MICROS Y LABORATORIO
GLCD SPI + RTC

ROBINSON JIMENEZ MORENO

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

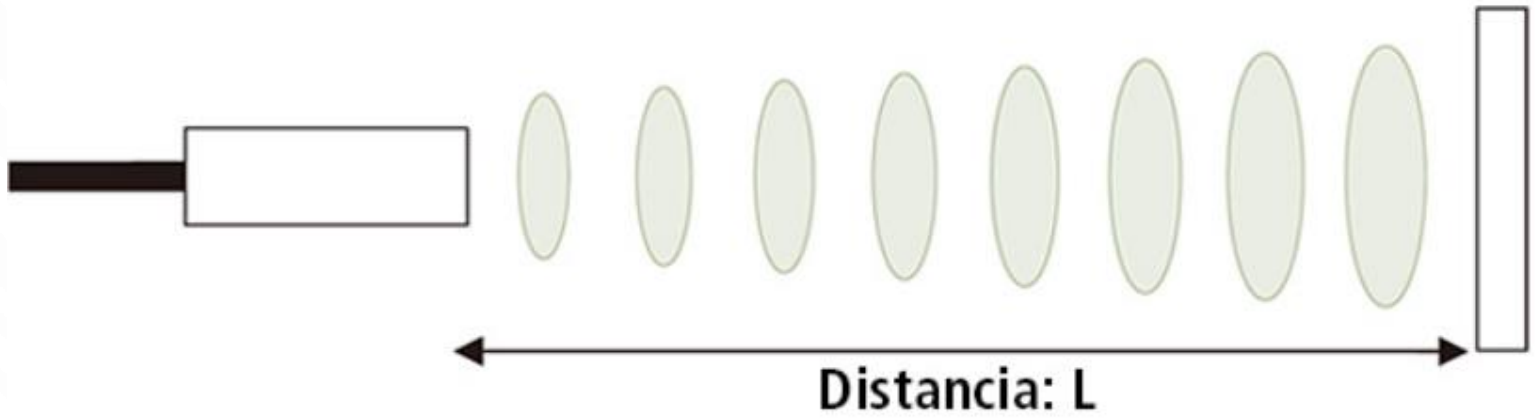
SENSOR ULTRASONIDO

Miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.

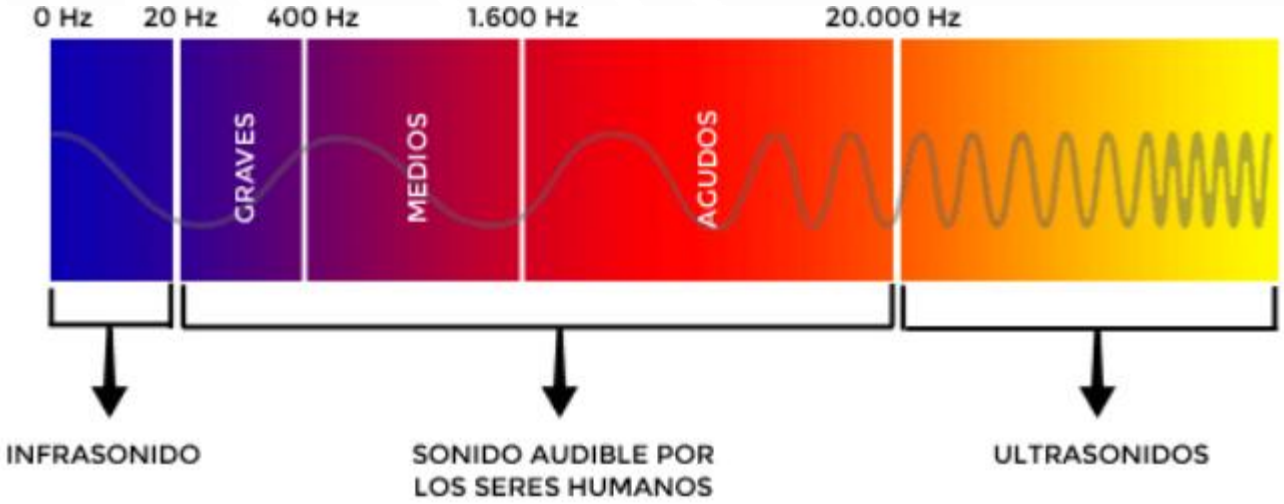
La distancia se puede calcular con la siguiente fórmula:

$$\textbf{Distancia } L = 1/2 \times T \times C$$

donde L es la distancia, T es el tiempo entre la emisión y la recepción, y C es la velocidad del sonido. (El valor se multiplica por 1/2 ya que T es el tiempo de recorrido de ida y vuelta).



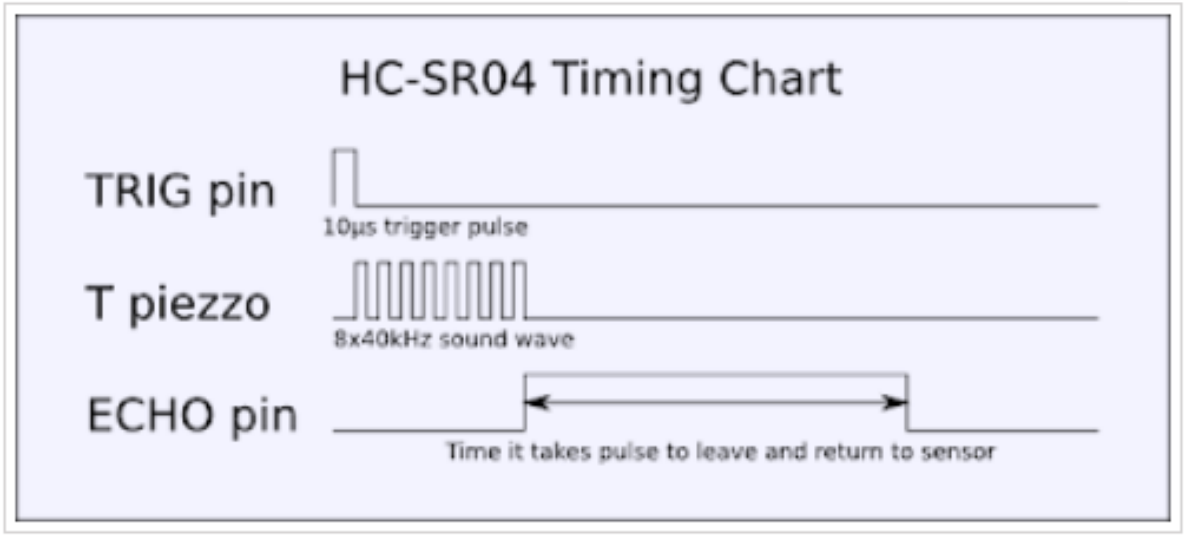
El sensor HC-SR04 tiene una sensibilidad muy buena del orden de los 3mm, es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm. Se requiere un microcontrolador para leer los datos que entrega. Tienen dos transductores, básicamente, un altavoz y un micrófono.



Pines de conexión:

- VCC
- Trig (*Disparo del ultrasonido*)
- Echo (*Recepción del ultrasonido*)
- GND

Distancia = {(Tiempo entre Trig y el Echo) * (V.Sonido 340 m/s)}/2



Funcionamiento:

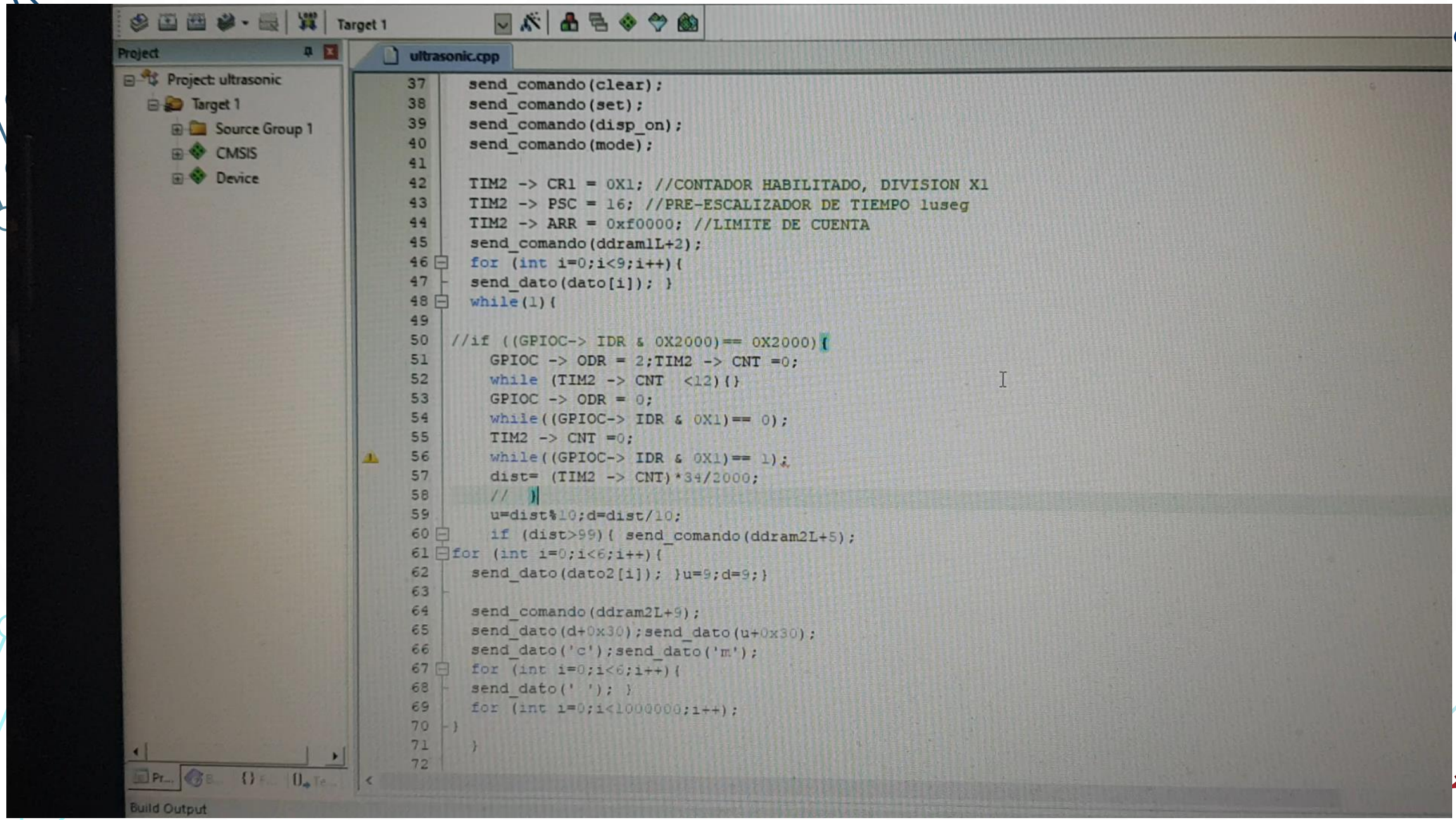
1.Enviar un Pulso "1" de al menos de 10uS por el Pin Trigger (Disparador).
2.El sensor enviará 8 Pulsos de 40KHz (Ultrasonido) y coloca su salida Echo a alto (seteo), se debe detectar este evento e iniciar un conteo de tiempo.
3.La salida Echo se mantendrá en alto hasta recibir el eco reflejado por el obstáculo a lo cual el sensor pondrá su pin Echo a bajo, es decir, terminar de contar el tiempo.
4.Se recomienda dar un tiempo de aproximadamente 50ms de espera después de terminar la cuenta.
5.La distancia es proporcional a la duración del pulso y puede ser calculada con las siguiente formula
6.Distancia en cm (centímetros) = Tiempo medido en us x 0.017

