

Notas sobre STM32

Ejemplos

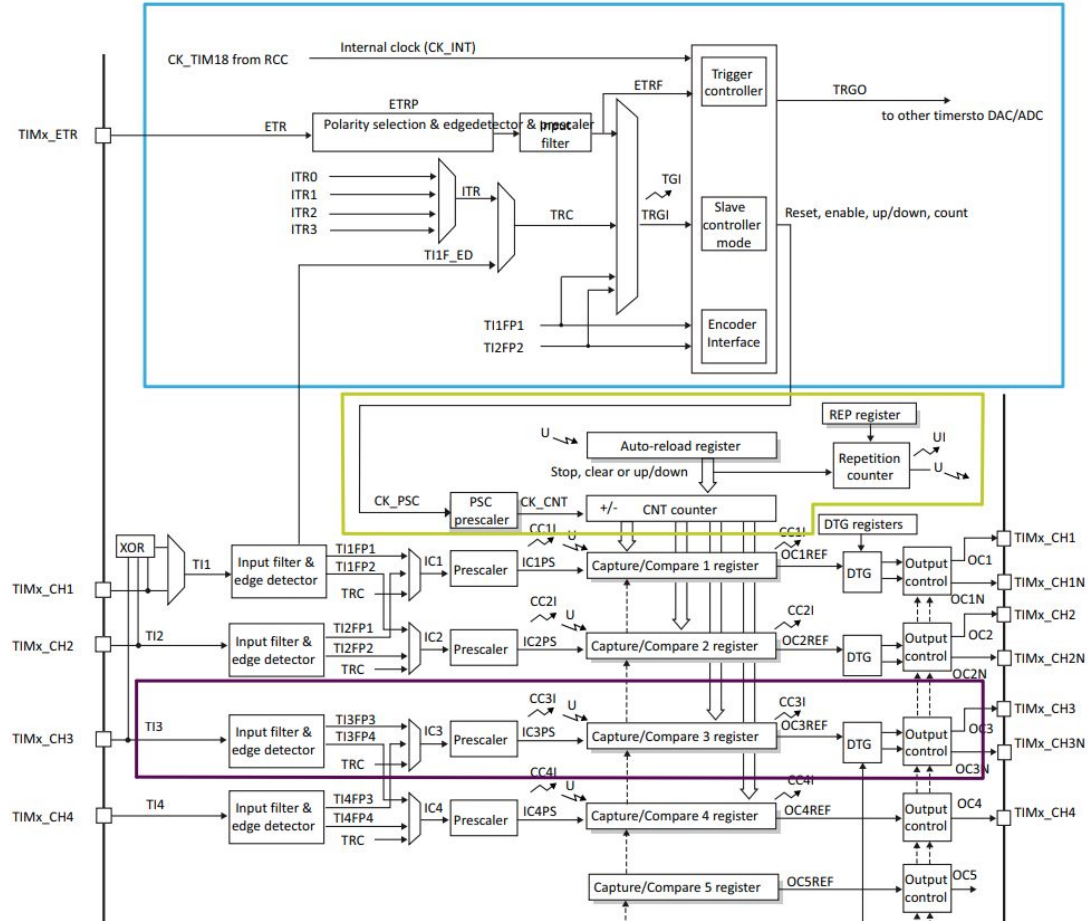
STM32 F1 HAL

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

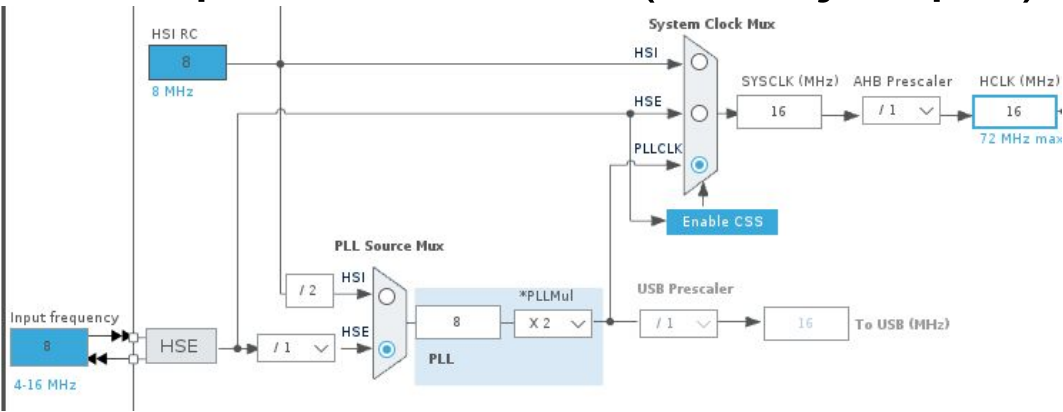
Contiene todas las referencias de las funciones de la HAL.

Timers

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



Interrupción de timer (mal ejemplo)



Mode

Slave Mode

Trigger Source

Clock Source

Channel1

Channel2

Channel3

Channel4

Combined Channels

☐ Activate-Break-Input

| | | |
|-----------|----------|-----------------|
| 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)

Counter Mode

Counter Period (AutoReload R...

Internal Clock Division (CKD)

Repetition Counter (RCR - 8 bi...

auto-reload preload

▼ Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)

Trigger Event Selection

| Parameter Settings | User Constants | NVIC Settings | DMA Settings |
|---|-------------------------------------|---------------|--------------|
| NVIC Interrupt Table | | | |
| Enabled | Preemption | Priority | Sub Priority |
| TIM1 break interrupt | <input type="checkbox"/> | 0 | 0 |
| TIM1 update interrupt | <input checked="" type="checkbox"/> | 0 | 0 |
| TIM1 trigger and commutation interrupts | <input type="checkbox"/> | 0 | 0 |
| TIM1 capture compare interrupt | <input type="checkbox"/> | 0 | 0 |

Interrupción de timer (mal ejemplo)

```
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 */
}
```

```
/* 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 | <input type="checkbox"/> | 0 | 0 |
| TIM1 update interrupt | <input type="checkbox"/> | 0 | 0 |
| TIM1 trigger and commutation interrupts | <input type="checkbox"/> | 0 | 0 |
| TIM1 capture compare interrupt | <input checked="" type="checkbox"/> | 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

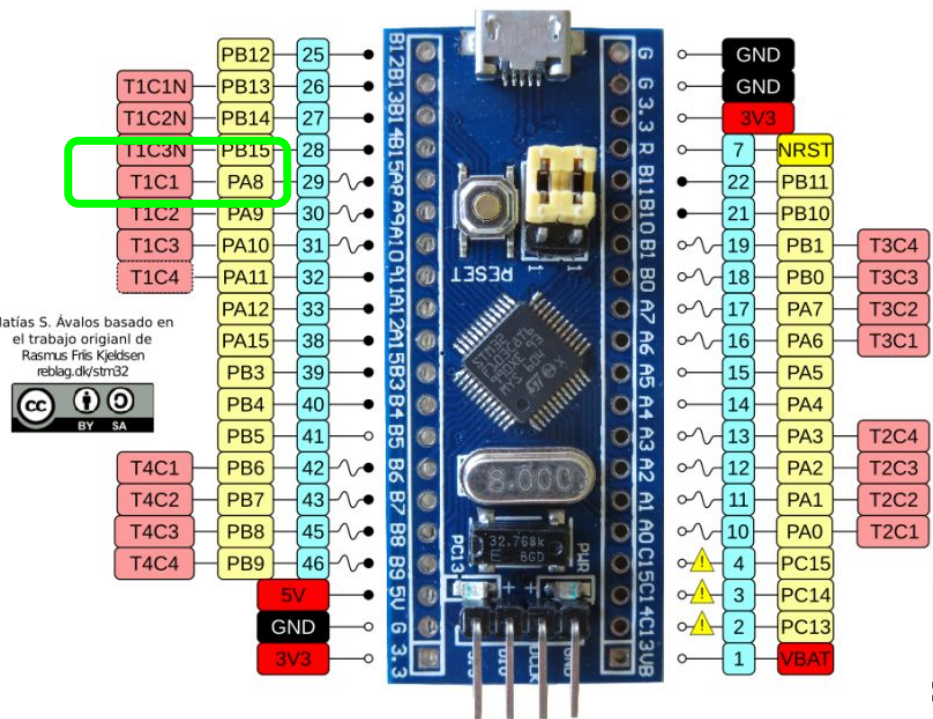
Interrupción de timer (usando Output Compare)

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

/* 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)

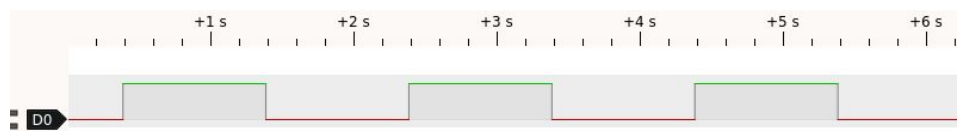


Matías S. Avalos basado en el trabajo original de Rasmus Fris Kjeldsen reblog.dv/stm32

| | |
|--------------------------|-----------------|
| 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

Trigger Source

Clock Source

Channel1

UART



EN STOCK



MÁS VENDIDO 10° en Programadores

\$ 1.849

en 6x \$ 308¹⁷ sin interés

[Ver los medios de pago](#)

Llega mañana por \$ 292⁴⁹ \$ 449⁹⁹

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 \$ 131⁶⁷ sin interés

[Ver los medios de pago](#)

Llega el martes por \$ 789⁹⁹

[Ver más formas de entrega](#)

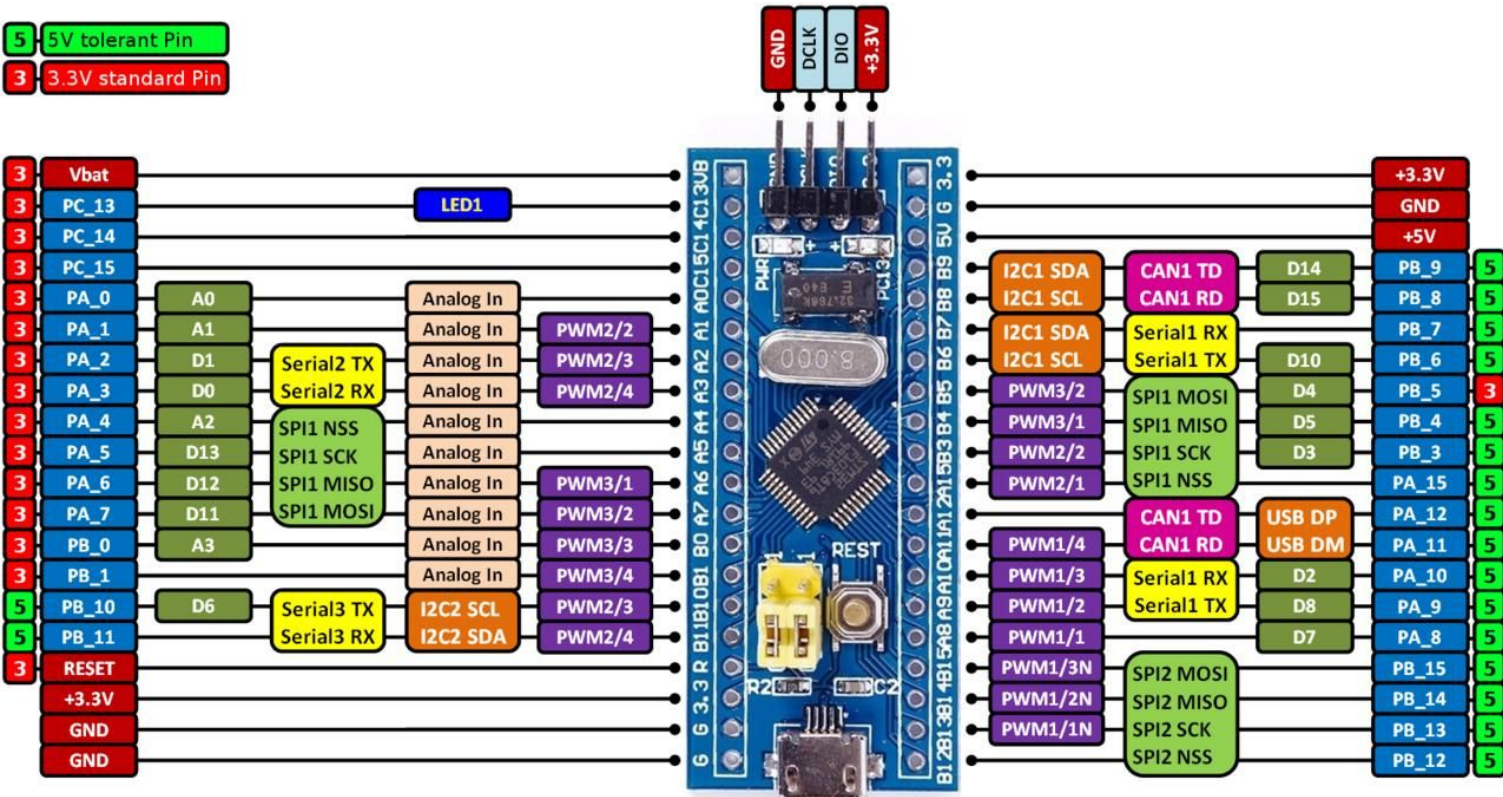
Devolución gratis

Tenés 30 días desde que lo recibís.

[Conocer más](#)

Stock disponible

UART



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)

⏪

⏩

i

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 | <input checked="" type="checkbox"/> | 0 | 0 |

PA11

PA10

PA9

PA8

PB15

PB14

USART1_RX

USART1_TX

TIM1_CH1

UART (TX modo bloqueante)

```
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));
    }
}

/* 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);
    }
}
```

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

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

```
/* USER CODE END 1 */
```

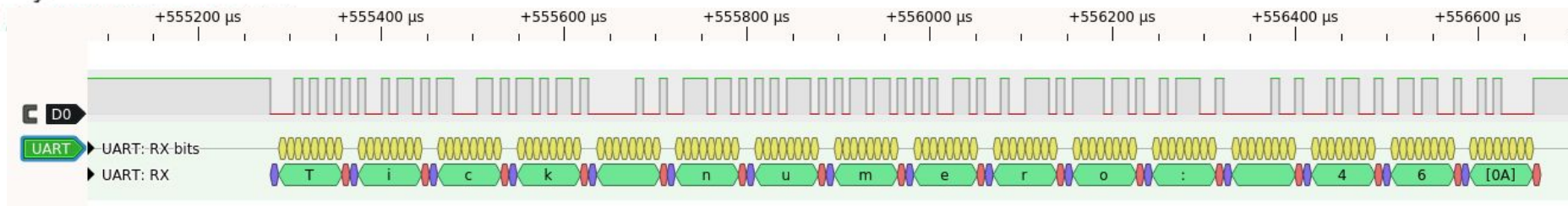
```
[19:51:11:177] Tick numero: 123 ↵
[19:51:12:184] Tick numero: 124 ↵
[19:51:13:175] Tick numero: 125 ↵
[19:51:14:182] Tick numero: 126 ↵
[19:51:15:174] Tick numero: 127 ↵
[19:51:16:181] Tick numero: 128 ↵
[19:51:17:172] Tick numero: 129 ↵
[19:51:18:179] Tick numero: 130 ↵
[19:51:19:186] Tick numero: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

