



Introduction

This application note describes a technique for driving liquid crystal displays (LCD) with STM32F10xxx microcontrollers (MCU), that is without any specific on-chip LCD driver hardware. This technique offers a solution for applications that require a display at low cost together with the versatile capabilities of standard STM32F10xxx MCUs. This application note also provides a technique to control the LCD contrast through firmware.

After an introduction on LCDs, [Section 1](#), [Section 2](#) & [Section 3](#) describe the typical waveforms required to drive an LCD with a multiplexing rate of 1 or 2 (duplex) and 4 (quadruplex). [Section 3](#) presents a solution based on a standard STM32F10xxx MCU directly driving a quadruplex LCD. This solution can be implemented with any STM32F10xxx MCU as it only requires the standard I/O ports and one timer, which are standard features on all STM32F10xxx MCUs. [Section 4](#) describes how to use the STM32F10xxx LCD glass library and gives a brief presentation of the LCD glass demo. [Section 5](#) describes low-power management using the LCD glass, and gives current and load CPU measurements. Finally, [Section 7](#) gives an example of the LCD connections to the STM32 MCU (two types of LCDs: 4 digits (64 segments: ref CT4-098-LV) and 8 digits (128 segments: ref: PD-878)).

With the STM32F10xxx LCD glass library, the CPU load for controlling the LCD is around 0.05% (at 72 MHz), and the number of external components is kept to the minimum (two external resistors per COM line). The number of I/Os involved is the same as in solutions using an on-chip LCD hardware driver or an external hardware LCD driver. With firmware contrast control, the STM32F10xxx MCU offers a very flexible solution that can be adapted easily to a range of applications.

The measured DC value depends on the used resistors (refer to [Section 3](#)) and the user configuration: LCD interruption priority configuration (refer to [Section 4.2](#)).

Contents

- 1 LCD principle 5**
- 2 LCD drive signals 7**
 - 2.1 Single backplane LCD drive 7
 - 2.2 Duplexed LCD drive 7
 - 2.3 Quadruplex LCD drive 9
 - 2.3.1 LCD mean voltage calculation 11
 - 2.3.2 Contrast calculation 11
- 3 Example of a quadruplex LCD with STM32F10xxx 13**
- 4 Firmware description 17**
 - 4.1 How to use the LCD library 18
 - 4.1.1 How to select the LCD glass by firmware 18
 - 4.1.2 How to select GPIOs to connect the LCD glass to STM32F10xxx 18
 - 4.1.3 How to select the timer interrupt priority that drives the LCD 20
 - 4.1.4 LCD contrast control 21
 - 4.2 LCD timer interrupt 21
 - 4.2.1 LCD with high interrupt priority 21
 - 4.2.2 LCD with low-high interrupt priority (using boosting priority) 22
 - 4.2.3 LCD interrupt always with lowest priority 23
 - 4.3 Default demo configurations 23
- 5 LCD glass in battery-powered and low-power applications 24**
 - 5.1 Hardware implementation 24
 - 5.2 LCD glass managed by the timer using the Sleep low-power mode 25
 - 5.2.1 Some current and CPU load measurements 25
 - 5.3 LCD glass managed by the RTC and using the Stop low-power mode 26
 - 5.3.1 Some current and CPU load measurements 27
- 6 Comparison of the timer and RTC methods 29**
 - 6.1 Timer method 29

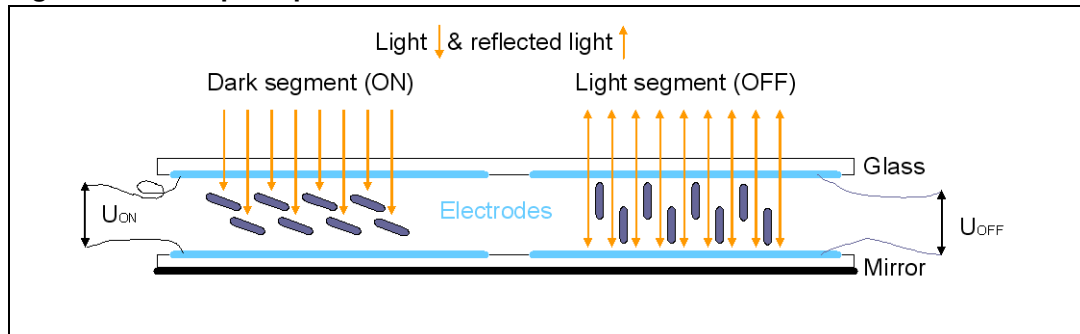
6.2	RTC method	29
7	Hardware implementation	30
7.1	Schematic	30
7.2	LCD segment line connections	31
8	Revision history	32

List of figures

Figure 1.	LCD principle	5
Figure 2.	Equivalent electrical schematic of an LCD segment.	6
Figure 3.	LCD signals for direct drive.	7
Figure 4.	Basic LCD segment connection in duplexed mode	8
Figure 5.	LCD signals for duplexed mode	8
Figure 6.	Basic LCD segment connection in quadruplexed mode	9
Figure 7.	LCD timing diagram for quadruplex mode	10
Figure 8.	LCD timing diagram for a single segment.	11
Figure 9.	Hardware connection diagram (1/2 bias control)	13
Figure 10.	LCD timing diagram with active and dead time (to decrease V_{rms})	14
Figure 11.	LCD timing diagram with active and dead time (to increase V_{rms})	15
Figure 12.	Screenshot of V_{com3} and V_{segx} (Segment1 of digit2)	16
Figure 13.	Screenshot of the four common lines	16
Figure 14.	Timing used by the STM32F10xxx LCD library	17
Figure 15.	LCD timing diagram for a single segment.	19
Figure 16.	Hardware connection diagram (1/2 bias control)	24
Figure 17.	Timer method state machine	25
Figure 18.	Timer method chronogram (in this case duty cycle = 50%)	26
Figure 19.	RTC method state machine	27
Figure 20.	RTC method chronogram	28
Figure 21.	LCD - STM32F10xxx connection example	30

1 LCD principle

Figure 1. LCD principle



An LCD panel is composed of many layers. A liquid crystal is filled between two of them (glass plates), that are separated by thin spacers coated with transparent electrodes and orientation layers. The orientation layer usually consists of a polymer (e.g. polyimide) that has been unidirectionally rubbed using, for instance, a soft cloth. As a result, the liquid crystal molecules are fixed, with their alignment more or less parallel to the plates, in the direction of rubbing. The crystal alignment directions at the surface of the two plates are perpendicular so that the molecules between the two plates undergo a homogeneous twist deformation in alignment to form a helix.

If no electric field is applied, the birefringent liquid crystal molecules keep their helical structure and rotate linearly polarized light waves passing through the plates. The transmitted light wave is then allowed through a crossed exit polarizer. As a result, the modulator has a bright appearance.

On the other hand, if an AC voltage of a few volts is applied, the resulting electric field forces the liquid crystal molecules to align themselves along the field direction and the twist deformation (the helix) is unwound. In this case, the polarization of the incident light is not rotated by the crystal molecules and the crossed exit polarizer blocks the light wave. As a result, the modulator appears dark.

The inverse switching behavior can be obtained with parallel polarizers. It must also be noted that gray scale modulation is easily achieved by varying the voltage between the crystal molecule reorientation threshold (reorientation is resisted by the elastic properties of liquid crystals) and the saturation field.

LCDs are sensitive to root mean square voltage ($V_{rms} = \sqrt{\text{Mean}(\text{Signal}^2)}$) levels. With a low root mean square voltage applied to it, an LCD is practically transparent (the LCD segment is then inactive or off). To turn an LCD segment on, causing the segment to turn dark (from light gray to opaque black), an LCD RMS voltage greater than the LCD threshold voltage is applied to the LCD. The LCD RMS voltage is the RMS voltage across the capacitor C in [Figure 2](#), which is equal to the potential difference between the SEG and COM values.

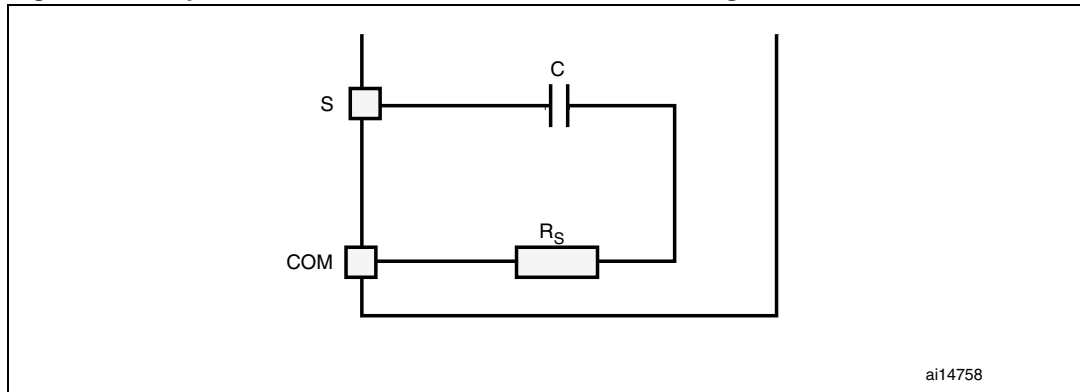
The LCD threshold voltage depends on the quality of the liquid used in the LCD and the temperature. The optical contrast is defined by the difference in transparency of an LCD segment that is on (dark) and an LCD segment that is off (transparent). The optical contrast depends on the difference between the RMS voltage on an on segment (V_{ON}) and the RMS voltage on an off segment (V_{OFF}). The higher the difference between $V_{ON(rms)}$ and $V_{OFF(rms)}$, the higher the optical contrast. The optical contrast also depends on the level of V_{ON} versus the LCD threshold voltage. If V_{ON} is below or close to the threshold voltage, the LCD is completely or almost transparent. If V_{OFF} is close or above the threshold voltage, the LCD is completely black.

In this document, contrast is defined as $D = V_{ON}(rms) / V_{OFF}(rms)$.

The applied LCD voltage must also alternate to give a zero DC value to prevent the electrolytic process and so, ensure a long LCD lifetime.

The higher the multiplexing rates, the lower the contrast. The signal period also has to be short enough to avoid visible flickering on the display.

Figure 2. Equivalent electrical schematic of an LCD segment



Note: The DC value should never be more than 100 mV (refer to the LCD manufacturer's datasheet), otherwise the LCD lifetime may be shortened. The frequency range is 30 Hz to 200 Hz typically. If it is less, the LCD flickers, if it is greater, power consumption increases.

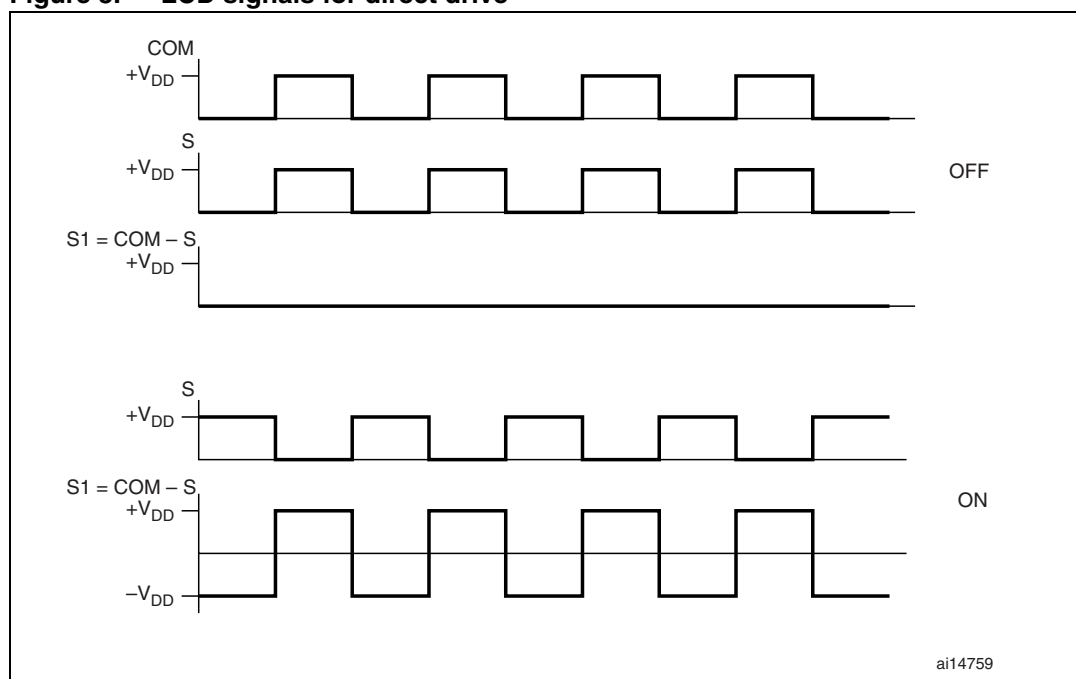
2 LCD drive signals

2.1 Single backplane LCD drive

In a single backplane drive, each LCD segment is connected to a segment line (S_x) and to a backplane (common line) common to all the segments. A display using S segments is driven with $S+1$ MCU output lines. The backplane is driven with a COM signal between 0 and V_{DD} with a duty cycle of 50%.

When switching a segment on, a signal with opposite polarity to COM is sent to the corresponding Segment pin. When the noninverted COM signal is sent to the Segment pin, the segment is off. Using an MCU, the I/O operates in output mode at either logic 0 or 1.

Figure 3. LCD signals for direct drive



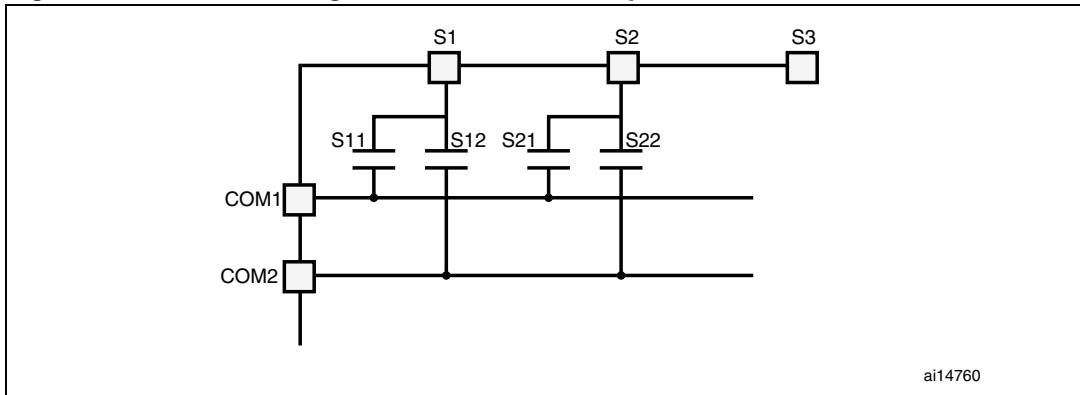
2.2 Duplexed LCD drive

In a duplexed drive, two backplanes are used instead of one. Each LCD segment line (S_x) is connected to two LCD segments, whose other side is connected to one of the two backplanes or common lines (refer to [Figure 4](#)). Thus, only $(S/2)+2$ MCU pins are necessary to drive an LCD with S segments.

Three different voltage levels have to be generated on the backplanes: 0, $V_{DD}/2$ and V_{DD} . The Segment voltage levels are 0 and V_{DD} only. [Figure 4](#) shows typical backplane, segment and LCD waveforms. The intermediate voltage $V_{DD}/2$ is only required for the backplane voltages. The STM32F10xxx I/O pins selected as backplanes are set by firmware to output mode for 0 or V_{DD} levels and to high impedance input mode for $V_{DD}/2$. When one backplane is active, the other is neutralized by applying $V_{DD}/2$ to it. This $V_{DD}/2$ voltage is determined by two resistors of equal value, externally connected to the I/O pin. By using an MCU with

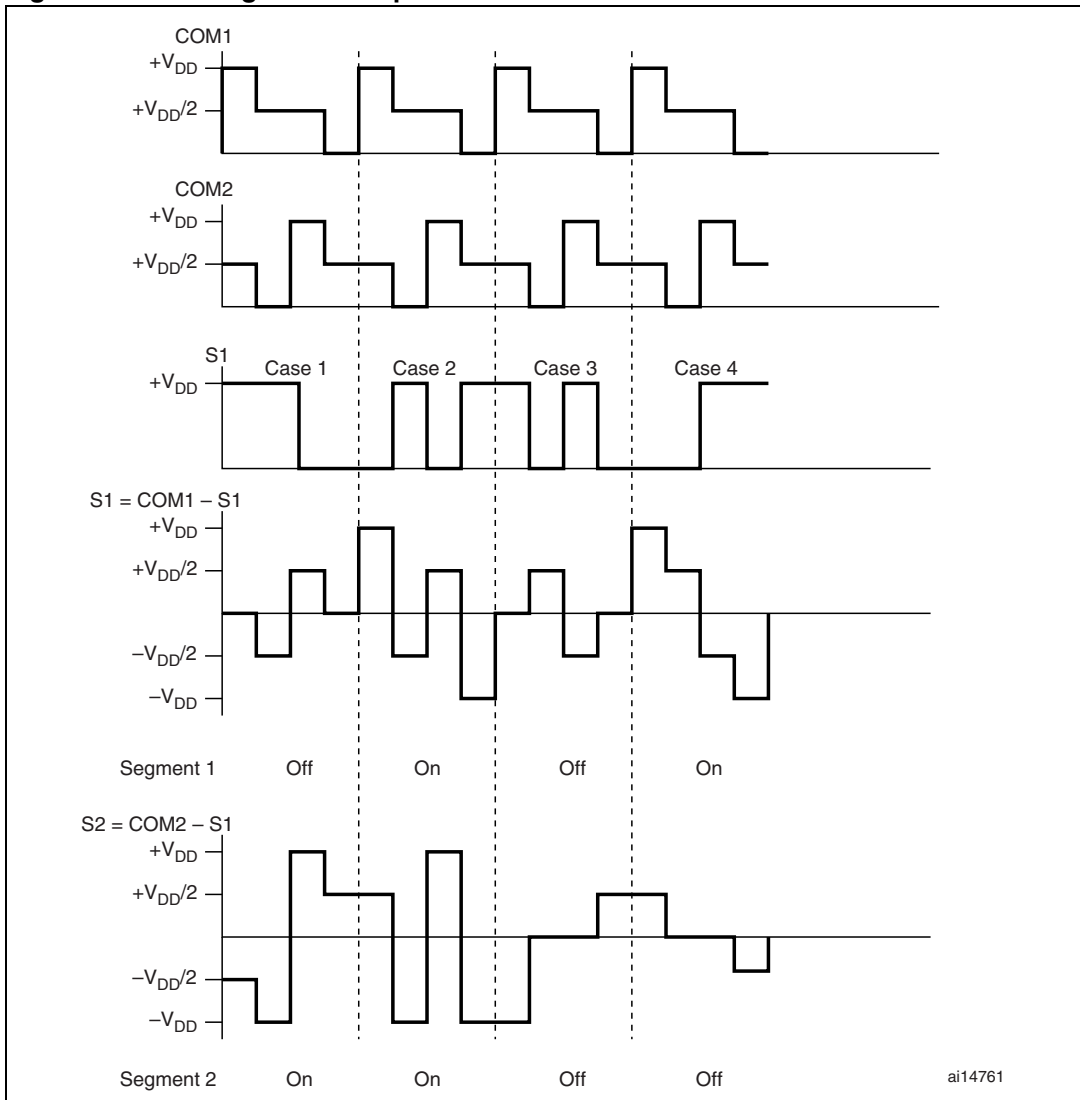
flexible I/O pin configuration, a duplexed LCD drive can be implemented with only 2 external resistor bridges (each on two COM lines).

Figure 4. Basic LCD segment connection in duplexed mode



ai14760

Figure 5. LCD signals for duplexed mode



ai14761

2.3 Quadruplex LCD drive

In a quadruplex LCD drive, four backplanes are used. Each LCD pin is connected to four LCD segments, whose other side is connected to one of the four backplanes. Thus, only $(S/4)+4$ MCU pins are necessary to drive an LCD with S segments. For example, to drive an LCD with 128 segments (32×4), only 36 I/O ports are required (32 I/O ports to drive the segments, 4 I/O ports to drive the backplanes).

Three different voltage levels have to be generated on the common lines: 0, $V_{DD}/2$, V_{DD} . The Segment line voltage levels are 0 and V_{DD} only. The LCD segment is inactive if the RMS voltage is below the LCD threshold voltage, and is active if the LCD RMS voltage is above the threshold. [Figure 7](#) shows typical backplane, Segment and LCD waveforms. The intermediate voltage $V_{DD}/2$ is only required for Backplane voltages. The MCU I/O pins selected as backplanes are set by firmware to output mode for 0 or V_{DD} levels and to the high-impedance input mode for $V_{DD}/2$. The $V_{DD}/2$ voltage is determined by two resistors of equal value, externally connected to the I/O pins. When one backplane or COM is active, the other ones are neutralized by applying $V_{DD}/2$ to them.

Figure 6. Basic LCD segment connection in quadruplexed mode

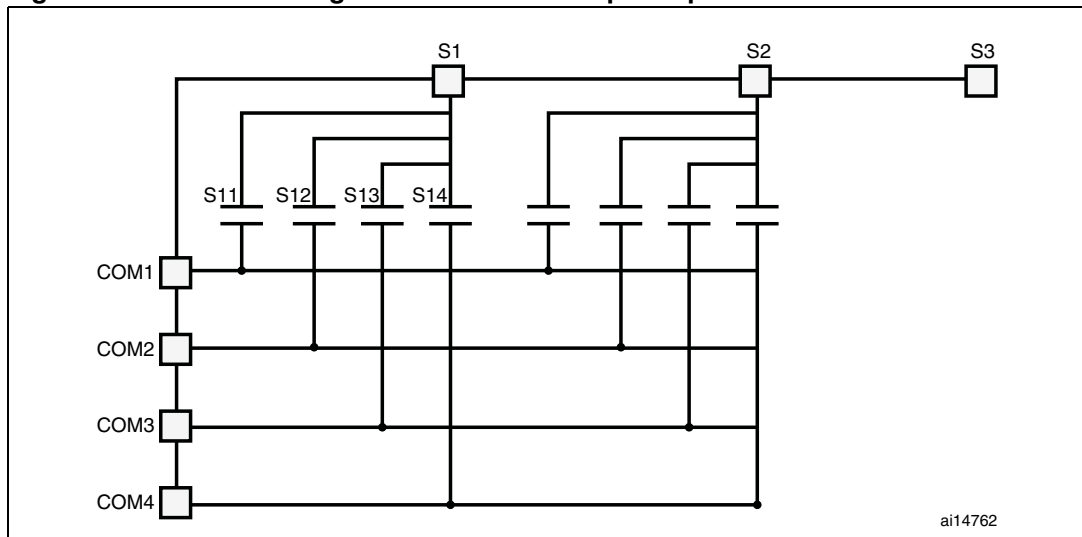


Figure 7. LCD timing diagram for quadruplex mode

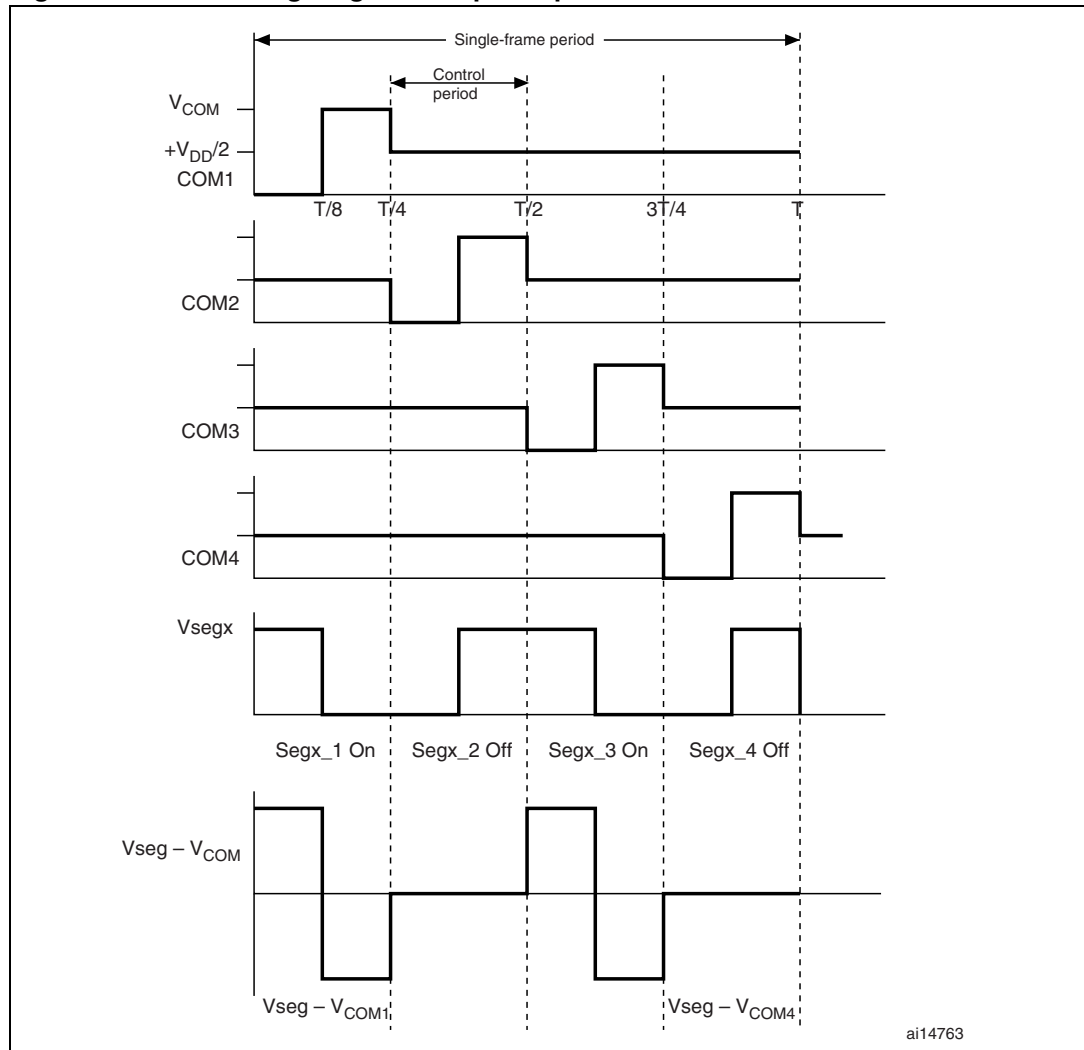
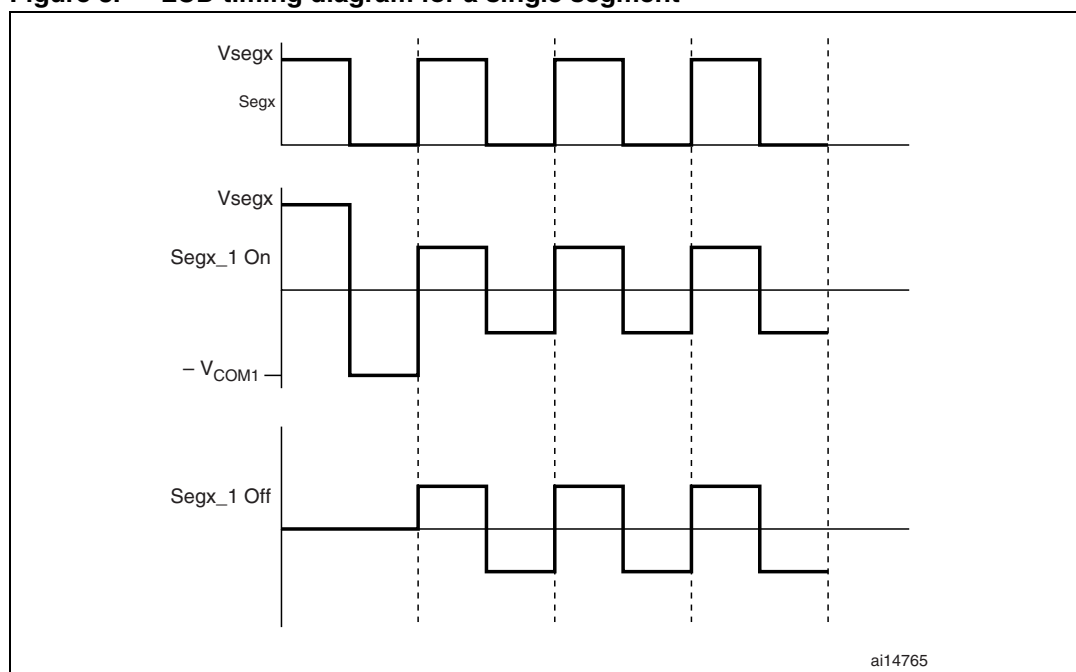


Figure 8. LCD timing diagram for a single segment



2.3.1 LCD mean voltage calculation

The LCD mean voltage must be very close to zero to guarantee long life to the LCD. The LCD mean voltage for On and Off periods can be calculated as shown below:

$$V_{\text{mean(On)}} = 1/8 V_{\text{seg}} + 1/8 (-V_{\text{COM}}) + 3(V_{\text{seg}} - V_r/2) + 3(-V_r/2) \quad (1)$$

$$V_{\text{mean(Off)}} = 3(V_{\text{seg}}/2) + 3(-V_r/2) \quad (2)$$

$V_{\text{mean(On)}}$ and $V_{\text{mean(Off)}}$ assume identical periods for each phase.

Equating equations (1) & (2) to zero, that is putting $V_{\text{mean(On)}} = 0$ and $V_{\text{mean(Off)}} = 0$, gives: $V_{\text{seg}} = V_{\text{COM}} = V_r = V_{\text{CC}}$, where:

- V_{COM} is the maximum voltage on the COM line
- $V_r/2$ is the voltage in the middle of the resistor bridge, applied on the COM line
- V_{seg} is the maximum voltage on the Segx line
- V_{CC} is the microcontroller power supply

2.3.2 Contrast calculation

Let the frame period be $T + xT$, where T is the active time and xT , the dead time. x is the proportion of dead time and V_x is the voltage during the dead time.

$$V_{\text{rms(ON)}} = \sqrt{\frac{1}{T + xT} \int_0^{(T + xT)} f(t)^2 dt}$$

$$V_{\text{rms(ON)}} = \sqrt{\frac{1}{T + xT} \left(\int_0^{\frac{T}{8}} (V_{\text{CC}})^2 dt + \int_{\frac{T}{8}}^{\frac{2T}{8}} (-V_{\text{CC}})^2 dt + \int_{\frac{2T}{8}}^T ((V_{\text{CC}})/2)^2 dt + \int_0^{\frac{xT}{8}} (V_x^2) dt \cdot 8 \right)}$$

$$V_{rms}(ON) = \sqrt{\frac{1}{T+xT} \left(V_{CC}^2 \cdot \frac{T}{8} + V_{CC}^2 \cdot \frac{T}{8} + \frac{V_{CC}^2}{4} \cdot \frac{6T}{8} + (Vx^2 \cdot \frac{xT}{8}) \cdot 8 \right)}$$

$$V_{rms}(ON) = \sqrt{\frac{1}{1+x} \left(\frac{(14V_{CC}^2)}{32} + (Vx)^2 \cdot x \right)}$$

Since $Vx = 0$ (in case of a decrease in V_{rms})

$$V_{rms}(ON) = \sqrt{\frac{1}{1+x} \cdot \frac{(14V_{CC}^2)}{32}}$$

$$V_{rms}(ON) = \sqrt{\frac{1}{1+x}} \cdot 0.661V_{CC}$$

$$V_{rms}(OFF) = \sqrt{\frac{1}{T+xT} \left(\int_0^{\frac{T}{8}} (0)dt + \int_{\frac{T}{8}}^{\frac{2T}{8}} (0)dt + \int_{\frac{2T}{8}}^T (V_{CC}/2)^2 dt + \int_0^{\frac{xT}{8}} (Vx^2)dt \cdot 8 \right)}$$

$$V_{rms}(OFF) = \sqrt{\frac{1}{1+x} \left(\frac{(6V_{CC}^2)}{32} + Vx^2 \cdot x \right)}$$

Since $Vx = 0$ (in case of a decrease in V_{rms})

$$V_{rms}(OFF) = \sqrt{\frac{1}{1+x}} \cdot 0.18V_{CC}$$

, where $Dx =$ Contrast calculation with contrast control method

$$\cdot 0.661V_{CC}$$

The contrast D , between V_{ON} and V_{OFF} is constant (quality of contrast). We only change the optical contrast by tuning V_{ON} close to the threshold value of the LCD.