MICRO

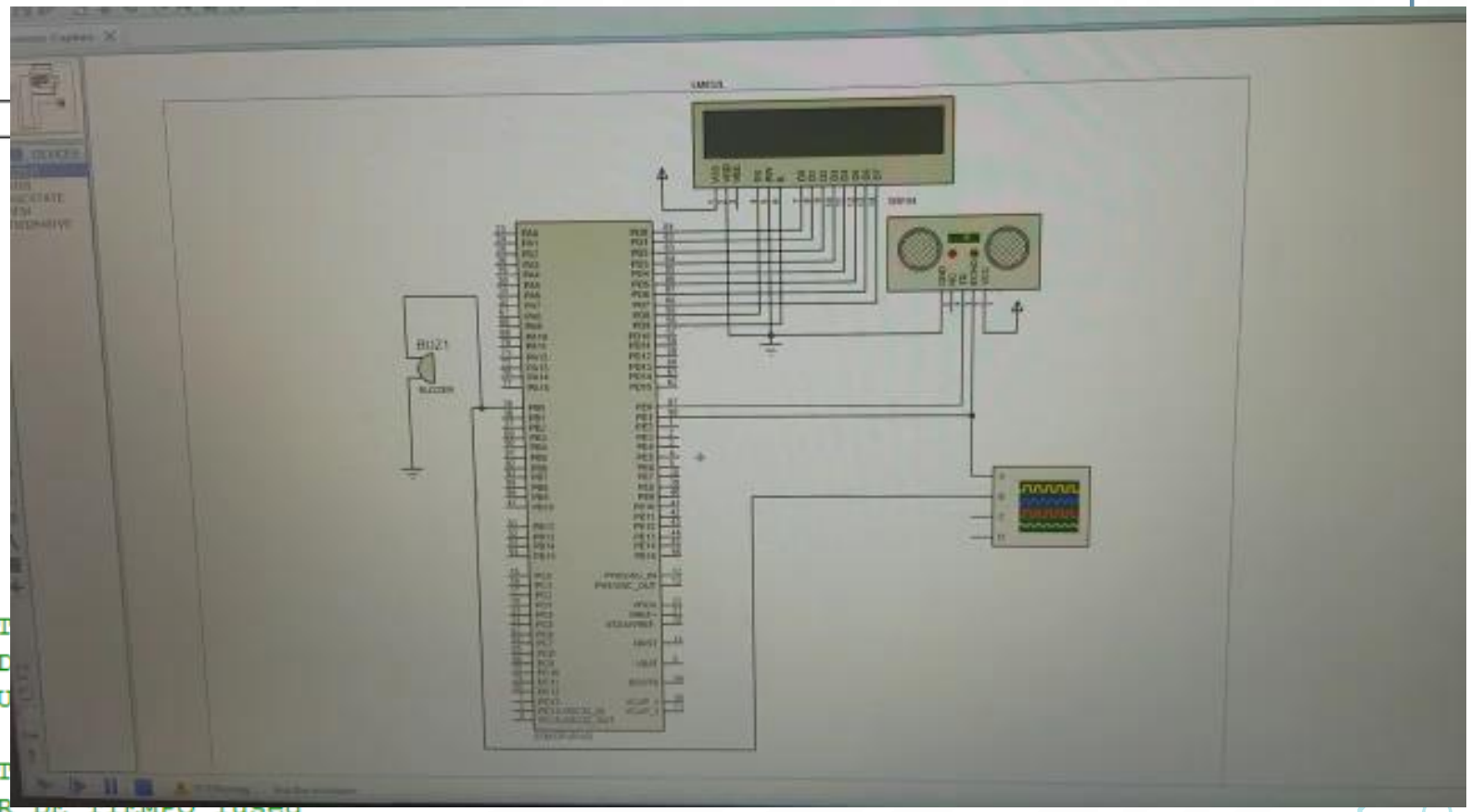
- CONFIGURACIÓN PINES ENTRADA SALIDA DIGITAL
- CONFIGURACIÓN TIMER CAPTURA DE TIEMPO
- ENVIAR PULSO DE 10 MICRO SEGUNDOS
- DETECTAR SEÑAL ECHO (FLANCO SUBIDA)
- INICIAR TIMER
- DETECTAR SEÑAL ECHO (FLANCO BAJADA)
- LEER TIMER
- CALCULAR DISTANCIA

```c
#include "stm32f7xx.h"

char dato[9]={'D','i','s','t','a','n','c','i','a'};
char dato2[6]={'M','a','x',' ',' ',' '};
char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char ddram1L=0x80;
char ddram2L=0xC0;
int u,d,dist=0;

void send_comando(char a){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= a;
  GPIOG->ODR &=~(1UL<<8);//RS=0
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int dl=0;dl<5000;dl++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}

void send_dato(char b){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= b;
  GPIOG->ODR |=(1UL<<8);//RS=1
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int dl=0;dl<5000;dl++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}
```

```c
int main(void){
  RCC -> AHB1ENR = 0X7f; //PUERTO B
  RCC -> APB1ENR = 0X1; //TIMER 2
  GPIOB -> MODER = 0X10004001; //LEDS TARJETA
  GPIOC -> MODER = 0x4;

  GPIOG->MODER =0x55555;
  send_comando(clear);
  send_comando(set);
  send_comando(disp_on);
  send_comando(mode);

  TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
  TIM2 -> PSC = 16; //PRE-ESCALIZADOR DE TIEMPO 1useg
  TIM2 -> ARR = 0xf0000; //LIMITE DE CUENTA
  send_comando(ddram1L+2);
  for (int i=0;i<9;i++){
  send_dato(dato[i]); }

  while(1){//if ((GPIOC-> IDR & 0X2000)== 0X2000){
    GPIOC -> ODR = 2;TIM2 -> CNT =0;
    while (TIM2 -> CNT  <12){}
    GPIOC -> ODR = 0;
    while((GPIOC-> IDR & 0X1)== 0);
    TIM2 -> CNT =0;
    while((GPIOC-> IDR & 0X1)== 1);
    dist= (TIM2 -> CNT)*17/1000;
    //  }
    u=dist%10;d=dist/10;
    if (dist>99){ send_comando(ddram2L+6);
  for (int i=0;i<6;i++){
  send_dato(dato2[i]); }}
  else {send_comando(ddram2L+6);
  for (int i=0;i<3;i++){send_dato(' ');}
  for (int i=0;i<800000;i++);
  send_comando(ddram2L+9);
  send_dato(d+0x30);send_dato(u+0x30);}
  send_dato('c');send_dato('m');
}
  }
```


