AN1712
APPLICATION NOTE

GENERATING A HIGH RESOLUTION SINEWAVE USING ST7 PWMART

By Microcontroller Division Applications

INTRODUCTION

The purpose of this application note is to present a software technique for generating a high resolution sinewave using ST7 PWMART, tunable in frequency and average amplitude. This application has been implemented using the ST72321J9 microcontroller. The PWMART (Autoreload timer peripheral embedded in the microcontroller) is used to generate a PWM signal and this PWM signal is then filtered by low pass filter (simple RC circuit) to generate a sinewave.

Rev. 1.0

1 GENERATING A SINUSOID

This section highlights the main features of the ST7 PWMART used to generate a PWM signal which is then filtered by low pass filter (a simple RC circuit in this example) to generate a sinusoid. Please refer to the ST7 datasheet for more details.

The ST7 PWMART consists of an 8-bit auto reload counter with compare/capture capabilities and a 7-bit clock prescaler.

1.1 PWM GENERATION

The free running 8-bit counter is fed by the output of the prescaler, and is incremented on every rising edge of the clock signal. It is possible to read or write the contents of the counter on the fly by reading or writing the Counter Access register (ARTCAR). When a counter overflow occurs, the counter is automatically reloaded with the contents of the ARTARR register (the prescaler is not affected).

The counter clock frequency is given by:

$$f_{COUNTER} = \frac{f_{INPUT}}{2^{CC[2:0]}}$$

The timer counter's input clock (f_{INPUT}) feeds the 7-bit programmable prescaler, which selects one of the 8 available taps of the prescaler, as defined by CC[2:0] bits in the ARTCSR Register. Thus the division factor of the prescaler can be set to 2^n (where n = 0, 1,..7). This f_{INPUT} frequency source is selected through the EXCL bit of the ARTCSR register and can be either the f_{CPU} or an external input frequency f_{EXT}. The clock input to the counter is enabled by the TCE (Timer Counter Enable) bit in the ARTCSR register. When TCE is reset, the counter is stopped and the prescaler and counter contents are frozen. When TCE is set, the counter runs at the rate of the selected clock source.

The timer compare function is based on four different comparisons with the counter (one for each PWMx output). Each comparison is made between the counter value and an output compare register (OCRx) value. This OCRx register can not be accessed directly, it is loaded from the duty cycle register (PWMDCRx) at each overflow of the counter.

This double buffering method avoids glitch generation when changing the duty cycle on the fly.

PWM mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins with minimum core processing overhead. This function is stopped during HALT mode. Each PWMx output signal can be selected independently using the corresponding OEx bit in the PWM Control register (PWMCR). When this bit is set, the corresponding I/O pin is

configured as output push-pull alternate function. The PWM signals all have the same frequency which is controlled by the counter period and the ARTARR register value.