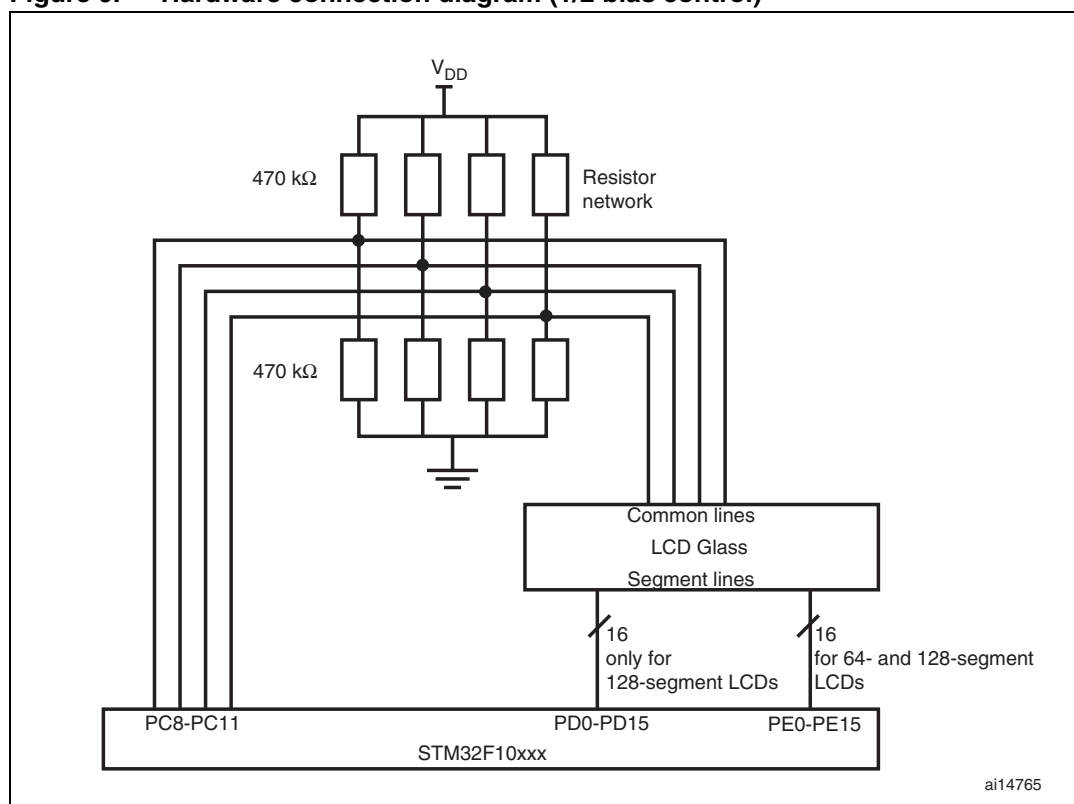
Note: *The LCD contrast must be controlled instantaneously by firmware depending on the external temperature and the supply voltage value. So the user has to measure both the ambient temperature using a temperature sensor and the microcontroller supply voltage using for example an ADC channel.*

3 Example of a quadruplex LCD with STM32F10xxx

The following example describes a drive for a quadruplex mode (4COM) LCD using the STM32F10xxx. Refer to [Figure 9](#). The only external components needed for driving the LCD are eight resistors of 470 k Ω each. To get the best DC voltage on the LCD, the user can use precision resistors to have a precise $V_{DD}/2$ when the GPIOs are in high impedance. The higher the resistor precision, the lower the DC value.

One I/O port per segment and one I/O port for each COM line are needed to drive the LCD. For example, to drive a quadruplex LCD that has 128 segments (with 32 segment lines and 4 COM lines) a total of 36 I/O ports only is required.

Figure 9. Hardware connection diagram (1/2 bias control)



The LCD timing is generated by a timer output-compare interrupt. Each cycle consists of four phases, one for each backplane. Each COM line generates its waveform during the corresponding phase, e.g. COM1 line during phase1. During other phases, it remains at level $V_{DD}/2$. Each phase consists of two parts:

1. Active time
2. Dead time

During the active time, segment lines and common lines are used to drive the LCD. During the dead time, Segment lines and common lines are used to tune the contrast.

Active time starts after the Output Compare 2 interrupt and dead time starts after the Output Compare 1 interrupt. A total of 16 interrupts are generated in each frame period with four

interrupts per control period. There are 2 Output Compare 2 events (CC2_1 and CC2_2) and 2 Output Compare 1 events in each phase. These are explained as follows:

- During CC2_1: V_{DD} is applied to the segments which have to be turned on and 0 is applied to the segments which have to be turned off. The Common line which corresponds to this phase is set to low level. Other Common lines are set to level $V_{DD}/2$.
- During CC1: to decrease V_{rms} , all segment lines and Common lines are inactive, that is set to low level (see Figure 10). To increase V_{rms} , Common lines are set low and segment lines are set to high level (see Figure 11).
- During CC2_2: Segment lines are supplied with voltage levels that are inverted compared to the one applied during CC2_1. The Common line that corresponds to this phase is set to high level. Other Common lines are set to $V_{DD}/2$.
- Again during CC1: to decrease V_{rms} , all Segments and Common lines are inactive, that is set to low level (see Figure 10). To increase V_{rms} , Common lines are set to high and Segments lines are set low level (see Figure 11).

Figure 10. LCD timing diagram with active and dead time (to decrease V_{rms})

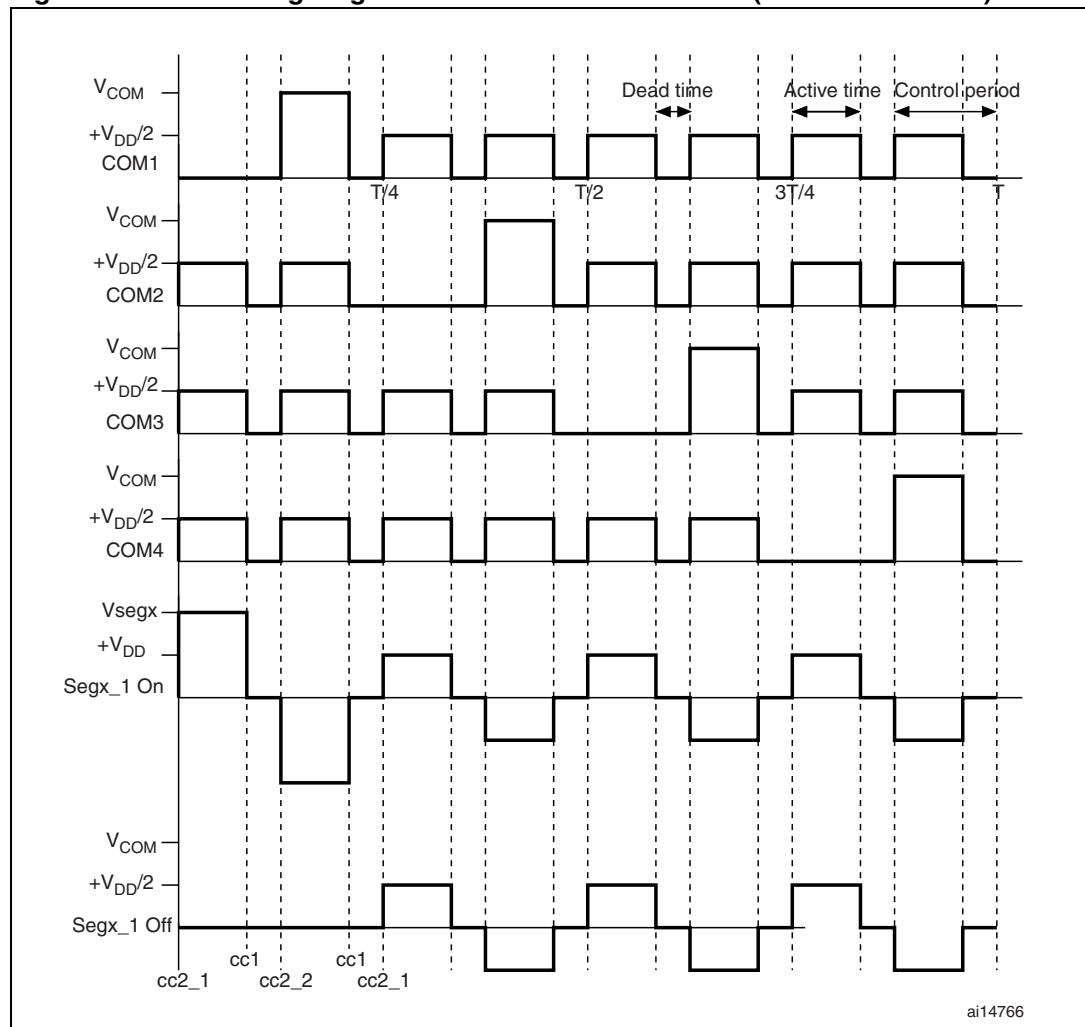


Figure 11. LCD timing diagram with active and dead time (to increase V_{rms})

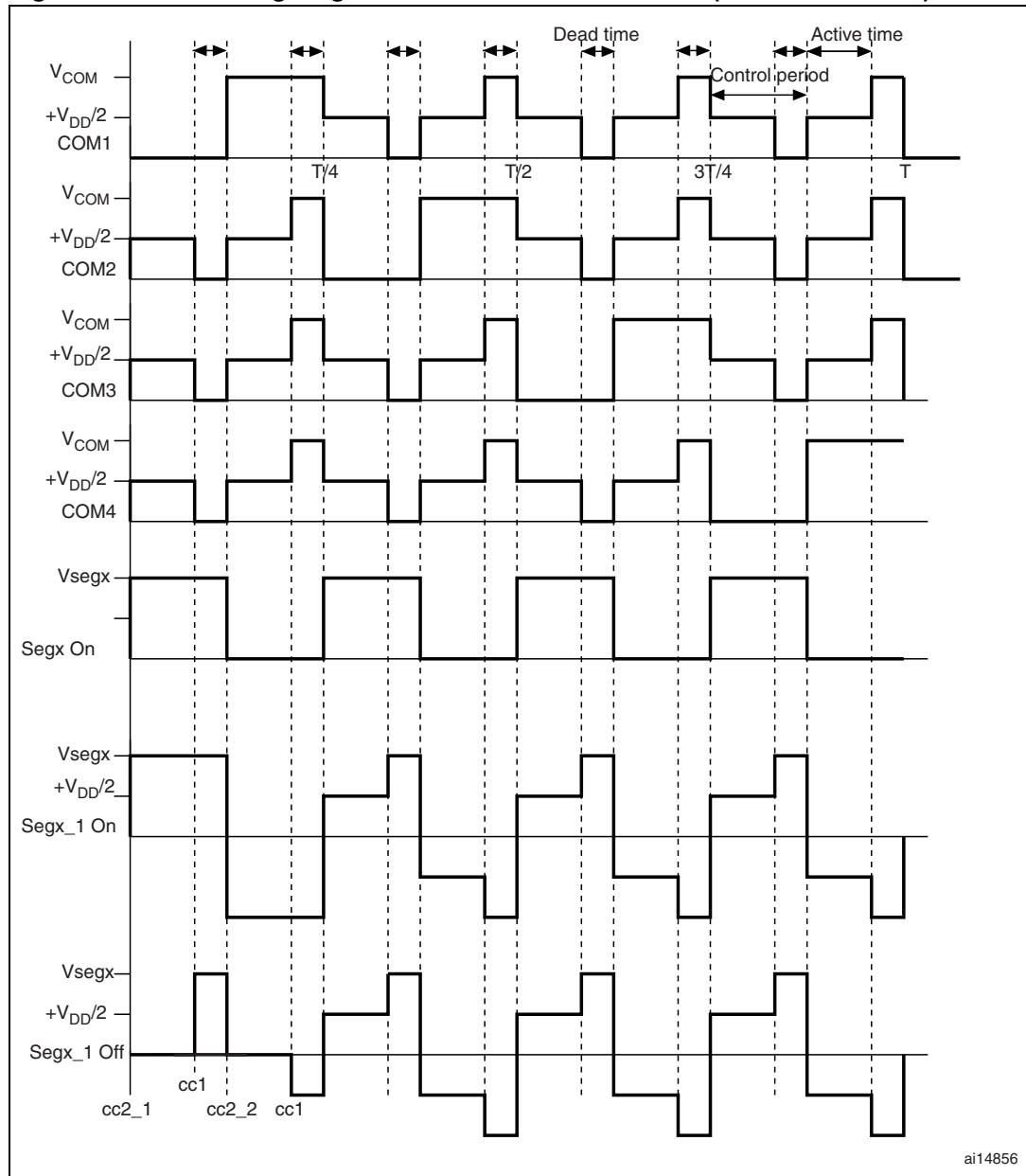


Figure 12. Screenshot of Vcom3 and Vsegx (Segment1 of digit2)