Target 1
Project
Project: ultrasonic
Target 1
Source Group 1
CMSIS
Device
ultrasonic.cpp
send_comando(clear);
send_comando(set);
send_comando(disp_on);
send_comando(mode);
TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
TIM2 -> PSC = 16; //PRE-ESCALIZADOR DE TIEMPO luseg
TIM2 -> ARR = 0xf0000; //LIMITE DE CUENTA
send_comando(ddram1L+2);
for (int i=0;i<9;i++){
send_dato(dato[i]); }
while(1){
//if ((GPIOC-> IDR & 0X2000)== 0X2000){
GPIOC -> ODR = 2;TIM2 -> CNT =0;
while (TIM2 -> CNT <12){}
GPIOC -> ODR = 0;
while((GPIOC-> IDR & 0X1)== 0);
TIM2 -> CNT =0;
while((GPIOC-> IDR & 0X1)== 1);
dist= (TIM2 -> CNT)*34/2000;
// }
u=dist%10;d=dist/10;
if (dist>99){ send_comando(ddram2L+5);
for (int i=0;i<6;i++){
send_dato(dato2[i]); }u=9;d=9;}
send_comando(ddram2L+9);
send_dato(d+0x30);send_dato(u+0x30);
send_dato('c');send_dato('m');
for (int i=0;i<6;i++){
send_dato(' '); }
for (int i=0;i<1000000;i++);
}
}
Build Output

```
  void TIM3_IRQHandler(void)   {
    TIM3->SR &= 0;
  if(a>1){   GPIOB -> ODR ^= 1; }//INTERMITENCIA LED 1
  }
}
void send_comando(char a){
void send_dato(char b){
int main(void){
  RCC -> AHB1ENR = 0X7f; //PUERTO B
  RCC -> APB1ENR = 0X3; //TIMER 2-3
  GPIOB -> MODER |= 0X10004001; //LEDS
  GPIOE -> MODER = 0x1;
  GPIOD->MODER =0x55555;

  send_comando(clear);
  for(int temp=0;temp<1000;temp++);
  send_comando(set);
  send_comando(disp_on);
  send_comando(mode);

  TIM2 -> CR1 = 0X1; //CONTADOR HABILIT
  TIM2 -> PSC = 16; //PRE-ESCALIZADOR D
  TIM2 -> ARR = 0xf0000; //LIMITE DE CU

  TIM3 -> CR1 = 0X1; //CONTADOR HABILIT
  TIM3 -> PSC = 1600; //PRE-ESCALIZADOR DE TIEMPO 1useg
  TIM3 -> ARR = 500; //frec 1Khz
  TIM3 -> DIER = 0X1;
  NVIC_EnableIRQ(TIM3_IRQn);
```



29 Real-time clock (RTC)

29.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

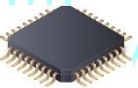
Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

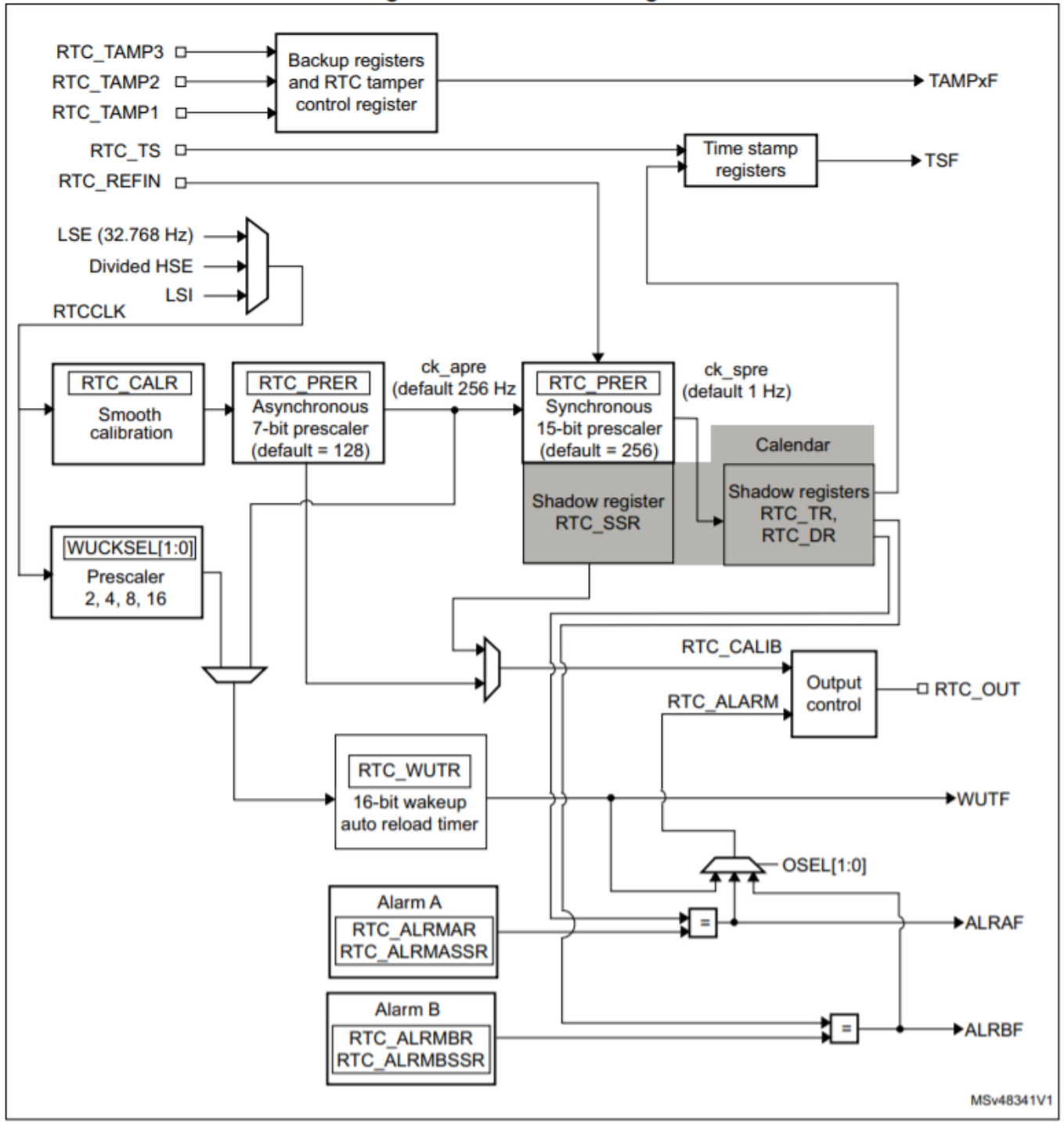
As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).



29.2 RTC main features

The RTC unit main features are the following (see *Figure 302: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- 32 backup registers.


RTC_TAMP3
RTC_TAMP2
RTC_TAMP1
Backup registers and RTC tamper control register
TAMPxF
RTC_TS
RTC_REFIN
Time stamp registers
TSF
LSE (32.768 Hz)
Divided HSE
LSI
RTCCLK
RTC_CALR
Smooth calibration
RTC_PRER
Asynchronous 7-bit prescaler (default = 128)
ck_apre (default 256 Hz
RTC_PRER
Synchronous 15-bit prescaler (default = 256)
ck_spre (default 1 Hz)
Calendar
Shadow register RTC_SSR
Shadow registers RTC_TR, RTC_DR
WUCKSEL[1:0]
Prescaler 2, 4, 8, 16
RTC_CALIB
RTC_ALARM
Output control
RTC_OUT
RTC_WUTR
16-bit wakeup auto reload timer
WUTF
OSEL[1:0]
Alarm A
RTC_ALRMAR
RTC_ALRMASSR
=
ALRAF
Alarm B
RTC_ALRMBR
RTC_ALRMBSSR
=
ALRBF
MSv48341V1

29.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 29.6.4: RTC initialization and status register (RTC_ISR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

29.3.7 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR_CR1 register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_TAMPCR, RTC_BKPxR, RTC_OR and RTC_ISR[13:8].