$$f_{PWM} = \frac{f_{COUNTER}}{256 - ARTARR}$$

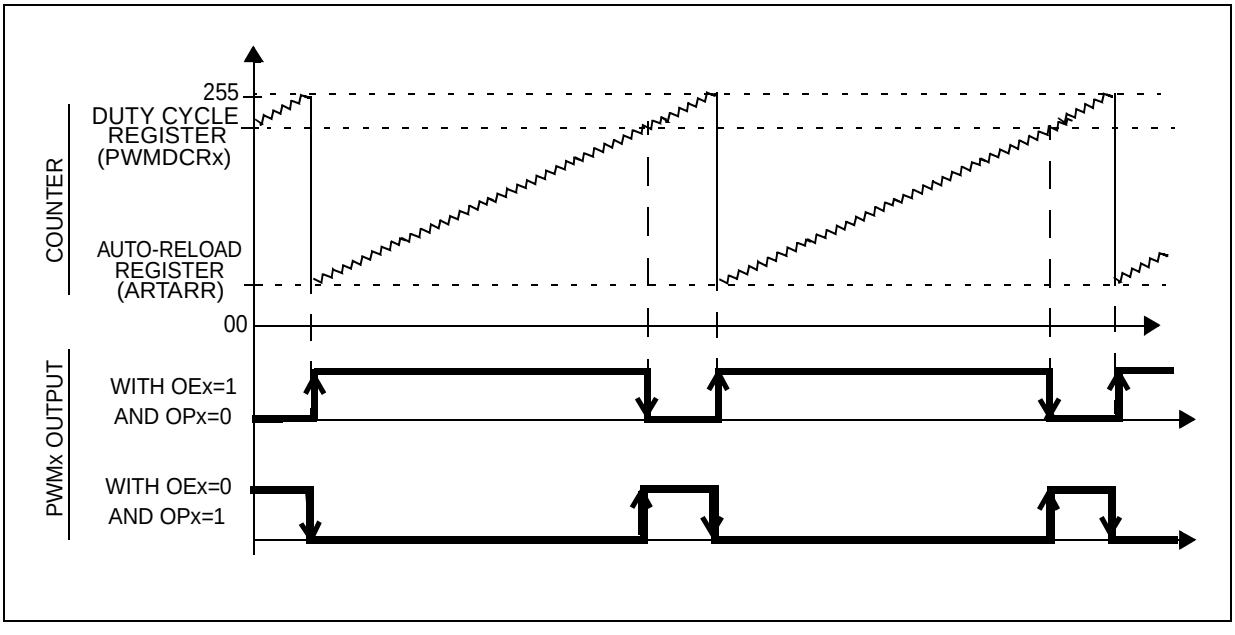
When a counter overflow occurs, the PWMx pin level is changed depending on the corresponding OPx (output polarity) bit in the PWMCR register. When the counter reaches the value contained in one of the output compare register (OCRx) the corresponding PWMx pin level is restored.

Note: The reload values will also affect the value and the resolution of the PWM output signal duty cycle. To obtain a signal on a PWMx pin, the contents of the OCRx register must be greater than the contents of the ARTARR register.

The resolution for the PWMx duty cycle is:

$$Resolution = \frac{1}{256 - ARTARR}$$

Figure 1. PWM Auto-reload Timer Function



On overflow, the OVF flag of the ARTCSR register is set and an overflow interrupt request is generated if the overflow interrupt enable bit, OIE, in the ARTCSR register, is set. The OVF flag must be reset by the user software. This interrupt is used as a time base in the application.

1.2 SINEWAVE GENERATION

At the start of the program:

– the PWMDCR0 register is initialized to obtain a 50% duty cycle

– the number of samples in a sinewave cycle is defined

– the counter reload value (ARTARR) is initialized

In the software provided with this application note, predefined initialization values for each frequency can be selected in the define.h file.

More than 18 samples in a sinewave cycle should be selected to generate a sinewave with Total Harmonic Distortion of less than 5%. So, depending on the number of samples in a sinewave cycle, the duty cycle register (PWMDCR0) is modified, taking care that no 0% and 100% PWM is generated for any of the sinewave samples (because the PWM duty cycle must be between ~99% to ~1% to generate an undistorted sinewave). The PWM duty cycle is changed after a certain number (“COUNTER”) of overflow cycles. So, the sinewave frequency depends on three parameters,

– the PWM frequency (f_{PWM})

– the number of samples in a sinewave cycle

– the number of overflow cycles after which the PWM duty cycle changes (“COUNTER”)

So, the sinewave frequency can be given by:

$$f_{SINE} = \frac{1}{t_{PWM} * \text{number of samples} * \text{COUNTER}}$$

See also Figure 2.

This PWM signal must be filtered with an external RC network selected for the filtering level required to generate a sinusoid. The cut off frequency of low pass RC filter is given as:

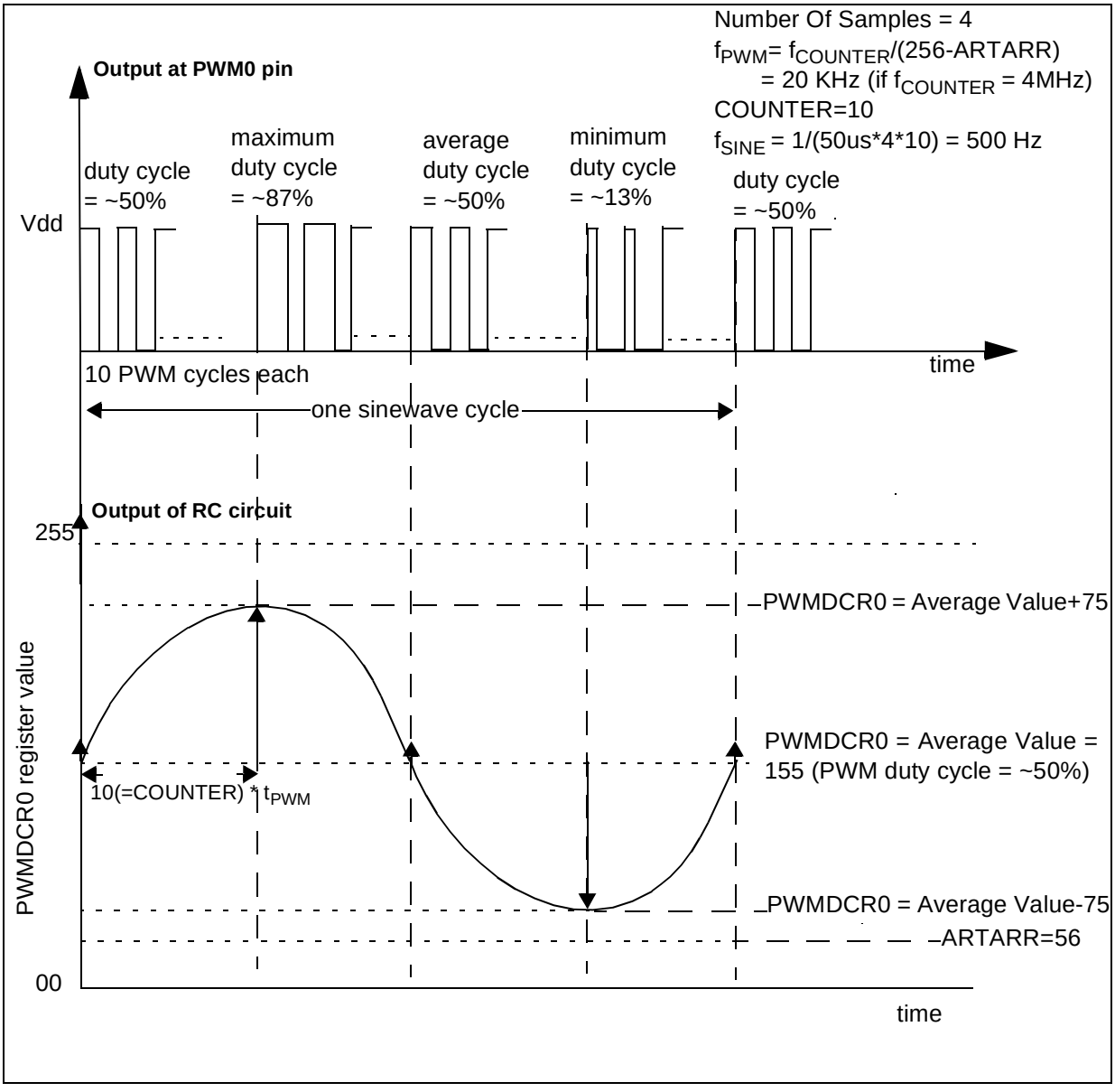
$$f_H = \frac{1}{2 * pi * R * C}$$

So, the value of R and C must be chosen in such a way that the output sinewave frequency should be less than this high cut off frequency (f_H).

The instantaneous value of sinewave depends on the duty cycle of PWM. So:

$$V_{SINE\ max,min} = \text{Max, Min PWM duty cycle} * V_{DD}$$
$$V_{SINE\ average} = \text{Average PWM duty cycle} * V_{DD}$$

Figure 2. Output at PWM0 pin and general calculation for finding sinewave frequency



2 IMPROVING SINEWAVE RESOLUTION

Sinewave resolution depends on three factors:

– t_{PWM}
– the number of samples in a sinewave cycle
– “COUNTER” value

By changing any of these parameters you can get a different resolution.

If $f_{COUNTER}$ is fixed, you can only change t_{PWM} by varying the ARTARR value. The minimum change you can do to ARTARR is 1. So, the minimum change in t_{PWM} is $t_{COUNTER}$. For example:

If $f_{COUNTER}$ is 4 MHz and ARTARR value is 56, then t_{PWM} is 50us. If the number of samples in a sinewave cycle is 40 and COUNTER is 10, f_{SINE} will be 50 Hz.

If you change the ARTARR value to 57, t_{PWM} will change to 49.75us. So, f_{SINE} will be ~50.25 Hz (assuming the number of samples and COUNTER value are fixed). So, the resolution is ~0.25 Hz.

Now to improve resolution, change ARTARR = 36, COUNTER = 7 and number of samples in a sinewave cycle = 52.

Assuming $f_{COUNTER}$ is still 4MHz, t_{PWM} will be 55us. In this case f_{SINE} will be 49.95 Hz, which gives improved resolution (~0.05Hz).

3 SOFTWARE CONFIGURATION

The software is developed in “C” language and gives you the option of using the ST7 software library or not. A header file “define.h” is supplied. This header file defines the sin structure. The sin structure has the three user defined datatypes for storing sinewave patterns depending on the number of samples in a sinewave cycle, the “index” which is used for counting the current sinewave sample and another datatype which is used to indicate whether the current sample is related to the upper half or lower half of the sinewave. Depending on the number of samples in a sinewave cycle, sample values are initialized for the sinewave envelope (X(n)=Asin((2*pi*n)/N)). where A is the sinewave amplitude, pi is 3.1416, n is the n^{th} sample of the sinewave, N is the number of samples in a sinewave cycle. N should be more than 18 in order to generate a sinewave with Total Harmonic Distortion of less than 5%. The value of A should be such that no 0% or 100% duty cycle is generated for any of the sinewave samples.

There are other define types for the sinewave frequency from 45 to 65Hz with a resolution of less than 0.1Hz. These define types contain five parameters:

– “no_of_samples_half” for defining the number of samples in half a sinewave cycle

– “COUNTER” value which defines the number of overflow cycles after which the PWM duty cycle changes

– Initialization value of ARTARR for f_{PWM},

– Initialization value of PWMDCR0 a 50% duty cycle

– “AVERAGE_AMP” to define the average sinewave amplitude.

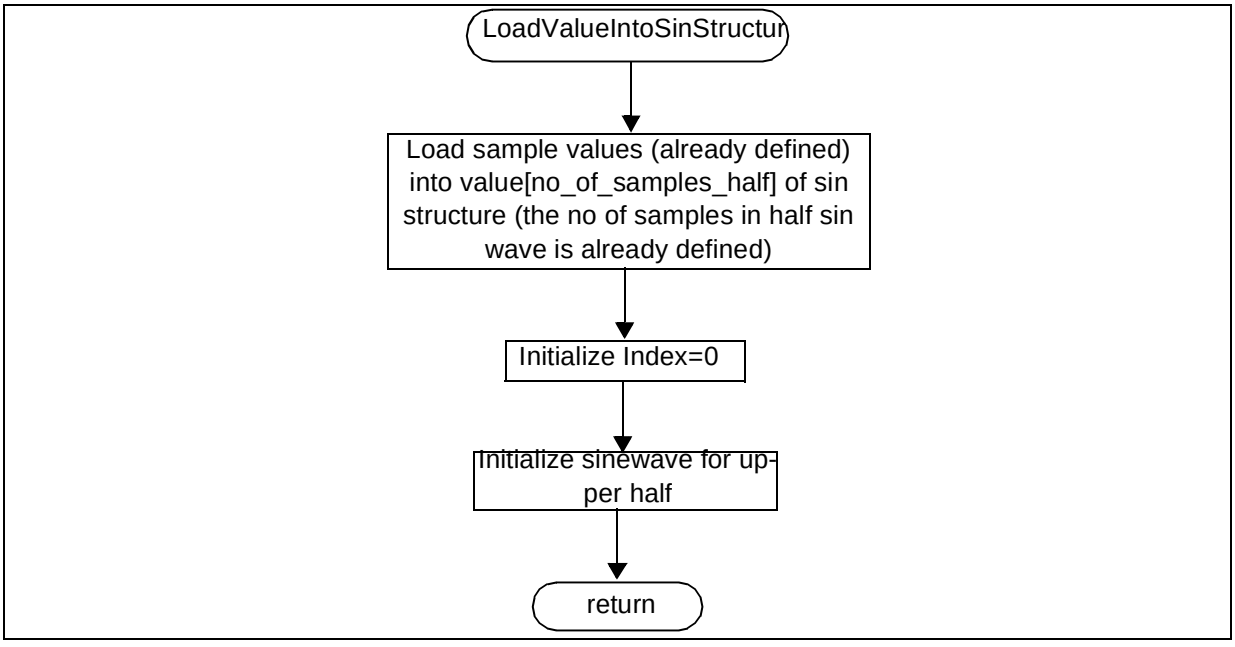
A sinewave with a resolution of less than 0.1Hz is obtained by varying the “no_of_samples_half”, the “COUNTER” value and ARTARR. The average sinewave amplitude is also software configurable by the “AVERAGE_AMP” parameter.