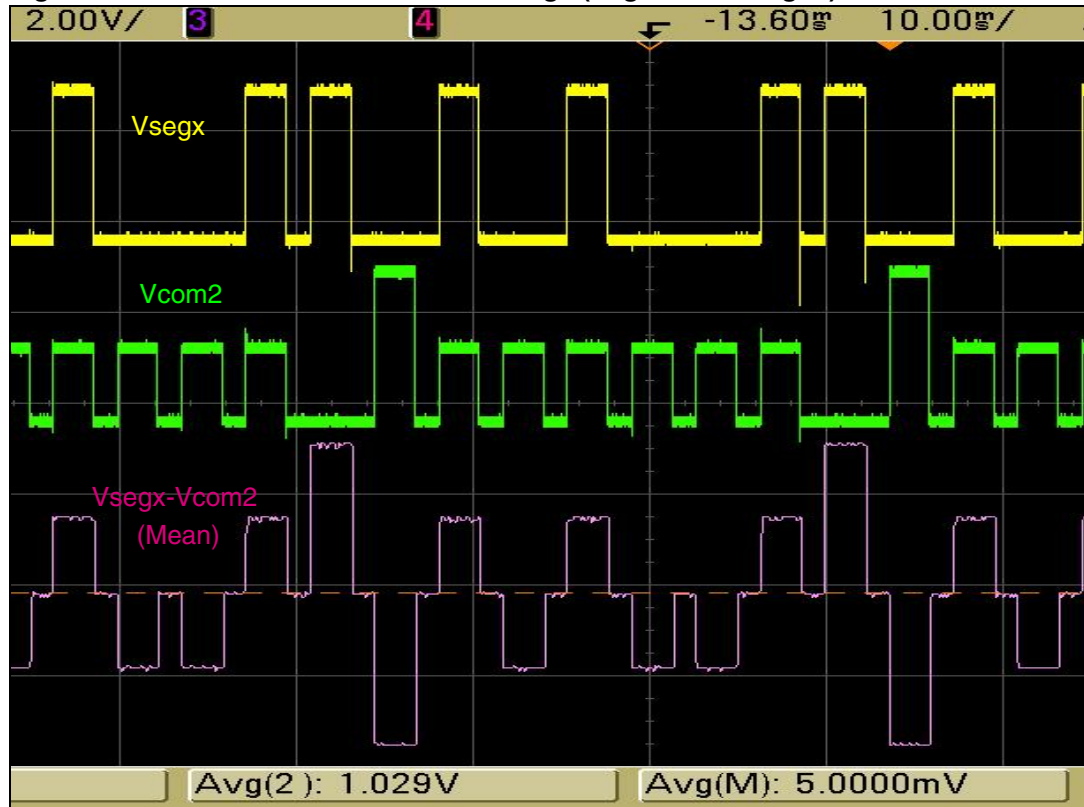
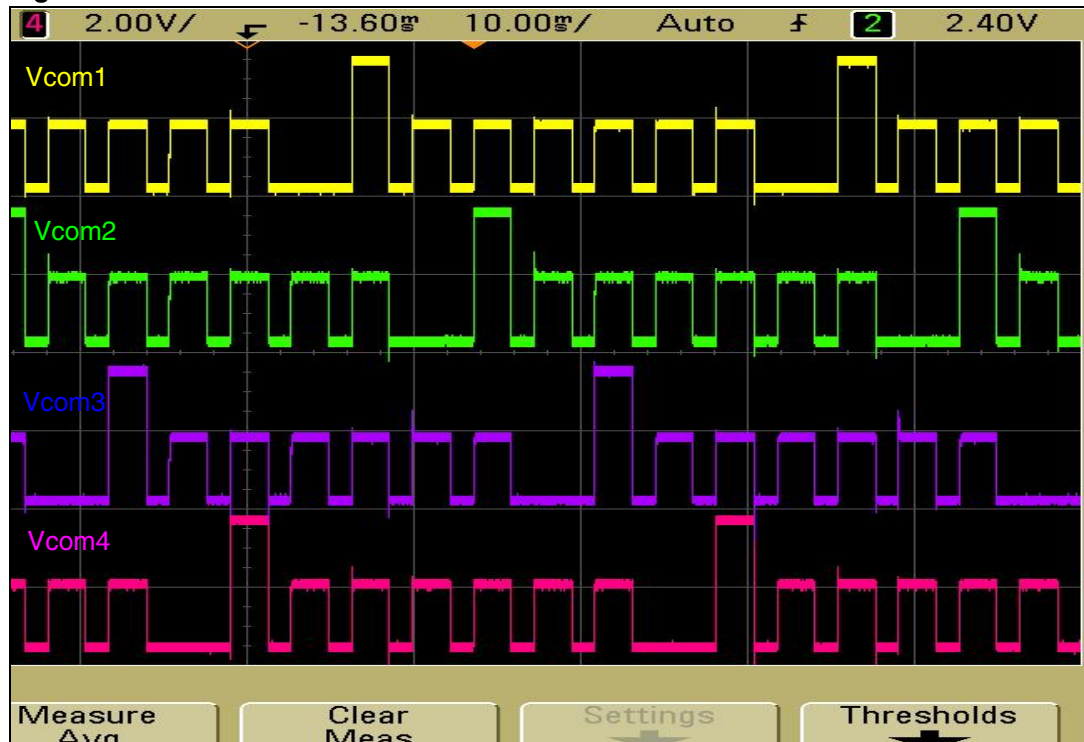


Figure 13. Screenshot of the four common lines



4 Firmware description

In the STM32F10xxx LCD demonstration, the PE0-PE15, PD0-PD15 port pins are connected to the 32 segment lines and are used to generate the segment signals. The PC8 to PC11 pins are connected to the 4 COM lines and used to generate the COM signals.

The LCD driver consists of five initialization routines (system clock configuration, NVIC configuration, GPIO initialization, timer configuration and EXTI configuration). To activate the LCD, these four initialization routines have to be called. After the routines have been called, the STM32F10xxx gets the timer Output Compare 1 (CC1) and Output Compare 2 (CC2) interrupts.

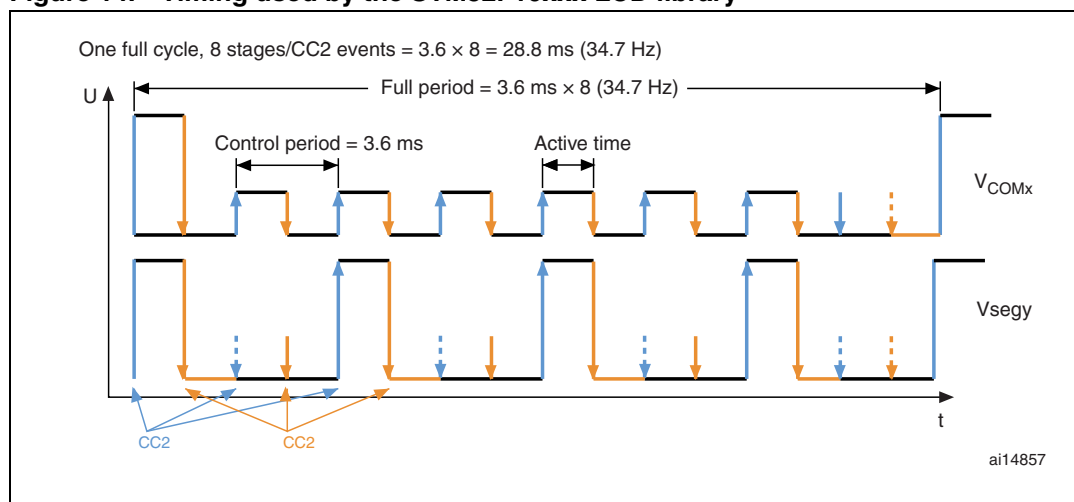
The ports are flexible, configurable by firmware, that is, the user can change the hardware implementation with minimum changes in firmware. This is done by changing the defines in the *glasslcd.h* file.

Example: the user can change the port to which the common lines are connected (port X), and its offset, according to PX0 (where X can be A, B, C, D or E). By default, in the firmware, common lines COM1 to COM4 are driven by PC8 to PC11, segment lines of lower quarter digits (Digit1 to Digit4) are connected to port E and segment lines of higher quarter digits (Digit5 to Digit8) are connected to port D.

The timer used by default in the firmware to drive the LCD is TIM3; it can be changed to another timer.

[Figure 14](#) presents the LCD timings used in the firmware. The LCD demo consists of displaying a message on LCD glass.

Figure 14. Timing used by the STM32F10xxx LCD library



In [Figure 14](#), the orange arrows represent CC1 events and the blue arrows represent CC2 events.

The timer interrupt routine is called every 3.6 ms by Output Compare 2 (CC2) to handle LCD segment switching.

The timer interrupt routine is called again after a delay specified by Output Compare 1 (CC1) to handle contrast control (to switch all voltages off for instance).

4.1 How to use the LCD library

The firmware is made to be flexible and configurable to adapt to the user's needs.

4.1.1 How to select the LCD glass by firmware

The implemented firmware can drive two reference types of glass LCD: 64- and 128-segment LCDs

- CT4-098: 64 segments (4 digits: 64 segments, manufacturer: ZHIYI Technology)
- PD 878: 128 segments (8 digits: 128 segments, manufacturer: Pacific Display)
- VIM 878: 128 segments (8 digits: 128 segments, manufacturer: Varitronix)

Note: PD 878 and VIM 878 LCDs are pin-to-pin compatible.

These LCDs are selectable by firmware by commenting and uncommenting the following lines in the *glasslcd.h* file:

```
#define USE_LCD_REF_PD_878
#define USE_LCD_REF_CT4_098
```

To use the first LCD (CT4-098), comment the `#define USE_LCD_REF_PD_878` line and uncomment `#define USE_LCD_REF_CT4_098`.

To use the second LCD (PD 878 or VIM 878), comment the `#define USE_LCD_REF_CT4_098` line and uncomment `#define USE_LCD_REF_PD_878`.

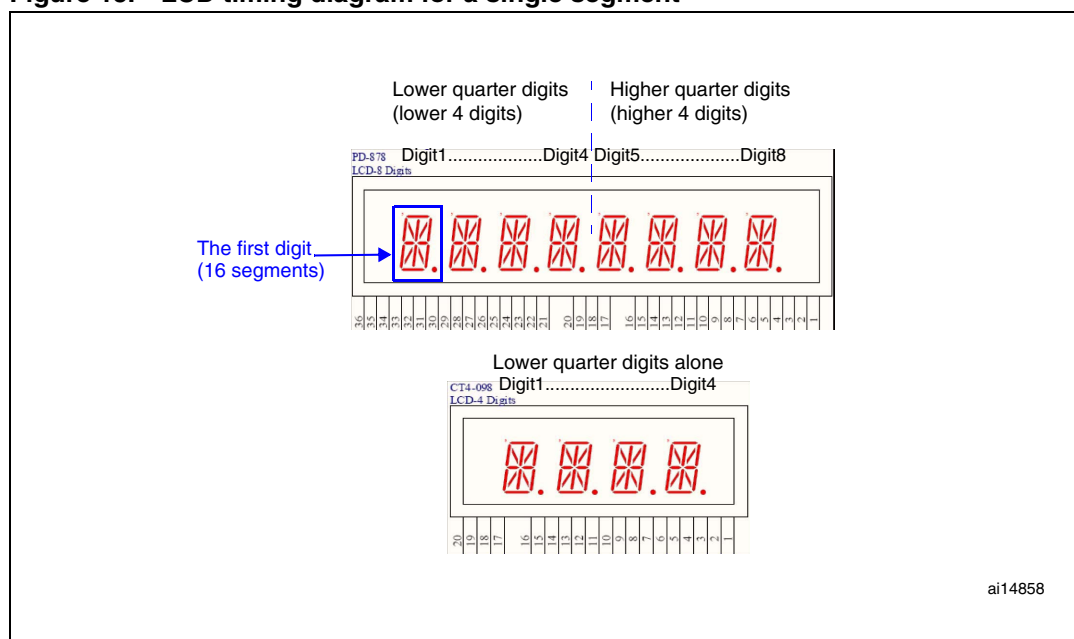
For each case, the letter and number map has to be initially selected in the preprocessor.

Note: The user has to comment one of the line and uncomment the other (select one type of LCD reference), otherwise the compiler generates an error.

4.1.2 How to select GPIOs to connect the LCD glass to STM32F10xxx

Each digit (16 segments) is driven by 4 common lines and 4 segment lines to form a matrix of 16 segments. So each digit occupies 4 GPIOs to drive the common lines and 4 others to drive the segment Lines. (common lines and segment lines are driven by different ports).

Figure 15. LCD timing diagram for a single segment



ai14858

Segment line configuration

(Refer to [Table 3](#) for more pinout details.)

CT4-098 has 16 segment lines and 4 common lines, so one port (16 GPIOs) is reserved for the segment lines and the common lines are connected to another GPIO port (refer to CT4-098 LCD Glass datasheet).

If PD-878 is selected, two ports are reserved for segment lines and the 4 common lines are connected to a third one (refer to PD-878 LCD Glass datasheet).

The configuration of segment lines is made by modifying the following lines:

```
#define SegmentsLines_LQD_Port    GPIOE
#define SegmentsLines_HQD_Port    GPIOD
```

The first line is used to select the port connected to the LCD segment lines of the first 4 digits (LQD: lower quarter digits), the second line is used to select the second 4 digits (HQD: higher quarter digits). Refer to [Figure 9](#).

In the configuration shown above port E drives digit 1 to digit 4 and port D drives digit 5 to digit 8.

Note: In the case where CT4-098 LCD Glass is selected, the second line has no effect.

Common line configuration:

The two LCDs have 4 common lines (COM1 to COM4). The firmware configuration is done in the glasslcd.h file using the three following lines:

```
#define CommonLines_Port          GPIOC
// #define CommonLines_EightHighPortPins
#define CommonLines_Pin_Offset    0
```

The first line defines the port to which common lines are connected (in the case shown above, the common lines are connected to port C, they cannot be connected to the same port as the segment lines).

The second line is used to determine to which eight port pins the common lines are connected.

- Lower eight port pins: PX0 to PX7
- Higher eight port pins: PX8 to PX15

So if `#define CommonLines_EightHighPortPins` is commented, COM1 to COM4 are connected to GPIOs with a range of 0 to 7. Otherwise (the line is uncommented), the common lines are connected to GPIOs with a range of 8 to 15.

The third line defines the offset pin regarding pin0 or pin8 (depending on whether the second line is commented or not). `CommonLines_Pin_Offset` value must not exceed 4 (0 to 4), otherwise the firmware does not work correctly.

In the configuration shown above, COM1 is connected to PC8, COM2 to PC9, COM3 to PC10 and COM4 to PC11.

If `#define CommonLines_EightHighPortPins` is uncommented, COM1 is connected to PC8, COM2 to PC9, COM3 to PC10 and COM4 to PC11.

The user has to configure the used GPIO APB clocks in the line below:

```
#define RCC_APB1Periph_Used_GPIO    RCC_APB2Periph_GPIOC \
                                     | RCC_APB2Periph_GPIOD \
                                     | RCC_APB2Periph_GPIOE
```

Timer selection

Three lines have to be configured by the user:

```
#define TIMER_LCD                    TIM3
#define TIMx_IRQChannel              TIM3_IRQChannel
#define RCC_APB1Periph_TIMx         RCC_APB1Periph_TIM3
```

The first line, determines which timer will drive the LCD. The second line determines which IRQ channel will be used to generate the interrupt. The third line configures the APB timer clock.

In the configurations shown above, TIM3 drives the LCD.

If the user wants to change the default timer, the 3 above mentioned lines have to be changed. Also, and all the code in the `TIM3_IRQHandler()` function has to be cut/pasted to the used timer IRQ handler. For example, if TIM2 is used, the code in `TIM3_IRQHandler()` must be moved into `TIM2_IRQHandler()`.

4.1.3 How to select the timer interrupt priority that drives the LCD

Refer to [Section 4.2](#) for more details.

4.1.4 LCD contrast control

The firmware contrast control is under pending patent from STMicroelectronics. The use of this technique with a non-STMicroelectronics microcontroller has to be agreed by STMicroelectronics.

LCD contrast is entirely controlled by firmware without the use of any external components. LCD contrast can be adjusted to the optimal value depending on the operating voltage of the LCD used. It is controlled by varying the dead phase timing as shown in the LCD timing diagram ([Figure 7](#) and [Figure 8](#)).

Dead time can be used to decrease or increase the rms voltage (V_{rms}) of the LCD. Dead time is the voltage compensation time to regulate the rms voltage up and down. Dead time can be implemented either after each control period or at the end of the frame. To avoid flickering, the dead time must be adjusted depending on the LCD quality and the frame frequency.

In the example shown in [Figure 10](#), the rms value of the LCD decreases when the dead time increases and rms value increases when the dead time decreases. In [Figure 11](#), it works in the opposite way.

The `LCD_SettingContrast()` function controls the LCD contrast that sets the channel2 pulse length.

Example

```
uint16_t ContrastValue = 0xE5C;
LCD_SettingContrast(ContrastValue);
ContrastValue has a range of 0x0000 to 0x1CB8.
```

The 0x0000 value corresponds to the maximum contrast: the LCD is completely dark (100% duty cycle).

The 0x1CB8 value corresponds to the minimum contrast: the LCD is completely transparent (0% duty cycle).

4.2 LCD timer interrupt

The LCD interrupt routine (or `TIMER_LCD`: the timer used to drive the LCD glass) is optimized to have the minimum DC voltage value. Direct access to registers is used to reduce the execution time and the jumping time arising from function calls.

4.2.1 LCD with high interrupt priority

In this configuration, the LCD always has the high interrupt priority (priority 0). It interrupts all other ongoing interrupts and no interrupt can interrupt an LCD interrupt.

To run the LCD in this configuration:

- Modify the value of the interrupt priority in the `glasslcd.h` file as follows:

```
#define LCD_Priority_Value 0 /* LCD priority value = 0 */
```
- Comment this line in the `glasslcd.h` file as follows:

```
// #define LCD_Use_Boost_Priority
```

Impact

In this case the Vseg and Vcom pulses are kept at the same width, so the DC voltage value is kept near 0 V (in the worst case 5 mV).

The PD-878 manufacturer specifies that the maximum allowed DC voltage is 50 mV. So here we have a DC voltage 10 times less than the value specified by the manufacturer.

4.2.2 LCD with low-high interrupt priority (using boosting priority)

When the LCD is configured in low-high interrupt priority, the LCD interrupt priority toggles between a low priority value and the highest priority value.

By default, the LCD interrupt is configured with the lowest priority value (configuration in main()). When the interrupt occurs, the LCD priority interrupt is boosted to have the highest priority (priority 0), so that no other interrupt can interrupt it. Before returning from the LCD interrupt, its priority returns to the previous lowest priority value defined by the user in the `glasslcd.h` file. This avoids a delay between the setting of the GPIO segment line and common line pins subsequent to decreasing the DC voltage value.

This boosting is performed using the priority boosting mechanism of the Cortex-M3 core on which the STM32F10xxx microcontroller is based.

To run the LCD in this configuration:

- Modify the value of the interrupt priority in the `glasslcd.h` file to the lowest priority value:

```
#define LCD_Priority_Value 6 /* LCD priority value = 6 */
```

The lowest user interrupt priority value in this case is 5. So the LCD interrupt priority value must be lower than the lowest user interrupt priority.

- Uncomment this line in the `glasslcd.h` file as follows:

```
#define LCD_Use_Boost_Priority
```

Impact

The user firmware is never interrupted by the LCD interrupt.

The DC voltage is higher than the DC value presented in the previous configuration (LCD in higher priority). This is due to user interrupts that have the highest priority and occur at the same time as the LCD interrupt (with lowest priority). So the LCD interrupt is a nested interrupt and is delayed until the user interrupt execution is finished. This causes an inequality between segment and com pulses (Vseg and Vcom pluses are not kept at the same width), consequently the DC voltage is increased in accordance with the number of user interrupts and their execution time.

So the higher the number of user interrupts that postpone the LCD execution, the higher the DC voltage. The higher the execution time of the user interrupt that postpones the LCD execution, the higher the DC voltage.

In this configuration, the measured DC value is 18 mV (depending on the number of user interrupts that postpone the LCD interrupt, and their execution time).

The PD-878 manufacturer specifies that the maximum allowed DC voltage is 50 mV. Here, we have a DC voltage 2.5 times less than the value specified by the manufacturer.

4.2.3 LCD interrupt always with lowest priority

The LCD interrupt always has the lowest priority. All user interrupts can interrupt the LCD execution, and the LCD interrupt cannot interrupt any other interrupt.

To run the LCD in this configuration:

- Modify the value of the interrupt priority in the `glasslcd.h` file to the lowest priority:

```
#define LCD_Priority_Value 6 /* LCD priority value = 6 */
```

The lowest user interrupt priority value in this case is 5. So the LCD interrupt priority value must be lower than the lowest user interrupt priority.
- Comment this line in `glasslcd.h` file as follows:

```
// #define LCD_Use_Boost_Priority
```

Impact

The DC voltage is higher than the DC value presented in the two previous configurations (LCD with higher priority and boosted LCD priority).

This is due to:

- The user interrupts that interrupt the LCD routine between the settings of segment line and common line GPIOs. An undesirable jitter appears on the Vseg-Vcom signals, and its average is added to the DC value of Vseg-Vcom, consequently the DC value is increased.
- The user interrupts that have highest priority and occur at the same time as the LCD interrupt (with lowest priority). The LCD interrupt is a nested interrupt and is delayed until user interrupt execution is finished. This causes the inequality of the segment line and common line pulses, consequently the DC voltage is increased according to the number of user interrupts and their execution time.

In this configuration, we can measure a DC value of about 50 mV (depending on the number of user interrupts that delay or interrupt the LCD interrupt and their execution time).

4.3 Default demo configurations

- The firmware runs from a CPU with a 2 MHz frequency.
- The used LCD is PD-878 (or VIM 878)
- Segment lines of the lower quarter digits are connected to port E
- Segment lines of the higher quarter digits are connected to port D
- Common lines are connected to port C
- The four common lines are connected as follows: COM1 to PC8, COM2 to PC9, COM3 to PC10 and COM4 to PC11
- TIM3 drives the LCD glass
- LCD priority value = 3
- Boost priority is used (at entry LCD interrupt the priority = 0, at exit the interrupt routine priority returns to 3)

When the user runs the demonstration, the “STM32” message is displayed on the LCD glass.