Clear

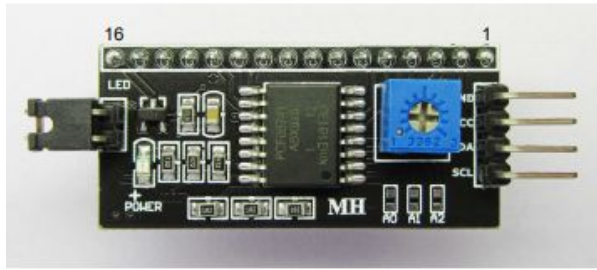
☐ Hex output

☐ Logging to: /home/edgardog/cutecom.log

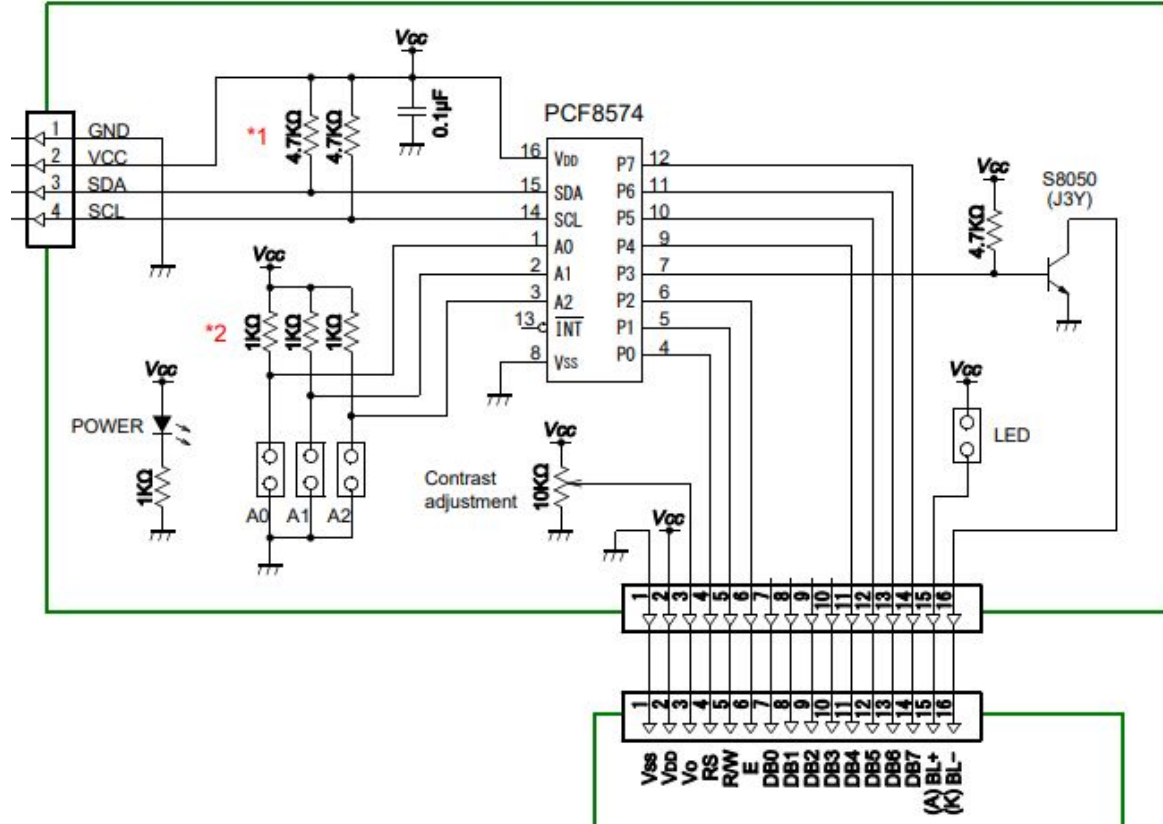
Device: FTDI FT232R USB UART @ttyUSB0 Connection: 115200 @ 8-N-1



