AN1613
APPLICATION NOTE

Microstepping Motor Drive by using an ST52x430 ICU and the L6208

Authors: G.Pitruzzello, G.Rascona'

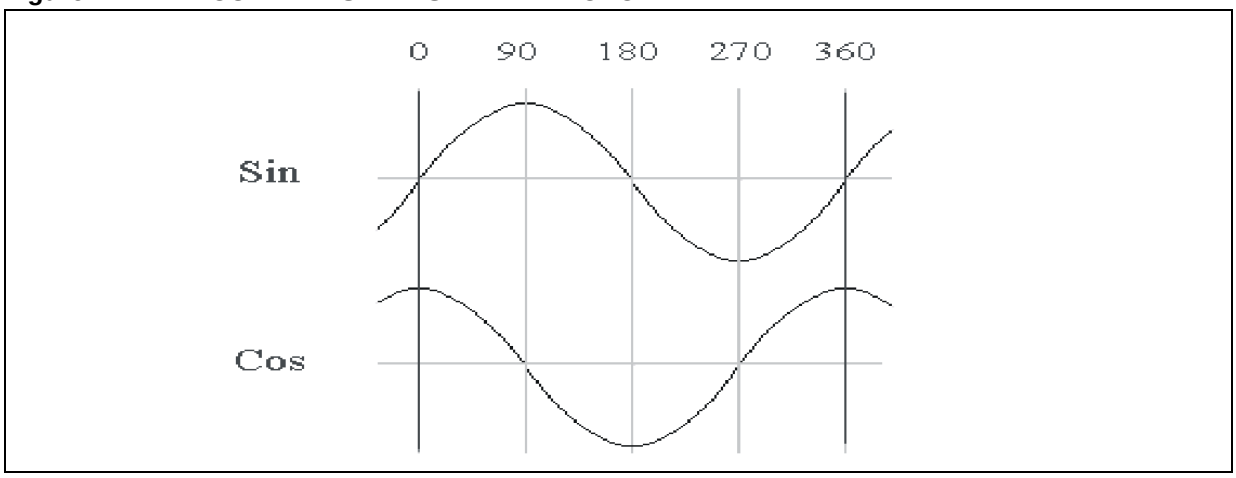
INTRODUCTION

This application note discusses a method of implementing the microstepping mode of operation to drive a stepper motor using the ST52x430 ICU. The reader should be somewhat familiar with stepper motor driving and the torque generation principles. Microstepping is a way of moving the stator flux of a stepper motor more smoothly than in full or half step drive modes, allowing the stepper motor to stop and hold a position between the full or half-step positions. As the jerky character of low stepping motor operation is reduced, there are fewer vibrations and problems with resonance, which makes noiseless stepping possible down to 0 Hz, smaller step angles and better positioning possible. Although some microstepping controllers offer hundreds of intermediate positions between steps, it is worth noting that microstepping doesn't generally offer great precision, both because of linearity problems and because of the effects of static friction. The microstepping technique can also be used to increase stepper motor position accuracy beyond the manufacturer's specification even when half and full step mode are used. A microprocessor based microstepping system is the best way to achieve this goal. Microstepping can replace or simplify gearboxes and mechanics for damping noise and vibrations. Also, motor selection becomes easier and more flexible. In a microprocessor based microstepping application, software, PWM timers or D/A converters may be used that are located within the microprocessor in order to replace an external microstepping controller to achieve the lowest possible microstepping hardware cost. The same hardware cost as in full and half step systems for similar motor sizes can be achieved. Finally, the most common problems that arise by using the microstepping technique for driving stepper motors, "microstepping position ripple" and "holding torque ripple" can be easily solved by providing a sine/cosine profile compensation.

MICROSTEPPING EXCITATION

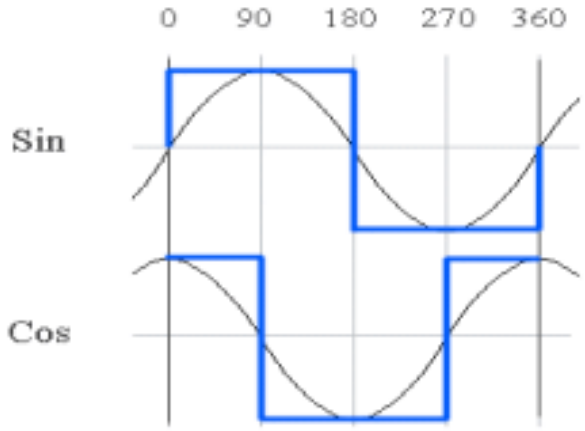
The ideal current waveform for driving a stepper motor is a sinewave. Two sinewaves in 'quadrature' (90 degrees out of phase) form the ideal drive current. If the two stepper coils follow the current waveforms shown in figure 1, the motor will run quietly and smoothly, which is the ideal condition. In fact, the "step" usually associated with stepper motors will disappear.

Figure 1. IDEAL CURRENTS IN A STEPPER MOTOR



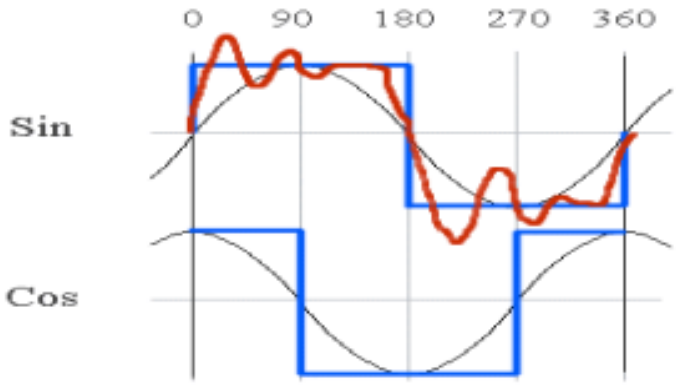
The sine/cosine waves allow the motor to move continuously from one pole (whole step position) to the next. As one coil increases in current, the other coil decreases. The rotor advances smoothly and the torque output is continuous at any given position. The torque stays the same for any angle because the current is always correctly proportional between the 2 coils. Now, let's compare this with current waveforms of the typical bipolar stepper driver (in Hi-Torque mode).

Figure 2. COMPARED TO TYPICAL WAVEFORMS



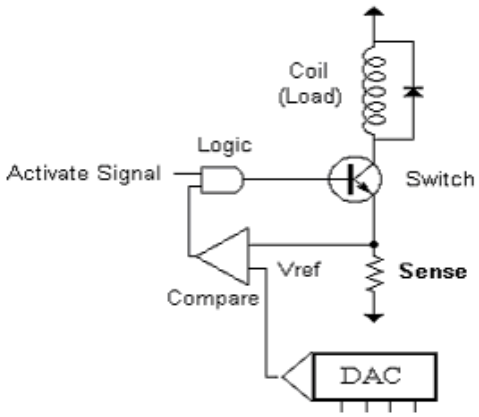
This is obviously not sinusoidal, as there are no smooth transitions. However, this is still useful for typical stepper applications, which only need to step the motor in full steps where the motor can (to a smaller degree) hold its position without current applied, using detent magnetic torque. The stepper driver attempts quantum leaps between coil states, which correspond to the motor's natural full-step positions. However, let's look at the real-world example of a coil excited by a square wave voltage; the resulting current waveform is more like what is shown in figure 3, with current overshoot and ringing, which can occur at "natural" frequencies of the motor and/or mechanical system to which it is attached. This phenomenon is known as "resonance" and can be a real problem in mechanical systems. Microstepping helps reduce resonance problems because it doesn't allow the current levels to get "out of control" in this manner. Clearly, the sinusoidal waveforms of diagram 1 would be preferable.

Figure 3. COMPARED TO REAL WAVEFORMS



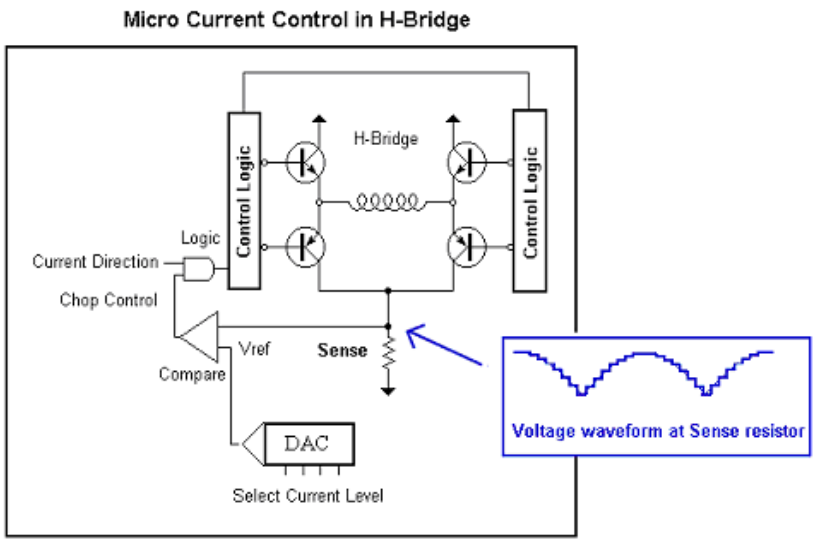
For this purpose, let's consider the circuit shown in Figure 4, a DAC (digital-to-analog converter), which can be used as voltage reference. The reference voltage can be set digitally and therefore the current can be regulated through the coil in discrete steps. The current in the coil is developed as a voltage across the sense resistor. Discrete steps are not quite as good as a smooth sinewave, but if the steps are small enough the waveform approaches a sinewave.

Figure 4. DIGITAL CURRENT CONTROL USING A DAC MODEL



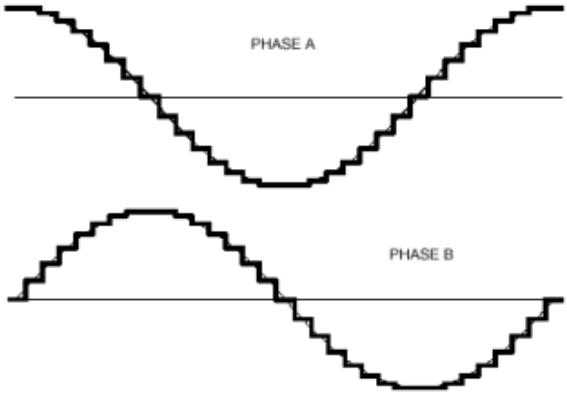
A controller that can generate fractional steps using this method of current regulation is called a *Microstepping driver.* A Microstepping driver can sub-divide the motor's step angle into many subdivisions (4 or more). Figure 5 is an example of a Microstepping driver (1/2 of the circuit is shown), a chopping H-Bridge with digital current control. This circuit controls one coil of the stepper motor, while an identical circuit controls the other coil. Current direction and current levels are synchronized to create the desired sine/cosine quadrature waveforms. The voltage waveform at the sense resistor is a rectified sinewave (absolute value of sinewave), since current is never negative in the sense resistor. Recall how the H-Bridge reverses the direction of current in the coil. It is usually convenient to look at the voltage level at the sense resistor to see the current waveform.

Figure 5. CURRENT CONTROL USING AN H-BRIDGE



By providing the bridge circuit with two 90 degree out of phase unipolar voltage waveforms, the current waveforms in the two coils are shown in figure 6.

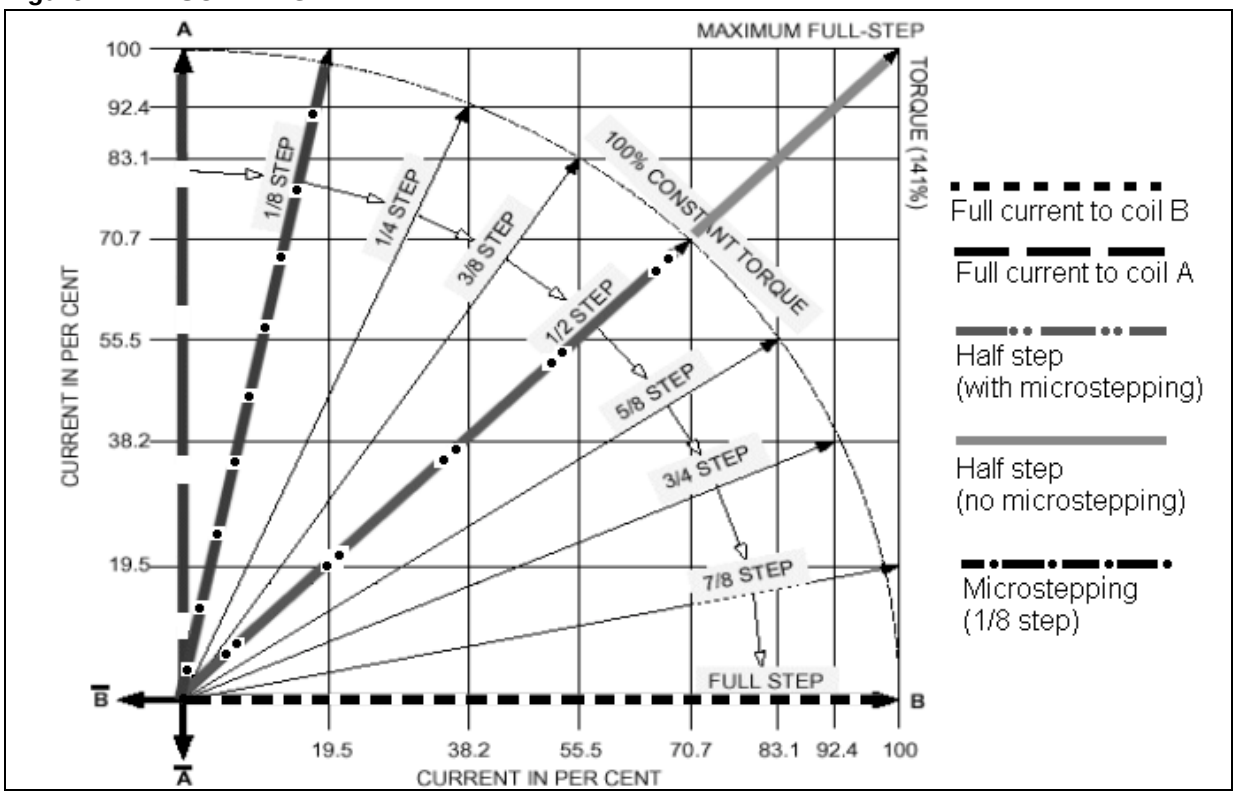
Figure 6. PHASE CURRENTS IN A MICROSTEPPER



PHASOR DIAGRAMS

Let's now look at what current ratios are needed to produce a particular step angle. The microstep angle can be graphically represented with a "phasor diagram" (see Figure 7). The X and Y axis indicate the current level in two respective coils A and B. A vector (ray from origin to coordinates X,Y) shows the resulting angle and Torque (magnitude of the vector) when current is applied to both coils. This diagram shows the 'sub-angle' between natural whole steps (poles) of the motors, which on a typical 200 step per revolution motor is 1.8 degrees. The graph below is an example of how that angle can be sub-divided further.

Figure 7. PHASOR DIAGRAM



Try to locate the following vectors and understand what they represent:

Dashed Line

Full current to coil B only. Rotor points to a natural 'pole' at a whole step position.

Long dash

Full current to Coil A only. Rotor points to a natural 'pole' also at a whole step position.

Dash-Dot-Dot Line

Now consider the "dashed double dotted line" vector. The angle is the same as the "continuous line" vector (rotor points midway between the natural poles), but the torque is now the same magnitude as the whole step positions. This was achieved by reducing the current by .707 (one half the square root of 2) in each coil. The result is even torque at the half step positions, precisely what we want.

Continuous Line

The "continuous line" vectors are the most important to grasp. These vectors show what happens when you take a "half" step with an ordinary (non-microstepping) controller by fully activating both coils. The angle is correct, since rotor points midway between the natural poles, but the magnitude (torque) is greater than at the whole step positions (dashed dotted line vectors). The magnitude is 1.414 times (square root of two) greater. The result is uneven torque, with resultant surges in current usage with each half step.

Dash-Dot line

Now carry this concept one step further and look at the "dashed dotted line" (1/8 step) vector. The current in the coils is precisely proportional to achieve an eighth step between the whole step positions with equivalent torque. You can tell that the torque is equivalent since the magnitude (length) of this ray is the same as the whole step vectors. All the microstep vectors have the same magnitude.

The phasor diagram summarizes how the microstepping controller uses the current levels in each of the two coils to achieve the desired sub-angle. The minimum microstep angle that one can take depends on the precision of the motor and the microstep driver. Generally speaking, 1/8th to 1/16th step is a good compromise that can be achieved with ordinary motors and a good controller. Let's look at the current ratios needed to achieve a particular 1/8th step angle.

Table 1. STEP SUBDIVISION 1/8th CURRENT RATIOS

Step Series	Current Phase A	Current Phase B	Step Position
0	100.0%	0.0%	full step
1	98.1%	19.5%	1/8 step
2	92.4%	38.2%	1/4 step
3	83.1%	55.5%	3/8 step
4	70.7%	70.7%	1/2 step
5	55.5%	83.1%	5/8 step
6	38.2%	92.4%	3/4 step
7	19.5%	98.1%	7/8 step
8	0.0%	100.0%	full step
x†	100.0%	100.0%	1/2 step

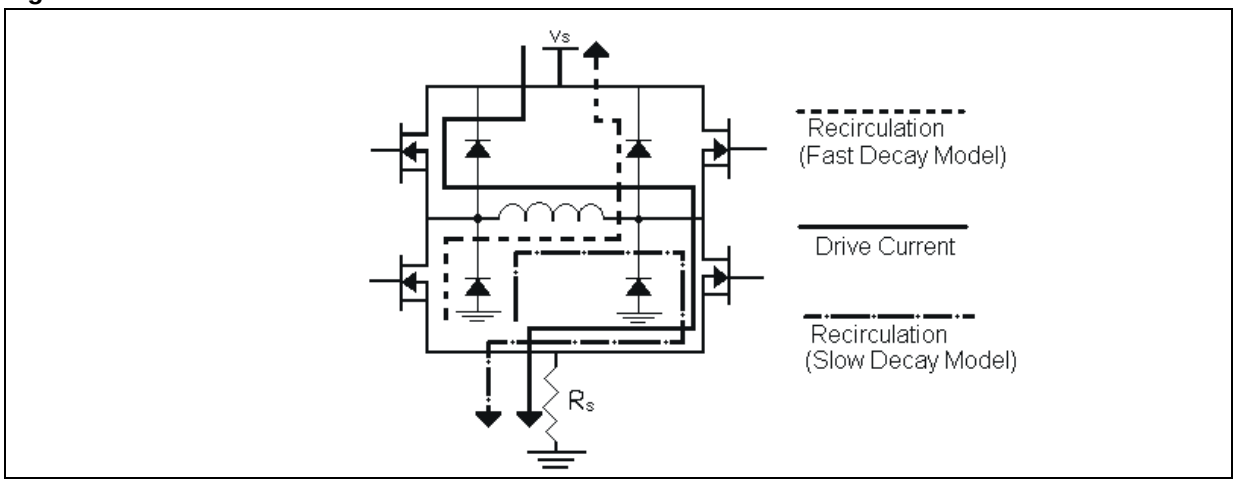
† 2-Phase drive at 100% ≈ 141% torque.

The numbers that are expressed are pretty precise. It takes a 3 bit linear DAC to get within 3% of these sinusoidal values. Now all the percentages can be represented with only 3 bits. A 98.1% value is thrown out and rounded to 100% in order to fit the 9 values in 3 bits (8 values), which is a reasonable compromise. As we will see, our microcontroller based system eliminates the need of using the DAC to get these voltage references, simplifying the complexity of the microstepping driver, reducing the overall system cost and allowing the user any other best suited value for these voltage references. The number of microsteps acting on the software stored in the microcontroller must be changed immediately without causing a change in the hardware.

CURRENT DECAY-RECIRCULATION MODES

The chopping current driver turns the H-Bridge sections on and off in order to energize the motor coil. This is shown as the continuous line path in the diagram above. When the appropriate transistors are activated, the current flows through the coil in the desired direction (continuous line). Now, when the current achieves a level that is too high (as developed across the sense resistor at the bottom), the chopping driver turns the coil off, in order to regulate the current. There are two possible ways (fig. 8): the dash-dot line provides for a slow decay, the dashed line allows a fast decay. One path or the other are chosen according to the motor speed and the location in the sinusoidal curve. The objective is to follow the sinewave as close as possible. When the sinewave is at a peak, the rate of change is 0, therefore the decay should be slow, but when it crosses zero it should be faster.

Figure 8. CURRENT DECAY-RECIRCULATION MODES

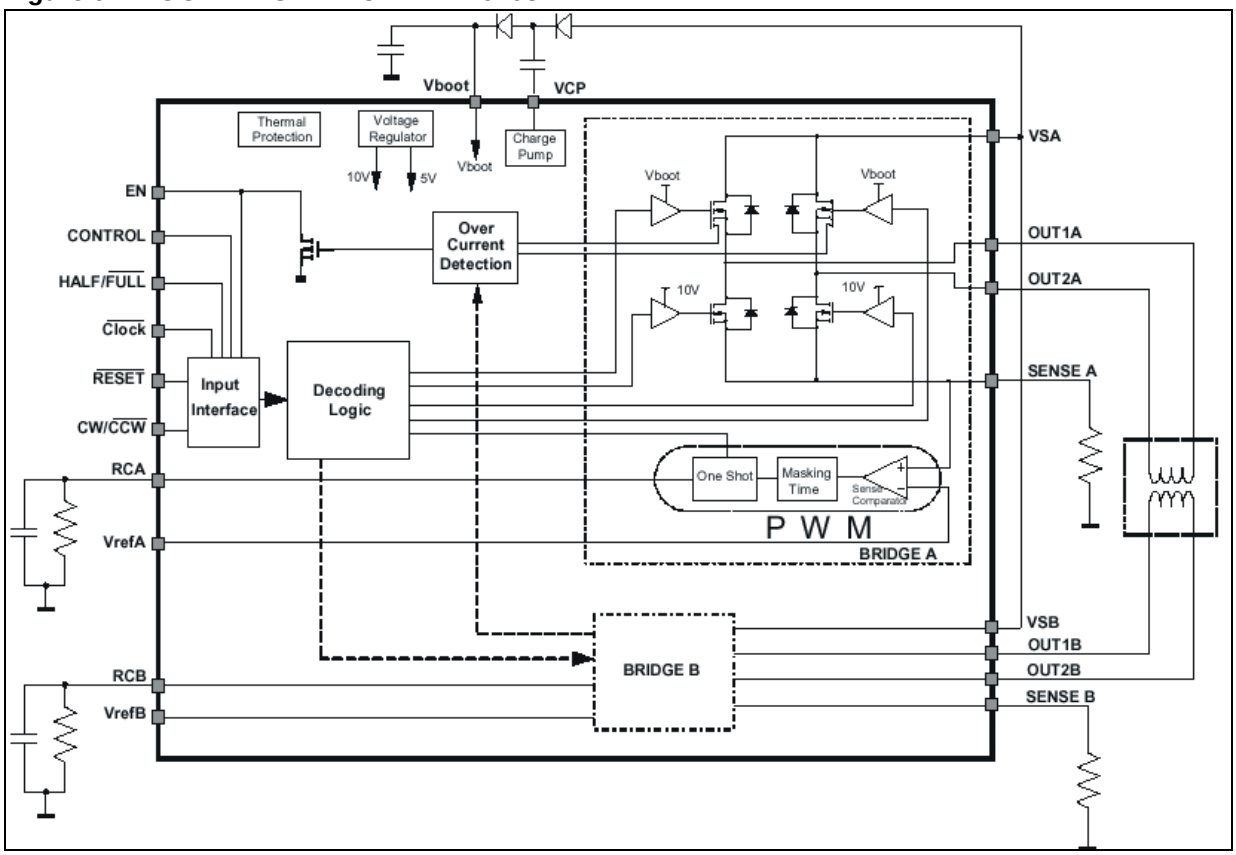


In general, slow decay makes for quieter operation. However, if the decay is too slow, sine tracking is lost and resonance problems could occur. When the decay is too fast, the motor is noisy and the efficiency of the chopper is decreased since it's switching more often in order to regulate the current. The best solution is a dynamic method, in which the decay mode tracks the current of change of the sinewave. This allows the most precise tracking and involves the least amount of "tuning" to a particular motor, which however, is somewhat tricky to implement. The decay "trajectory" must be modified in each quadrant and must actually be inverted depending on which direction the motor is spinning. A microcontroller is the best suited solution to implement such a "dynamic" solution.

THE L6208 IC STEPPER MOTOR DRIVER

The L6208 is a fully integrated bipolar stepper motor driver with two full bridges having power DMOS transistors. All the circuitry to implement the phase generation (decoding logic) is integrated, as well as a "constant Toff PWM control" for the current, separately for any of the two windings of the driven motor. Below is the block diagram of the IC.

Figure 9. BLOCK DIAGRAM OF THE L6208 DRIVER



The internal decoding logic generates all the signals for implementing the full (one or two phase ON) and half step modes to drive a stepper. The constant Toff PWM current control consists of a sense comparator and a monostable. When the current in each phase of the motor reaches the value set by the corresponding Vref voltage (Vref/Rsense) it will be forced to decrease for a constant Toff time, set by the RC network applied to the RC_A and RC_B pins.

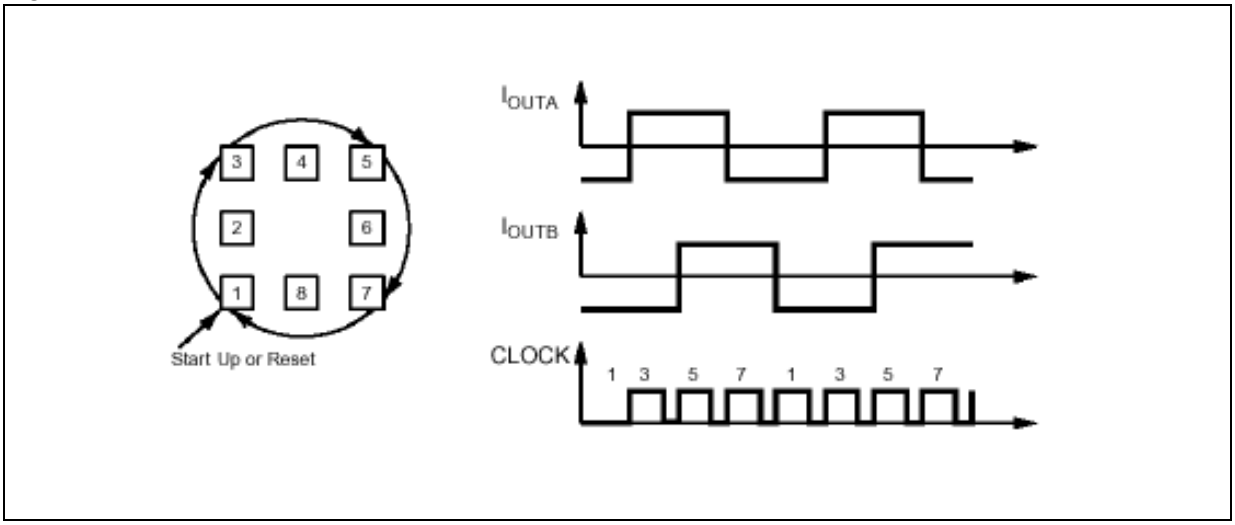
If the "Control" pin is at High logic level, during the off-time the voltage applied to the motor phase will be approximately 0V, turning the high-side Mosfets of the bridge ON (slow decay recirculation); if control is Low, instead, the voltage applied to the phase will be reversed, turning the low side Mosfet OFF that was on and turning on the opposite low-side (fast decay recirculation).

In our system the principle to get the microstepping driving mode of operation is developed by setting the L6208 to generate the current waveforms shown in figure 2 and letting the voltage references change in the sinusoidal manner as shown in figure 5 by using the ST52x430 microcontroller as reference generator. The "normal drive mode" has to be selected for the stepper driver to achieve this purpose.

CONNECTING THE ST52X430 WITH L6208 FOR GENERATING THE MICROSTEPPING MODE OF OPERATION (1/8TH STEP)

If the “normal drive mode” (Full step two-phase-on) was chosen for the L6208, the following ideal waveforms are found in the two currents flowing in each coil of the motor:

Figure 10. NORMAL DRIVE MODE



The two currents in the two motor coils always differ by zero and can only assume their rated (positive or negative) values according to the current control strategy imposed by the internal logic of the L6208. These values are voltages and are provided at the two inputs labeled “Vrefa” and “Vrefb”. In this manner, there are only four possible configurations for the two currents as shown in figure 10. The rotor will be moved one step per time for each transition in the currents (these changes are periodically produced by an external clock signal).

But, if inputs Vrefa and Vrefb are allowed to change in a sinusoidal manner (unipolar) as shown in figure 5, then the position assumed by the rotor will be a fractional part of one complete step. For nine different voltage level references there will be nine different positions for the rotor within one complete step (these positions were described by the phasor diagram). In this manner, a 1/8th microstepping driver is achieved and similarly, changing the number of levels (by software), any other number of microsteps could be developed.

Therefore, the result that is obtained by providing the nine levels to the Vrefa and Vrefb inputs with an appropriate external clock signal are the current waveforms shown in figure 6.

HOW TO PRODUCE THE VOLTAGE REFERENCES FOR VrefA and VrefB INPUTS

In order to obtain a voltage reference at pin 'VrefA' (or similarly at VrefB) for implementing a 1/8th microstep driver, a filtered PWM signal at 78 KHz generated by the fully programmable Timer/PWM peripheral was used. The nine levels listed in table 2 are simply obtained by setting as many values for the duty cycle of the PWM signal. The table below shows the nine duty cycle values used to generate all the eight microsteps in which each step is subdivided. As stated above, a step is swept by the rotor when the electrical angle of the current varies by 90 degrees. Therefore, 1/8th of step is obtained by dividing the total step electrical angle by 8:

$$\mu StepAngle = (StepAngle)/8 = 90/8 = 11.25°(electrical)$$

Table 2. Duty cycle values used in the program to generate the voltage reference levels

Step Angle	degrees	sin	VrefA (V)	Duty cycle
0	0	0	0	0
1/8	11.25	0.195	0.975	50
2/8	22.5	0.382	1.91	97
3/8	33.75	0.555	2.775	142
4/8	45	0.707	3.535	180
5/8	56.25	0.831	4.155	212
6/8	67.5	0.923	4.615	235
7/8	78.75	0.98	4.9	250
8/8	90	1	5	255

By filtering the PWM signal (via a low pass filter), each one of the above listed values of the duty cycle will give one of the nine levels of the sinusoidal voltage reference and beginning from this one, the current control strategy implemented in the L6208 will produce a sinusoidal current in the motor coil.

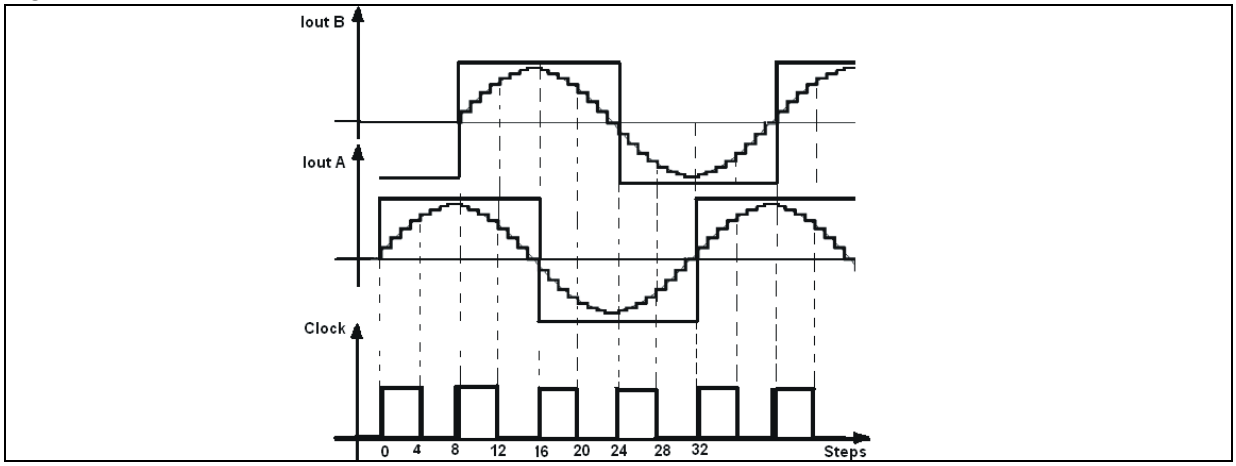
Note: while the reference voltage levels are always positive values, the decoding logic of the L6208 driver programmed for working in "full step mode", generates a step current that can be both positive or negative according to the quadrant in which it is working.

The same concept applies to produce the waveform for the other VrefB inputs. The only important factor is that a 90 degree phase displacement be kept between these two signals.

CLOCK TIMING

Figure 11 shows how the voltage reference and the current in a motor coil vary according to the clock signal at the input of the L6208. The clock signal is generated by the microcontroller firmware and obviously, it affects the motor speed (one motor step every clock pulse).

Figure 11. CURRENTS DURING MICROSTEPPING AND CLOCK SIGNAL



SIMPLE CRITERIA FOR CHOOSING THE LOW PASS FILTER TYPE

If a PWM signal is inserted in the VrefA pin (or VrefB), the reference voltage level filtering has to be as strong as possible according to the rate of change of the reference itself. The procedure described below could be used to calculate the cut frequency for a first order RC filter (20 dB/dec).

Name the PWM frequency as f_{PWM} and suppose that you want a ripple of 10% maximum. This means that the cut frequency has to be fixed one decade lower than f_{PWM}:

$$f_{CUT1} = f_{PWM}/10$$

In microstepping mode it is necessary to sinusoidaly change the voltage reference in time and the rate of these changes related with the motor speed. If an additional 10% error can be accepted for each reference level, the cut frequency can be further reduced by a factor of 2.3:

$$f_{CUT2} = f_{CUT1}/2.3$$

In this manner for a 1/8th microstep, the step frequency will be equal to:

$$f_{STEP} = f_{CUT2}/8$$

In a motor with a number of steps per round equals to NStep, the motor speed is:

$$Speed = f_{STEP}/N_{STEP}Hz$$

Looking back at the previous formulas, we see that the maximum final speed (with rounding performed), depends on the frequency of the PWM once the N_{STEP} has been fixed by the motor characteristics.

A second order filter (LC or a double RC) should be preferred to improve the performance either in terms of maximum motor speed or in terms of minimum ripple (better reference level definition).

An LC low pass filter (with a PWM frequency of 78 KHz) with the following characteristics was chosen:

$$L = 2700\mu H$$
$$C = 4.7\mu F$$
$$f_{CUT2} = 1.4kHz$$

Whereas, the sample motor we used has the following characteristics:

number of phase:	2;
step angle:	1.8 degrees;
winding resistance:	6.6 ohm;
holding torque:	3.2 Kgf*cm

The N_{STEP} is 200 steps/round and with that LC cut frequency the maximum motor speed we obtained was 52 rpm.

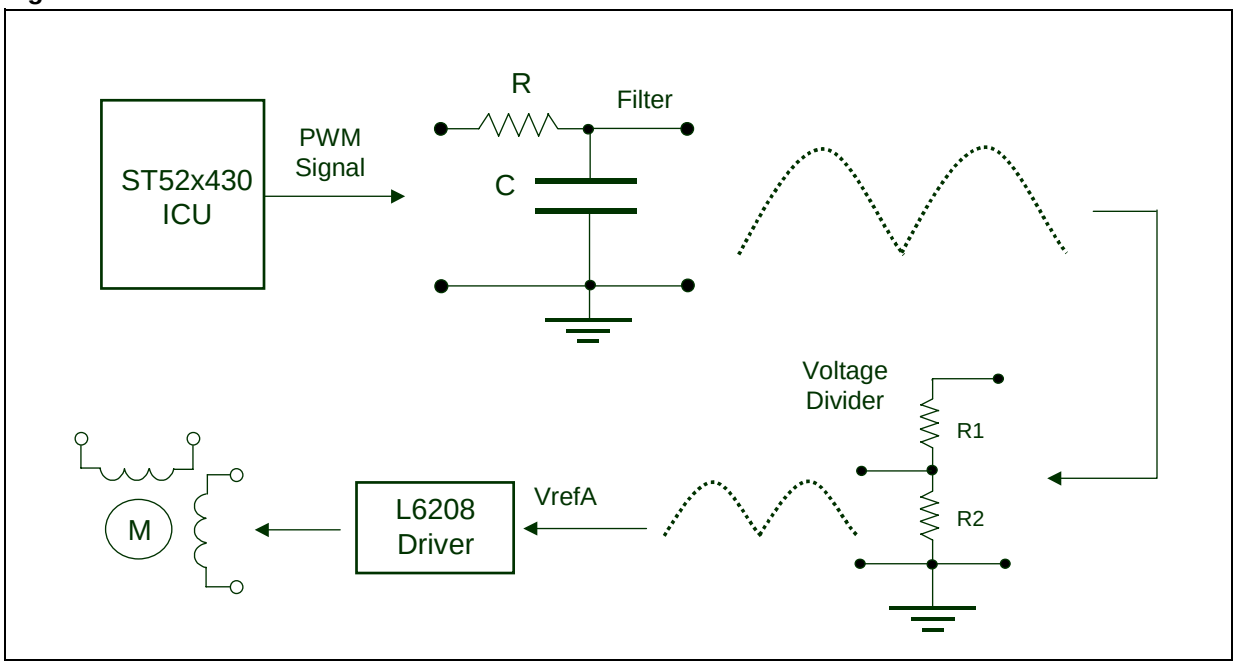
HOW TO COMPARE THE REFERENCE VOLTAGE GENERATED BY THE MICROCONTROLLER WITH THE VOLTAGE GENERATED BY R_{SENSE}.

By looking at the functional block diagram of figure 9 we see a “sense comparator” with two inputs. This component is the most important part involved in the “Toff constant” current control strategy performed by the L6208. In its normal mode of operation, this IC limits the maximum value of current flowing on the motor coil by means of this “sense comparator”, comparing the values present at its two inputs: one of them is the current in the motor coil read through a sense resistor, R_{sense} and the other is a voltage level set by an external source. In our case, this latter value is provided by the microcontroller and is formed by the nine level voltage waveforms discussed above.

The two signals entering the sense comparator have to match one another. According to the hardware motor characteristics, the adequate R_{sense} is fixed. The peak current is calculated for each coil from the rms current value absorbed by the motor. Multiplying this value for the R_{sense} we achieve the maximum voltage level ($V_{ref,\ max}$) that the sense comparator can read to ‘+’ input. The input ‘-’ coming from VrefA should not exceed this latter value.

Looking at table 2, we see that the maximum output voltage from the filter is 5V that could be much greater than $V_{Ref,\ max}$. If this is the case, the need is to reduce the voltage levels coming from the low pass filter, adding a voltage divider for scaling the reference levels to the appropriate values. Figure 12 shows these concepts graphically.

Figure 12. BLOCK DIAGRAM FOR MICROSTEPPING

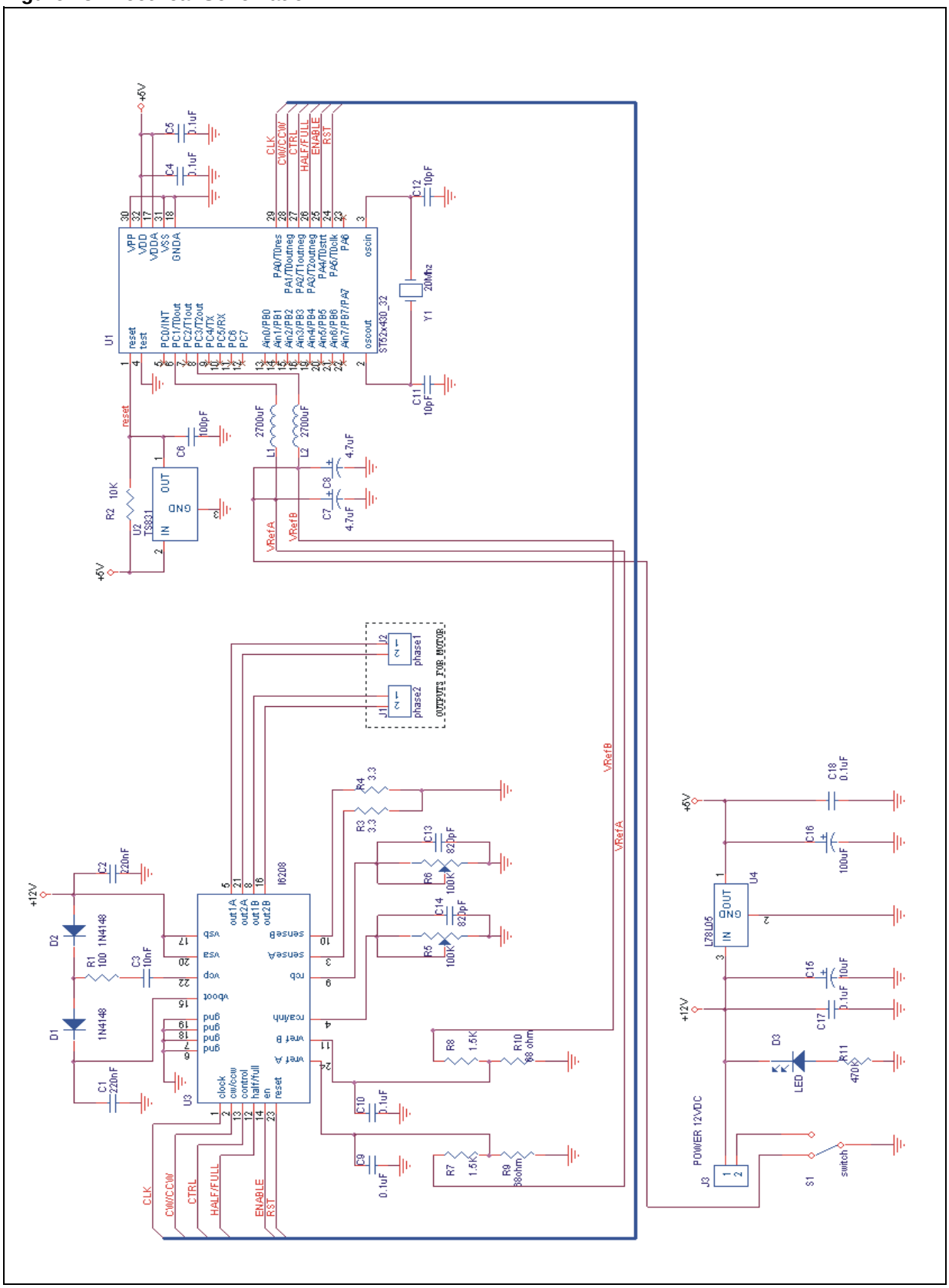


SCHEMATIC OF THE APPLICATION

Figure 13 shows the electrical schematic of the application. The ST52x430 microcontroller provides the PWM signal to the LC filters via the PC1/T0out and PC3/T2out pins, whose outputs generate the voltage reference at the Vref_A and Vref_B input of the L6208 driver. The LC filter output is scaled by two resistive voltage dividers (R7/R9 and R8/R10). The ST52x430 provides other signals to the L6208 for configuring the motor operations via some pins of PORT_A:

- **PA0** **CLK** = the clock signal;
- **PA1** **CW/CCW** = set the rotation direction (clockwise and counterclockwise);
- **PA2** **CTRL** = set the decay mode (Fast or Slow decay);
- **PA3** **HALF/FULL** = set the Half or Full step mode;
- **PA4** **ENABLE** = enable the L6208 driver;
- **PA5** **RST** = reset the driver;

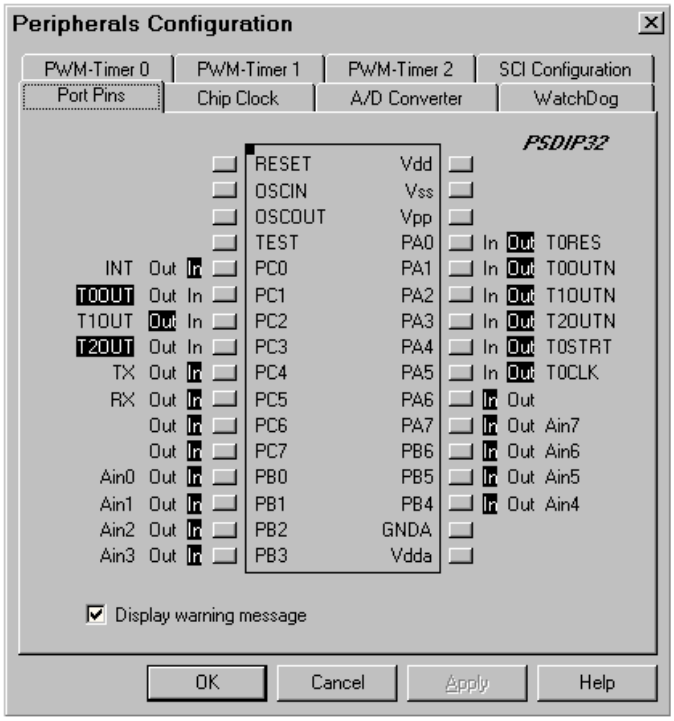
Figure 13. Electrical Schematic



SOFTWARE DESCRIPTION

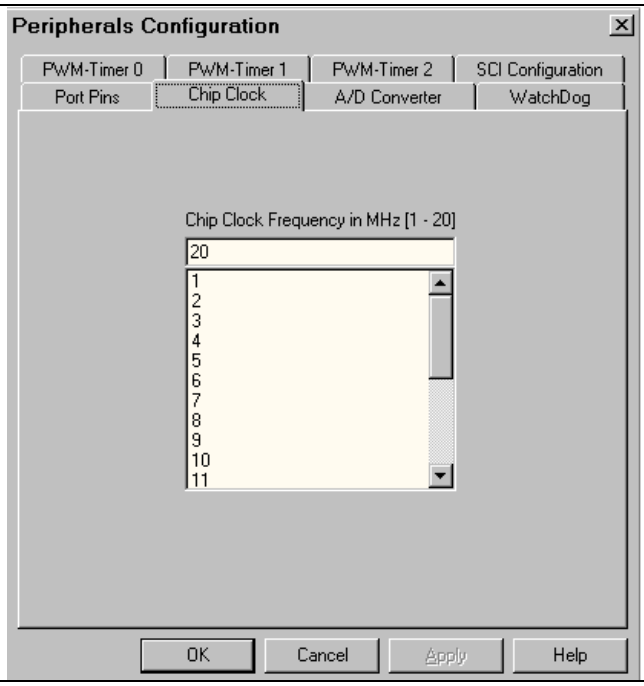
Visual FIVE, a very easy to use tool for programming the micro's of the ST52 family, was used to write the simple firmware for producing the signals to provide the L6208 driver. This tool uses a visual approach to configure and to program all the micro's functions. A series of windows below will show the various steps to be performed for programming the ST52x430.

Figure 14. PORT PIN CONFIGURATION



In order to obtain a PWM signal with an oscillator frequency of 78KHz (see figure 16) the system clock was set to 20 MHz (see Figure 15).

Figure 15. CLOCK MASTER CONFIGURATION



Figures 16 and 17 show the windows to configure one timer for counting each microstep duration and two PWM timers to generate the duty cycle pattern files.

Figure 16. PWM CONFIGURATION

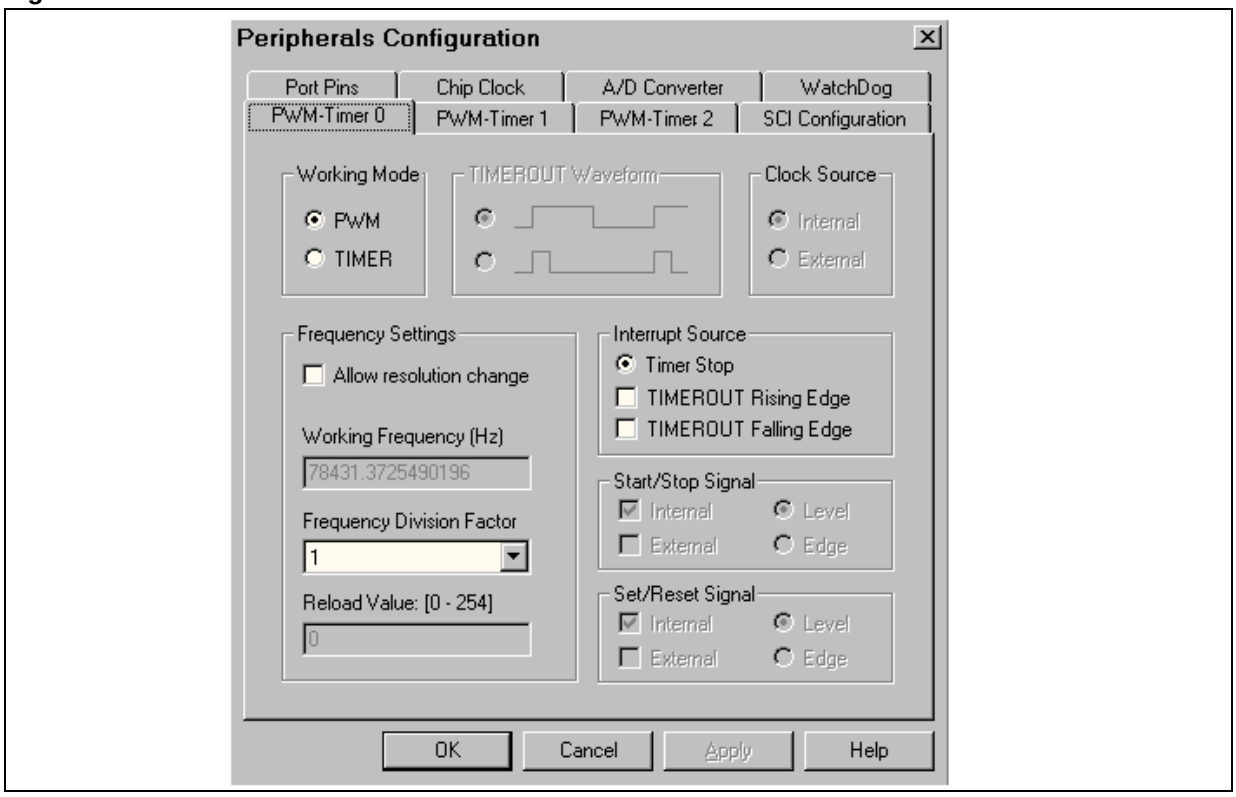


Figure 17. TIMER CONFIGURATION

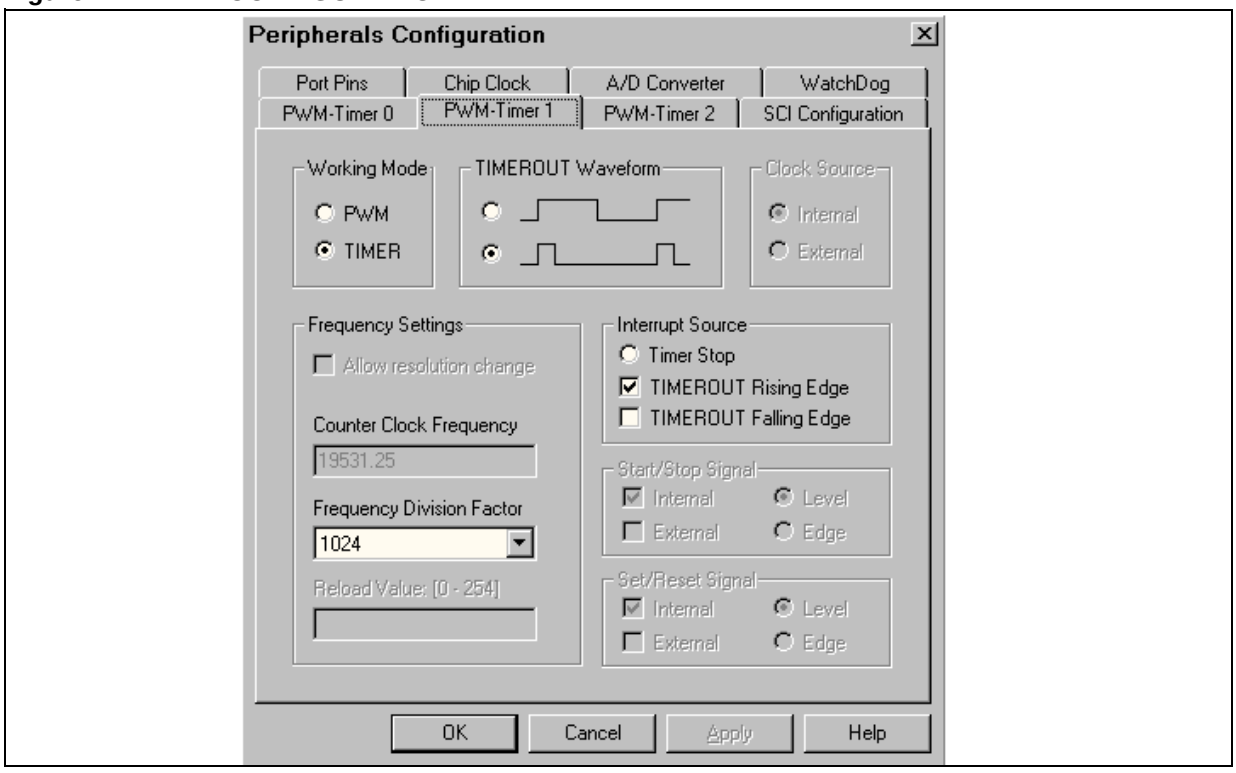
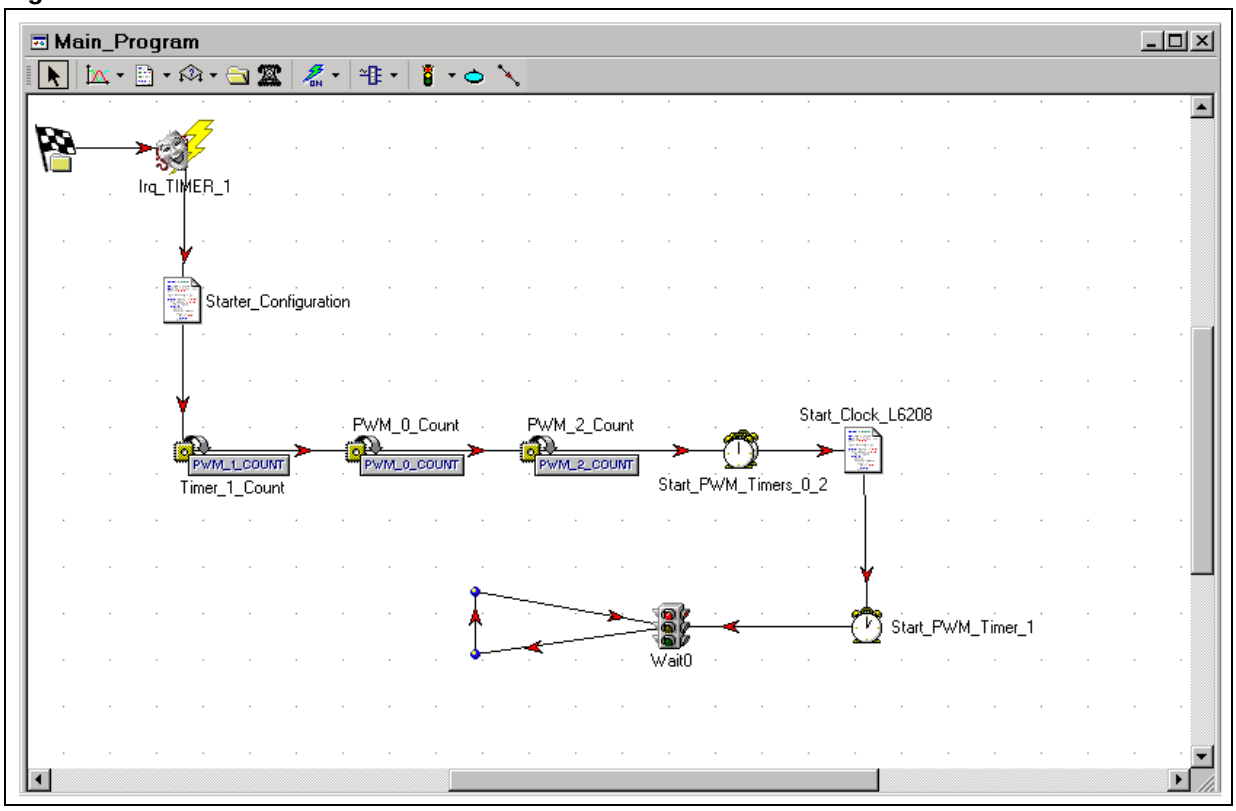


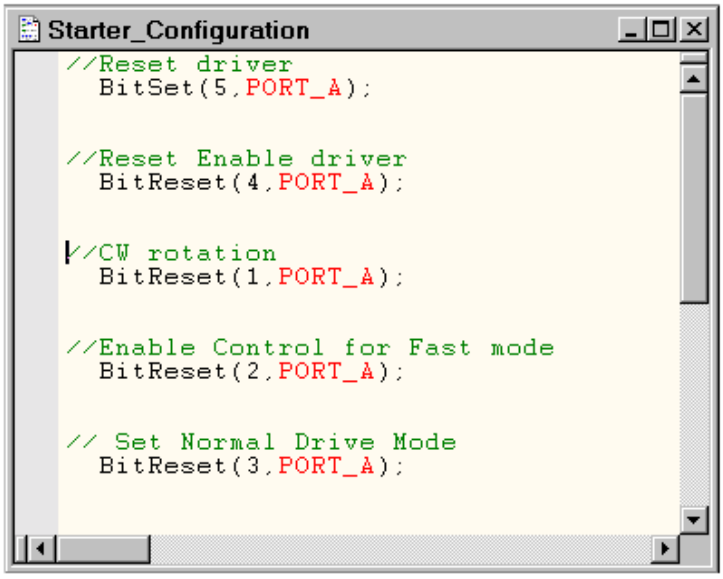
Figure 18 shows how the main program operates. The **Irq_Timer** block enables the micro to receive an interrupt from Timer_1 (microstep duration). In our example this time interval equals 1.64 ms.

Figure 18. MAIN PROGRAM



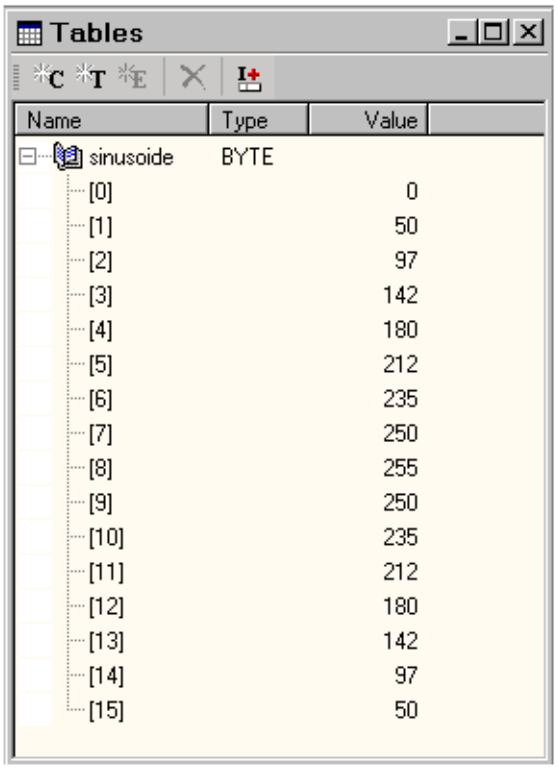
As shown in Figure 19, the **Starter_Configuration** block allows the micro to configure the L6208 driver.

Figure 19. CONFIGURATION OF THE L6208



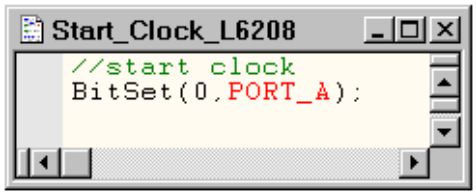
The **Timer_1_Count** block sets the right value for the internal **Timer_1** register for counting the 1.64ms. The following two blocks '**PWM_0Count'** and '**PWM_2_Count'** set the initial value for the duty cycle in each phase. All other values for the duty cycle are stored in a table that develops the unipolar sinusoidal profile voltage reference (see Figure 20).

Figure 20. PWM SIGNAL VALUES



The **Start_PWM_Timers_0**, **Start_PWM_Timer_2** start the two PWM peripherals and **Start_Clock_L6208** gives the first rising edge to the clock input of the L6208 (see Figure 21).

Figure 21. START CLOCK FOR THE L6208



Wait0 waits for an interrupt coming from the TIMER1 peripheral (every 1.64ms). Figure 22 shows the code of the TIMER interrupt routine. In this routine the two PWM duty cycles are refreshed with the values stored in the table.

Figure 22. Vref AND CLOCK SIGNAL GENERATION

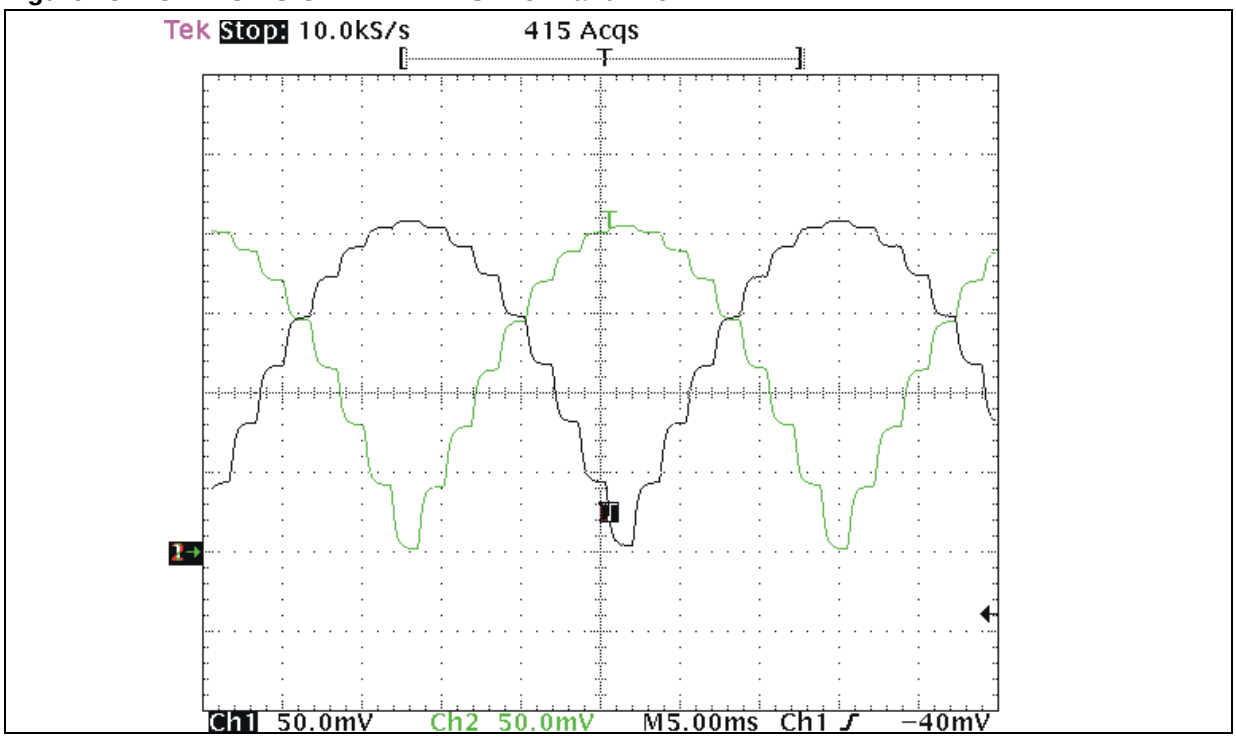
Voltage_ref_and_Clock

```
//output of the micro
PWM_2_COUNT= sinusoide[indice];
PWM_0_COUNT= sinusoide[indice90];


//clock driver
if(clk==4 || clk==12)
  BitReset(0,PORT_A);

if(clk==0 || clk==8)
  BitSet(0,PORT_A);
clk++;


indice++;
indice90++;


if(clk>15)
  clk=0;


if(indice>15)
  indice=0;

if(indice90>15)
  indice90=0;
```

EXPERIMENTAL RESULTS

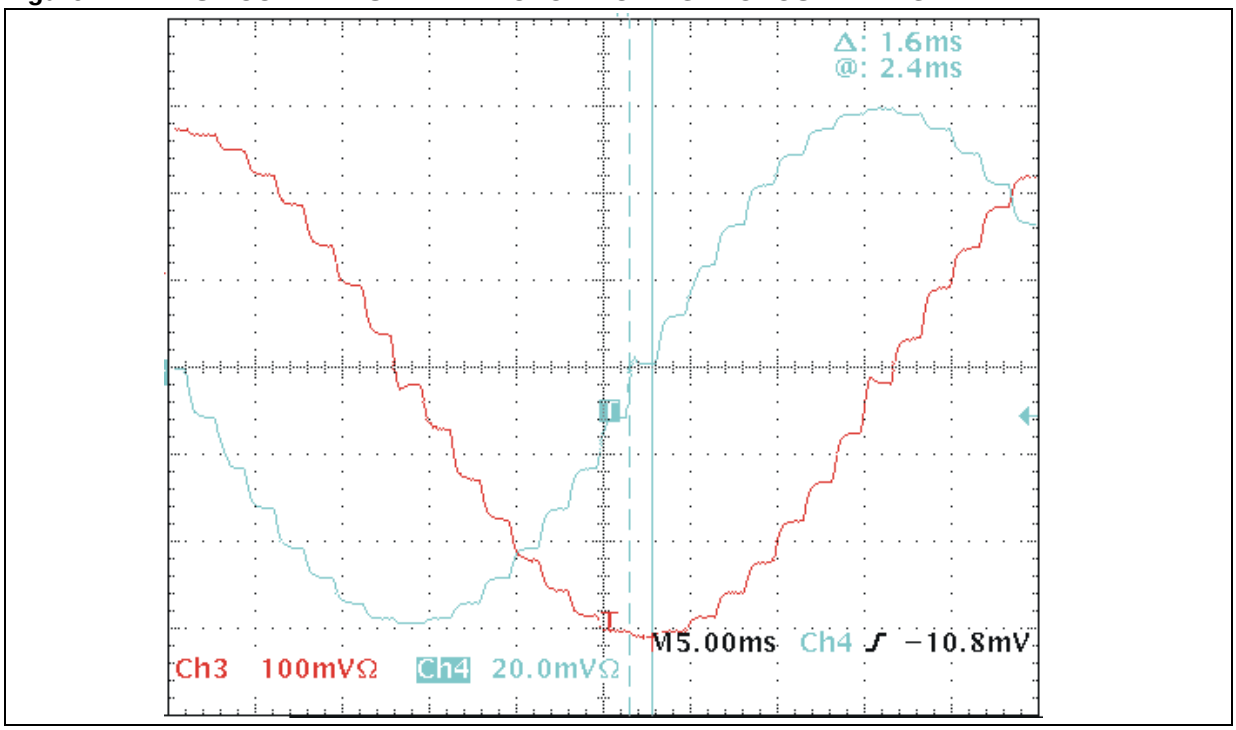
The following graphs show the waveforms obtained with the hardware and software described above for a 1/8th microstepping driving mode. In figure 23 the voltage reference levels measured at the output of the voltage divider and applied at the two inputs VrefA and VrefB of the L6208 can be seen.

Figure 23. VOLTAGE SIGNAL AT PINS VrefA and VrefB



The two phase currents in the motor coils are shown below.

Figure 24. PHASE CURRENTS IN THE MOTOR DURING MICROSTEPPING



CONCLUSION

A 1/8th microstepper driver implementation has been shown in this application note, using a micro of the ST52 ICU family (ST52x430). Although it's relatively easy to find several devices on the market that develop the same function, our approach offers more flexibility than a dedicated IC. By simply modifying the firmware of the micro, the most common issues relative to this type of motor can be resolved. Moreover, while the micro drives the stepper motor, many other operations can be performed inherent to the motor, or not, by using all the other resources that the ST52x430 possesses: A/D converters, SCI interface and so on.

REFERENCES

1. ST52x430 Datasheet
2. Visual FIVE 5.0, STMicroelectronics, 2002
3. L6208 stepper motor
4. Desoer-Kuh. "Fondamenti di teoria dei circuiti"
5. "Power Electronics" by Ned Mohan, Tore Undeland and William Robbins.