1. Write '0xCA' into the RTC_WPR register.
2. Write '0x53' into the RTC_WPR register.

```
RTC->WPR =0xCA;
RTC->WPR =0x53;
```

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

29.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

```
RTC->CR = 0x40;
```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
							rw	rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC_TS input rising edge generates a time-stamp event

1: RTC_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected

11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled

1: Wakeup timer enabled

Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

Bit 9 **ALRBE**: *Alarm B enable*

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE:** Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

29.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in *RTC register write protection on page 917*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF
														rc_w0	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

Bit 9 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).

This flag is cleared by software by writing 0.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).

This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

29.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

```
RTC->WPR =0xCA;
RTC->WPR =0x53;
```

5.3.20 RCC backup domain control register (RCC_BDCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	Res.	Res.	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw						rw	rw				rw	rw	rw	r	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **BDRST:** Backup domain software reset
This bit is set and cleared by software.
0: Reset not activated
1: Resets the entire Backup domain
Note: The BKPSRAM is not affected by this reset, the only way of resetting the BKPSRAM is through the Flash interface when a protection level change from level 1 to level 0 is requested.

Bit 15 **RTCEN:** RTC clock enable
This bit is set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

```
RCC->BDCR |=0x8100;
```

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection
These bits are set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.
00: No clock
01: LSE oscillator clock used as the RTC clock
10: LSI oscillator clock used as the RTC clock
11: HSE oscillator clock divided by a programmable prescaler (selection through the RTCPRE[4:0] bits in the RCC clock configuration register (RCC_CFGR)) used as the RTC clock

29.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 917* and *Reading the calendar on page 918*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 917*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

```
RTC->TR = 0x515950;      //hora  11:59:50 pm
```

29.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 917* and *Reading the calendar on page 918*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 917*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden
001: Monday
...
111: Sunday

```
RTC->DR = 0x22E330;        //fecha
```

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

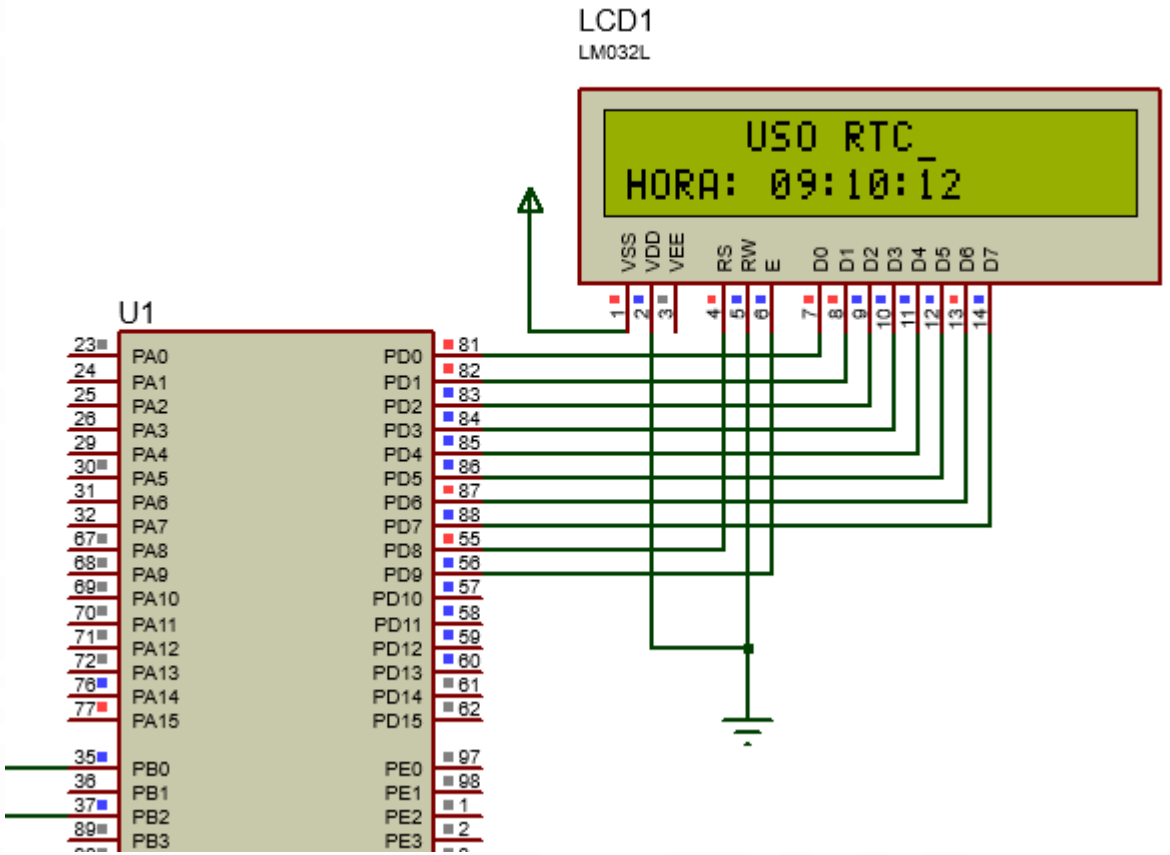
```
////////////////////////////////  LCD MAS RTC INTERNO STM 32  ////////

#include <stdio.h>
#include "STM32F7xx.h"

int hora=0;
int fecha=0;
char Fecha[]={'F','e','c','h','a',':',' ',' ',' ',' ',' ','/',' ',' ',' ','/'};
char Hora[]={'H','o','r','a',':',' ',' ',' ',':',' ',' ',' ',':'};
int us=0, ds=0,um=0,dm=0,uh=0,dh=0,ud=0,dd=0,uM=0,dM=0,ua=0,da=0,pm=0;

char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char ddram1L=0x80;
char ddram2L=0xC0;

void send_comando(char a){
  GPIOG ->ODR = a;
  GPIOG->ODR &=~(1UL<<8);//RS=0
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}

void send_dato(char b){
  GPIOG ->ODR = b;
  GPIOG->ODR |=(1UL<<8);//RS=1
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}
```

```
int main(void){
  //configuracin RTC
  RCC->APB1ENR |= (1UL<<28); // ACTIVA reloj de PWR
  PWR->CR1 |= 0x100;  // habilita el bit para 8 acceso a escritura
  RCC->BDCR |=0x1;    //habilita el reloj LSE
  for(int i=0;(RCC->BDCR &= 0x3)==0x1;i++){};  //retardo estabilizacion reloj LSE
  RCC->BDCR |=0x8100;      //habilitA reloj de RTC y fuente de reloj RTC (LSE)
  RTC->WPR =0xCA;          //claves para acceso a escritura
  RTC->WPR =0x53;
  RTC->ISR = 0x80;          //habilitA modo inicializacion en el bit 7
  for(int i=0;(RTC->ISR &= 0xC0)==0x80;i++){};//retardo para entrar modo inicializacion
  RTC->TR = 0x515950;     //hora
  RTC->DR = 0x20E330;     //fecha
  RTC->CR = 0x40;         //formato
  RTC->ISR &= 0xffffff7f;  //sale del modo inicializacion borrando el bit 7
  RTC->WPR =0xff;          //deshabilita acceso a escritura
  PWR->CR1 &= 0xfffffEff;  //deshabilito el bit 8 del RTC
  //configuracion LCD
  RCC->AHB1ENR |= 0xffff;
  GPIOG->MODER = 0x55555;
  send_comando(set);
  send_comando(disp_on);
  send_comando(mode);

  for(int i=0;i<11;i++){
    send_dato(Hora[i]);
  }
  send_comando(0xc0);
  for(int i=0;i<14;i++){
    send_dato(Fecha[i]);
  }
```

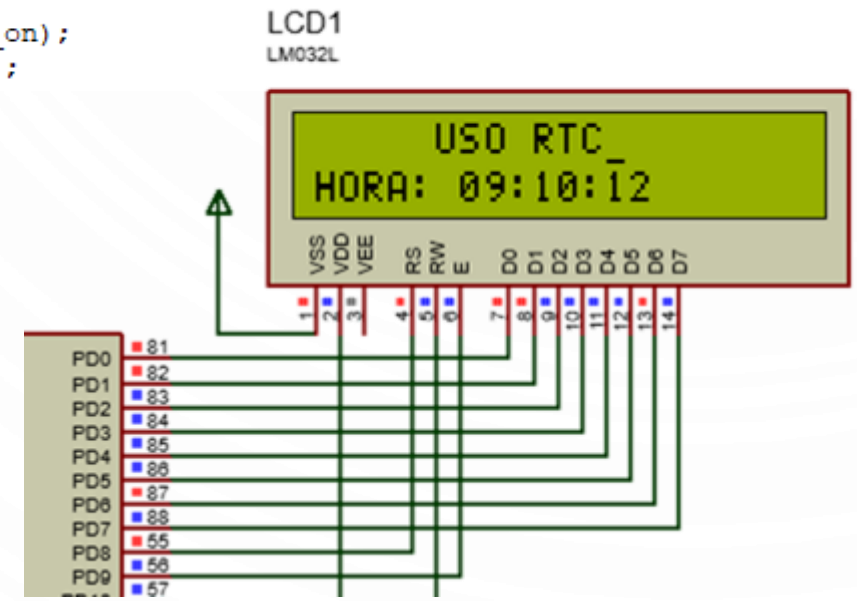
```
while(true){
  hora=RTC->TR;
  fecha=RTC->DR;

  us=hora & 0xf;
  ds=(hora>>4)&0xf;
  um=(hora>>8)&0xf;
  dm=(hora>>12)&0xf;
  uh=(hora>>16)&0xf;
  dh=(hora>>20)&0xf;
  dh=dh&0x3;
  pm=(hora>>22);

  ud=fecha & 0xf;
  dd=(fecha>>4)&0xf;
  uM=(fecha>>8)&0xf;
  dM=(fecha>>12)&0x1;
  ua=(fecha>>16)&0xf;
  da=(fecha>>20)&0xf;
```

```
    send_comando(0x8C);
    send_dato(us+48);
    send_comando(0x8B);
    send_dato(ds+48);
    send_comando(0x89);
    send_dato(um+48);
    send_comando(0x88);
    send_dato(dm+48);
    send_comando(0x86);
    send_dato(uh+48);
    send_comando(0x85);
    send_dato(dh+48);

    send_comando(0xc9);
    send_dato(ud+48);
    send_comando(0xc8);
    send_dato(dd+48);
    send_comando(0xcc);
    send_dato(uM+48);
    send_comando(0xcb);
    send_dato(dM+48);
    send_comando(0xcf);
    send_dato(ua+48);
    send_comando(0xcE);
    send_dato(da+48);

  }
}
```

```c
#include "stm32f4xx.h"

int hora = 0;
char horalim[7]={'U', 'S', 'O', ' ', 'R', 'T', 'C'};
char horact[16] ={'H', 'O', 'R', 'A', ':', ' ', ' ', ' ', ':', ' ', ' ', ':', ' ', ' '};
char lim[6]={};
char clear=0x01;
char set = 0x38;
char disp_on = 0x0E;
char mode = 0x06;
char ddram1L = 0x80;
char ddram2L = 0xC0;

void send_comando(char a){
GPIOD -> ODR = a;
GPIOD -> ODR &=~(1UL<<8); //RS=0
GPIOD -> ODR |=(1UL<<9); //Enable 1
for(int Cont=0; Cont<10000; Cont++);
GPIOD -> ODR &=~(1UL<<9); //Enable 0
}

void send_dato(char b){
GPIOD -> ODR = b;
GPIOD -> ODR |=(1UL<<8); //RS=1
GPIOD -> ODR |=(1UL<<9); //Enable 1
for(int Cont=0; Cont<10000; Cont++);
GPIOD -> ODR &=~(1UL<<9); //Enable 0
}
```



```c
int main(void){
RCC -> AHB1ENR |= 0xFF;
GPIOD->MODER = 0x5555555;
//Configuracion RTC
RCC -> APB1ENR |= (1UL<<28); //ACTIVA reloj de PWR
PWR -> CR |= 0x100;  //habilita el bit para acceso a escritura
RCC -> BDCR |= 0x1;  //habilita el reloj LSE
for (int i=0; (RCC -> BDCR &= 0x3)== 0x1; i++){}; //Retardo estabilizacion reloj LSE
RCC -> BDCR |= 0x8100; //habilita reloj de RTC y fuente de reloj RTC (LSE)
RTC -> WPR = 0xCA; //clave para acceso a escritura
RTC -> WPR = 0x53;
 RTC -> ISR = 0x80; //habilita modo inicializacion en el bit 7
for (int i=0; (RTC -> ISR &= 0xC0)==0x80; i++){};  //retardo para entrar modo inicializacion
RTC -> TR = 0x491010; //HORA
RTC -> CR = 0x40; //FORMATO
RTC -> ISR &= 0xFFFFFF7F; //SALE DEL MODO INICIALIZACION BORRANDO EL BIT 7
RTC -> WPR = 0xFF; //DESHABILITA ACCESO A ESCRITURA
PWR -> CR &= 0xFFFFFEFF; //DESHABILITO EL BIT 8 DEL RTC
//CONFIGURACION LCD

send_comando(set);
send_comando(disp_on);
send_comando(mode);
```

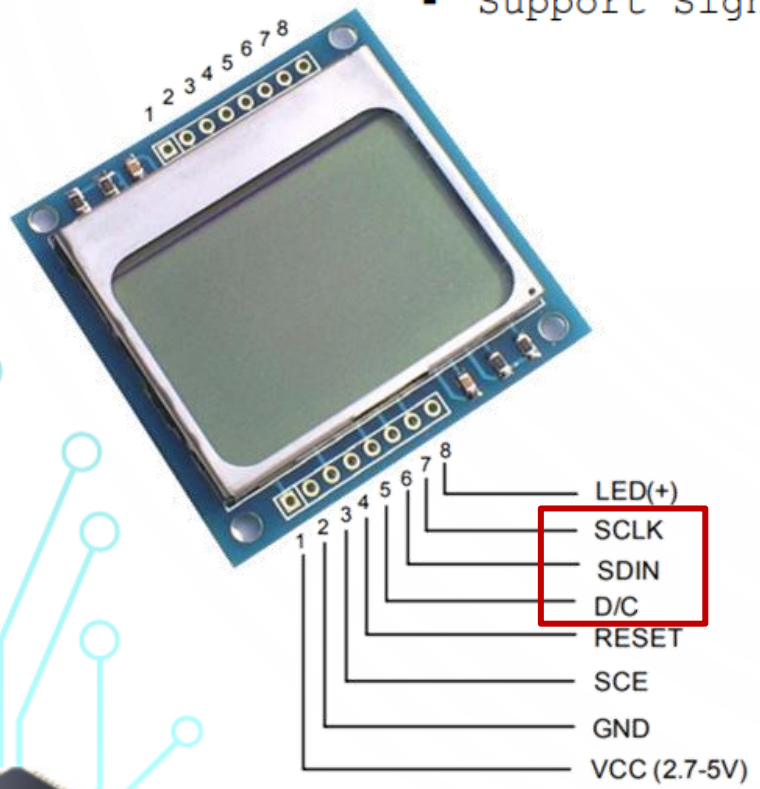



```c
  while(true){
hora = RTC -> TR;
horact[6]=((hora>>20)&0x3)+48;
horact[7]=((hora>>16)&0xF)+48;
horact[9]=((hora>>12)&0xF)+48;
horact[10]=((hora>>8)&0xF)+48;
horact[12]=((hora>>4)&0xF)+48;
horact[13]= (hora & 0xF) +48;

send_comando(ddram2L);
    for (int i=0;i<16;i++){
     send_dato(horact[i]);}

send_comando(ddram1L+5);
  for (int i=0;i<7;i++){
send_dato(horalim[i]);}

 for (long i=0;i<500000;i++)__NOP;

}
}
```

LCD GRAFICA

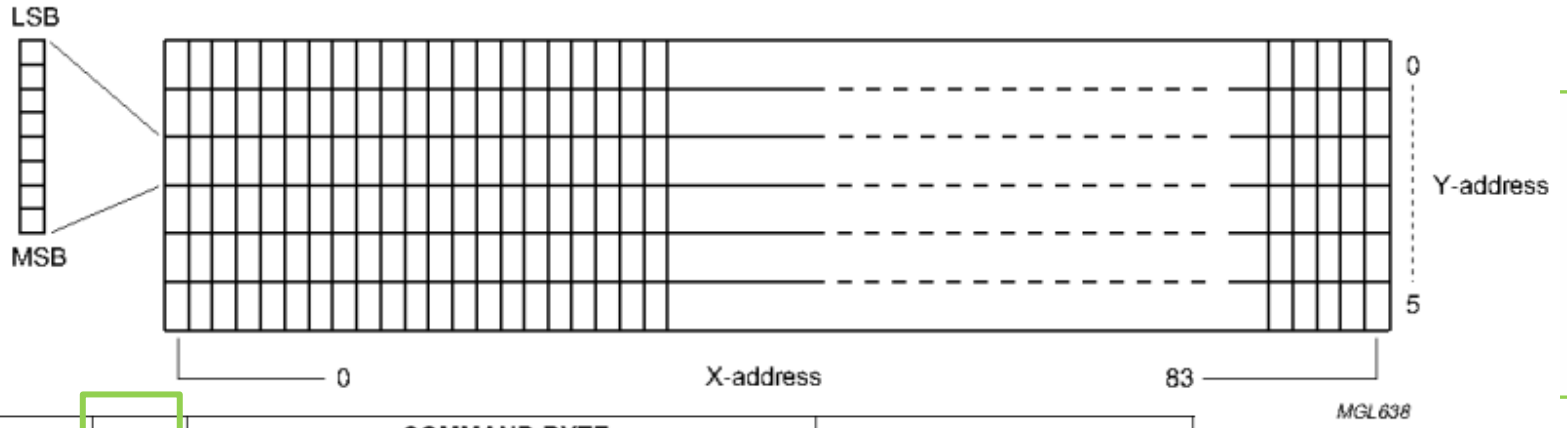
Specifications of LCD 5110

- 48 x 84 Dot LCD Display
- Serial Bus Interface with maximum high speed 4.0 Mbits/S
- Internal Controller No.PCD8544
- LED Back-Light
- Run at Voltage 2.7 -5.0 Volt
- Low power consumption; it is suitable for battery applications
- Temperature range from -25˚C to +70˚C
- Support Signal CMOS Input



Handling of LCD Address (Addressing)

The address arrangement of memory that is shown on LCD Display (DDRAM) is Matrix that consists of 6 rows (Y Address) from Y-Address 0 to Y-Address 5 and 84 columns (X Address) from X-Address 0 to X-Address 83. If user wants to access to the position of displaying result on LCD Display, must refer to the relationship between X-Address and Y-Address. Data that will be sent to display is 8 bit (1 Byte) and it will be arranged as vertical line; in this case, Bit MSB will be lower and Bit LSB will be upper as shown in the following picture;

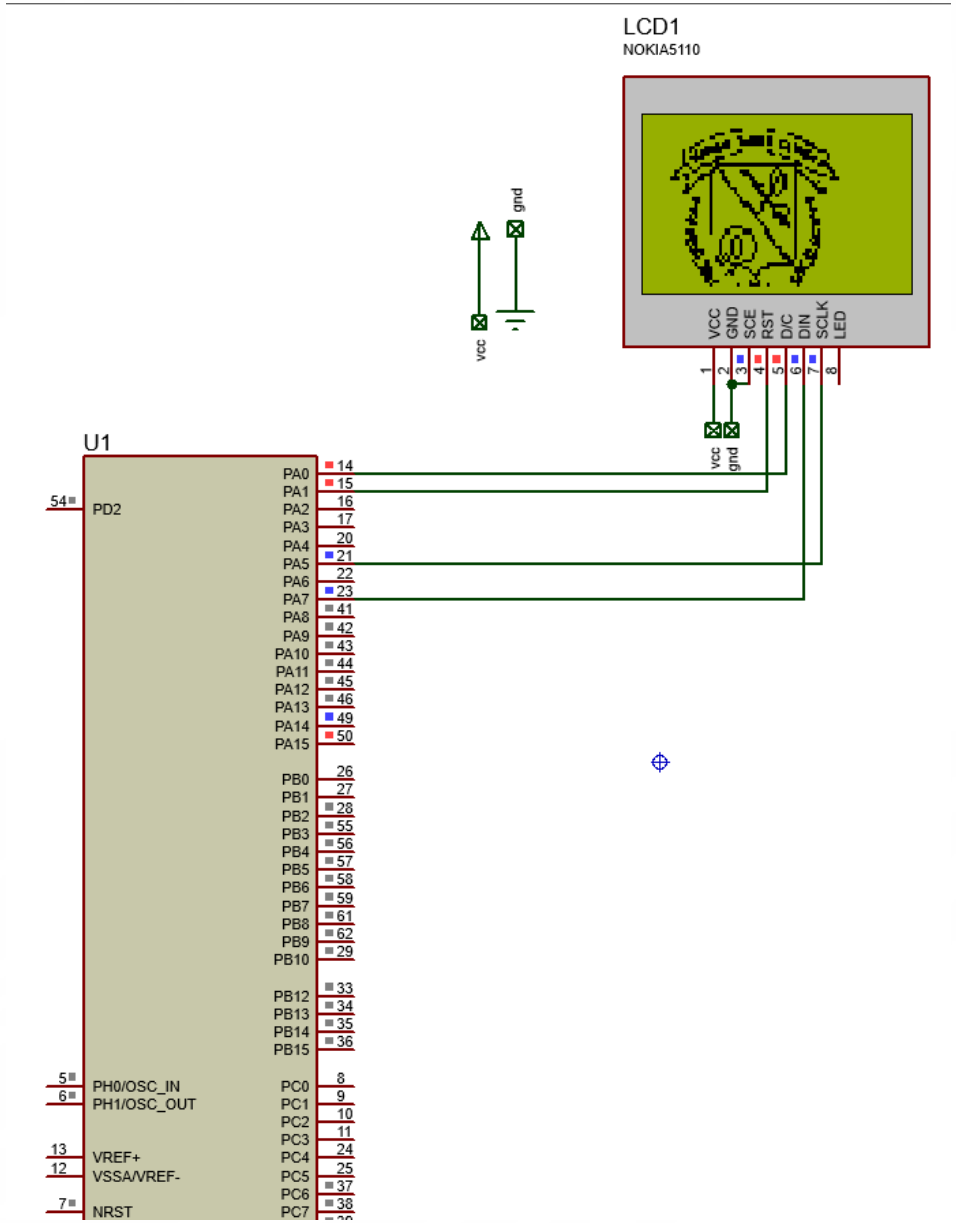


INSTRUCTION	D/$\overline{C}$	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	writes data to display RAM
(H = 0)										
Reserved	0	0	0	0	0	0	1	X	X	do not use
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Reserved	0	0	0	0	1	X	X	X	X	do not use
Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	do not use
	0	0	0	0	0	0	0	1	X	do not use
Temperature control	0	0	0	0	0	0	1	TC_1	TC_0	set Temperature Coefficient (TC_x)
Reserved	0	0	0	0	0	1	X	X	X	do not use
Bias system	0	0	0	0	1	0	BS_2	BS_1	BS_0	set Bias System (BS_x)
Reserved	0	0	1	X	X	X	X	X	X	do not use
Set V_{OP}	0	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	write V_{OP} to register

PD	chip is active
V	horizontal addressing
H	use basic instruction set
D and E	
00	display blank
10	normal mode
01	all display segments on
11	inverse video mode
TC_1 and TC_0	
00	V_{LCD} temperature coefficient 0
01	V_{LCD} temperature coefficient 1
10	V_{LCD} temperature coefficient 2
11	V_{LCD} temperature coefficient 3

```
TxSPI(0x20);
TxSPI(0x0C);
TxSPI(0x80);
```

```c
#include "stm32f401xe.h"

const unsigned char Imagen[504] =
{
const unsigned char Imagen2[504] =
{
void TxSPI(unsigned char d){
 SPI1->DR=d;//Transmisión del dato
 while(SPI1->SR&SPI_SR_BSY);//Espera fin de la transmisión
}

// Función principal Main //
int main(void){
 RCC->AHB1ENR|=(1UL<<0);
 GPIOA->MODER|=(2<<10)|(2<<14)|(1<<0)|(1<<2);
 GPIOA->AFR[0]|=(5UL<<28)|(5UL<<20);//AF5

 //Configuracion SPI 1 //
 RCC->APB2ENR|=RCC_APB2ENR_SPI1EN;
 SPI1->CR1=0;
 SPI1->CR1=(7<<3)|SPI_CR1_MSTR |SPI_CR1_SPE;;

 //Inicio de la pantalla MANUAL: PCD8544.pdf//
 GPIOA->ODR=2;

 for(int r=0;r<10000;r++)__ASM("NOP");
 TxSPI(0x20);
 TxSPI(0x0C);
 TxSPI(0x80);
 GPIOA->ODR=3;

 for(int n=0;n<504;n++)//Envio de mapa de bits
 TxSPI(Imagen[n]);

 while(1){ }
}
```



TAREA próxima clase:

Emplear una LCD grafica por SPI para visualizar la temperatura del sensor LM 35 este último con valor numérico y por barra tipo termómetro (vertical), se debe visualizar la hora y fecha mediante el RTC.