Freescale Semiconductor
Application Note

Document Number: AN3817
Rev. 1, 2/2009

Interfacing Stepper Motor with MC9S08LG32

by: Sunaina Srivastava
Reference Design and Applications Engineering
Microcontroller Solutions Group

1 Introduction

The MC9S08LG32 is a low cost microcontroller and can be applied to several applications using different input signals. This application note:

- Describes how to control a low current stepper motor using the MC9S08LG32.
- Explains how to drive a stepper motor when there is no stepper motor driver within the MCU.
- Provides an example that illustrates the hardware connections and software driver for a stepper motor driven through PWM.
- Describes how to configure MC9S08LG32 to drive a stepper motor.
- Describes how the input signal is managed, implemented, and used.

Figure 1 shows the block diagram for the MC9S08LG32 and stepper motor interface. The MC9S08LG32 contains 69 GPIOs (80-pin package) and eight channels of TPM

Contents

that can be used to drive a stepper motor. You can operate the stepper motor in the following modes when interfaced with the MC9S08LG32:

- GPIOs — All four stepper motor pins are connected to the GPIOs of the MC9S08LG32. The waveform required to move the motor in a clockwise or anti-clockwise direction is provided through the GPIOs.
- TPM + GPIOs — Various channels of TPM can be used in combination with GPIOs to drive the motor. TPM channels are configured to provide the required waveform in synchronization with the GPIOs.

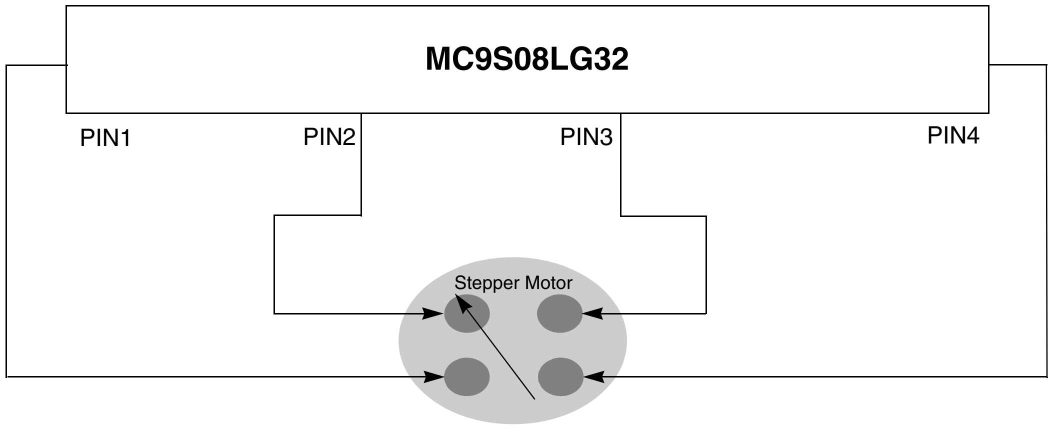

Figure 1. MC9S08LG32 and Stepper Motor Interface Block Diagram

2 Stepper Motor Management

A stepper motor is a small brushless synchronous electric motor that can divide a full rotation into a large number of steps. If it is electronically connected to the MCU, the motor's position can be controlled with precision without any feedback mechanism.

Steppers exhibit more vibration than other motor types. The discrete step tends to snap the rotor from one position to another. This vibration can cause the motor to lose torque at some speeds. The effect can be mitigated by accelerating quickly through the problem speed range, physically dampening the system, or using a micro-stepping driver. Motors with a greater number of phases have a smoother operation than those with fewer phases.

There are two basic arrangements for the electromagnetic coils: unipolar and bipolar. This application note focuses on a bipolar motor.

A bipolar motor is built with two different coils, which in this document are named coil A and coil B. Since each coil has two wires, a bipolar stepper motor has four different wires.

Bipolar stepper motors operate differently from traditional DC motors. Stepper motors have multiple toothed electromagnets arranged around a central metal gear. The electromagnets are energized by an external control circuit, such as a microcontroller. To make the motor turn, the following steps are required:

1. Coil A is connected to the power that causes the gear's teeth to be magnetically attracted to the electromagnet's teeth.

2. The gear's teeth are aligned to the first electromagnet. Next, they are aligned to the second electromagnet.
3. Coil B is then turned on and coil A is turned off.
4. The gear then rotates to be aligned with the next gear.
5. Repeat the process described in steps 1 to 4.

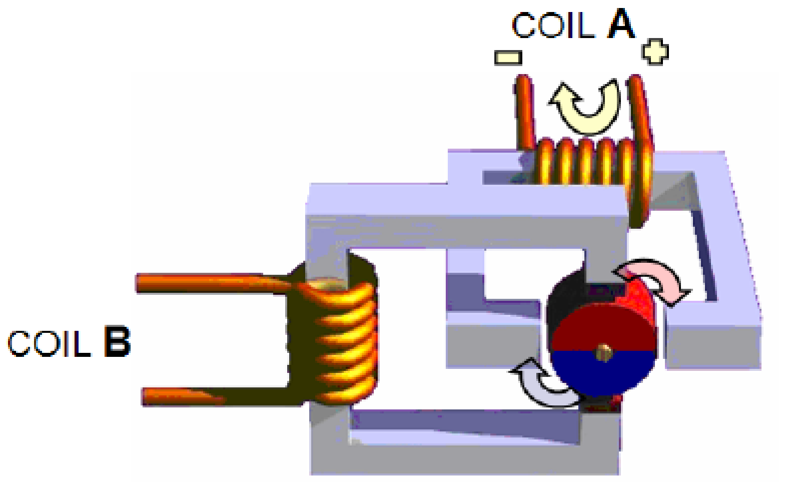

Figure 2. Stepper Motor Coil A

Each of these rotations is called a step. This is how the motor can be rotated through a precise angle.

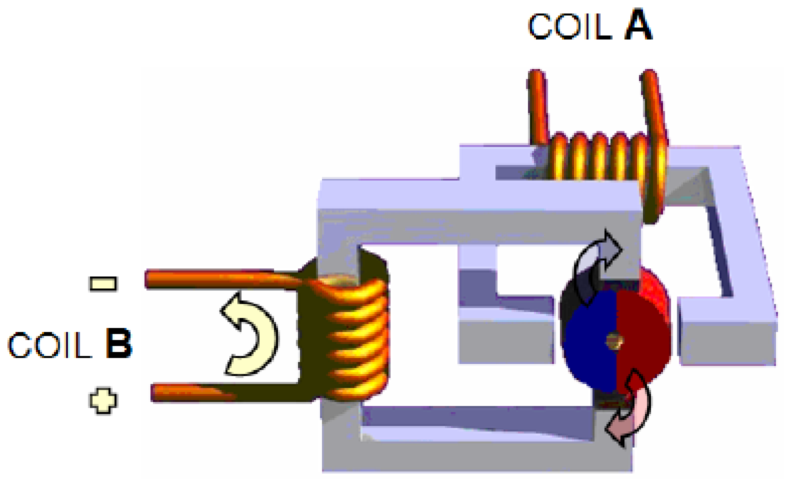

Figure 3. Stepper Motor Coil B

The main characteristic of the stepper motor used to implement this application is that it has a maximum current consumption of 20 mA. The torque provided by the motor is small. The maximum static torque = 4 mNm, with maximum dynamic torque = 1.4 mNm, which is strong enough to move standard gauges. The S08 microcontroller MC9S08LG32 can drive up to 10 mA. The motor is connected to the microcontroller through a current driver IC. The system requires V_{CC}, GND, and input signals.

3 Operation and Functionality of Stepper Motor

To set the basis, the steps of the motor are managed by the duty cycle of a PWM input signal. The percentage of the duty cycle allows the microcontroller to communicate the specific position.

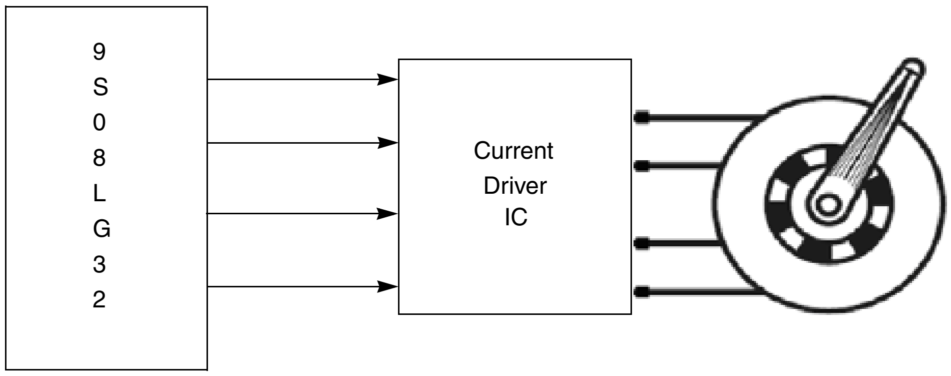

Figure 4. Complete Implemented System

Figure 5 is an example of the implemented system.

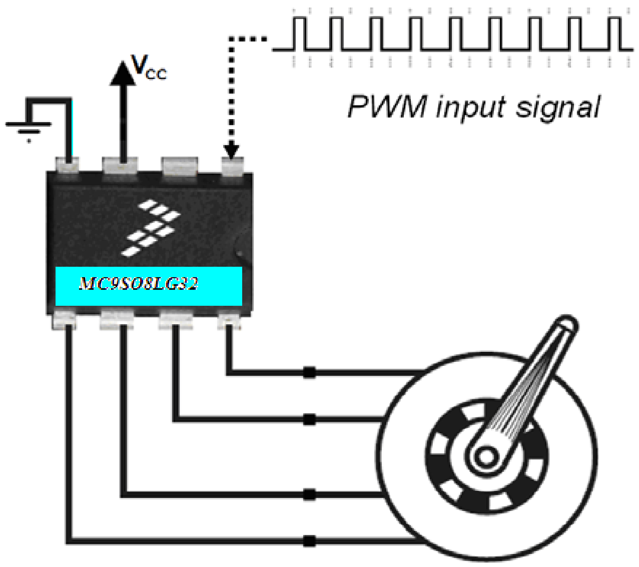

Figure 5. Complete Block Diagram of Motion Control

In the automotive industry, there are several sensors or applications that have a PWM signal as an output that is the result of a particular action or event. For example, there are complex temperature sensors that generate a PWM signal as output. Another example is an electronic central unit (ECU) that measures the speed of a vehicle and has an output pin where you can read a PWM signal. The variations of this PWM signal can be translated into positions to indicate temperature or speed, such as the application described in this document.

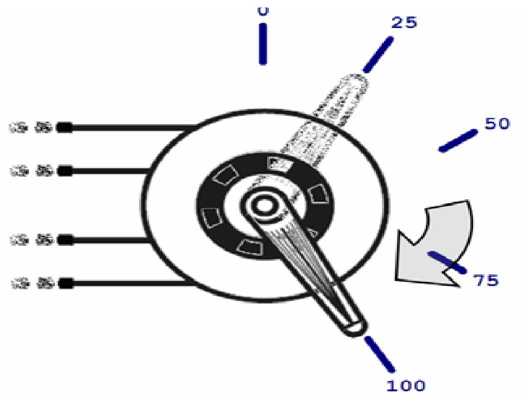

Figure 6. Stepper Motor with Gauge

To define how many steps make up one complete cycle, you must first establish how many positions are available. In this example there are 100 different positions. In case of a temperature sensor, this could be from 0 °C to 100 °C. In case of speed, this could be 0 kph to 100 kph. This means that 1% of the duty cycle is associated with one step of the stepper motor, or one degree centigrade, or one kph. See Figure 7 and Figure 8.

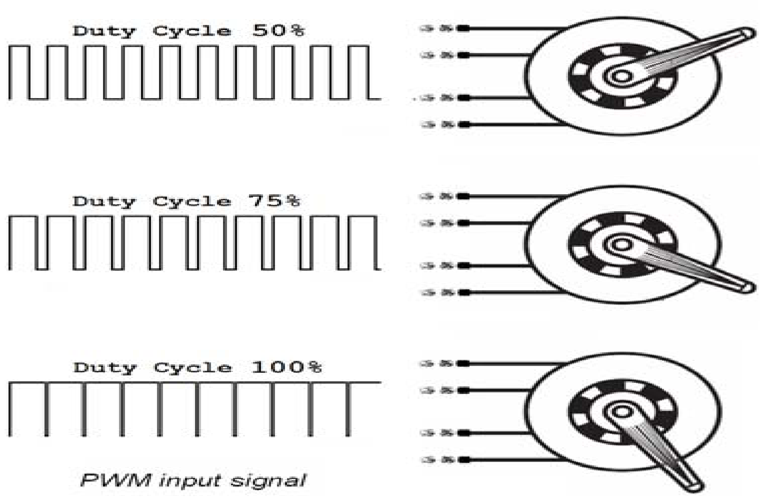

Figure 7. Stepper Motor Gauges at Different Duty Cycles

The position of the gauge depends on the duty cycle of the PWM signal. There are different ways to translate duty cycles to steps. For example, the average voltage can be measured during one period of time using the ADC of the microcontroller. This document explains how to use the timer/pulse width modulator (TPM) module.

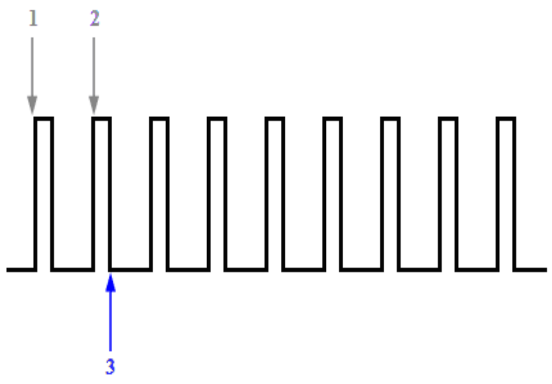

Figure 8. Input Signal Data Measurements

The measured time between point 1 and point 2 is called the frequency. The measured time between point 2 and point 3 is called Duty_Cycle_H. To find the duty cycle of the signal, calculate:

Duty Cycle = ((Duty_Cycle_H × 100) ÷ Frequency) ***Eqn. 1***

After the duty cycle is calculated, the coils of the motor are controlled through port A, channels 0 to 3. As established previously, there are one hundred positions available. This means that the required position is equal to the duty cycle.

Required Position = Duty Cycle ***Eqn. 2***

If the microcontroller detects a change in the value of the duty cycle, it moves the stepper to the required position. Each time the duty cycle varies, channels are strictly toggled in a specific sequence.

4 Hardware Interface Description

Table 1. Table1.TPM Channel / PIN Descriptions

Pins	Direction	Description
TPM1CH0/PTH5	I/O	TPM1 Channel 0/Port F GPIO
TPM1CH1/PTH4	I/O	TPM1 Channel 1/Port F GPIO
TPM2CH0/PTI5	I/O	TPM2 Channel 0/Port A GPIO
TPM2CH1/PTI4	I/O	TPM2 Channel 1/Port A GPIO
TPM2CH2/PTF0	I/O	TPM2 Channel 2/Port A GPIO
TPM2CH3/PTF5	I/O	TPM2 Channel 3/Port F GPIO
TPM2CH4/PTF4	I/O	TPM2 Channel 4/Port F GPIO
TPM2CH5/PTF3	I/O	TPM2 Channel 5/Port F GPIO

4.1 Pin Connections

The stepper motor has four pins that needs to be connected to the SoC to operate the motor in both clockwise and anti-clockwise directions. These pins are numbered 1, 2, 3, and 4, and decide the amount and direction of the rotation of the motor. Figure 24 shows the sample connection of stepper motor pins with the MC9S08LG32 SoC.

4.2 Hardware Abstraction Layer

This section describes the registers that you need to operate a stepper motor in all possible driving modes with the MC9S08LG32.

4.2.1 GPIO Registers

4.2.1.1 Port x Data Direction Register (PTxDD)

	7	6	5	4	3	2	1	0
R W	PTxDD7	PTxDD6	PTxDD5	PTxDD4	PTxDD3	PTxDD2	PTxDD1	PTxDD0
Reset:	0	0	0	0	0	0	0	0

Figure 9. Port x Data Direction Register (PTxDD)

Table 2. PTxDD Register Field Descriptions

Field	Description
7:0 PT[x]DD[y]	Data direction bit for Port x Pin y 0 Input pin 1 Output pin

4.2.1.2 Port x Data Register (PTxD)

	7	6	5	4	3	2	1	0
R W	PTxD7	PTxD6	PTxD5	PTxD4	PTxD3	PTxD2	PTxD1	PTxD0
Reset:	0	0	0	0	0	0	0	0

Figure 10. Port x Data Register (PTxD)

Table 3. PTxD Register Field Descriptions

Field	Description
7:0 PT[x]D[y]	Data bit for Port x Pin y (when configured as output pin)

4.2.2 TPM Registers

4.2.2.1 TPM Status and Control Register (TPMxSC)

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 11. TPM Status and Control Register (TPMxSC)

Table 4. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select 0 All channels operate as input capture, output compare, or edge-aligned PWM mode 1 All channels operate in center-aligned PWM mode
4–3 CLKS[B:A]	Selects the clock to TPM block 00 TPM counter disable 01 Bus clock 10 Fixed system clock 11 External source
2–0 PS[2:0]	Prescale factor select 000 1 001 2 010 4 011 8 100 16 101 32 110 64 111 128

4.2.2.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMxCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 12. TPM Counter Register High (TPMxCNTH)

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMxCNTL clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 13. TPM Counter Register Low (TPMxCNTL)

4.2.2.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 14. TPM Counter Modulo Register High (TPMxMODH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 15. TPM Counter Modulo Register Low (TPMxMODL)

4.2.2.4 TPM Channel n Status and Control Register (TPMxCnSC)

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0						▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 16. TPM Channel n Status and Control Register (TPMxCnSC)

Table 5. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n interrupt flag 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable 0 Channel n interrupt requests disabled 1 Channel n interrupt requests enabled

Table 5. TPMxCnSC Field Descriptions (continued)

Field	Description
5–4 MSn[B:A]	Mode select for TPM channel n. Refer to the summary of channel mode and setup controls in Table 6.
3–2 ELSn[B:A]	Edge/level select bits. Refer to Table 6 for the summary of these bits.

Table 6. Mode, Edge, and Level Selection

CPWMS	MSn[B:A]	ELSn[B:A]	Mode	Configuration
X	XX	00	Pin not used for TPM, but used as general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses
		X1		Low-true pulses
1	XX	10	Center-aligned PWM	High-true pulses
		X1		Low-true pulses

4.2.2.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 17. TPM Channel Value Register High (TPMxCnVH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 18. TPM Channel Value Register Low (TPMxCnVL)

4.3 Modes of Operation

A stepper motor can be driven in any of the following modes when interfaced with the MC9S08LG32:

4.3.1 4 GPIO Mode

This mode is used when no TPM channels are available for stepper mode operation. In this mode all stepper motor signals are driven through GPIOs.

When a GPIO is used for this purpose, it cannot also be used for any other functionality that is muxed with it. These pins need to be programmed as output pins, using the configuration registers.

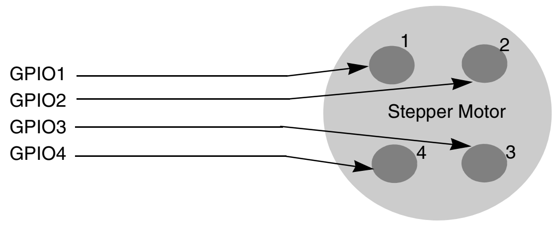

Figure 19. 4 GPIO Mode

4.3.2 2 TPM + 2 GPIO Mode

Using TPM channels and GPIOs, this mode is used when fewer TPM channels are available for stepper motor operation. In this mode, the stepper motor's two input signals are connected with two TPM channels and the remaining two inputs are connected to GPIO pins. This mode is capable of driving the motor in micro, partial, and full steps.

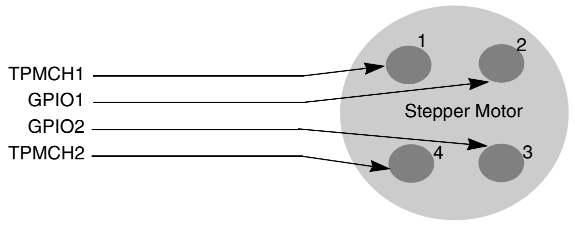

Figure 20. TPM and 2 GPIO Mode

4.3.3 3 GPIO Mode

This mode uses the 3-pin mode of the stepper motor. In this case three GPIOs are used to connect to the four pins of the motor, as the negative pins (2 and 3) of both motor coils are clubbed into one pin.

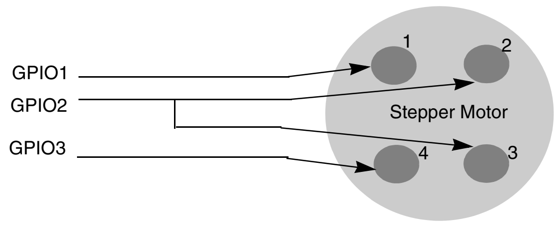

Figure 21. 3 GPIO Mode

4.3.4 2 TPM + 1 GPIO Mode

This uses the 3-pin mode (negative pins (2 and 3) of both the coils connected together) of the stepper motor. In this mode, two TPM pins are connected to positive pins (1 and 4) of both the coils and one GPIO pin is connected to the third pin of the motor.

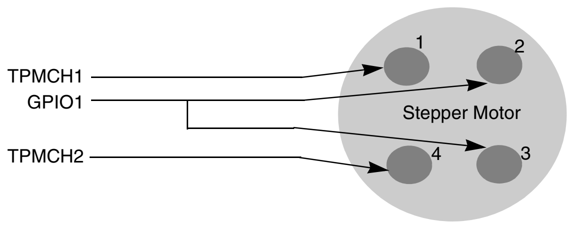

Figure 22. TPM and 1 GPIO Mode

NOTE

The most preferred mode of operation is two TPM channels and two GPIOs (see Section 4.3.2, "2 TPM + 2 GPIO Mode" for details). If you don't have enough GPIOs, then use the 2 TPM + 1 GPIO mode.

5 Software Interface Description

Following are the external APIs available, utilizing all the stepper motor functionalities, in the released software driver.

- **InitStepper** — Initializes the working mode of the stepper as PIN1–TPM1CH0, PIN2–PTI0, PIN3–PTI1, PIN4–TPM1CH1. It initializes all the required TPM and GPIO registers, but does not enable the TPM clock.
- **MotorCurrentDriverCE_Toggle** — Toggles the ChipSelect signal (active low) for the current driver IC that is required to operate the stepper motor.
- **DriveStepper_to_zero** — Follows a specified process to get the stepper to initial zero position. This function might be different for different stepper motors. It basically follows a predefined movement to avoid jittery and noisy motion.
- **Count_pulses** — Captures and counts the number of pulses coming from a digital sensor on a TPM channel that is operating in input capture mode. This measures the sensor's output-pulses value.
- **Move_motor_microstep** — Function providing movement of the stepper motor in micro-steps, which are 1/12 degree each.
- **Move_motor_partialstep** — Function providing movement of the stepper motor in partial steps, which are 1/3 degree each.
- **Move_motor_fullstep** — Function providing movement of the stepper motor in full steps, which are 1 degree each.

6 Software Operation Modes

The stepper motor can be driven in any of the following modes when interfaced with MC9S08LG32:

- Full step mode
- Partial step mode
- Micro-stepping mode
- Acceleration process at power-on or power-on reset

  In most of the VID29-XX applications, the angular range of the instrument dial is less than 300°. This allows you to use mechanical stop to define the zero position. Generally the pointer will be reset to the zero position at each power-up process. In the power-up process, we recommend a frequency acceleration process to speed up the VID29 step motor until a high speed is reached. This will quickly drive the pointer to its initial stop position without creating visible and audible jitter of the pointer.

The position of the gauge depends on the duty cycle of the PWM signal.

Figure 23 shows the waveform on the four pins of the stepper motor for its rotation in the possible modes:

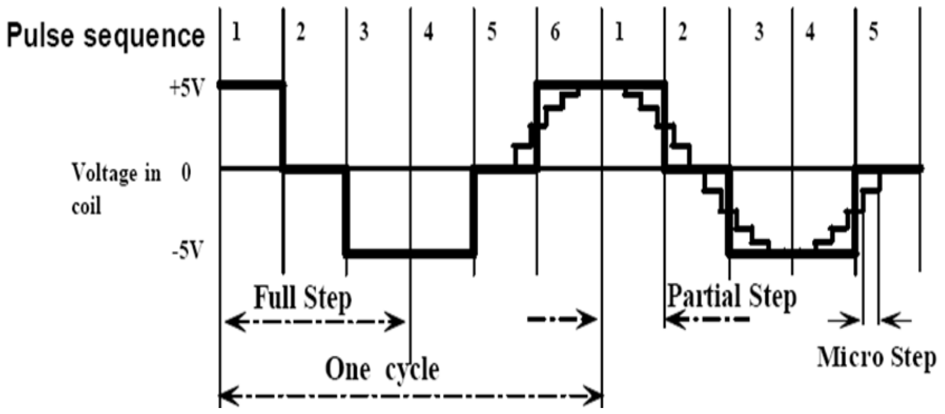

Figure 23. Driving Pulse in Full, Partial, and Micro Step

NOTE

In order to make the motor run with more stability and to decrease the noise, micro-stepping technology is recommended. The micro-pulse sequence is more precise and nearer to a sine wave than the others.

7 How to Interface Stepper Motor with MC9S08LG32

This section describes an example of interfacing the VID29-05 stepper motor with the MC9S08LG32. It shows how to configure the TPM block of the SoC to control motion of the stepper motor in 4-pin mode (2 TPM + 2 GPIO).

7.1 Description of Motor (VID29 Series)

The VID29-XX and VID29-XXP series are precise stepping motors of patent design, with a gear reduction ratio of 180/1, mainly used in dashboard instrumentation or other digital indicator equipment, and are also used to transfer digital signals directly and accurately to an analog display output. Driven by two sequential logic pulse signals in the 5 V to 10 V range, the output shaft can reach a stepping angle resolution of 1/12°.

The angular speed can reach more than 500 Hz. The new and modern design makes for high efficiency, high position accuracy, and an extremely robust gear system. The special gear shape helps to decrease friction and noise. Special materials have been chosen for each component to increase the reliability and safety of the motor.

7.1.1 Features of the Motor

- Wide working voltage: 5 V to 10 V
- Low current consumption: less than 20 mA, 5 V, 2 × 100 mW
- Wide working temperature: –40 °C to 105 °C
- Extremely robust construction: 30 mm × 7.6 mm
- High micro-step resolution: 1/12°
- Directly driven by MCU

7.2 Hardware Connections

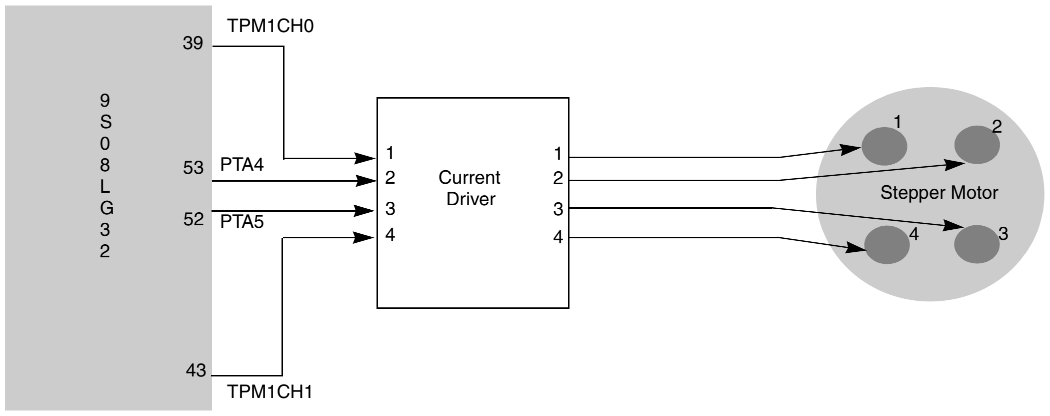

Figure 24. Configuration Example

7.3 Register Configurations

Table 7 describes the control and data configurations for the stepper motor operation in the mode shown above:

Table 7. TPM and GPIO Register Configurations

Register	Value	Configuration description
SCGC1_TPM1	1	Disable clock gating to TPM1 block
PINPS2_TPM10	0	Enable TPM1CH0 to come out at PTF1
PINPS2_TPM11	1	Enable TPM1CH1 to come out at PTH4
TPM1CNT	1	Clear the TPM1 counter
TPM1MOD	Value	Initialize the modulo register with a value of the period of the TPM cycle
TPM1C0SC_MS0x	2	Configure TPM1CH0 channels to operate in edge-aligned mode

Table 7. TPM and GPIO Register Configurations (continued)

Register	Value	Configuration description
TPM1C1SC_MS1x	2	Configure TPM1CH1 channels to operate in edge-aligned mode
TPM1C0SC_ELS0x	2	Configure TPM1CH0 channels to follow high true pulses
TPM1C1SC_ELS1x	2	Configure TPM1CH1 channels to follow high true pulses
TPM1SC_TOIE	1	Enable the TPM1 overflow interrupt
PTADD_PTADD4	1	Configure PTA4 to act as output pin
PTADD_PTADD5	1	Configure PTA5 to act as output pin

7.3.1 Initialization Procedure

You can follow these steps to configure the TPM block of the SoC to operate the motor in 4-pin mode, operating in all of the step modes, using micro, partial, and full steps.

1. Disable the clock gating for TPM module.

   SCGC1_TPM1 = 1
2. Write appropriate value to pin position register so as to take out TPM1 channels at the intended pins.

   PINPS2_TPM10 = 0

   PINPS2_TPM11 = 1
3. Clear the TPM1 counter and configure the TPM1 modulo register, to suit the stepper motor waveform period.

   TPM1CNT = 0x01

   TPM1MOD = 862
4. Configure the MS bits in both channels to operate in TPM edge-aligned mode.

   TPM1C0SC_MS0x = 0x2

   TPM1C1SC_MS1x = 0x2
5. Configure the ELS bits in both channels to follow high true pulses.

   TPM1C0SC_ELS0x = 0x2

   TPM1C1SC_ELS1x = 0x2
6. Enable the TPM1 block overflow interrupt.

   TPM1SC_TOIE = 1
7. Configure both the GPIOs to operate as output pins.

   PTADD_PTADD4 = 1

   PTADD_PTADD5 = 1

After the above configuration, the stepper motor is configured properly to be operated in micro, partial, and full step modes.

8 References

See S08LG Product Summary Page for more information and the documents released for MC9S08LG32.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3817
Rev. 1
2/2009