I²C LCD



PCF8574 I2C LCD1602 Adapter board



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

I²C LCD

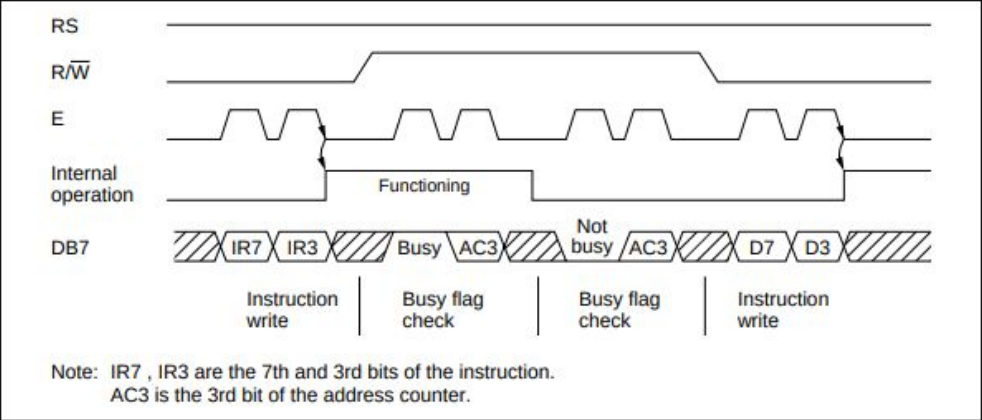


Figure 17 Example of 4-Bit Data Transfer Timing Sequence

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. |

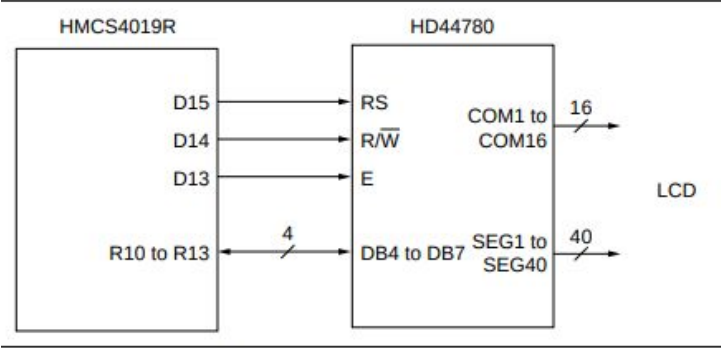
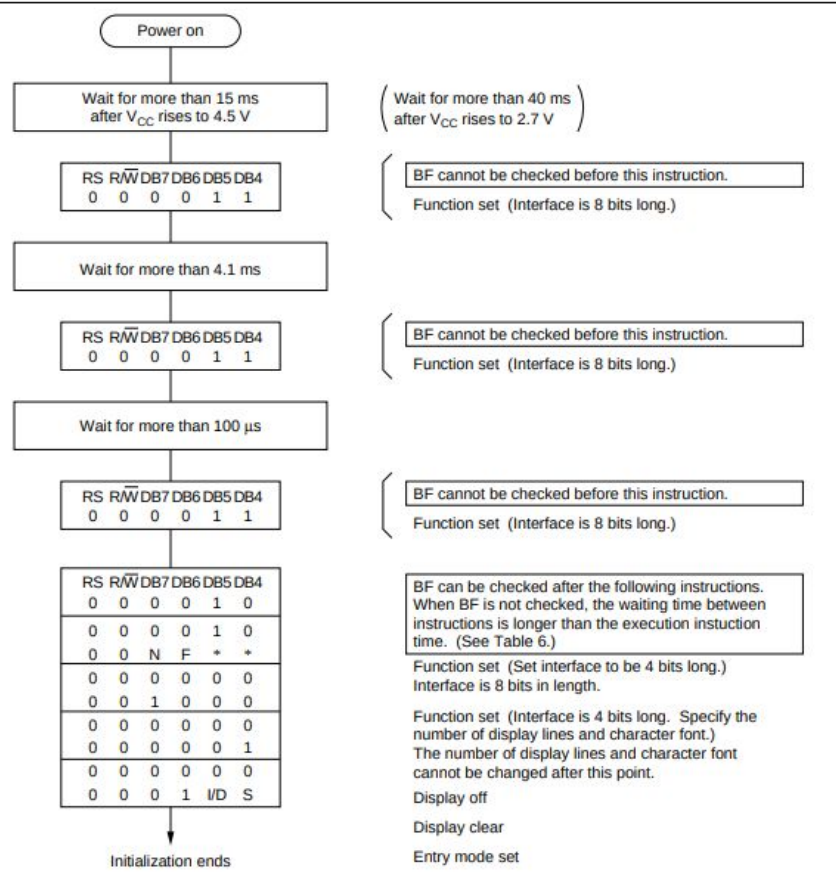


Figure 18 Example of Interface to HMCS4019R

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

I²C LCD



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

Display on/off control

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

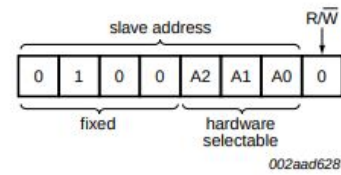
| RS | $\overline{R/W}$ | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
|----|------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B |

Code

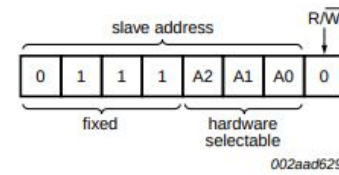
I²C LCD

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

```
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);
}
```



a. PCF8574



b. PCF8574A

| | 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

```
void I2CLCD_Init(I2CLCDDisplay display ){
    for(int i=0;i<sizeof(initArray);i++){
        sendNibbleCmd(display,initArray[i]);
        HAL_Delay(5);
    }
}
```

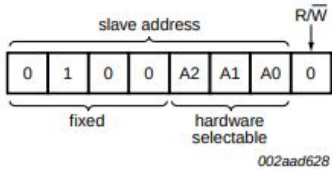
I²C LCD

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

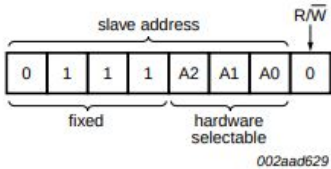
```
void sendNibbleData(I2CLCDDisplay display, uint8_t lower_nibble){  
    uint8_t shifted[2];  
    shifted[0] = (lower_nibble <<4);  
    shifted[1] = (lower_nibble <<4);  
    if (display.Backlight){  
        shifted[0] |= 0x08;  
        shifted[1] |= 0x08;  
    }  
    shifted[0] |= 0x05;  
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

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



a. PCF8574



b. PCF8574A

| | 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 |

I²C LCD

```
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++;
    }
}
```

Memorias SD por SPI

http://elm-chan.org/docs/mmc/mmc_e.html

Las memorias SD soportan el protocolo SDIO (Serie pero con 4 bits en paralelo con velocidades de hasta 208MHz).

También soporta el protocolo SPI.


Las memorias utilizan 3,3V. Los adaptadores

Suelen tener voltage converters.



SDC

| No | SD | SPI |
|----|------|------|
| 8 | DAT1 | |
| 7 | DAT0 | DO |
| 6 | Vss2 | |
| 5 | CLK | SCLK |
| 4 | Vcc | |
| 3 | Vss1 | |
| 2 | CMD | DI |
| 1 | CAT3 | CS |
| 9 | DAT2 | |



MMC

| No | MMC | SPI |
|----|------|------|
| 7 | DAT | DO |
| 6 | Vss2 | |
| 5 | CLK | SCLK |
| 4 | Vcc | |
| 3 | Vss1 | |
| 2 | CMD | DI |
| 1 | RES | CS |

Nuevo | 48 vendidos

Modulo Micro Sd Ci
Con Adaptador 3v3

\$ 319

en 6x \$ 53⁷ sin interés

Ver los medios de pago

 Llega hoy por \$ 449⁹⁹

Comprando dentro de los próximos 18 min.

Beneficio Mercado Puntos

Ver más formas de entrega

 Devolución gratis

Tenés 30 días desde que lo recibís.

Conocer más

Stock disponible

http://elm-chan.org/docs/mmc/mmc_e.html

Mediante SPI se utiliza:

- Chip Select (Activo en Bajo)
- Clock (se inicializa en 250KHz pero luego se puede aumentar)
- MISO
- MOSI

Existen memorias

- MMC (casi en desuso)
- **SDSC (de 8MB a 2GB)**
- SDHC (de 2GB a 32GB)
- **SDXC (de 32GB a 2TB)**
- SDUC (de 2TB a 128TB)

YMMV = Your mileage might vary.