

GENERATION AND IMPLEMENTATION OF SINE WAVE TABLE

SYED TAHMID MAHBUB

A sine wave is a wave with formula $y=\sin(x)$. Now we need to make this sine wave digitally, that is, “using square waves”. Let’s explain with an example.

When digitally producing sine wave, we do so using a fixed number of square wave pulses. The larger the number, the cleaner the sine wave. So, for example’ sake, we’ll use 10 samples, though, keep in mind, in practice, 10 is too less and usually at least 32 are taken.

So, we have $y=\sin(x)$. x varies between 0° to 360° . But, we’ll only focus on one half and then reverse this to have the negative half. Since we have 10 samples, we divide the positive half of the sine wave into 10 parts. The first part is 0° to 18° . We’ll send pulses, one at 0° and the other at 18° , the next pulse at 36° , then at 54° and so on. So, the values we need are:

1. $\sin(0) = 0$
2. $\sin(18) = 0.31$
3. $\sin(36) = 0.59$
4. $\sin(54) = 0.81$
5. $\sin(72) = 0.95$
6. $\sin(90) = 1$
7. $\sin(108) = 0.95$
8. $\sin(126) = 0.81$
9. $\sin(144) = 0.59$
10. $\sin(162) = 0.31$

We don’t take $\sin(180)$ as this value will be next called when we implement $\sin(0)$, which both equal 0.

Now, we know how to find the values. Now, how to implement this. So, we’re using PIC PWM. 1 means 100% duty cycle, 0 means 0% and 0.5 means 50%. So, from the above table, we know that:

1. $\sin(0) = 0\%$
2. $\sin(18) = 31\%$
3. $\sin(36) = 59\%$
4. $\sin(54) = 81\%$
5. $\sin(72) = 95\%$
6. $\sin(90) = 100\%$
7. $\sin(108) = 95\%$
8. $\sin(126) = 81\%$

9. $\sin(144) = 59\%$

10. $\sin(162) = 31\%$

Now, we need to put these values into a proper sine table. For that we need to know the value of PR2.

$$\text{PWM Period} = [(PR2) + 1] * 4 * TOSC * (\text{TMR2 Prescale Value})$$

We know, PWM period, TOSC and prescale value, so putting this and rearranging gives us,

$$PR2 + 1 = \text{PWM Period} / (4 * TOSC * \{\text{TMR2 Prescale Value}\})$$

Therefore,

$$PR2 = [\text{PWM Period} / (4 * TOSC * \{\text{TMR2 Prescale Value}\})] - 1$$

*Formula derived from that given in datasheet.

Now, we need to decide at which frequency we are going to do the PWM (carrier frequency). Say, we have an oscillator of 16MHz and we decide to use a carrier frequency of 16kHz. Then, if you put in the values into the above formula, we have:

$$PR2 = [(1/16,000)/(4 * \{1/16,000,000\} * 1)] - 1$$

Giving us,

$$PR2 = 249$$

So we assign 249 to PR2. The duty cycle register is CCPR1L. When CCPR1L = 0, duty cycle = 0, when CCPR1L = (PR2 + 1), duty cycle = 100%. This is because (PR2 + 1) = Period

$$\text{So, } CCPR1L / (PR2 + 1) = \text{Duty cycle}$$

So, the table:

1. $\sin(0) = 0\%$
2. $\sin(18) = 31\%$
3. $\sin(36) = 59\%$
4. $\sin(54) = 81\%$
5. $\sin(72) = 95\%$
6. $\sin(90) = 100\%$
7. $\sin(108) = 95\%$
8. $\sin(126) = 81\%$
9. $\sin(144) = 59\%$
10. $\sin(162) = 31\%$

Can be rearranged as:

1. $\sin(0) = 0 * 250$
2. $\sin(18) = 0.31 * 250$
3. $\sin(36) = 0.59 * 250$
4. $\sin(54) = 0.81 * 250$
5. $\sin(72) = 0.95 * 250$
6. $\sin(90) = 1 * 250$
7. $\sin(108) = 0.95 * 250$
8. $\sin(126) = 0.81 * 250$
9. $\sin(144) = 0.59 * 250$
10. $\sin(162) = 0.31 * 250$

Here we multiplied by 250 as 250 is the peak of the sine wave $[y=\sin(x) * 250]$ as 250 is the period.

So we have:

1. $\sin(0) = 0$
2. $\sin(18) = 77.5$, take as 77
3. $\sin(36) = 147.5$, take as 147
4. $\sin(54) = 202.5$, take as 202
5. $\sin(72) = 237.5$, take as 237
6. $\sin(90) = 250$
7. $\sin(108) = 237.5$, take as 237
8. $\sin(126) = 202.5$, take as 202
9. $\sin(144) = 147.5$, take as 147
10. $\sin(162) = 77.5$, take as 77

There, we have now found out our sine table:

[0,77,147,202,237,250,237,202,147,77]

That's 10 values. You can repeat the process for as many values as you need.

Now, how to use this for implementation of sine wave:

$PR2 = 249$

Initially, $CCPR1L = 0$

We enable CCP1 interrupt, by writing one to CCP1IE and GIE and PEIE. The interrupt occurs at the end of every period, ie, when $TMR2 = PR2$, ie, when TMR2 counts 250 steps, ie 6.25us, ie at a frequency of 16kHz – our carrier frequency.

We set the sine table as an array. Let's take the variable as "sinval".

So,

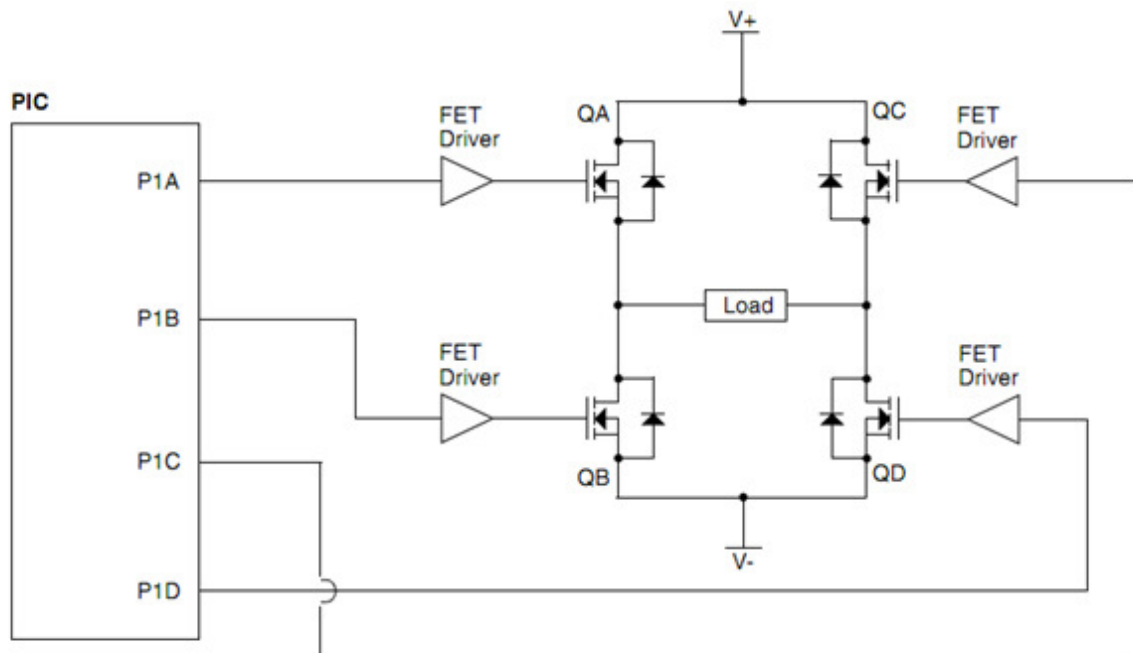
`sinval = [0,77,147,202,237,250,237,202,147,77]`

We'll take a variable to act as the array index. Let's take that as "ind". Initially, ind = 0. We'll also take another variable that signals the end of the index. Let's take that as "endi". We know there are 10 values, sinval[0] to sinval[9], so we'll assign endi 10. So, initially, endi = 10

In the ISR, we first use ind to determine the current position of the array. So, at first ind=0. We call sinval with the index being ind. So, when ind=0, we call sinval[0], which is 0. We pass this value to CCPR1L. Then, we increment the index variable "ind" and compare this against "endi". Ind does not equal endi, so we continue. Repeat the procedure until all are done.

So when ind = 9, sinval[9] is called. We have retrieved the value 77. We pass this to CCPR1L. Then after incrementing, ind=10. We always compare the value against endi. So ind = endi, so, we reverse the direction of the full bridge by toggling bit 7 of ECCP1CON, which is P1M1 or EPWM1M1.

We continue again, this time clearing ind, so that we have ind = 0 again, and when ind = endi, we reverse the direction again.



So, what we have achieved is, while at first QA and QD are on, we modulated in one direction, and then with QA and QD off and QB and QC on the other direction. If we use LC filter, we'll achieve a sine wave. But.... There is one thing remaining. The frequency isn't 50Hz. Why?

Our carrier frequency is 16kHz and in a total wave, we have 20 samples. So, the frequency is $16/20 = 0.8$ kHz = 800Hz.

800Hz is the frequency. So period is 1.25ms. For 50Hz, we need a period of 20ms. So, we must prolong our period $(20/1.25) = 16$ times. This can be done, by sending each sample 16 times, thus prolonging the period and so, we have a 50Hz frequency.

This would mean sending each value of `sinval` 16 times in the ISR and then increment the index register "ind".

So `sinval[0]` would be sent 16 times, `sinval[1]` would be sent 16 times, `sinval[2]` 16 times and so on. This means, for 16 interrupts, the same value would be passed to CCPR1L and then "ind" would be incremented.

To make the process of obtaining the sine table easy, I wrote a software in VB called "Sine Wave". This requires you to have .NET framework installed. Have the latest version of .NET 3 installed. And you can easily get a sine table. I will upload this software for free for those who want to make sine wave table for their use.

Here's a screenshot:

