
PWM resolution enhancement through a dithering technique for STM32 advanced-configuration, general-purpose and lite timers

Introduction

Nowadays power-switching electronics exhibit remarkable performance enhancements, and switching frequencies are constantly increasing to meet the requirements of modern power-conversion systems. This presents increasing challenges in the field of control techniques.

Digitally-controlled Pulse Width Modulation (PWM) generators make a trade-off between switching frequency and duty-cycle fine tuning. Achieving both high resolution and high switching frequency implies that the control circuitries operate at high frequencies.

This application note presents a dithering technique that enhances the PWM resolution while keeping the same switching frequency and operating frequency for the control circuitry.

The PWM dithering technique demonstration covered in this application note uses one of the STM32 16-bit general purpose timers. The implementation and the associated X-CUBE-PWM-DITHR software expansion for STM32Cube both target the NUCLEO-F302R8 Nucleo board, but can easily be tailored to any of the STM32 MCUs, regardless of the hardware development environment.

Contents

- 1 PWM dithering technique concept 5**
 - 1.1 Trade-off between switching frequency and PWM resolution 5
 - 1.2 PWM dithering 5
 - 1.2.1 Introduction 5
 - 1.2.2 Concept 5
 - 1.2.3 N-added-resolution-bit PWM dithering 7

- 2 PWM dithering technique implementation for STM32 timers 9**
 - 2.1 STM32 timers features for PWM dithering implementation 9
 - 2.1.1 STM32 timer time-base 9
 - 2.1.2 STM32 timer channels 9
 - 2.1.3 Channel register preload feature 10
 - 2.1.4 Timer DMA burst feature 10
 - 2.2 System-level features and peripherals used in PWM dithering implementation 11
 - 2.2.1 Timer update-event-triggered DMA transfer 11
 - 2.2.2 DMA half-transfer interrupt 11
 - 2.3 PWM dithering implementation using STM32 MCUs 11

- 3 Demonstration 14**
 - 3.1 Demonstration hardware environment 14
 - 3.2 Demonstration firmware architecture 15
 - 3.2.1 Utilization of the microcontroller hardware resources 15
 - 3.2.2 Demonstration firmware description through flowcharts 16
 - 3.2.3 Firmware Footprint 17
 - 3.3 Experimental results 18
 - 3.3.1 Dithering effect on PWM resolution enhancement 18
 - 3.3.2 Low-pass filter dimensioning considerations 19

- 4 References 20**

- 5 Revision history 21**

List of tables

Table 1.	PWM frequency versus PWM Resolution	5
Table 2.	Possible patterns for 1-bit PWM dithering	7
Table 3.	PWM dithering pattern for generating 1/2 LSB dither effect (highest ripple magnitude).	8
Table 4.	PWM dithering pattern for generating 1/2 LSB dither effect (lowest ripple magnitude)	8
Table 5.	PWM dithering patterns, 3-added-resolution-bit PWM (lowest output ripple)	8
Table 6.	Firmware Footprint	18
Table 7.	Document revision history	21

List of figures

Figure 1.	Duty-cycle adjustment for PWM dithering technique	7
Figure 2.	STM32 timers PWM modes	10
Figure 3.	PWM dithering technique implementation bloc diagram	13
Figure 4.	STM32F3 Nucleo board setup	15
Figure 5.	Application's main function	16
Figure 6.	Peripherals initialization flowchart	17
Figure 7.	DMA interrupt handler flowchart	17
Figure 8.	Dithering effect applied on the rising slope of the triangular waveform	18
Figure 9.	Low-pass filter cut-off frequency effect on the dithering technique efficiency	19

1 PWM dithering technique concept

1.1 Trade-off between switching frequency and PWM resolution

There is a tight relationship between the switching frequency which is equivalent to the PWM frequency and the PWM resolution. Increasing the PWM resolution implies to decrease the PWM frequency while maintaining a constant timer clock frequency. For a general purpose STM32 timer, the minimal clock frequency required to clock the timer to achieve a given PWM frequency and resolution is given by the formula below:

$$\text{Timer clock} = \text{PWM frequency} \times 2^{\text{PWM resolution}}$$

[Table 1](#) provides some examples based on the above formula.

Table 1. PWM frequency versus PWM Resolution

STM32 16-bit timer	PWM resolution	PWM frequency
72 MHz	16 bit	~1.1 kHz
72 MHz	14 bit	~4.4 kHz
72 MHz	12 bit	~17.5 kHz
72 MHz	10 bit	~70 kHz
72 MHz	8 bit	~281 kHz
72 MHz	6 bit	~1.125 MHz
72 MHz	4 bit	~4.5 MHz

1.2 PWM dithering

1.2.1 Introduction

To overcome the trade-off presented above, the PWM dithering technique proposes to enhance the PWM resolution while maintaining a constant PWM frequency and without any need to increase the timer clock frequency.

Theoretically, any desired number of added resolution bits is possible. Practically, the added resolution bits are limited to a few ones, in particular due to the firmware footprint exponential increase (See [Section 3.2.3: Firmware Footprint](#)).

1.2.2 Concept

The PWM dithering is done by making the PWM duty-cycle not constant any more. The idea is to adjust the duty-cycle by one LSB with a repetitive pattern over a given number of consecutive PWM periods. An external low-pass filter is used to cut the extra switching

components and the result consists in an added DC offset with a resolution less than one duty cycle LSB. The duty-cycle adjustment can be made following two possible approaches.

- The first approach consists in subtracting one LSB from the periods where a duty-cycle adjustment should be made. In this case the DC offset is going to be negative.
- The other way is by adding one LSB to the periods where a duty-cycle adjustment should be made. In this case the DC offset is going to be positive.

The added DC offset is proportional to the ratio between the number of PWM periods where duty-cycle adjustment is made, divided by the number of consecutive PWM periods making one PWM dithering pattern. Whatever the duty-cycle adjustment pattern, the obtained ratio is always a positive sub-one fractional value. With this dithering technique, external devices can be controlled with a resolution higher than the original PWM resolution.

The resulting PWM resolution is given by the below formula:

$$PWM_{Effective_Resolution} = PWM_{Resolution} + PWM_{Dither_Resolution}$$

$$PWM_{Resolution} = \text{Timer clock frequency} / \text{PWM frequency}$$

The $PWM_{Dither_Resolution}$ is given by the below formula

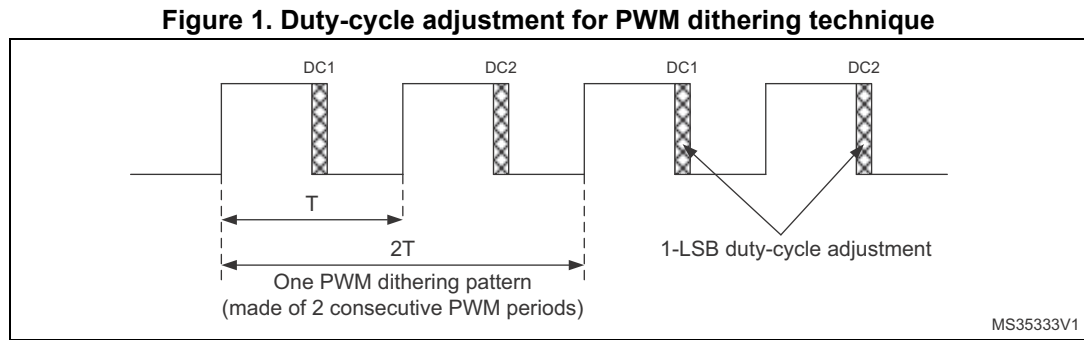
$$N_{PWM_Adjustment_Periods} = 2^{PWM_{Dither_Resolution}}$$

where $N_{PWM_Adjustment_Periods}$ is the number of PWM periods required to constitute one duty-cycle adjustment pattern (also called, PWM dithering pattern).

The required number of periods for a complete duty-cycle adjustment pattern is a power-of-2 of the number of resolution bits, added with the dithering technique.

Just for simplicity sake, the below example is given for 1-bit enhanced PWM resolution. The concept remains the same for larger numbers of added resolution bits.

Within this example we want to add only one extra bit of resolution using the dithering technique. The duty-cycle adjustment pattern will occupy 2 PWM periods. A sequence of PWM periods is presented in [Figure 1](#) below. The PWM dithering technique consists in adjusting the duty-cycle of two consecutive PWM periods. The pattern could be the same for the next two PWM periods to maintain the same output value or it can be changed to a new one.



For 1-added-resolution-bit PWM dithering, four patterns are possible. These patterns are listed in [Table 2](#) below.

Table 2. Possible patterns for 1-bit PWM dithering

DC1	DC2	$(DC1 + DC2)/2$
0 LSB	0 LSB	0 LSB ⁽¹⁾
0 LSB	1 LSB	$\frac{1}{2}$ LSB ⁽²⁾
1 LSB	0 LSB	$\frac{1}{2}$ LSB ⁽²⁾
1 LSB	1 LSB	1 LSB ⁽¹⁾

1. Not of interest because achievable by hardware.
2. PWM dithering effect: for 1-bit PWM dithering, the fine tuning step is $\frac{1}{2}$ LSB.

The first and the last patterns give a fine tuning of one full LSB but this is already achievable by hardware. The second and the third patterns are the ones targeted by the PWM dithering technique. These patterns allow to control the PWM resolution by a smaller step ($\frac{1}{2}$ LSB). For 1-added-resolution-bit PWM dithering, the two dithering-effect patterns are similar. Any one of them can be used for achieving the $\frac{1}{2}$ LSB fine-tuning step.

1.2.3 N-added-resolution-bit PWM dithering

For larger numbers of added resolution bits, the same concept as described above remains applicable. For N-added-resolution-bit PWM dithering, 2^N patterns are possible. As it is the case for the 1-added-resolution-bit PWM dithering, the first and the last patterns (those which have a fine tuning step of 1 LSB) are not of interest. All the remaining patterns present the dithering effect. When choosing which pattern to use, two considerations must be taken into account:

- The first one, which is intuitive, is the desired dither value.
- The second one is to choose the pattern that generates as low as possible ripple while keeping the required dither effect.

Let's consider a 3-added-resolution-bit PWM dithering example. In this case, each PWM dithering pattern is composed of $2^3 = 8$ PWM periods and the fine tuning step is $\frac{1}{8}$ LSB. Let's assume the desired average dither effect should be $\frac{1}{2}$ LSB. [Table 3](#) and [Table 4](#) provide such pattern examples producing a $\frac{1}{2}$ LSB fine-tuning. The patterns in [Table 3](#) are characterized by a high ripple magnitude whereas the pattern in [Table 4](#) is showing a minimal output ripple.

Table 3. PWM dithering pattern for generating 1/2 LSB dither effect (highest ripple magnitude)

Average value	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
4/8 = 1/2 LSB	1	1	1	1	0	0	0	0
4/8 = 1/2 LSB	0	0	0	0	1	1	1	1

Table 4. PWM dithering pattern for generating 1/2 LSB dither effect (lowest ripple magnitude)

Average value	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
4/8 = 1/2 LSB	1	0	1	0	1	0	1	0

Carefully selecting the PWM dithering pattern helps reducing the output ripple induced by the PWM dithering technique. Nevertheless, a slight degradation of the output ripple margin is unavoidable. [Table 5](#) lists the PWM dithering patterns that produce the lowest output ripple for each average value achievable with a 3-added-resolution-bit PWM dithering implementation.

Table 5. PWM dithering patterns, 3-added-resolution-bit PWM (lowest output ripple)

Average value	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8	Ripple
0 LSB	0	0	0	0	0	0	0	0	None
1/8 LSB	0	0	0	0	0	0	0	1	Highest
2/8 LSB	0	0	0	1	0	0	0	1	↓
3/8 LSB	0	0	0	1	0	1	0	1	
4/8 LSB	0	1	0	1	0	1	0	1	Lowest
5/8 LSB	0	1	0	1	1	0	1	1	↑
6/8 LSB	0	1	1	1	0	1	1	1	
7/8 LSB	0	1	1	1	1	1	1	1	Highest
1 LSB	1	1	1	1	1	1	1	1	None

2 PWM dithering technique implementation for STM32 timers

2.1 STM32 timers features for PWM dithering implementation

The PWM dithering technique can be applied to any of the STM32 advanced configuration, general purpose and lite timers, regardless of their resolution: 16-bit or 32-bit timers.

The STM32 timers present a set of common features that make the dithering technique implementation relatively easy and with a small impact on the overall system performance. Below is a brief presentation of the STM32 timer features that are needed for the dithering technique implementation. For further details on these features or any other STM32 timer features, refer to the corresponding microcontroller reference manual.

2.1.1 STM32 timer time-base

The time-base unit is a basic component of the STM32 timers. It is included in every STM32 timer. The time-base unit contains the counter register, the auto-reload register and several other components. Depending on the configured counting direction, the counter is either incrementing from zero to the auto-reload register value or is decrementing from the auto-reload register value to zero. The time-base unit is used to define the timer counting periodicity.

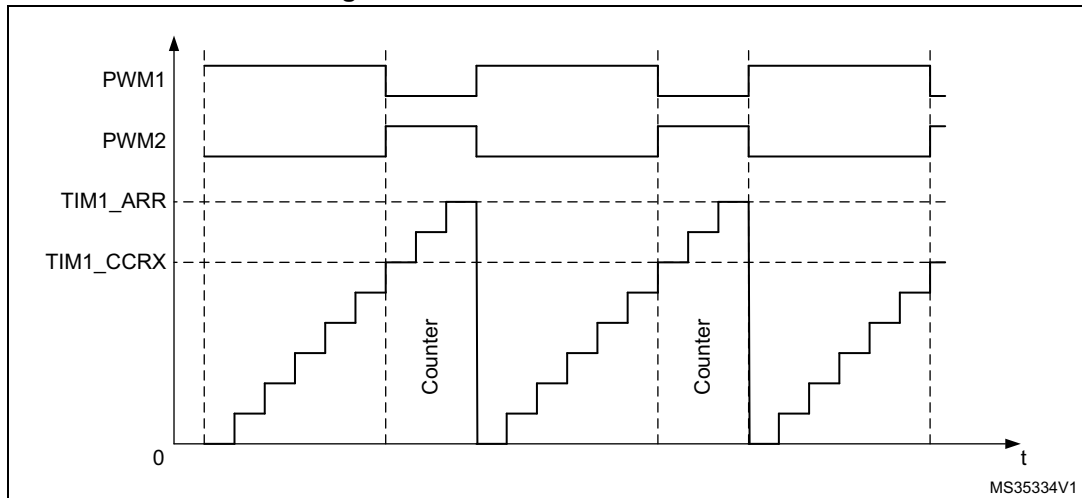
2.1.2 STM32 timer channels

The STM32 timers implement different number of channels ranging from 0 to 6. The number of channels depends on the type of timer and on the product into which it has been embedded. The timers with zero channel (only a time-base unit) are called basic timers. These timers are not concerned by this application note because at least one channel (one timer output) is required by the PWM dithering technique.

The STM32 timer channels can be used as input channels or as output channels. When configured in output mode, STM32 channels can be configured to output PWM signals. The PWM output configuration is the one used by the dithering technique.

When configured in PWM mode, the duty-cycle of the output waveform is controlled by the counter register value and the channel register value. The timer is continuously carrying on a comparison between the counter value and the timer channel value. Depending on the comparison result and the configured PWM mode (PWM1 or PWM2 mode), the channel output is either high or low. The resulting signal, as shown in [Figure 2](#), is a PWM signal with a period duration controlled by the auto-reload register value and a duty-cycle controlled by the channel register value.

Figure 2. STM32 timers PWM modes



2.1.3 Channel register preload feature

As the channel output level depends on the continuous comparison between the counter register value and the timer channel register value, any change to the channel register may induce immediate channel output level change. As a consequence, writing directly to the channel register in the middle of a PWM period may generate spurious waveforms. To overcome this problem, the STM32 timer channels embed two channel registers: an active channel register and a shadow channel register. These two channel registers are controlled by the preload feature. When preload feature is disabled, any write operation is made directly into the active channel register used for carrying the comparison operation. As explained above, this configuration may generate spurious waveforms if the write access is not well synchronized with the counter value (this is practically difficult to achieve). When preload is activated, any write access is made to the shadow channel register while the channel active register remains unchanged (no perturbation on the ongoing PWM period). As soon as the timer counter reaches its counting boundary an update event is generated by the timer and the shadow register content is transferred into the active register which will be used during the next PWM period for carrying the comparison operation and define the duty-cycle ratio.

In summary, the preload feature guaranties that no spurious waveforms are generated while accessing the channel register. The channel register write access can be performed anytime within a PWM period.

For further details regarding the preload feature, refer to the corresponding microcontroller reference manual.

2.1.4 Timer DMA burst feature

In order to apply dithering effect on several timer outputs simultaneously (it is of high interest when designing multi-phase power converters), it is required to update several channel registers using the same DMA channel (on some products this is also called stream). The DMA burst feature built into the STM32 timers makes possible to update the content of several timer registers following a single event generated by the timer. In particular, this allow to update several channel registers simultaneously, following the timer update event.

2.2 System-level features and peripherals used in PWM dithering implementation

PWM dithering technique consists in refreshing the duty-cycle value for each PWM period. In other terms, this technique consists in refreshing the channel register value on each PWM period.

To offload the CPU, the STM32 DMA peripheral can be used to perform the timer channel register refresh thanks to the features described in [Section 2.2.1](#) and [Section 2.2.2](#).

2.2.1 Timer update-event-triggered DMA transfer

The STM32 timers feature the capability to trigger DMA transfers following the occurrence of particular timer events. For PWM dithering implementation, the timer update event can be used to start a DMA transfer. The timer update event is a periodic event that occurs each time the timer counter reaches its counting boundary. This event is used to trigger the channel register refresh at the start of each new PWM period.

2.2.2 DMA half-transfer interrupt

As explained above, in order to generate the PWM dithering effect, the duty-cycle should be updated at each PWM period for a given range of periods and following a given pattern. The pattern is a series of successive values that will be transferred one by one into the channel register at each new PWM period. For a given dither value, a dedicated pattern should be calculated. As a consequence, when a pattern is being transferred, the next one should be calculated. One way to do this is by splitting the pattern table into two halves. When DMA is transferring values from one part of the table, the firmware can calculate the other part of the table and prepare the pattern for the next PWM period series (the duty-cycle adjustment pattern) corresponding to the desired dither value.

DMA half-transfer interrupt and transfer-complete interrupt can be used for implementing this mechanism. When the half-transfer interrupt occurs this means that the DMA has completed the transfer of the values from the first half of the table and is going to transfer values corresponding to the next series of PWM periods from the second half of the table. When transfer-complete interrupt occurs this means that DMA has completed the transfer of values from the second part of the table and is going to switch back to the first half of the table for transferring pattern values for the next series of PWM periods.

2.3 PWM dithering implementation using STM32 MCUs

This section explains how to assemble the above described features in one application in order to enhance the resolution of one timer channel configured in PWM mode.

First of all, the programmer should reserve the needed space in the SRAM for the table of patterns. The size of the table of patterns can be calculated using the below formula:

$$\text{Table_size} = 2 \times 2^N$$

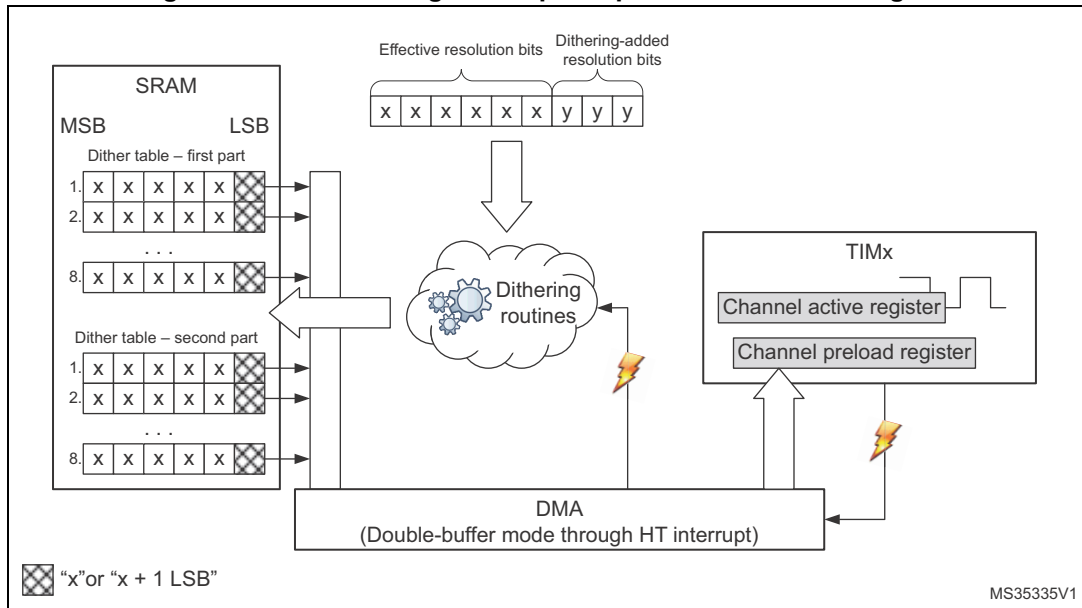
where N is the PWM dithering effect added resolution bits.

When using a 16-bit timer, the table of patterns should be a table of half-words, otherwise if using a 32-bit timer, the table of patterns should be a table of words.

The firmware routines that initialize the peripherals needed for PWM dithering should respect the following recommendations.

- The DMA should be configured to transfer values from the table of patterns located in SRAM memory into the timer channel register. The DMA source address increment should be configured in circular mode in order to roll over when it reaches the end of the table of patterns and points again on the table top. The DMA destination address should be fixed. The DMA half-transfer interrupt and transfer complete interrupt should be enabled for the relevant channel (on some products it is called stream). The firmware example provided with this application note presents a proposal on how to configure the DMA for implementing the PWM dithering technique. For further details about DMA modes and configurations, refer to the microcontroller corresponding reference manual.
- The timer should have its time-base initialized with appropriate values in order to generate the required PWM frequency. The “preload” function should be activated for the timer channel register. The timer should be configured to send DMA requests following each update event.
- The dithering patterns should be calculated based on the desired duty-cycle. If the timer possible hardware resolution for duty-cycle is x-bit length and the desired dither-effect added resolution is y-bit length, the duty-cycle passed to the dithering routines should be (x+y)-bit length. The DMA half-transfer and transfer-complete interrupts can be used as triggers for dithering patterns calculation routines. These DMA interrupts are useful to know which part of the table of patterns is used by DMA and which one is free and can be updated by the dithering patterns generation routines. The demonstration firmware provided with this application note shows an example of possible implementation of dithering patterns generation routines. This example, while being functional, might be further enhanced in order to improve performance (e.g. reduce footprint, reduce CPU load, etc.). [Figure 3](#) illustrates how to assemble the STM32 MCU features together in order to implement the required timer PWM output resolution.

Figure 3. PWM dithering technique implementation bloc diagram



3 Demonstration

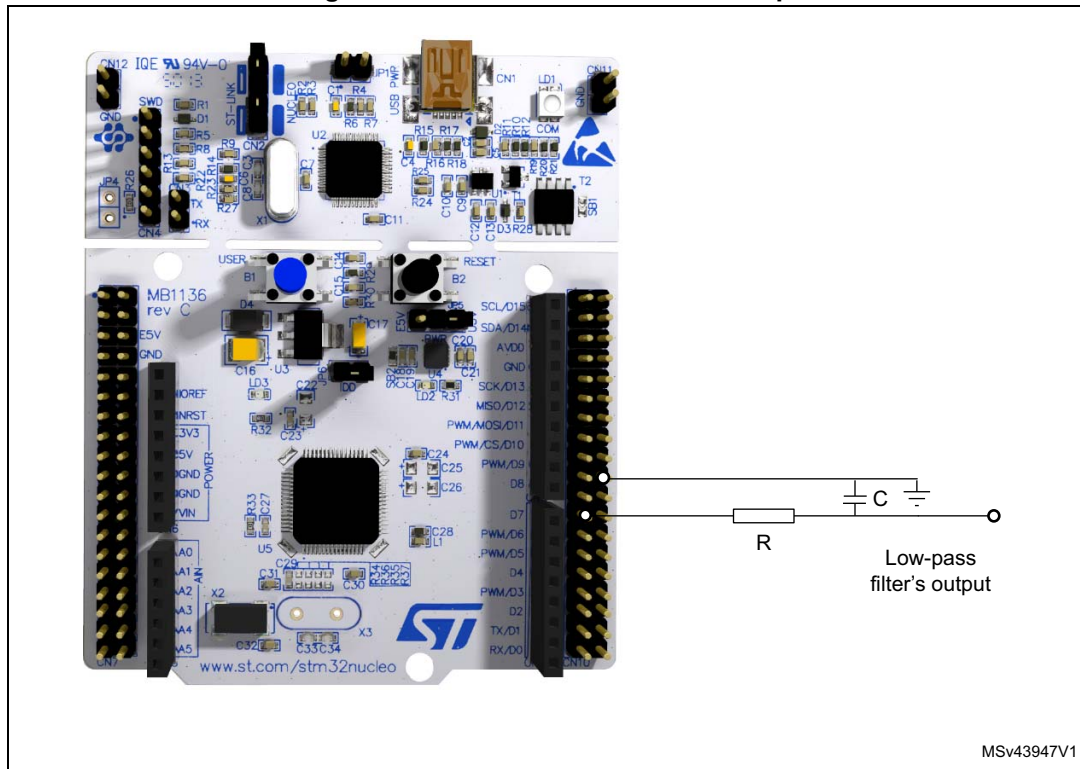
This section details a 3-added-resolution-bit PWM dithering implementation. It provides insights about the firmware architecture used for the implementation. The demonstration example is given for 3 added resolution bits, but can be easily extended for different number of added resolution bits. The provided demonstration firmware is targeting the STM32F3 series of products. It can be easily migrated to any other product in the STM32 microcontrollers family.

The demonstration firmware generates a triangular waveform. The idea is to demonstrate the PWM dithering effect on the staircase rising slope of the generated waveform without affecting the falling slope. The lower the PWM resolution, the higher the rising staircase steps. Applying the dithering technique on the PWM signal used to generate the triangular waveform adds a certain number of extra resolution bits. If properly implemented, the PWM dithering technique should make the output steps smaller (dithering effect magnitude depends on the number of added resolution bits). The dithering effect should be observed at the output of the external low pass filter (see [Section 3.3.2: Low-pass filter dimensioning considerations](#)), rather than directly at the output of the timer.

3.1 Demonstration hardware environment

The demonstration firmware provided with this application note is intended to be run on the NUCLEO-F302R8 Nucleo board. The user manual for this board and all related documentation are available on www.st.com. Before the board is used for the demonstration, it needs to be slightly reworked to mount the low-pass filter on the timer output used for the demonstration.

Figure 4. STM32F3 Nucleo board setup



MSv43947V1

The low-pass filter cut-off frequency calculation is based on several parameters including the PWM period. For this demonstration example the cut-off frequency is around 160 kHz ($R = 1 \text{ k}\Omega$, $C = 1 \text{ nF}$). Our PWM switching frequency is around 1.125 MHz which is definitely in the rejected bandwidth of the low-pass filter. The resistor and the capacitor components should be mounted as close as possible to the timer output pin on the Nucleo board. If the low-pass filter is placed on a daughter board, the connections between the daughter board and the Nucleo board should be as short as possible in order to avoid distortion of the filter parameters and the resulting waveform (e.g. switching noise may become more important due to the inductance of the connection wires).

3.2 Demonstration firmware architecture

3.2.1 Utilization of the microcontroller hardware resources

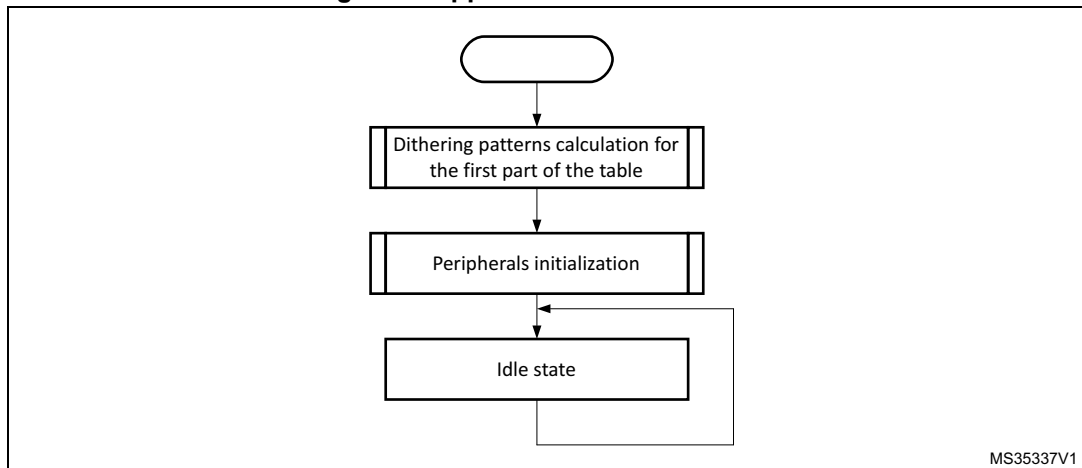
This demonstration firmware example is developed for the STM32F302R8T6 microcontroller present on the NUCLEO-F302R8 board. The hardware resources used for the PWM dithering implementation are listed below:

- DMA1 channel 5 is used for timer channel register refresh
- Timer1 channel1 is used for the PWM resolution enhancement through dithering technique
- Timer1 channel1 output is configured to be outputted on the pin PA.09 that belongs to the GPIO port A.

3.2.2 Demonstration firmware description through flowcharts

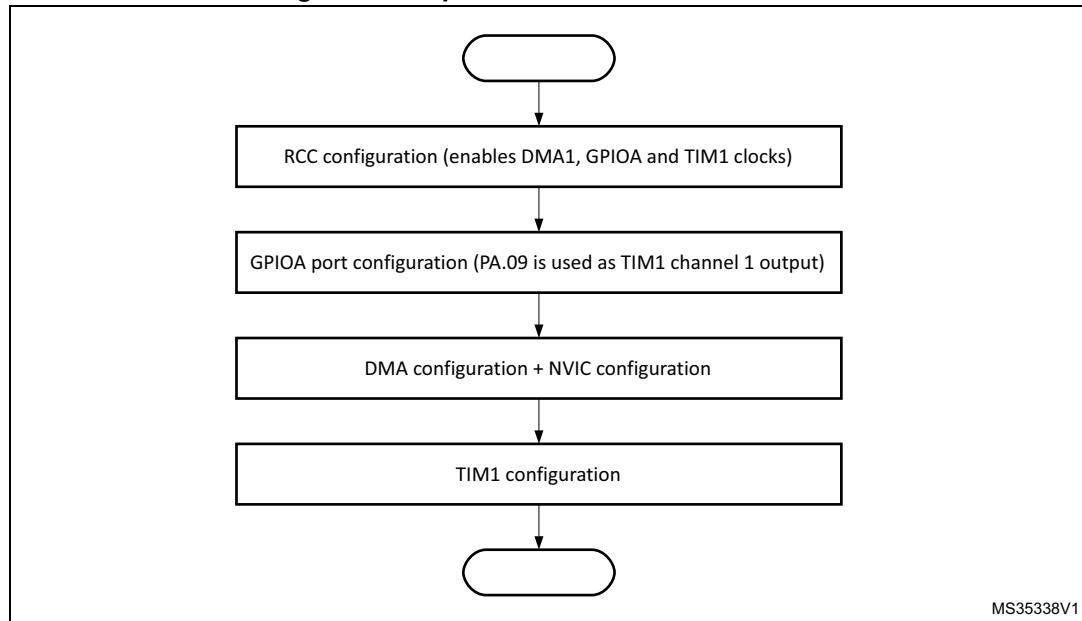
The CPU completes the system clock configuration and branches to the application main routine. The application main function is described by the flowchart in [Figure 5](#):

Figure 5. Application's main function



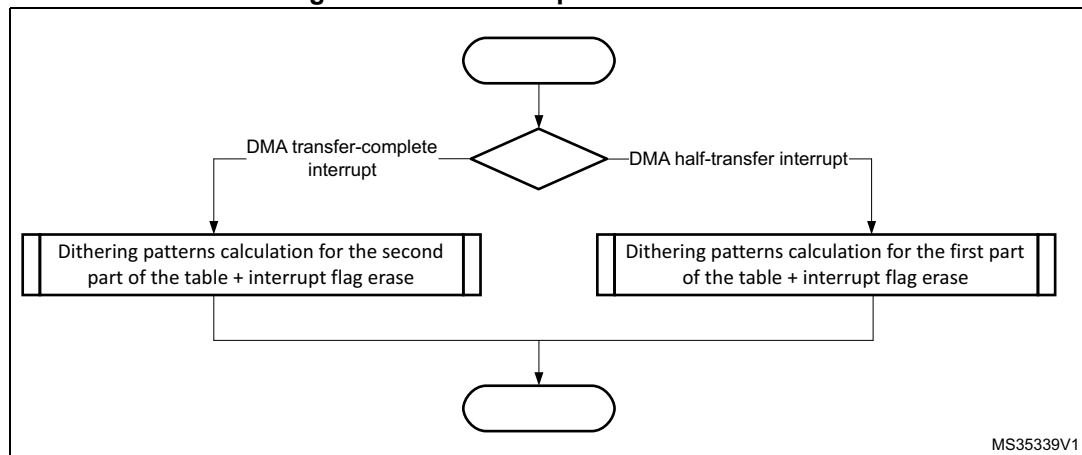
Before initializing the timer and DMA peripherals and start refreshing the timer channel register, the first half of the table of patterns should be initialized. The following table updates are handled by the DMA interrupt handler. The peripherals initialization function is described in the flowchart [Figure 6](#).

Figure 6. Peripherals initialization flowchart



After completing the peripherals initialization, the CPU goes into an infinite loop. Dithering patterns calculation task is handled by the DMA interrupt service routine (ISR) following each DMA half-transfer or transfer-complete interrupt. The flowchart in [Figure 7](#) describes the DMA interrupt handler function (ISR).

Figure 7. DMA interrupt handler flowchart



3.2.3 Firmware Footprint

The firmware routines and variables declaration associated with the PWM dithering technique implementation are located in the C source code files named *“main.c”* and *“stm32f30x_it.c”*. The footprint of these two files (obtained with IAR EWARM v7.80

toolchain) is shown in [Table 6](#).

Table 6. Firmware Footprint

File Name	Read-only code memory	Read-only data memory	Read/write data memory
main.c	624 bytes	9 bytes	196 bytes
stm32f30x_it.c	40 bytes	0 byte	0 byte

3.3 Experimental results

3.3.1 Dithering effect on PWM resolution enhancement

[Figure 8](#) shows the dithering effect on the rising slope of the triangular waveform. The rising slope looks more like a continuous oblique line where the falling slope still have the staircase shape. The falling slope has its duty-cycle controlled using a 6-bit resolution granularity with the timer clocked at 72 MHz and the PWM frequency set at 1.125 MHz (see [Table 1](#), for details). When the PWM dithering is applied on the rising slope, the duty-cycle control resolution is extended by 3 bits to reach 9-bit resolution. The output control granularity is reduced by a ratio of $2^3 = 8$ and the output step becomes smaller (almost invisible on [Figure 8](#)).

Figure 8. Dithering effect applied on the rising slope of the triangular waveform



3.3.2 Low-pass filter dimensioning considerations

For the example shown in *Figure 9*, the low-pass filter cut-off frequency has been deliberately altered (cut-off frequency too close to the PWM frequency divided by $2^3 = 8$). The switching added by the dithering-effect is not well averaged and the expected fine control may not be obtained. Nevertheless, *Figure 9* is interesting to better understand the effects of the dithering technique on the raw (unfiltered) PWM output signal.

Figure 9. Low-pass filter cut-off frequency effect on the dithering technique efficiency



4 References

1. "Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters" by Angel V. Peterchev, Student Member, IEEE, and Seth R. Sanders, Member, IEEE.
2. STM32F30xxx and STM32F31xxx datasheets and reference manuals.

5 Revision history

Table 7. Document revision history

Date	Revision	Changes
25-Jun-2014	1	Initial release.
17-Jan-2017	2	Following changes in the whole document: – STSW-STM32151 replaced by X-CUBE-PWM-DITHR – STM32F3DISCOVERY replaced by NUCLEO-F302R8 – STM32F303VCT6 replaced by STM32F302R8T6. Updated Figure 4 .

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved