08 ,
0x00 ,
0x08 ,
0x00,
0x01 ,
0x00,
0x06,
//Extra
0x00,
0x0C };
```

1. Display clear
2. Function set:
   DL = 1; 8-bit interface data
   N = 0; 1-line display
   F = 0; 5 × 8 dot character font
3. Display on/off control:
   D = 0; Display off
   C = 0; Cursor off
   B = 0; Blinking off
4. Entry mode set:
   I/D = 1; Increment by 1
   S = 0; No shift

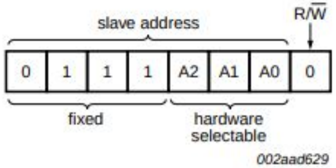
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Display on/off control	Code	0	0	0	0	0	0	1	D	C	B

I²C LCD

```c
typedef struct I2CLCDDisplay {
    uint8_t Address;
    uint8_t Backlight;
    I2C_HandleTypeDef Bus;
}I2CLCDDisplay;
```



a. PCF8574



b. PCF8574A

```c
void sendNibbleCmd(I2CLCDDisplay display, uint8_t lower_nibble){
    uint8_t shifted[2];
    shifted[0] = (lower_nibble <<4);
    shifted[1] = (lower_nibble <<4);
    if (display.Backlight){
        shifted[0] |= 0x08;
        shifted[1] |= 0x08;
    }
    shifted[0] |= 0x04;
    shifted[1] &= 0xFC;
    HAL_I2C_Master_Transmit(&display.Bus,display.Address,shifted,2,10);
}
```

	P0	P1	P2	P3	P4	P5	P6	P7
	RS	RW	E	BL	DB4	DB5	DB6	DB7
	P7	P6	P5	P4	P3	P2	P1	P0
	DB7	DB6	DB5	DB4	BL	E	RW	RS
0X	0	0	0	0	a	b	c	d
X0	a	b	c	d	0	0	0	0
| 8	a	b	c	d	BL	0	0	0
0 | 4	a	b	c	d	BL	1	0	0
1&fc	a	b	c	d	BL	0	0	0

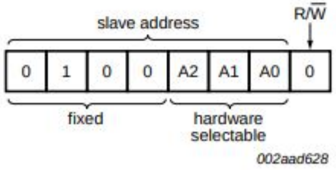
Si recibo: 0000abcd , Shifted[0] = abcd BL 1 00
Shifted[1] = abcd BL 0 00

Genera un pulso en E

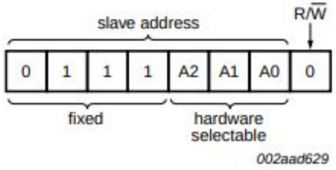
```c
void I2CLCD_Init(I2CLCDDisplay display ){

        for(int i=0;i<sizeof(initArray);i++){
            sendNibbleCmd(display,initArray[i]);
            HAL_Delay(5);
        }
}
```

I^{2}C LCD

```c
typedef struct I2CLCDDisplay {
    uint8_t Address;
    uint8_t Backlight;
    I2C_HandleTypeDef Bus;
}I2CLCDDisplay;
```



a. PCF8574



b. PCF8574A

```c
void sendNibbleData(I2CLCDDisplay display, uint8_t lower_nibble){
    uint8_t shifted[2];
    shifted[0] = (lower_nibble <<4);
    shifted[1] = (lower_nibble <<4);
    if (display.Backlight){
        shifted[0] |= 0x08;
        shifted[1] |= 0x08;
    }
    shifted[0] |= 0x05;
    shifted[1] &= 0xFC;
    HAL_I2C_Master_Transmit(&display.Bus,display.Address,shifted,2,10);
}
```

Si recibo: 0000abcd , Shifted[0] = abcd BL 1 01
Shifted[1] = abcd BL 0 00

Genera un pulso en E

```c
void sendDataByte(I2CLCDDisplay display, uint8_t byte ){
    sendNibbleData(display,(byte>>4));
    sendNibbleData(display,(byte));
}
```

	P0	P1	P2	P3	P4	P5	P6	P7
	RS	RW	E	BL	DB4	DB5	DB6	DB7
	P7	P6	P5	P4	P3	P2	P1	P0
	DB7	DB6	DB5	DB4	BL	E	RW	RS
0X	0	0	0	0	a	b	c	d
X0	a	b	c	d	0	0	0	0
| 8	a	b	c	d	BL	0	0	0
0 | 5	a	b	c	d	BL	1	0	1
1&fc	a	b	c	d	BL	0	0	0

I^{2}C LCD

```c
void I2CLCD_WriteLine(I2CLCDDisplay display, uint8_t lineNumber,char *data){
    uint8_t offset=0;
    switch (lineNumber){
    case 0:
        offset = 0;
        break;
    case 1:
        offset = 40;
        break;
    case 2:
        offset = 20;
        break;
    case 3:
        offset = 84;
        break;
    }
    offset |=0x80;
    sendNibbleCmd(display,(offset>>4));
    sendNibbleCmd(display,offset);
    uint8_t *pointer = (uint8_t*)data;
    while (*pointer != '\0'){
        sendDataByte(display,*pointer);
        pointer++;
    }
}
```