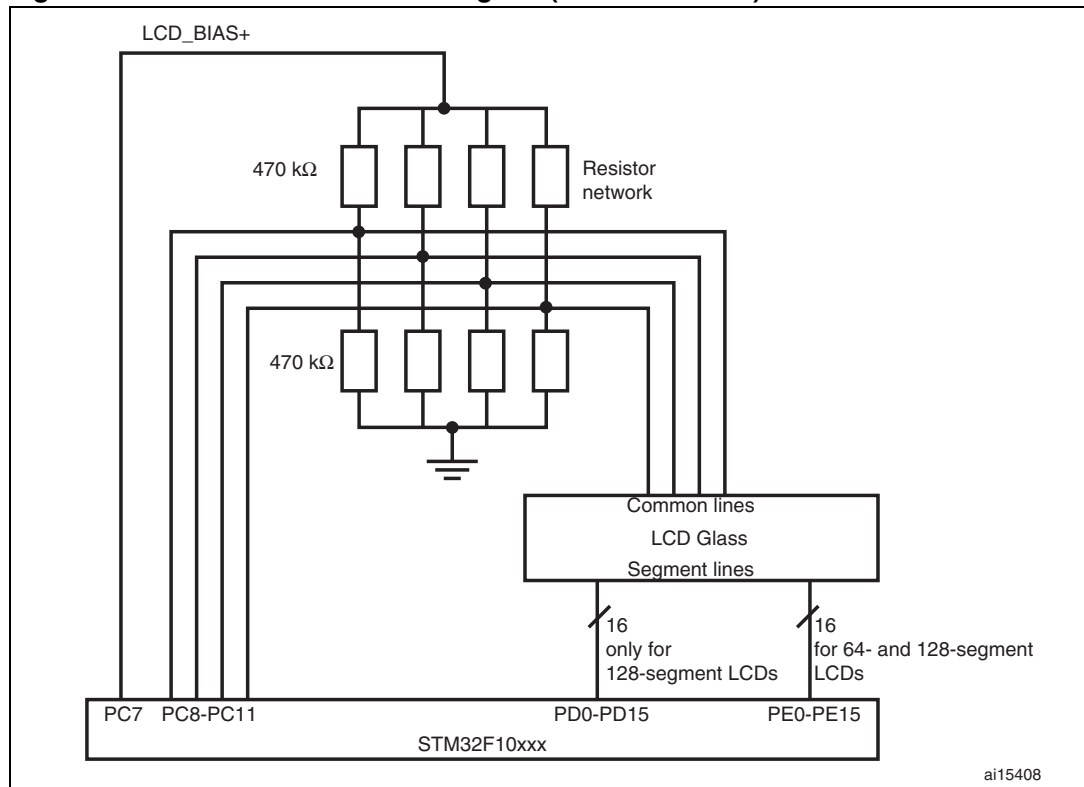
5 LCD glass in battery-powered and low-power applications

This section describes how to manage the LCD glass in low-power designs like battery-powered applications. It is divided into two parts. The first part describes, the timer method for driving the LCD glass using the STM32F10xxx Sleep mode. The second part describes how to drive the LCD glass using the RTC feature combined with the STM32F10xxx Stop mode to have minimum power consumption.

5.1 Hardware implementation

The hardware is the same as the one described in [Section 3](#), except for an additional line called LCD_Bias+, that is added to power off the LCD, and so save power, when the application enters a low-power mode. All segments and all common lines are set to 0 V, thus preventing any Leakage current on the GPIOs.

Figure 16. Hardware connection diagram (1/2 bias control)



LCD bias pin configuration by software

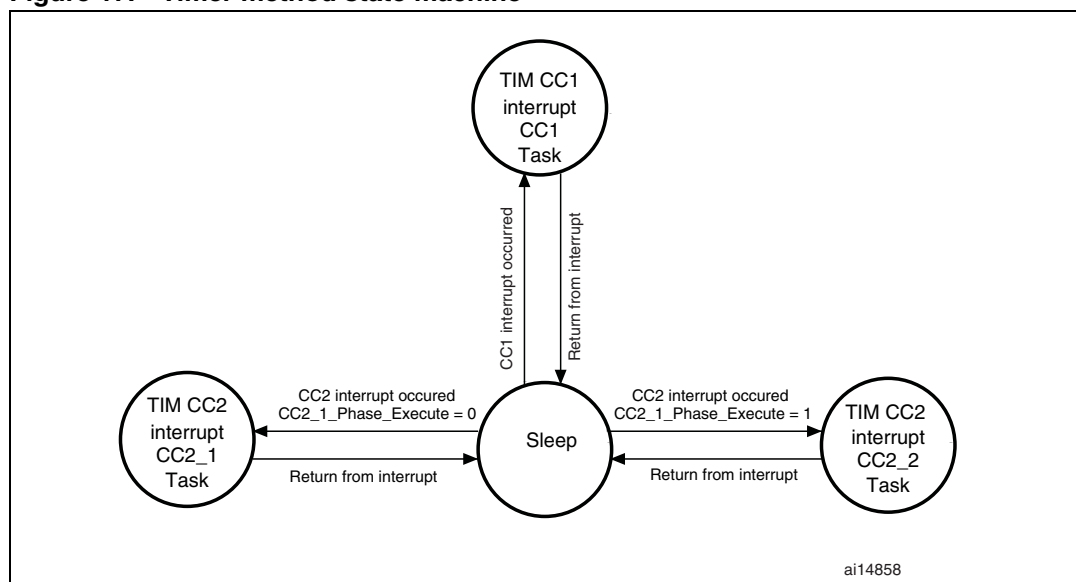
```
#define LCD_Bias_Port      GPIOC
#define LCD_BiasPlus_Pin  GPIO_Pin_7
```

The first line defines the port to which the LCD_BIAS+ pin is connected to power on/off the LCD glass. The second line defines the GPIO pin to be connected to LCD_BIAS+.

5.2 LCD glass managed by the timer using the Sleep low-power mode

As described previously, the timer method is based on the output compare interrupt (CC1 and CC2). CC2 is divided into two tasks called CC2_1 and CC2_2, which are managed (enabled) by a software flag (CC2_1_Phase_Execute). *Figure 17* shows the timer method state machine that explains the interactions of the different tasks.

Figure 17. Timer method state machine



5.2.1 Some current and CPU load measurements

Some current measurements are provided in *Table 1*. These measurements correspond to the (microcontroller + resistor network + LCD glass) consumption.

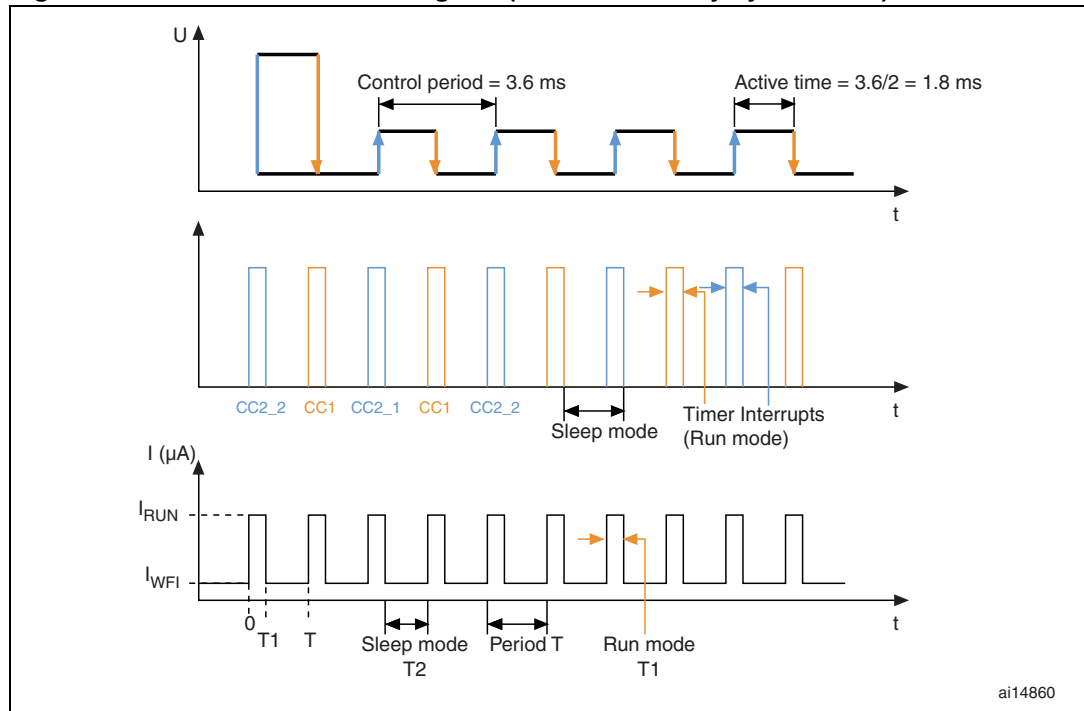
The measurement conditions are:

- Hardware environment: STM32F103xx is used, LCD ref: PD-878, HSI (internal RC) clock is used, power supply = 3 V, HCLK = PCLK1 = PCLK2
- Tool chain: IAR 4.42, configuration: high optimization on size, Inline threshold = 16.

Table 1. Current consumption vs. frequency

CPU load/ Consumption		HCLK frequency		
		1 MHz	2 MHz	4 MHz
Execution from Flash	CPU load	4.68%	2.33%	1.18%
	Consumption	629 μA	742 μA	882 μA
Execution from RAM	CPU load	5.91%	3.39%	1.57%
	Consumption	584 μA	676 μA	866 μA

Figure 18. Timer method chronogram (in this case duty cycle = 50%)



Period T is the sum of the interrupt execution time and the Sleep time:

$$T = T1 + T2$$

Mean current I_m can be calculated as shown below:

$$I_m = \frac{1}{T} \int_0^T i(t)dt = \frac{1}{T} \int_0^{T1} I_{RUN} dt + \frac{1}{T} \int_{T1}^T I_{Sleep} dt = \frac{T1}{T} I_{RUN} + \frac{T - T1}{T} I_{Sleep} = \frac{T1}{T} I_{Run} + \frac{T2}{T} I_{Sleep}$$

5.3 LCD glass managed by the RTC and using the Stop low-power mode

The RTC method is based on RTC Alarm interrupt and software flags. There are three tasks Task1, Task2 and Task3 instead of CC1, CC2_1 and CC2_2, respectively, in the timer method.

The RTC alarm register is updated each time Task1, Task2 or Task3 is executed, to have the timings needed to drive the LCD glass.

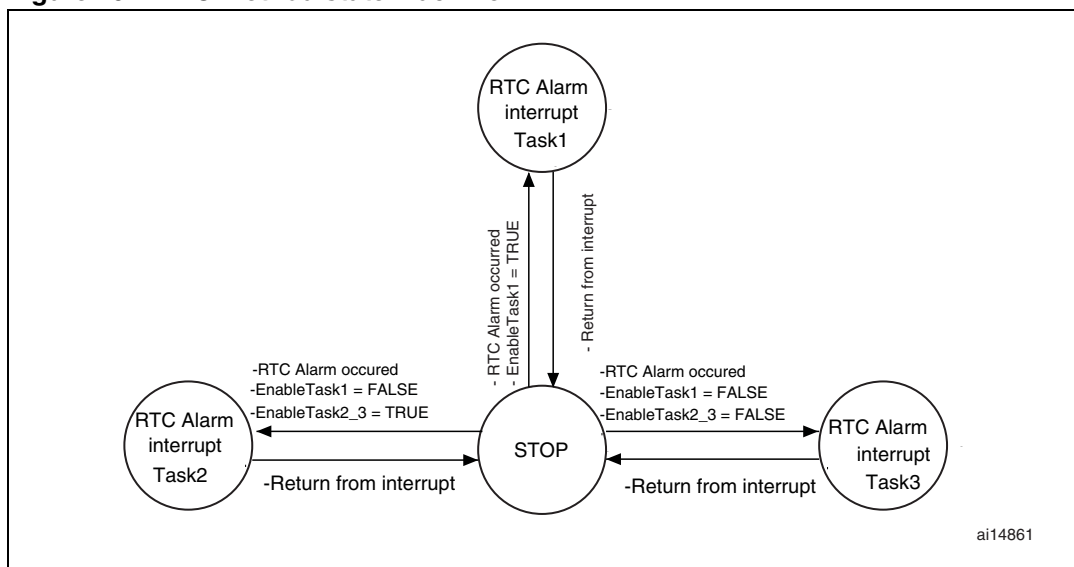
The RTC Alarm interrupt has the capability to wake up the core from the Stop mode (EXTI line 17).

Figure 19 shows how the RTC method works and how the tasks are organized to generate the adequate signals to drive the LCD glass using the Stop mode to minimize power consumption. The sequence of the three tasks is managed by two software flags: EnableTask1 and EnableTask2_3.

The contrast is defined in the *glasslcd_RTC.h* file and can be modified by the user (the value is in percent):

```
#define LCDContrastValue 50 /* Set LCD contrast to 50% */
```

Figure 19. RTC method state machine



5.3.1 Some current and CPU load measurements

Some current measurements are presented in [Table 2](#). They correspond to the (microcontroller + resistor network + LCD glass) consumption.

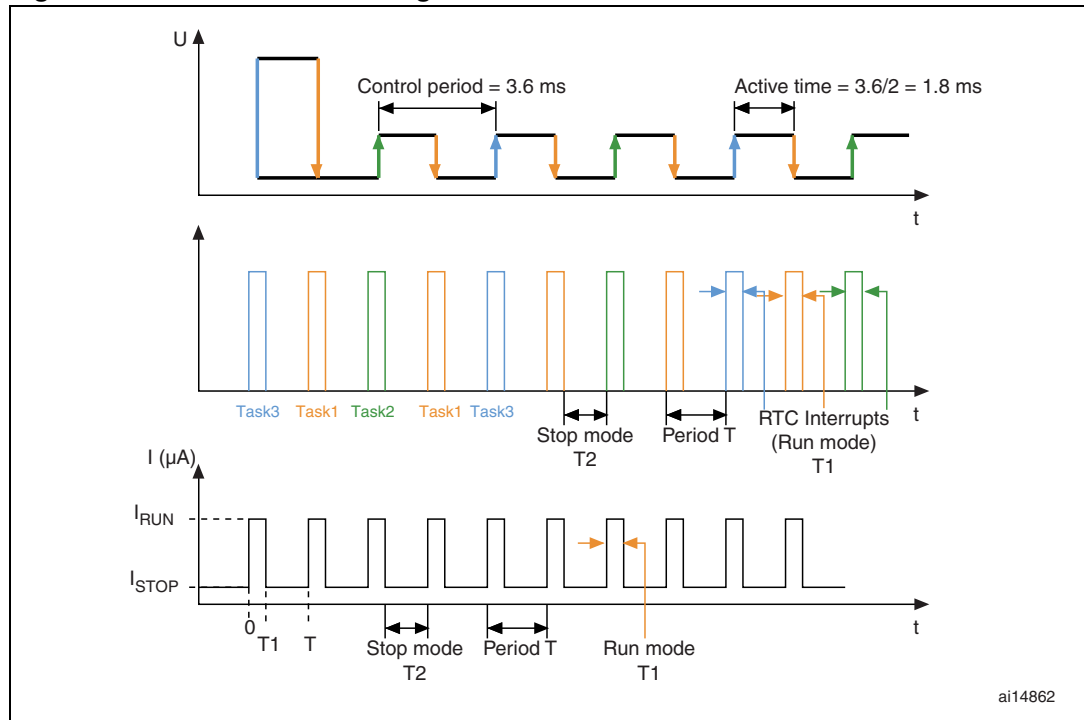
The measurement conditions are:

- Hardware environment: STM32F103xx is used, LCD ref: PD-878, HSI (high-speed internal RC) clock is used, LSI (low-speed internal) clock is used, power supply = 3 V, HCLK = PCLK1 = PCLK2
- Tool chain: IAR 4.42, configuration: high optimization on size, Inline threshold = 16.

Table 2. Current consumption vs frequency

CPU load/consumption		HCLK frequency		
		1 MHz	2 MHz	4 MHz
Flash Execution	CPU load	17.55%	13.42%	10.52%
	Consumption	280 μ A	258 μ A	290 μ A
RAM Execution	CPU load	20.31%	15.02%	10.63%
	Consumption	193 μ A	174 μ A	178 μ A

Figure 20. RTC method chronogram



Period T is the sum of the interrupt execution time and the Stop time:

$$T = T1 + T2$$

Mean current I_m is expressed by the following equation

$$I_m = \frac{1}{T} \int_0^T i(t) dt = \frac{T1}{T} I_{Run} + \frac{T2}{T} I_{STOP}$$

6 Comparison of the timer and RTC methods

6.1 Timer method

- One timer is used (two Output Compare): the two remaining Output Compare are free and can be used for another application.
- Very low CPU load: 4.6% at 1 MHz

However:

- The timer configuration depends on the APB1 clock configuration
- In low-power mode, the only supported mode is Sleep mode

6.2 RTC method

- Stop mode supported: minimum power consumption at 2 MHz ~ 174 μ A typical
- RTC configuration independent of the APB clock configuration.

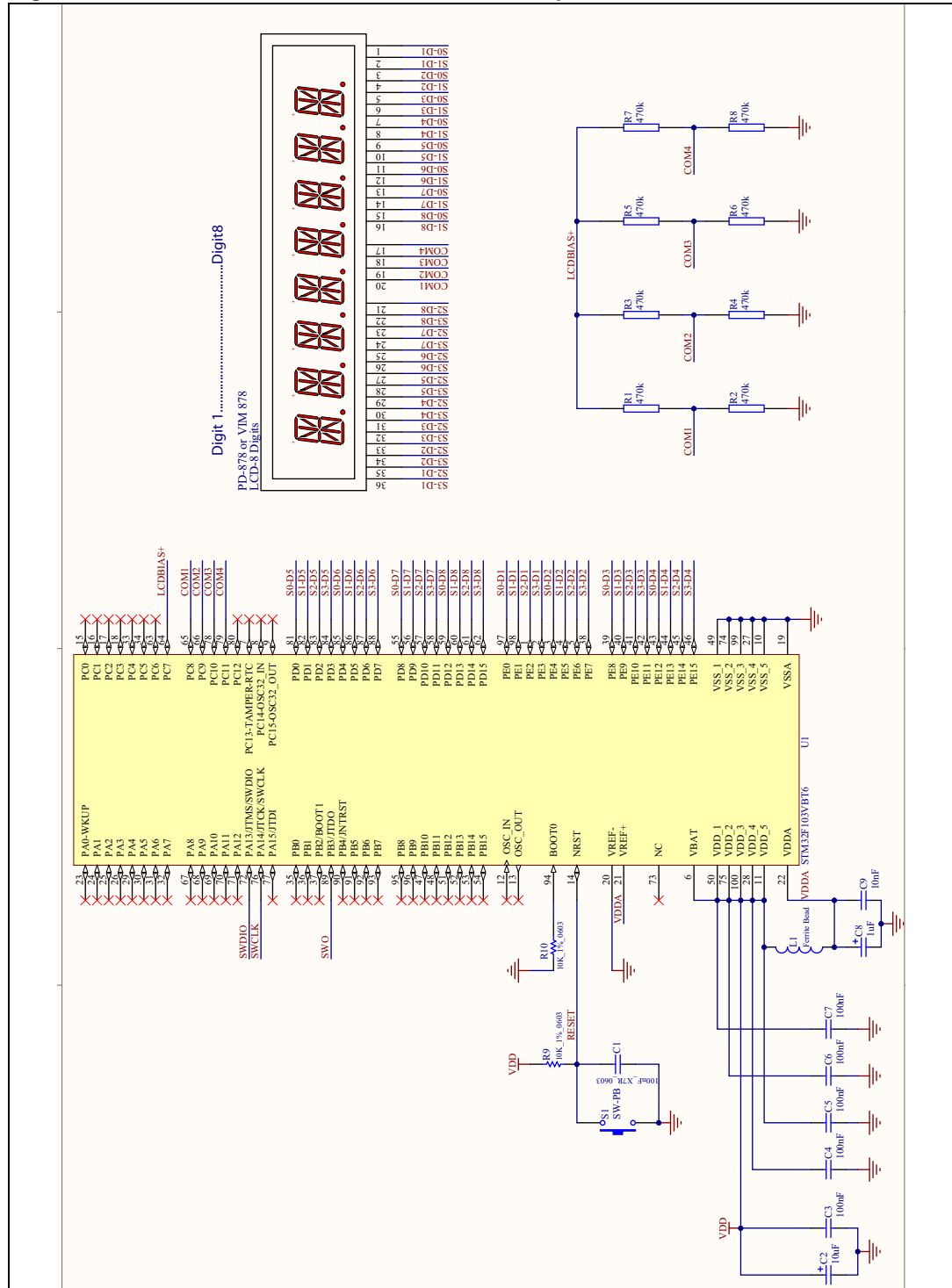
However:

- RTC is not available for the calendar feature.
- More instructions are required to execute in RTC Alarm interrupt (enter the configuration mode, poll on the RTOFF flag, etc.): higher CPU load.
- Some peripherals do not support the Stop mode to wake up the core: additional software is required to manage this feature.

7 Hardware implementation

7.1 Schematic

Figure 21. LCD - STM32F10xxx connection example



7.2 LCD segment line connections

Table 3. LCD segment line mapping

Digit	Segment line ⁽¹⁾	LCD pin number	
		CT4-098	PD-878
LCD reference ->		CT4-098	PD-878
1	S0 (PX.0)	1	1
	S1 (PX.1)	2	2
	S2 (PX.2)	3	35
	S3 (PX.3)	4	36
2	S0 (PX.4)	5	3
	S1 (PX.5)	6	4
	S2 (PX.6)	7	33
	S3 (PX.7)	8	34
3	S0 (PX.8)	9	5
	S1 (PX.9)	10	6
	S2 (PX.10)	11	31
	S3 (PX.11)	12	32
4	S0 (PX.12)	13	7
	S1 (PX.13)	14	8
	S2 (PX.14)	15	29
	S3 (PX.15)	16	30
5	S0 (PY.0)	-	9
	S1 (PY.1)	-	10
	S2 (PY.2)	-	27
	S3 (PY.3)	-	28
6	S0 (PY.4)	-	11
	S1 (PY.5)	-	12
	S2 (PY.6)	-	25
	S3 (PY.7)	-	26
7	S0 (PY.8)	-	13
	S1 (PY.9)	-	14
	S2 (PY.10)	-	23
	S3 (PY.11)	-	24
8	S0 (PY.12)	-	15
	S1 (PY.13)	-	16
	S2 (PY.14)	-	21
	S3 (PY.15)	-	22

1. X and Y represent the ports to which the segment lines are connected, X is different from Y, and X,Y can be: A,B,C,D or E.

8 Revision history

Table 4. Document revision history

Date	Revision	Changes
22-Jul-2008	1	Initial release.
30-Apr-2009	2	Example modified in Section 4.1.4: LCD contrast control . Titles of Figure 18 and Figure 19 modified.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2009 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com