It should be noted that the value of A and the value of the “AVERAGE_AMP” is chosen in such a way that no 0% or 100% duty cycle is generated for any of the sinewave samples.

3.1 SIN STRUCTURE INITIALIZATION

The sin structure has the three user defined datatypes for storing the sinewave pattern according to the number of samples in a sinewave cycle, the “index” which is used for counting the current sinewave sample and another datatype which is used for indicating whether the current sample is related to the upper half or lower half of the sinewave.

Figure 3. Loading of sample values and initialization of sin structure



3.2 AUTO-RELOAD TIMER INITIALIZATION

The counter is initialized by:

– Writing to the ARTARR register to set the PWM frequency.

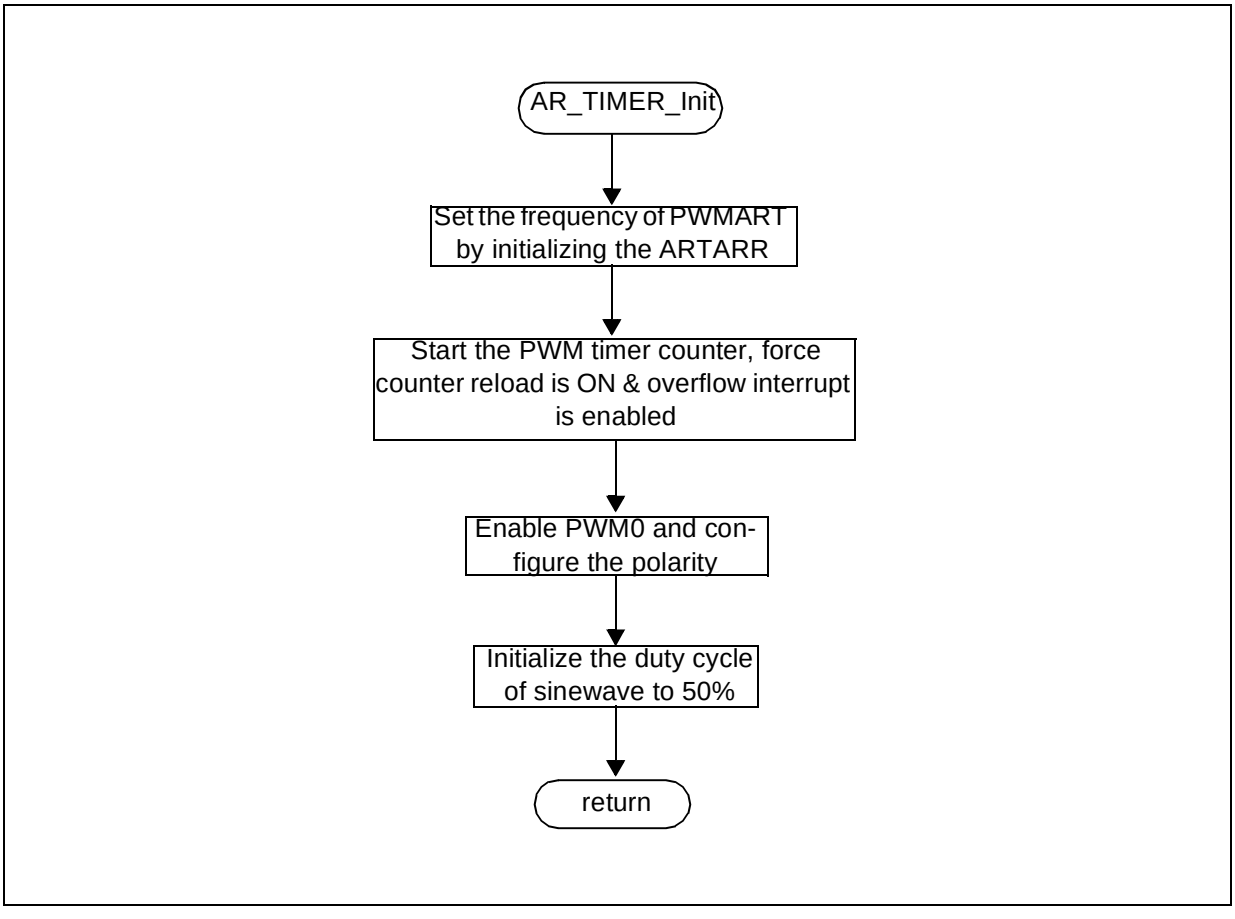
– Setting the FCRL (Force Counter Re-Load), the TCE (Timer Counter Enable) and OIE (Overflow Interrupt Enable) bits in the ARTCSR register.

In this case, the f_{INPUT} is CPU clock (f_{CPU}) and $f_{COUNTER} = f_{INPUT}$ (= 4MHz in this particular application).

– Enable PWM0 and configure the polarity in the PWMCR register.

– Initialize the PWMDCR0 register to define the PWM duty cycle.

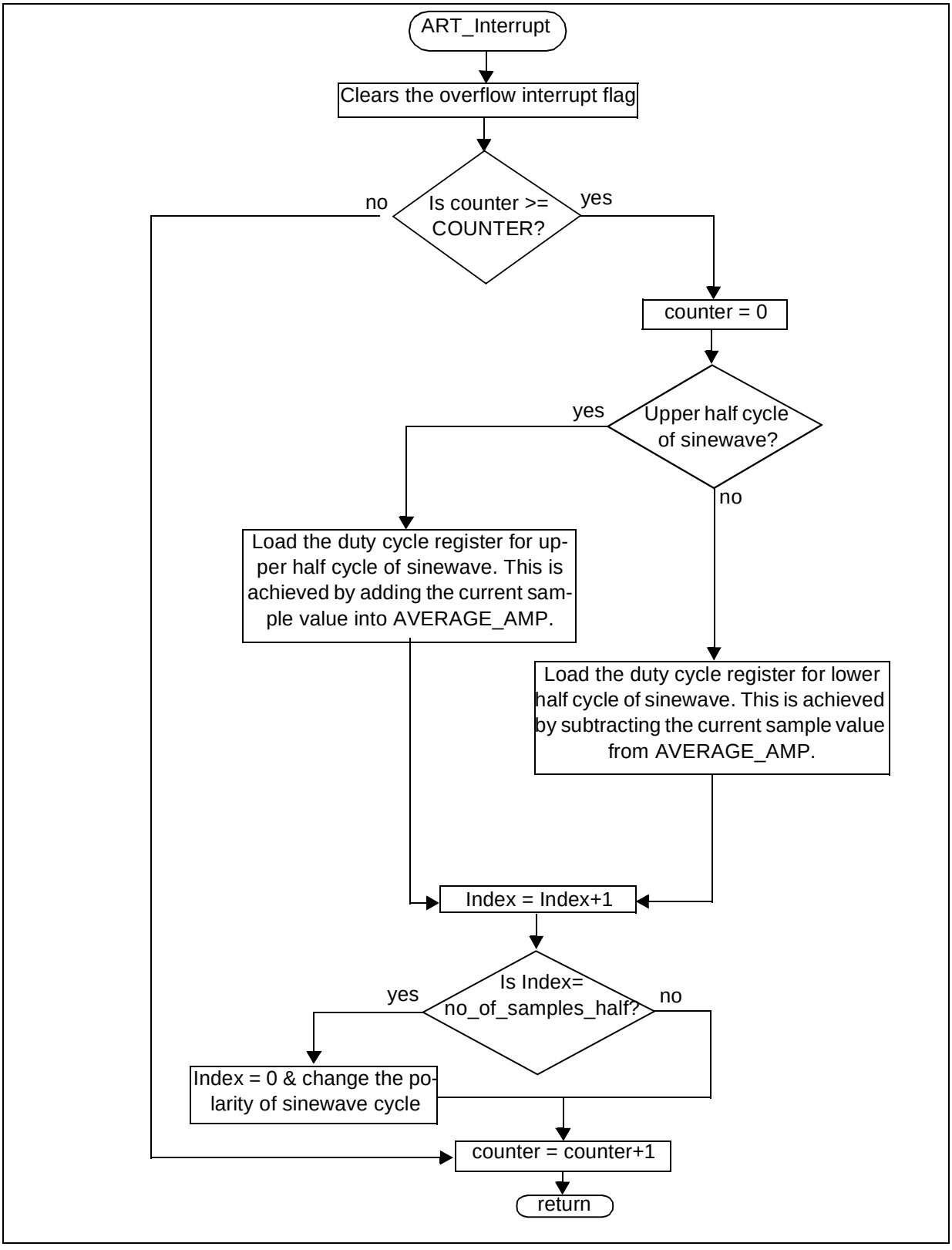
Figure 4. PWMART timer initialization



3.3 PWMART INTERRUPT SERVICE ROUTINE

This is the interrupt service routine for the PWMART interrupt. Every time an overflow occurs, an interrupt is generated (because overflow interrupt is enabled). The PWMART duty cycle is changed after every “COUNTER” number of overflow cycles. The duty cycle can vary from ~99% to ~1% depending on the sinewave sample values.

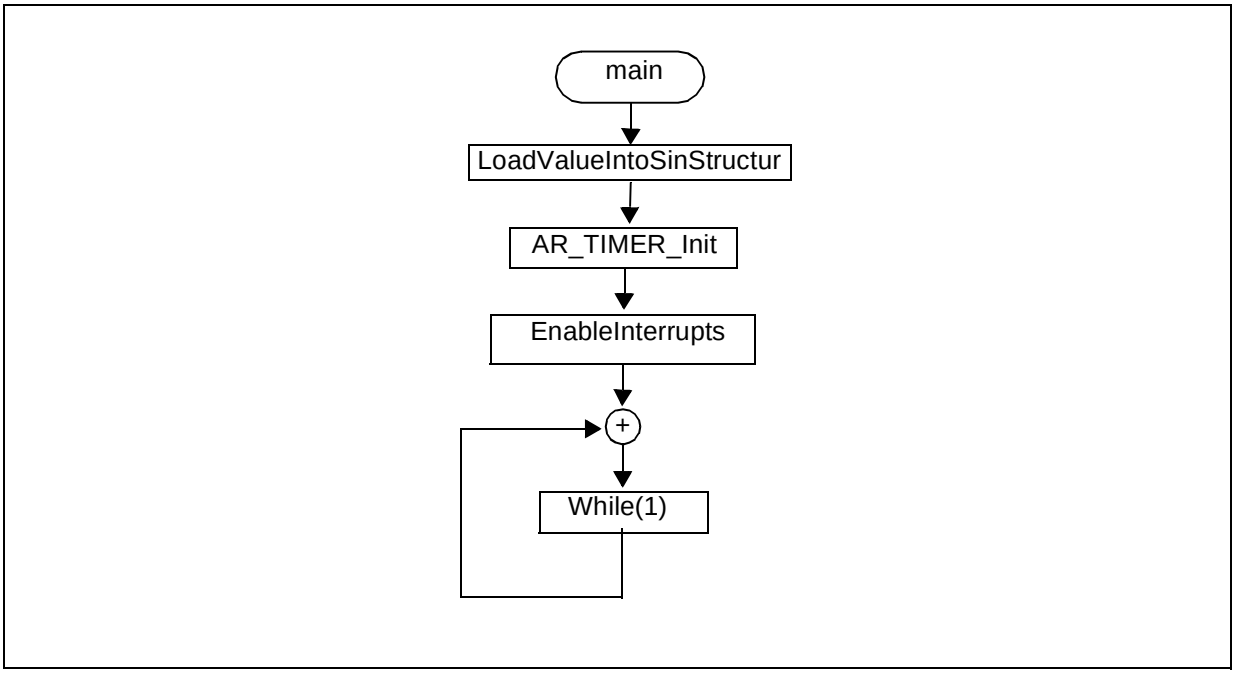
Figure 5. PWMART interrupt service routine



3.4 MAIN ROUTINE

The main routine calls LoadValueIntoSinStructur and AR_TIMER_Init. After this, the initialization interrupts are enabled (RIM is executed) so that the microcontroller can go into the interrupt routine and an infinite while loop is called.

Figure 6. Main routine

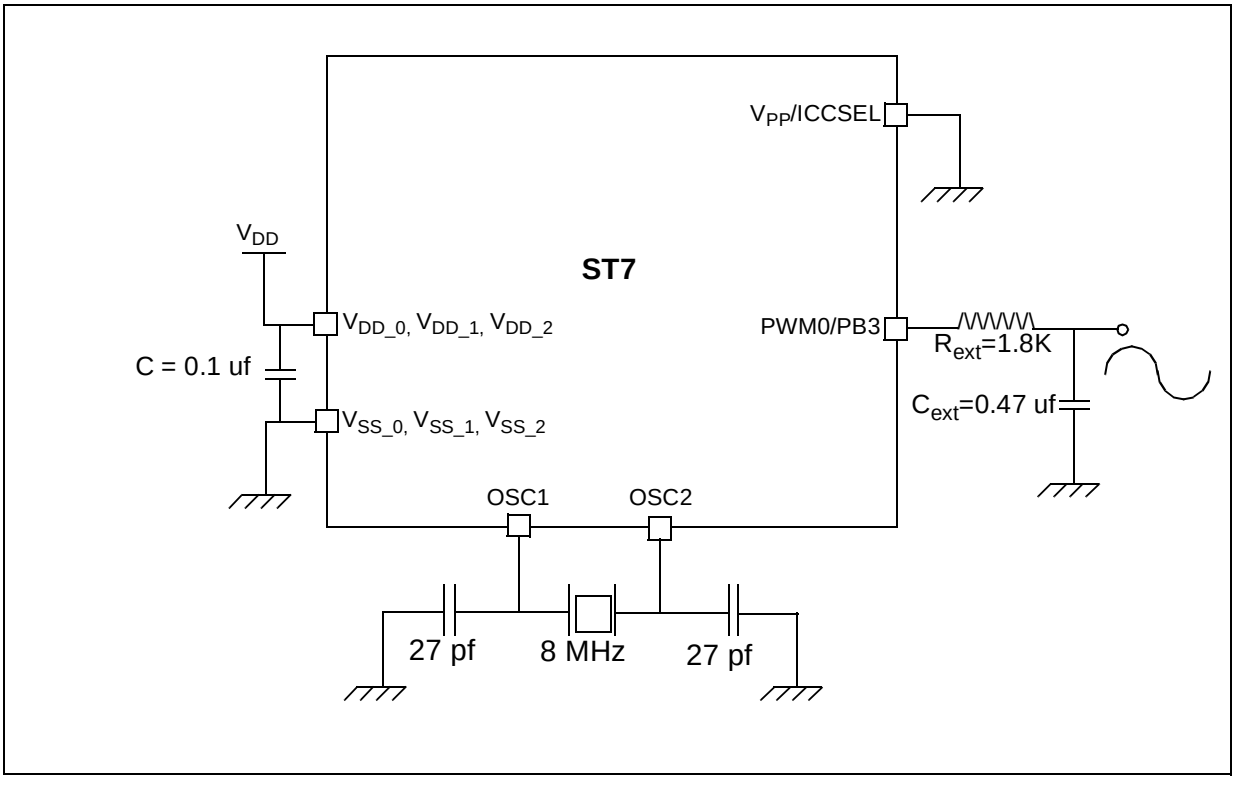


4 HARDWARE CONFIGURATION

This application runs on an ST72F321 microcontroller. The PWM0 channel is used to generate the PWM signal which is then filtered by low pass filter (a simple RC circuit in this example) to generate a sinusoid. The R_{ext} used is 1.8K and C_{ext} is 0.47uf. The values of R_{ext} and C_{ext} decides the filtering level.

The selected crystal for this example has a frequency of 8 MHz which gives f_{CPU} = 4 MHz because the PLL is disabled and slow mode is not selected. It gives the $f_{COUNTER}$ = 4 MHz for the reset value of Counter Clock Control bits in ARTCSR register.

Figure 7. Generation of sinewave: Application circuitry



5 SOFTWARE

All the source files in “C” language with the option of using the ST7 software library or not (ST7 software library version 1.1) are given in the zip file with this application note.

The source files are for guidance only. STMicroelectronics shall not be held liable for any direct, indirect or consequential damages with respect to any claims arising from use of this software.

THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS."