AN11483

How to design and program the PCA9629A advanced stepper motor controller

Rev. 1 — 27 June 2014 Application note

Document information

Info	Content
Keywords	1 MHz Fast-mode Plus (Fm+) I2C-bus, PCA9629A advanced stepper motor controller, programmable GPIOs
Abstract	The PCA9629A is a highly integrated design for stepper motor control to off-load CPU usage and enhanced stepper motor drive control logic with three pulse width drive formats plus ramp-up and ramp-down features. The motor drive outputs can program one-phase (wave drive), two-phase, and half-step drive format logic level outputs for stepper motor control modes as well as General Purpose Output (GPO) in bypass mode. There are four additional General Purpose Input/Outputs (GPIOs) — two for interrupt based motor control on P0 and P1 inputs, and two for general purpose on P2 and P3.

Revision history

Rev	Date	Description
1.0	20140627	Application note; initial release.

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

1. Introduction

Stepper motors are brushless and very reliable, since there are no contact brushes inside the motor. Therefore, the life of the motor is simply dependent on the life of the mechanical bearing. A stepper motor is a digital version of the electric motor that moves one step with one digital input pulse at a time. The rotor moves in discrete steps as commanded, rather than rotating continuously like a conventional motor. When the stepper motor is stopped but energized, a stepper motor holds its load steady with a holding torque and has full torque at standstill.

With stepper motors, precise positioning and repeatability of movement are achievable. They offer excellent response to starting, stopping, and reversing. Since the motors respond to digital input pulses, open-loop control of the motor is possible, making the motor simpler and less costly to control. Due to these reasons stepper motors are extensively used in many applications including gaming, robotics, industrial control, toys, medical equipment, door control, security cameras, vending machines, printers, etc. This application note explains how to design and program PCA9629A to drive unipolar stepper motor. Driving a bipolar motor needs additional external logic and is not explained in this application note.

2. Overview of PCA9629A advanced stepper motor controller

The PCA9629A is a 1 MHz Fast-mode Plus (Fm+) I^2C-bus controlled low-power CMOS device that provides all the logic and control required to drive a four-phase unipolar stepper motor as shown in Figure 1. PCA9629A can drive up to 25 mA/5 V outputs and is intended to be used with external high current drivers to drive the motor coils that need higher driving current and voltage. The PCA9629A supports three stepper motor drive formats: one-phase (wave drive), two-phase, and half-step. The PCA9629A does not support micro-stepping mode.

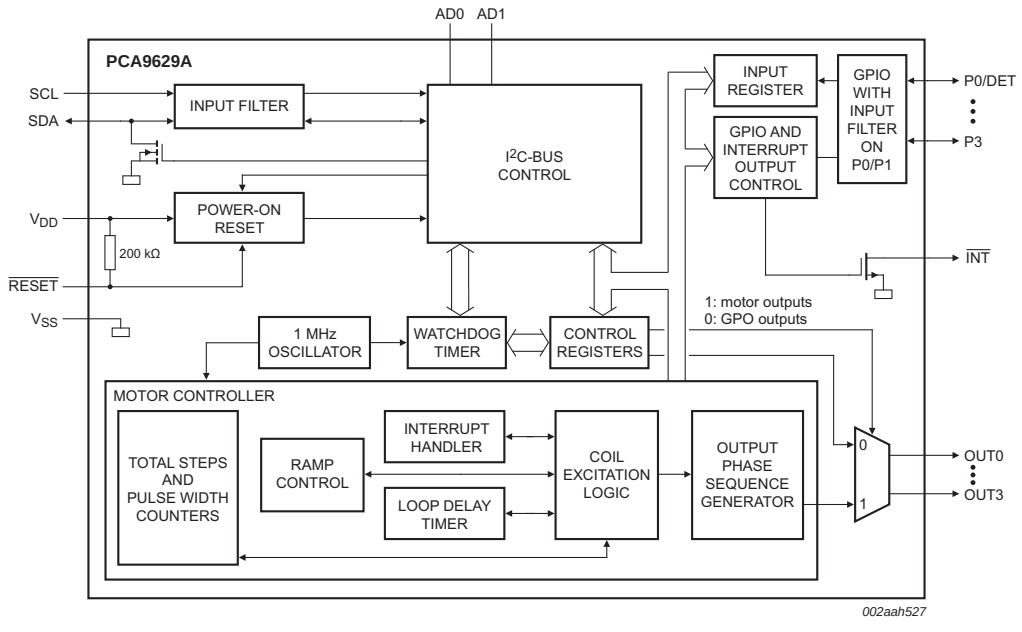


Fig 1. PCA9629A block diagram

Output wave train is programmable using control registers. All control registers are programmed via the two-wire I^2C-bus. Features built into the PCA9629A provide highly flexible control of stepper motor, off-load bus master/micro and significantly reduce I^2C-bus traffic. These include control of step size, number of steps, number of actions, and direction of rotation per single command. Restart motor to change to new speed and operation without waiting for the motor to stop. A ramp-up from motor start and/or ramp-down to motor stop are also provided with re-enable ramp-up or ramp-down to change the ramp rate curve on the fly. PCA9629A can be programmed to stop the motor automatically, restart motor, enable extra steps or reverse the direction of rotation of motor based on the P0 and P1 of GPIO inputs to generate interrupt for motor control.

2.1 PCA9629A key features

- Highly integrated design for stepper motor control to off-load CPU usage
- Enhanced stepper motor drive control logic with Pulse Width Modulation (PWM) technique for ramp-up, ramp-down and motor speed control
- Generate one-phase (wave drive), two-phase, and half-step drive format logic level outputs for stepper motor control modes
- Built-in oscillator (1 MHz) requires no external components
- Four programmable GPIOs (P0 to P3) with filter on P0 and P1 inputs to generate interrupt for initiating motor stop, restart, reverse or extra steps
- Interrupt-based motor control from P0 and P1 inputs to perform extra steps, reverse of direction, restart and stop motor without microcontroller handling
- Programmable step pulse width to control speed of motor (step rate 333.3 Kpps to 0.3 pps with ±3 % accuracy)
- Programmable motor action in multiple times in the range of 1 to 255 or continuously based motor operation settings
- Programmable start, emergency stop, extra steps, ramp-up/ramp-down or reverse the direction of rotation control of stepper motor without microcontroller interactions required
- Generate an interrupt when motor stop; no polling necessary to off-load CPU bandwidth
- Motor outputs OUT[0:3] can be configured as general purpose outputs to support bypass mode
- Dual loop delay timers for motor reversal mode to allow asymmetrical delay in motor reverse operation
- Programmable restart motor to change new speed and operation on the fly without stopping motor
- Programmable re-enable ramp-up or ramp-down to change ramp rate curve on the fly
- Single command to bring motor in home position from P0 input state
- Selectable active hold (last state), power-on or power-off for motor brake/stop control and time-out timer for overheat protection
- 16 programmable slave addresses using two address input pins
- All Call address allows programming and operation of more than one device at the same time with the same parameters
- Hardware active LOW $\overline{\text{RESET}}$ input to recover from bus stuck condition

- 32-bit step counter (clear after read) to count output step pulses continuously
- Programmable watchdog timer with option to generate interrupt, reset device or stop motor
- Input voltage range from 4.5 V to 5.5 V and −40 °C to +85 °C operating temperature
- ESD protection exceeds 2000 V HBM and 1000 V CDM
- 16-pin TSSOP16 package

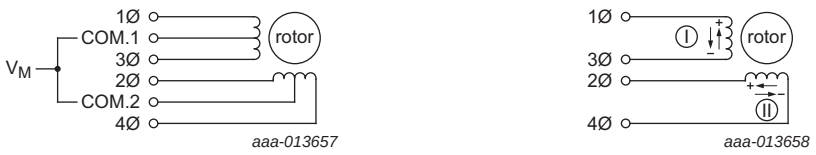
3. Hardware design example for driving unipolar stepper motor

3.1 Principle of unipolar and bipolar stepper motor operations

A unipolar stepper motor has two identical sets of windings, each winding with center tap that is tied to a power supply and the ends of the coils are alternately grounded. The current is allowed to flow in one direction only through the motor winding, and it is referred to as a ‘four-phase motor’. Each section of windings is switched on for each direction of magnetic field.

The torque output of the unipolar wound motor is lower than the bipolar motor (for motors with the same winding parameters) since the unipolar motor uses only 50 % of the available winding while the bipolar motor uses the entire winding as shown in Figure 2. Unipolar stepper motor features are:

- Required five or six wires
- Also called 4-phase
- The current is allowed only to flow in one direction through the motor windings
- For high speed application



a. Unipolar motor

b. Bipolar motor

Fig 2. Motor winding configurations

Bipolar motors have no center taps. The advantage to not having center taps is that current runs through an entire winding at a time instead of just half of the winding. As a result, bipolar motors produce more torque than unipolar motors of the same size. The drawback of bipolar motors, compared to unipolar motors, is that more complex control circuitry is required by bipolar. Current flow in the winding of a bipolar motor is bidirectional. A control circuit, known as an H-bridge, is used to change the polarity on the ends of one winding. Every bipolar motor has two windings, therefore, two H-bridge control circuits are needed for each motor.

Bipolar stepper motor features are:

- Required four wires
- Also called 2-phase (one winding/phase)
- The current is allowed to flow in both directions through the motor windings
- For high torque application

3.2 Example design circuit to drive unipolar stepper motor using a PCA9629A

Figure 3 shows an example application circuit to control a 12 V, 1.25 Amps unipolar stepper motor (48 steps per rotation). The PCA9629A stepper motor controller has a fixed I^2C-bus slave address 40h (jumper setting on J1) and is controlled by microcontroller (master device) for motor operation. The 4-channel high-current, high voltage Darlington driver with inverter is capable of handling continuous current of 1.25 A required by the motor coils and more than 12 V supply voltage, four logic level 25 mA push-pull outputs (OUT0 to OUT3) are from PCA9629A to the Darlington driver inputs (I1 to I3) and inverted outputs (O1 to O3) to drive the unipolar motor.

For proper operation of GPIO pins as input or output in this example, user must program P0 and P1 as inputs with optional filter to suppress noise on P0 or P1 input to allow sensing of logic level output from optical interrupter modules and generate active LOW interrupt signal on the $\overline{INT}$ pin of PCA9629A. This is a useful feature in sensing home position of motor shaft or reference for step pulses. Upon interrupt, the PCA9629A can be programmed to stop the motor automatically, restart motor, enable extra steps or reverse the direction of rotation of motor. Both P2 and P3 are programmed as outputs (25 mA push-pull) to drive external LEDs for status indicators.

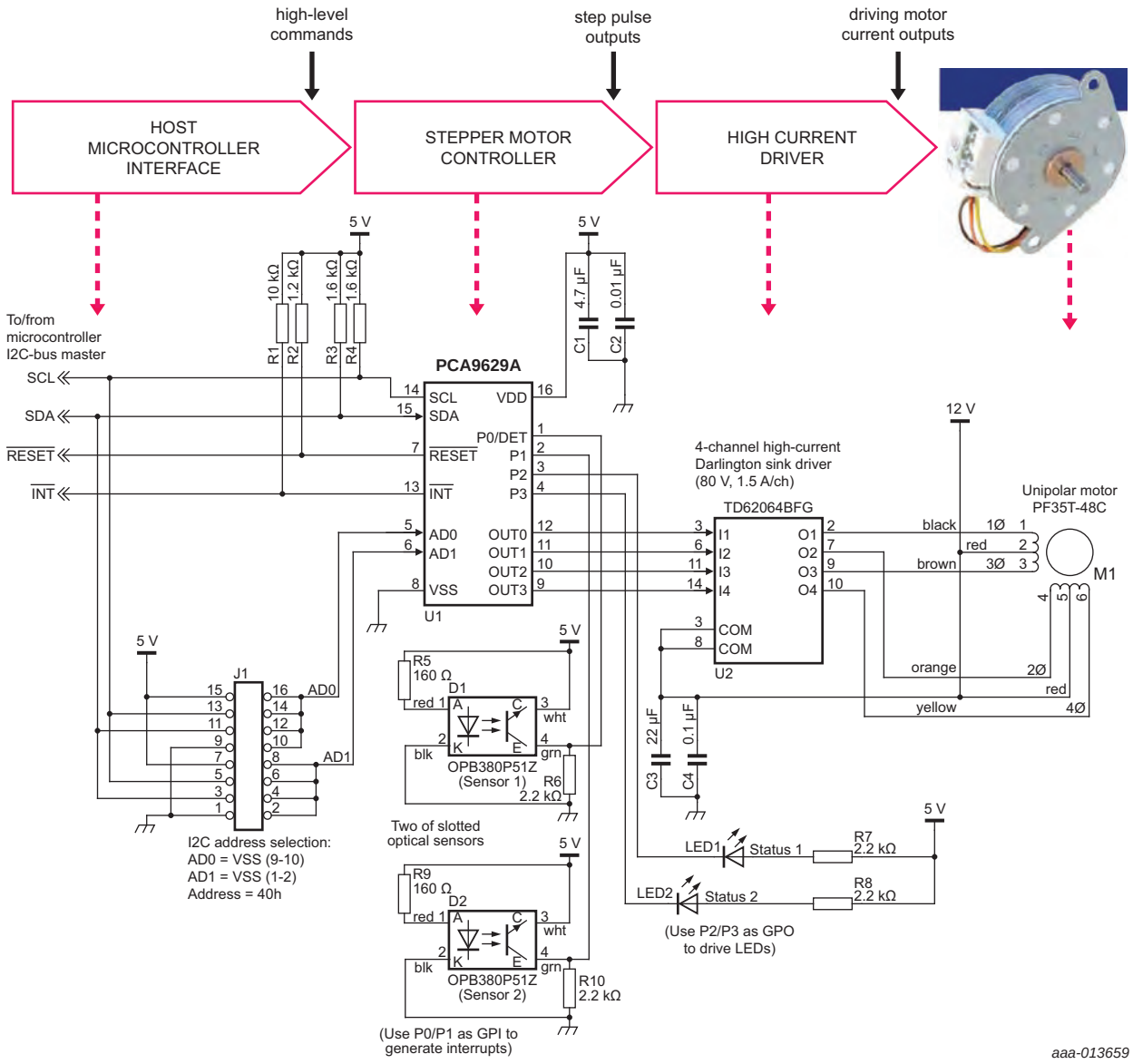


Fig 3. Unipolar stepper motor driver application using PCA9629A

4. How to program and control the PCA9629A internal registers to drive stepper motor

The PCA9629A has total 35 internal registers to store data for control motor operation as shown in Table 1. There are ten groups of registers:

- **Chip access control:** to set device operation mode such as normal or sleep mode, interrupt enable or disable, motor outputs change on I^2C-bus ACK or STOP condition, three subaddresses and one all-call address values with response or no response control.
- **Watchdog control:** to set watchdog time-out interval value, enable/disable and interrupt operation.
- **GPIO and interrupts control:** to configure P0-P3 general purpose I/O operation mode and to set interrupt triggering edge, interrupt enable/disable, interrupt flag status (source), and interrupt motor operation control based on P0-P1 inputs.
- **Motor output extra steps control:** to set extra steps values up to 255 steps when P0 or P1 interrupt has occurred.
- **Motor operation control:** to set motor output phase format (half-step, one or two-phase), motor output bypass mode (like GPO), output state when motor is stopped, motor performs multiple of actions up 255 times or continuously, motor start/stop, restart, emergency stop, conditional-start based on P0 input state, direction of motor operation in clockwise, counter-clockwise or both.
- **Ramp operation control:** to set ramp-up or ramp-down rate, enable/disable or re-enable to change ramp rate curve on the fly.
- **Loop delay timer control:** to set amount of time (up to 1.02 second in resolution of 4 ms) before reversing motor direction from clockwise to counter-clockwise or from counter-clockwise to clockwise rotation.
- **Motor output steps control:** to set the number of steps the motor should turn in clockwise or counter-clockwise direction.
- **Motor output step pulse width control:** to set step pulse width or speed for motor phase sequence output waveform in clockwise or counter-clockwise direction.
- **Motor output steps counter:** to count continuously the total number of step pulses that drive the motor coils from output ports OUT0 to OUT3. This 32-bit counter will be cleared after they are read, overflow, power-on reset, or hardware/software reset.

Table 1. PCA9629A register summary and groups

Address		Name	Function	Group
0	0x00	MODE	Device operation MODE setting	Chip access control
1	0x01	WDTOI	Watchdog time-out interval (1 second to 255 seconds)	Watchdog control
2	0x02	WDCNTL	Watchdog enable and interrupt control	
3	0x03	IO_CFG	General purpose I/O configuration for P0 - P3 (1 = input, 0 = output) and output logic levels	GPIO and interrupts control
4	0x04	INTMODE	Interrupt occurs edge selection for P0 - P3 and input filter control for P0 and P1 inputs	
5	0x05	MSK	Interrupt mask control for P0 - P3 and motor stop (0 = enabled, 1 = disabled)	
6	0x06	INTSTAT	Interrupt status for P0 - P3, motor stop, watchdog (read only)	
7	0x07	IP	Input port reflects the incoming logic levels on P0 - P3 (read only)	
8	0x08	INT_MTR_ACT	Interrupt motor actions control based on P0 and P1 inputs	
9	0x09	EXTRASTEPS0	Extra steps count up to 255 for INTP0	Motor output extra steps control
10	0x0A	EXTRASTEPS1	Extra steps count up to 255 for INTP1	
11	0x0B	OP_CFG_PHS	Output port configuration (either motor drive or GPO output) and motor drive phase control	Motor operation control
12	0x0C	OP_STAT_TO	Motor stop output state and time-out control	
13	0x0D	RUCNTL	Ramp-up control	Ramp operation control
14	0x0E	RDCNTL	Ramp-down control	
15	0x0F	PMA	Perform multiple of motor actions control (1 - 255 times or continuously when set to 0x00)	Motor operation control
16	0x10	LOOPDLY_CW	Loop delay time for reversing from CW to CCW (delay time = set value $\times$ 4 ms)	Loop delay timer control
17	0x11	LOOPDLY_CCW	Loop delay time for reversing from CCW to CW (delay time = set value $\times$ 4 ms)	
18	0x12	CWSCOUNTL	Number of steps count for clockwise (CW) low and high bytes [0:15]	Motor output steps control
19	0x13	CWSCOUNTH		
20	0x14	CCWSCOUNTL	Number of steps count for counter-clockwise (CCW) low and high bytes [0:15]	
21	0x15	CCWSCOUNTH		
22	0x16	CWPWL	Clockwise (CW) step pulse width control low and high bytes [0:15]	Motor output step pulse width control
23	0x17	CWPWH		
24	0x18	CCWPWL	Counter-clockwise (CCW) step pulse width control low and high bytes [0:15]	
25	0x19	CCWPWH		
26	0x1A	MCNTL	Motor start, restart, conditional-start, stop, emergency stop and direction of motor rotate control	Motor operation control
27	0x1B	SUBADR1	I^2C-bus subaddress 1, 2 and 3 setting	Chip access control
28	0x1C	SUBADR2		
29	0x1D	SUBADR3		
30	0x1E	ALLCALLADR	I^2C-bus All Call address setting	
31	0x1F	STEPCOUNT0	Four-byte (32-bit) step counter [0:3] (read only)	Motor output steps counter
32	0x20	STEPCOUNT1		
33	0x21	STEPCOUNT2		
34	0x22	STEPCOUNT3		

4.1 How to program motor operation without ramp control

To drive motor running in normal operation without ramp control, user must program the following registers step-by-step to control motor:

4.1.1 Step 1: Set step pulse width in CWPWH/L and CCWPWH/L registers

Set the step pulse width in CWPWH/L and CCWPWH/L registers for motor running speed in clockwise or counter-clockwise direction.

a. Step pulse width for clockwise rotation

b. Step pulse width for counter-clockwise rotation

Fig 4. Motor output step pulse width registers setting

The prescaler (3-bit) selects one of the eight dynamic pulse width ranges.

The Step_Pulse_Width (13-bit) sets the number of steps in each prescaler value.

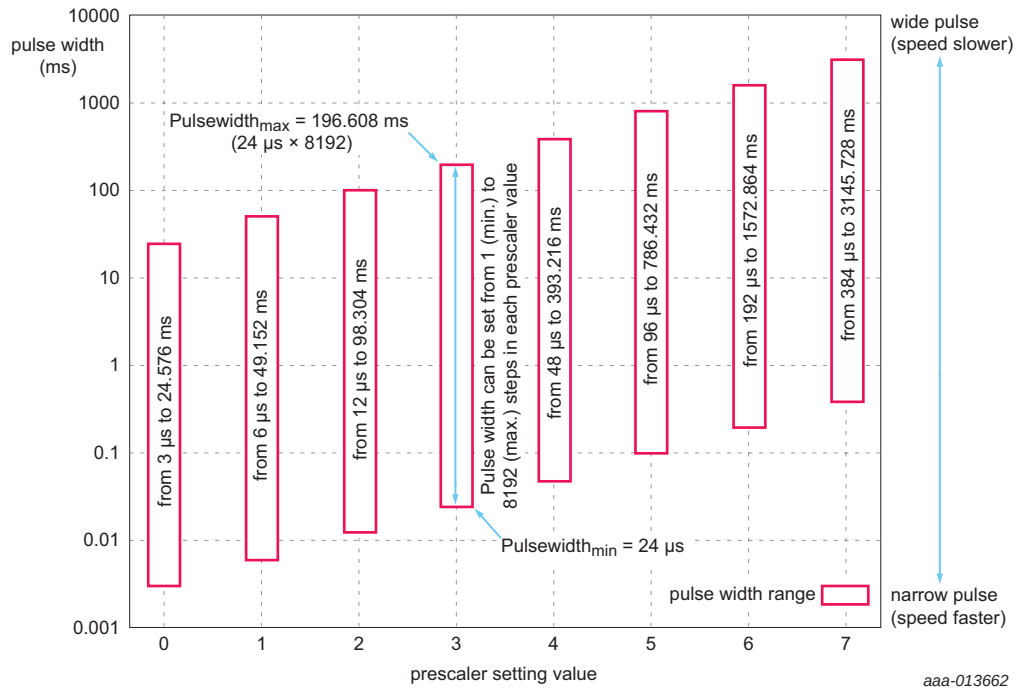


Fig 5. Step pulse width versus prescaler value

The motor always starts running in this final speed since there is no ramp control. The following equation is used to calculate the motor output step pulse width.

$$Pulsewidth_{min} = 2^{prescaler} \times 3\ \mu s$$

$$Pulsewidth_{max} = Pulsewidth_{min} \times 2^{13}$$

$$Pulsewidth_{final} = Pulsewidth_{min} \times (\text{STEP\_PULSE\_WIDTH} + 1)$$

For example, if set CWPWH = 0x04h and CWPWL = 0xFFh, the prescaler is set to ‘0’ and the STEP_PULSE_WIDTH (0x4FFh) is set to ‘1279’.

$$Pulsewidth_{min} = 2^{0} \times 3\ \mu s = 3\ \mu s$$

$$Pulsewidth_{final} = 3\ \mu s \times (1279 + 1) = 3.84\ ms\ (\text{output step pulse width})$$

Three output waveforms are shown for this example with one-phase (Figure 6), two-phase (Figure 7) and half-step (Figure 8) drive outputs in clockwise rotation.

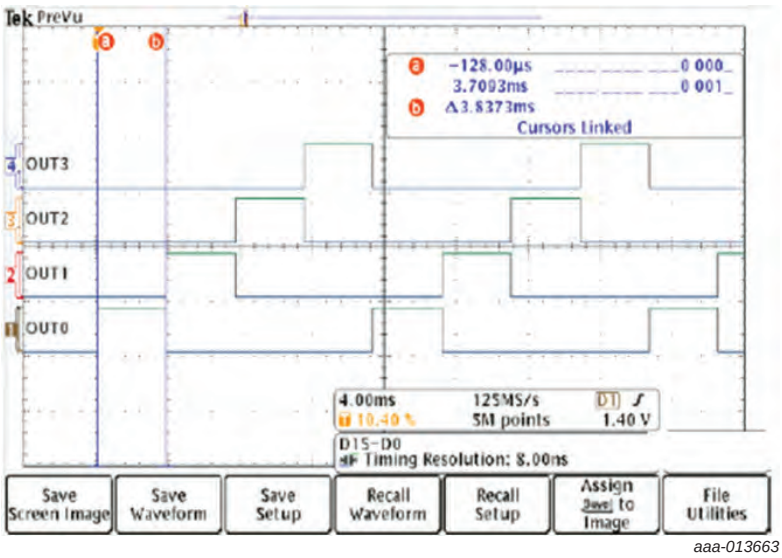


Fig 6. One-phase drive sequence output waveform

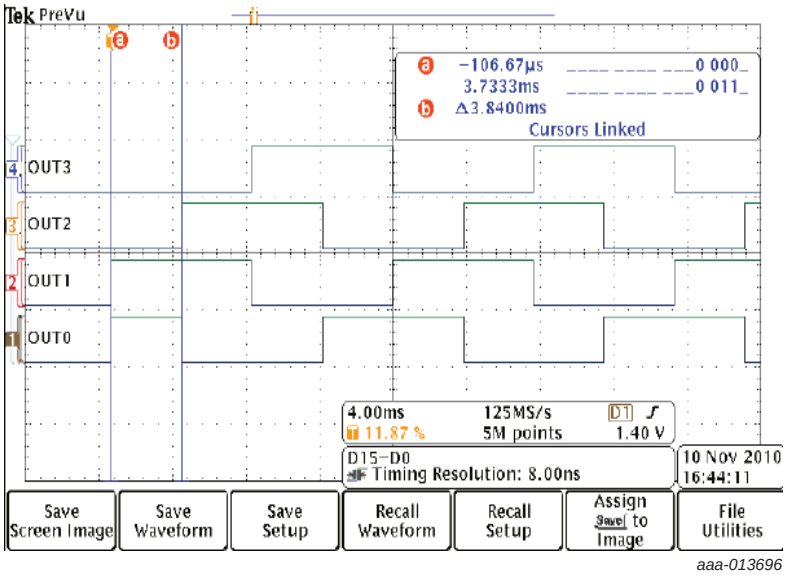


Fig 7. Two-phase drive sequence output waveform

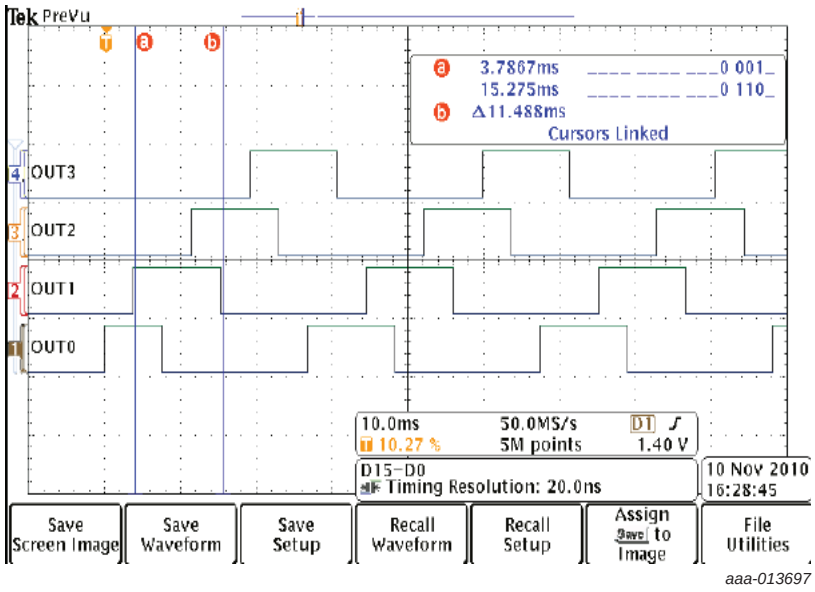
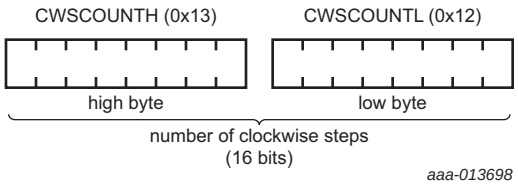


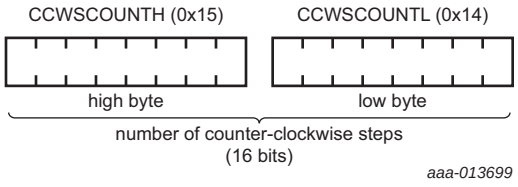
Fig 8. Half-step drive sequence output waveform

4.1.2 Step 2: Set number of steps in CWSCOUNTH/L and CCWSCOUNTH/L registers

Set the number of steps in CWSCOUNTH/L and CCWSCOUNTH/L registers for motor turning steps in clockwise or counter-clockwise direction.



a. Clockwise rotation steps count



b. Counter-clockwise rotation steps count

Fig 9. Motor output number of steps count registers setting

The number of steps count registers (16-bit) can be programmed from 1 (0x0001h) to 65535 (0xFFFFh) steps. If the number of steps is set to zero, the motor does not start.

4.1.3 Step 3: Set perform multiple of actions in PMA register and set motor operation control in MCNTL register

Set perform multiple of actions in PMA register for motor repeat operation times from 1 (0x01h) to 255 (0xFFh) or continuously (0x00h), and set the motor operation control in MCNTL register for motor start, stop, and type of rotations (clockwise, counter-clockwise, or both).

a. Perform multiple of action control

b. Motor control

Fig 10. Motor operation control registers setting

Remark: In the motor control register (MCNTL), the emergency stop (set bit 5 = 1) has the highest priority to stop motor immediately, normal stop motor (set bit 7 = 0 while the bit 5 = 0) will stop motor after ramp operation completed if ramp control is enabled, set restart motor (set bit 6 = 1) valid only when motor is still running (bit 7 = 1).

The following examples show how the motor operation based on these registers settings without ramp control.

- CWPWH = 0x04h; CWPWL = 0xFFh (set CW step pulse width = 3.84 ms)
- CCWPWH = 0x04h; CCWPWL = 0xFFh (set CCW step pulse width = 3.84 ms)
- CWSCOUNTH = 0x00h; CWSCOUNTL = 0x05h (set number of CW steps = 5)
- CCWSCOUNTH = 0x00h; CCWSCOUNTL = 0x06h (set number of CCW steps = 6)
- PMA = 0x05h (set perform motor action five times specified in bit[1:0] of MCNTL
- LOOPDLY_CW; LOOPDLY_CCW (set motor reversing direction loop delay timer)
- EXTRASTEPS0; EXTRASTEPS1 (if set non-zero, extra steps feature is enabled when P0 or P1 interrupt occurred)
- MCNTL = 0x80h, 0x81h, 0x82h, 0x83h as shown in Figure 11 through Figure 16 for the motor operation.

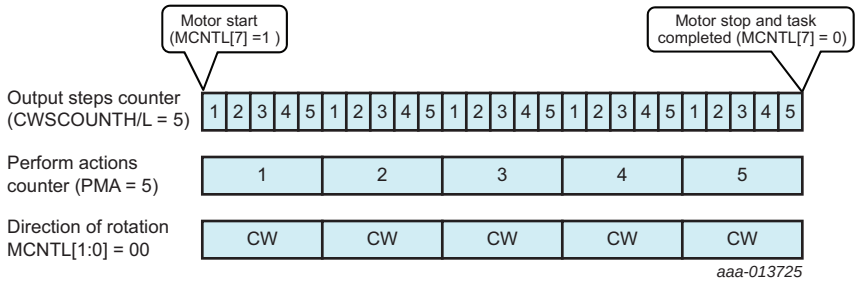


Fig 11. Set MCNTL = 0x80h to start motor and rotate clockwise (CW) only

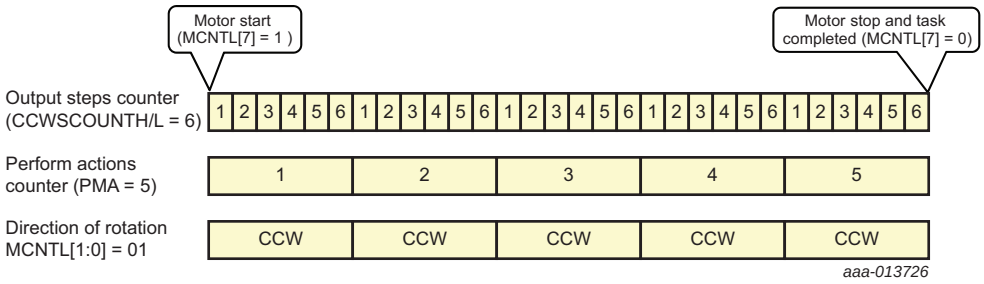


Fig 12. Set MCNTL = 0x81h to start motor and rotate counter-clockwise (CCW) only

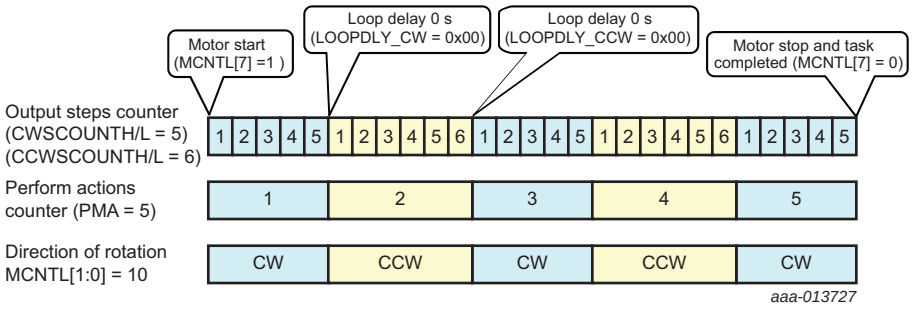


Fig 13. Set MCNTL = 0x82h to start motor and rotate clockwise first, then counter-clockwise

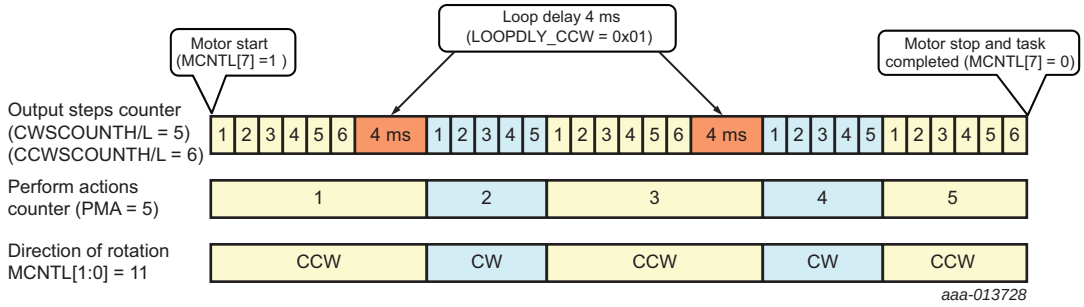


Fig 14. Set MCNTL = 0x83h to start motor and rotate counter-clockwise first, then clockwise

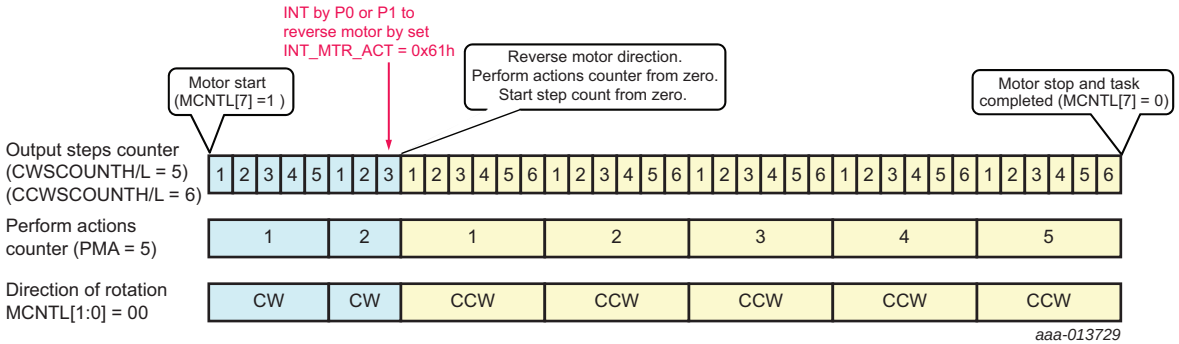


Fig 15. Set MCNTL = 0x80h to start motor and an interrupt P0 or P1 to reverse motor direction

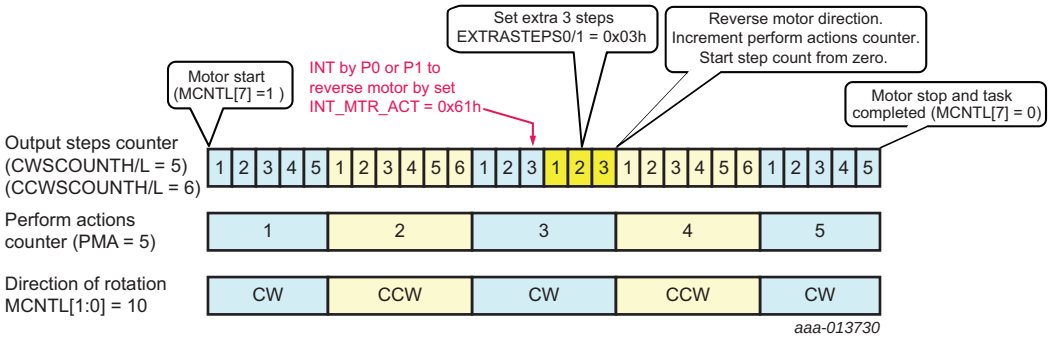


Fig 16. Set MCNTL = 0x82h to start motor and an interrupt P0 or P1 to reverse motor direction with 3 extra steps

4.2 How to program motor operation with ramp control

The PCA9629A can be programmed to ramp-up from motor start to final speed and ramp-down from final speed to motor stop.

The ramp-up control starts in speed of maximum_pulse_step, which is the maximum value ($pulsewidth_{max}$) of the selected range given in Figure 17. The ramp-up is completed to final speed when the pulse width gets the width that is set by CWPWL/CWPWH or CCWPWL/CCWPWH registers. During ramp-up, the step pulse width is automatically decremented (from the maximum value for step pulse width in the chosen range) until the value in CWPWH/L or CCWPWH/L register is reached, depending on the direction of rotation.

The ramp-down control ends in speed of maximum_pulse_step, which is the maximum value ($pulsewidth_{max}$) of the selected range given in Figure 17. During ramp-down, the step pulse width is automatically incremented from the current value in CWPWH/L or CCWPWH/L, depending on the direction of rotation, until it reaches the maximum value for step pulse width in the chosen range.

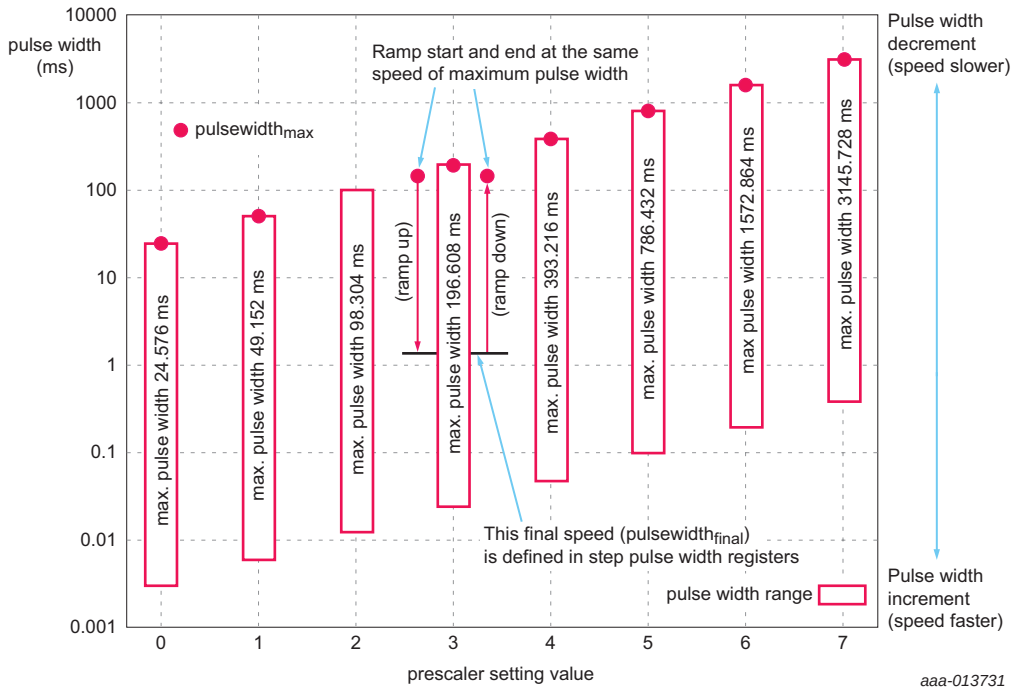


Fig 17. Ramp-up to and ramp-down from the final speed operation

When user set the motor steps in step count registers CWSCOUNTH/CCWSCOUNTH, CWSCOUNTL/CCWSCOUNTL and repeat operation times register PMA, it should be the count of steps that operated in final (top) speed. The total number of the motor operation steps is the sum of ramp-up steps, final speed steps and ramp-down steps as shown in Figure 18.

$$Steps_{total} = Steps_{ramp\text{-}up} + Steps_{final} + Steps_{ramp\text{-}down}$$

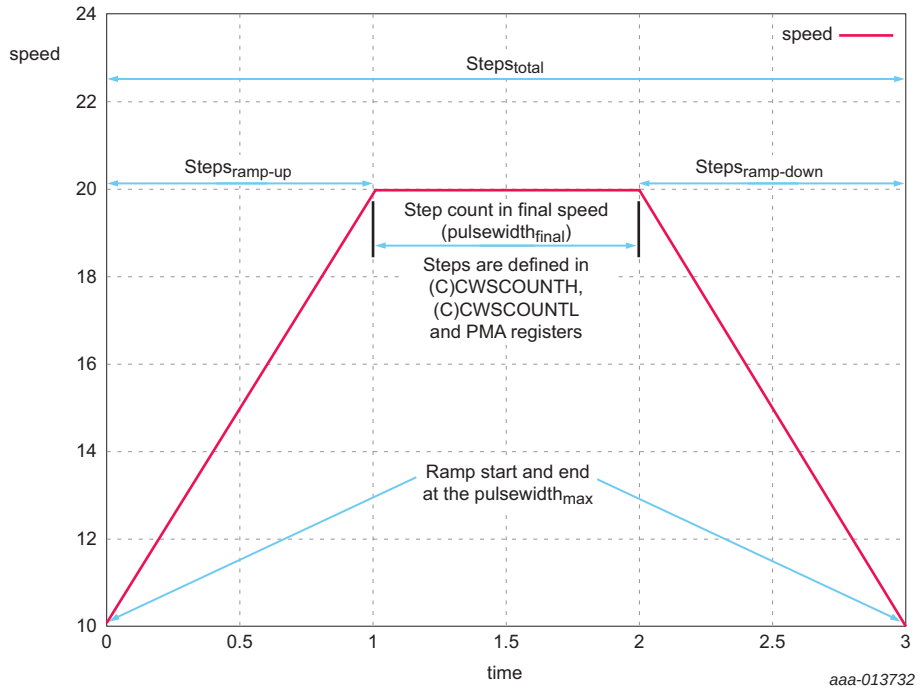


Fig 18. Ramp operation total steps calculation

For example, user can program the following registers to control motor ramp operation:

1. Set the ramp start and ramp end speed ($Pulsewidth_{max}$) and the final motor speed ($Pulsewidth_{final}$) after ramp-up and before ramp-down in CWPWH/L or CCWPWH/L registers for motor running speed in either clockwise or counter-clockwise direction.

   $$Pulsewidth_{min} = 2^{prescaler} \times 3\ \mu s$$

   $$Pulsewidth_{max} = Pulsewidth_{min} \times 2^{13}$$

   $$Pulsewidth_{final} = Pulsewidth_{min} \times (STEP\_PULSE\_WIDTH + 1)$$

   For example, if set CWPWH = 0x06h and CWPWL = 0x82h, the prescaler is set ‘0’ and the STEP_PULSE_WIDTH (0x682h) is set ‘1666’ for clockwise direction.

   $$Pulsewidth_{min} = 2^{0} \times 3\ \mu s = 3\ \mu s$$

   $$Pulsewidth_{max} = 3\ \mu s \times 2^{13} = 3\ \mu s \times 8192 = 24.576\ ms$$

   $$Pulsewidth_{final} = 3\ \mu s \times (1666 + 1) = 5.001\ ms\ \text{(the final motor speed)}$$

2. Set number of steps in CWSCOUNTH/L and CCWSCOUNTH/L registers for motor turning steps and perform multiple of actions in PMA register for motor repeat operation times in the final speed.

   For example, if set CWSCOUNTH/L = 0x0003h and CCWSCOUNTH/L = 0x0000h, PMA = 0x01h. The motor runs clockwise direction for 3 steps once in final speed.

3. Set ramp-up and ramp-down control registers as shown in Figure 19. The lower 4-bit is multiplication factor which defines the acceleration (pulse width decrement) rate for ramp-up operation or decelerating (pulse width increment) rate for ramp-down operation respectively. The multiplication factor can be set in the range of 0 to 13 (14 and 15 are reserved, do not use).

   For example, if set RUCNTL/RDCNTL = 0x2Ah to enable ramp-up when motor is starting and ramp-down when motor is stopping, the multiplication_factor is set 10. The pulse width decrement (ramp-up) and increment (ramp-down) per step can be calculated as below, the $Pulsewidth_{min}$ is set to 3 µs.

$$Pulsewidth\ inc/dec\ per\ step = Pulsewidth_{min} \times 2^{mulitplication\_factor} = 3\ \mu s \times 2^{10} = 3.072\ ms$$

   The pulse width of $Pulsewidth_i$ (ramp-up pulse width per step) can be calculated in the next formula:

$$Pulsewidth_i = \left(\frac{2^{13}}{2^{multiplication\_factor}} - i \right) \times Pulsewidth_{min} \times 2^{muliplication\_factor}$$

$$Pulsewidth_0 = ((2^{13} \div 2^{10}) - 0) \times 3\ \mu s \times 2^{10} = 24.576\ ms$$

$$Pulsewidth_1 = ((2^{13} \div 2^{10}) - 1) \times 3\ \mu s \times 2^{10} = 21.504\ ms$$

$$Pulsewidth_2 = ((2^{13} \div 2^{10}) - 2) \times 3\ \mu s \times 2^{10} = 18.432\ ms$$

$$Pulsewidth_3 = ((2^{13} \div 2^{10}) - 3) \times 3\ \mu s \times 2^{10} = 15.36\ ms$$

$$Pulsewidth_4 = ((2^{13} \div 2^{10}) - 4) \times 3\ \mu s \times 2^{10} = 12.288\ ms$$

$$Pulsewidth_5 = ((2^{13} \div 2^{10}) - 5) \times 3\ \mu s \times 2^{10} = 9.216\ ms$$

$$Pulsewidth_6 = ((2^{13} \div 2^{10}) - 6) \times 3\ \mu s \times 2^{10} = 6.144\ ms$$

$$Pulsewidth_{final} = 5.001\ ms$$

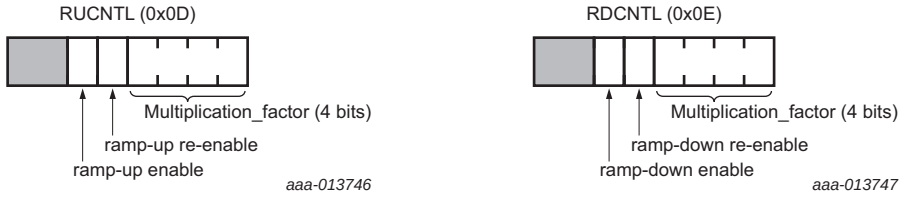
   The number of steps for the ramp-up and ramp-down can be calculated in next formula and shown in Figure 20.

$$Steps_{rampup} = (2^{13} - STEP\_PULSE\_WIDTH) \div 2^{multiplication\_factor}$$ (Round up to next integer number.)

$$Steps_{rampdown} = Steps_{rampup} - 1$$ (if the multiplication_factor is same for both ramp-up and ramp-down).

   The last ramp-down step pulse width is between $Pulsewidth_0$ and $Pulsewidth_1$. For example, STEP_PULSE_SWIDTH = 1666; multiplication_factor = 10, as set before.

   $Steps_{rampup} = (2^{13} - 1666) \div 2^{10} = 6.37$ (round up to next integer) = 7 (total ramp-up steps) and $Steps_{rampdown} = 7 - 1 = 6$ (total ramp-down steps) as shown in Figure 20.



a. Ramp-up control register b. Ramp-down control register

Fig 19. Ramp-up and ramp-down control registers setting

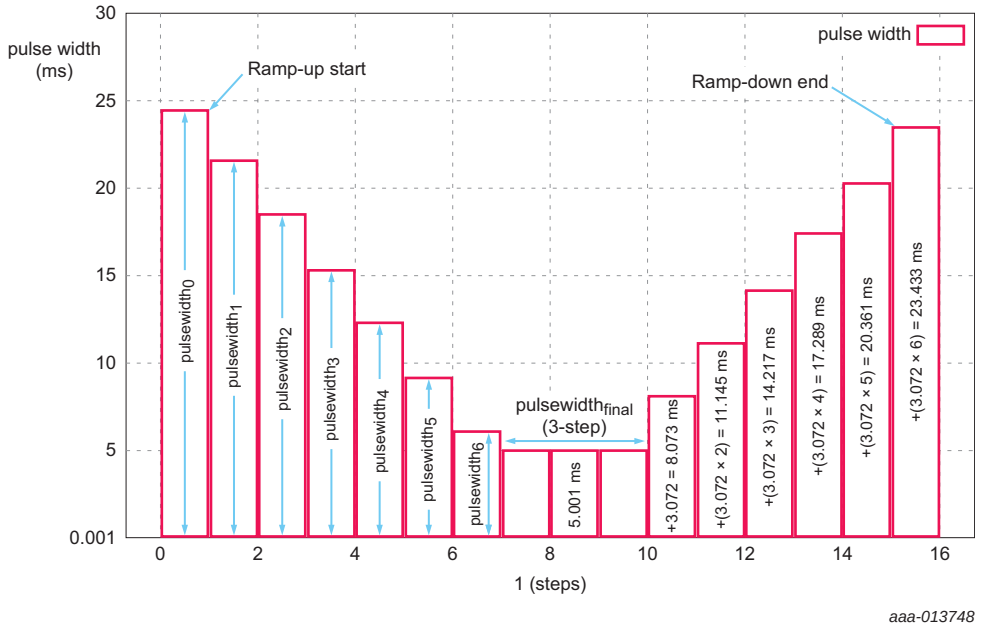


Fig 20. Ramp-up and ramp-down steps and pulse width

Figure 21 shows the actual output waveforms from PCA9629A for this ramp operation. The upper four waveforms are OUT0 to OUT3 and lower two waveforms are I^2C-bus, the ramp-up is completed by decreasing 7 pulses to final speed with 3 pulses having 5.001 ms pulse width, then ramp-down is completed by increasing 6 pulses to stop motor.

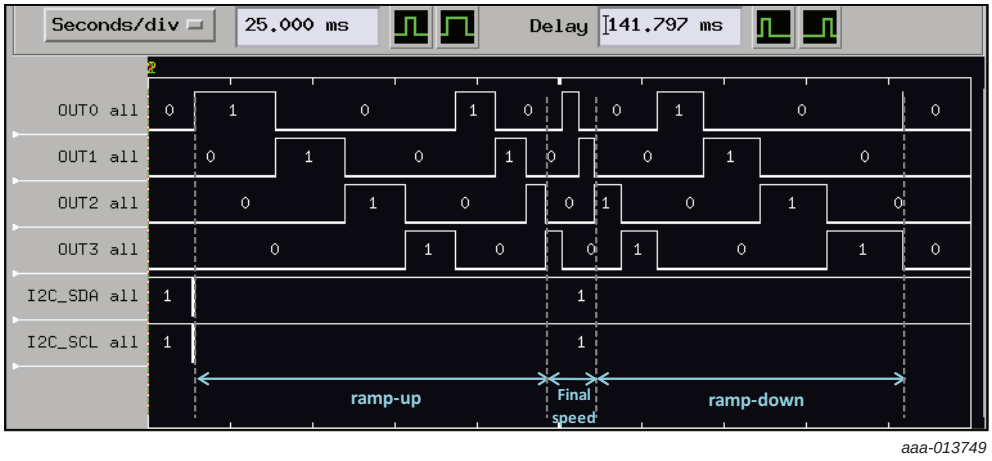


Fig 21. Output waveforms for ramp-up and ramp-down operation

5. Examples to control motor operation by setting internal registers

5.1 Example 1: Start and restart motor without ramp operation

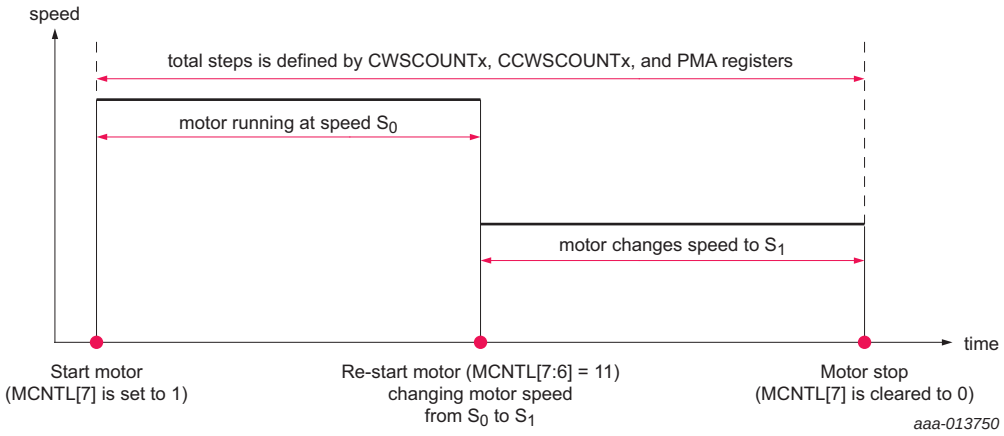


Fig 22. Motor start and restart operations without ramp control

- **CWPWH/L, CCWPWH/L** — Set the step pulse width for S_0 speed
- **CWSCOUNTH/L, CCWSCOUNTH/L** — Set number of steps for clockwise or counter-clockwise rotation
- **PMA** — Set perform multiple of actions specified in bit [1:0] of MCNTL register
- **RUCNTL, RDCNTL** — Disable both ramp-up and ramp-down operation
- **MCNTL** — Set bit 7 = 1 to start motor at S_0 speed
- **CWPWH/L, CCWPWH/L** — Change the step pulse width for S_1 speed
- **MCNTL** — Set bit 6 = 1 to restart motor, the motor speed changes from S_0 to S_1 immediately
- The motor stops when it either finishes the number of total steps, or set bit 7 = 0 to stop motor (same as set bit 5 = 1 to emergency-stop motor) in MCNTL register.

5.2 Example 2: Start motor with ramp-up/ramp-down disable and restart motor with ramp-up/ramp-down enable

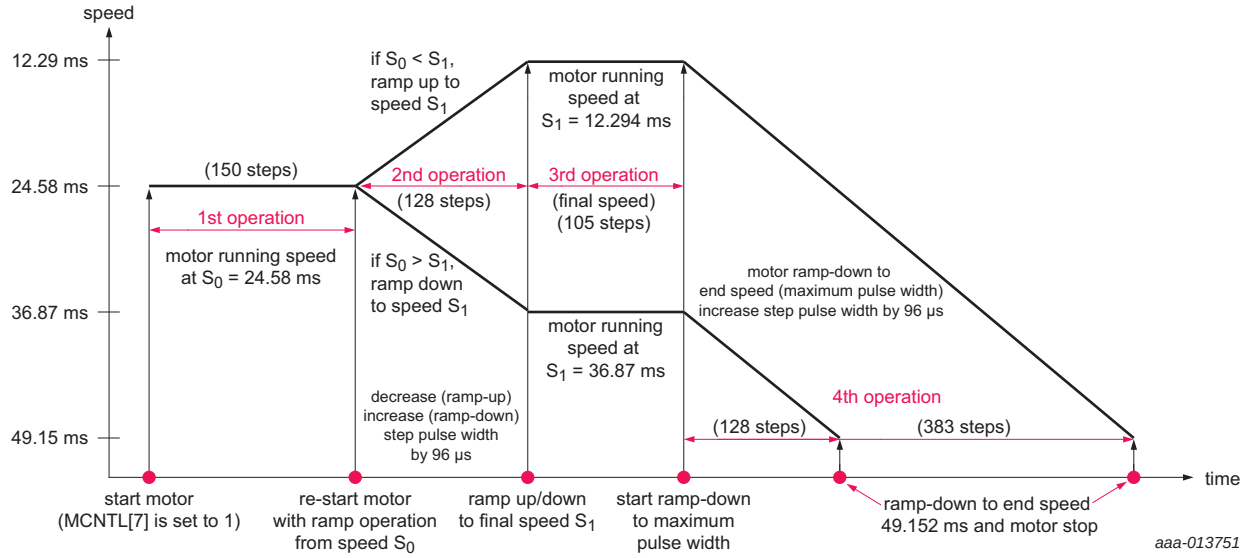


Fig 23. Motor start and restart operations with ramp control

- **CWPWH = 0x30h, CWPWL = 0x00h** — Set the clockwise step pulse width 24.582 ms (6 µs × 4097) for S_0 speed
- **CWSCOUNTH = 0x00h, CWSCOUNTL = 0xFFh** — Set 255 steps for clockwise rotation
- **PMA = 0x01h** — The motor operation specified in bits [1:0] of MCNTL register is executed once
- **RUCNTL/RDCNTL = 0x00h** — Disable both ramp-up and ramp-down operation
- **MCNTL = 0x80h** — Start motor with clockwise rotation at S_0 speed as shown in the first operation

If new motor speed S_1 is faster than S_0 and ramp operation is enabled, then motor starts ramp-up operation until it reaches the speed of S_1:

- **CWPWH = 0x28h, CWPWL = 0x00h** — Change the clockwise step pulse width to 12.294 ms (6 µs × 2049) for S_1 speed
- **RUCNTL/RDCNTL = 0x24h** — Enable ramp operation with pulse width decrement or increment rate 96 µs (6 µs × 16)
- **Wait for 150-step (24.582 ms × 150 = 3.687 s) of S_0 speed, then set MCNTL = 0xC0h** — Restart motor, the motor starts ramp-up from S_0 to S_1 as shown in the second operation with 128-step ((24.582 ms – 12.294 ms) ÷ 96 µs)
- After ramp-up to the final speed of S_1, motor runs another 105-step (total 255-step minus 150-step in first operation) and then motor starts ramp-down to end speed ($Pulsewidth_{max}$ = 6 µs × 2^{13} = 49.152 ms) as shown in the fourth operation with 383-step ((49.152 ms – 12.294 ms) ÷ 96 µs)

If new motor speed S_1 is slower than S_0 and ramp operation is enabled, then motor starts ramp-down operation until it reaches the speed of S_1:

- **CWPWH = 0x38h, CWPWL = 0x00h** — Change the clockwise step pulse width to 36.87 ms (6 µs × 6145) for S_1 speed
- **RUCNTL/RDCNTL = 0x24h** — Enable ramp operation with pulse width decrement or increment rate 96 µs (6 µs × 16)
- **Wait for 150-step (24.582 ms × 150 = 3.687 s) of S_0 speed, then set MCNTL = 0xC0h** — Restart motor, the motor starts ramp-down from S_0 to S_1 as shown in the second operation with 128-step ((36.87 ms – 24.582 ms) ÷ 96 µs)
- After ramp-down to the final speed of S_1, motor runs another 105-step (total 255-step minus 150-step in first operation) and then motor starts ramp-down to end speed ($Pulsewidth_{max}$ = 6 µs × 2^{13} = 49.152 ms) as shown in the fourth operation with 128-step ((49.152 ms – 36.87 ms) ÷ 96 µs)

5.3 Example 3: Motor home-position control from P0/DET (option 1) or P0/P1 (option 2) inputs

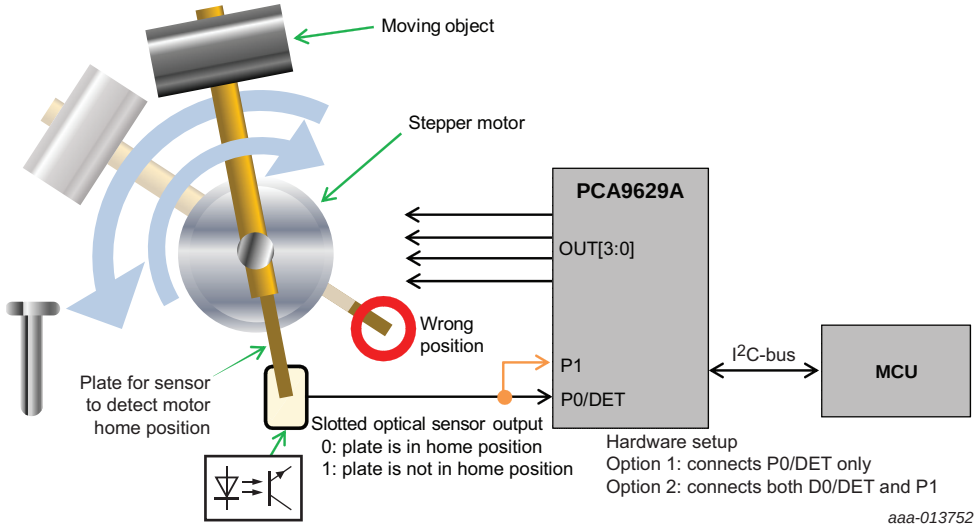


Fig 24. Control and monitor motor home-position implementation

The moving object such as the hammer must be kept in home position of optical sensor slot when motor is stopped, but the moving object can be moved out of home position due to the motor driving output off to prevent heat-up or vibration on the machine. The system must monitor the motor position periodically and get the moving object back to normal position while it is not in operation. The PCA9629A has implemented a conditional-START command based on P0/DET (pin 1) input state to bring motor back in right position, 2 bits in motor control register (MCNTL) are designed to enable and control the motor home-position operation without polling the P0 input state.

To make this conditional-START command work, the following control bits must be set:

- P0 (option 1) or both P0/P1 (option 2) pins must be configured as input (power-on default setting) in IO_CFG (= 0x0Fh) register
- Set P0 interrupt to occur on falling edge (option 1), or both P0/P1 interrupts to occur on different edge (option 2) with 1 ms noise suppressed in INTMODE (= 0x21h) register
- Enable P0 interrupt (option 1) in MSK (= 0x1Eh) or both P0/P1 interrupts (option 2) in MSK (= 0x1Ch) register
- P0 interrupt status flag bit must be cleared by access (read or write) the INTSTAT (= 0x00h) register
- Set enable interrupt based control of motor and stop motor on interrupt caused by P0 in INT_MTR_ACT (= 0x01h) register for option 1. Set enable interrupt based control of motor, P0 and P1 auto clear each other, stop motor on interrupt caused by P0 in INT_MTR_ACT (= 0x09h) register for option 2.

When the motor is stopped (START bit 7 = 0), the microcontroller can set bit 7 = 1 and bits [4:3] = 10 in MCNTL register to control motor operation based on input state of P0:

- If P0 input state is LOW, then motor START operation is ignored (motor is detected in right position and no movement).
- If P0 input state is HIGH, then motor is started until the P0 input state is detected as LOW (motor is moved to right position).

When the motor is stopped (START bit 7 = 0), the microcontroller can set bit 7 = 1 and bits [4:3] = 11 in MCNTL register to control motor operation based on input state of P0:

- If the P0 input state is HIGH, then motor START operation is ignored (motor is detected in right position and no movement).
- If P0 input state is LOW, then motor is started until the P0 input state is detected as HIGH (motor is moved to right position).

Remark: The input filter for P0 and P1 input state change detection can be enabled to suppress a spike or noise in the range of 500 µs to 10 ms in the control bits [6:4] of INTMODE register.

5.3.1 Option 1: Connects sensor output to P0/DET (pin 1) only for implementing the home position control

- **Write INTSTAT = 0x00h** — to clear all interrupts status flag bits
- **Write MCNTL = 0x90h** — to conditional-start motor if P0 input state is HIGH, then motor is started until the P0 input state is detected as LOW (motor is moved to right position) as shown in Figure 25 and Figure 26 for control signals and waveforms.

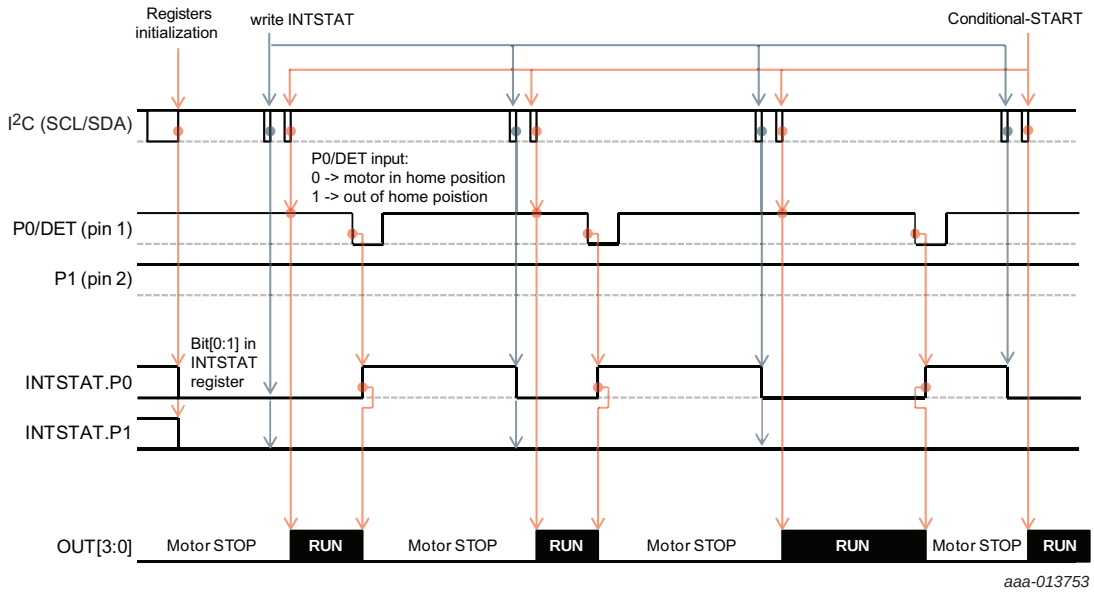


Fig 25. Motor home-position control (option 1) signals with internal interrupt status bits

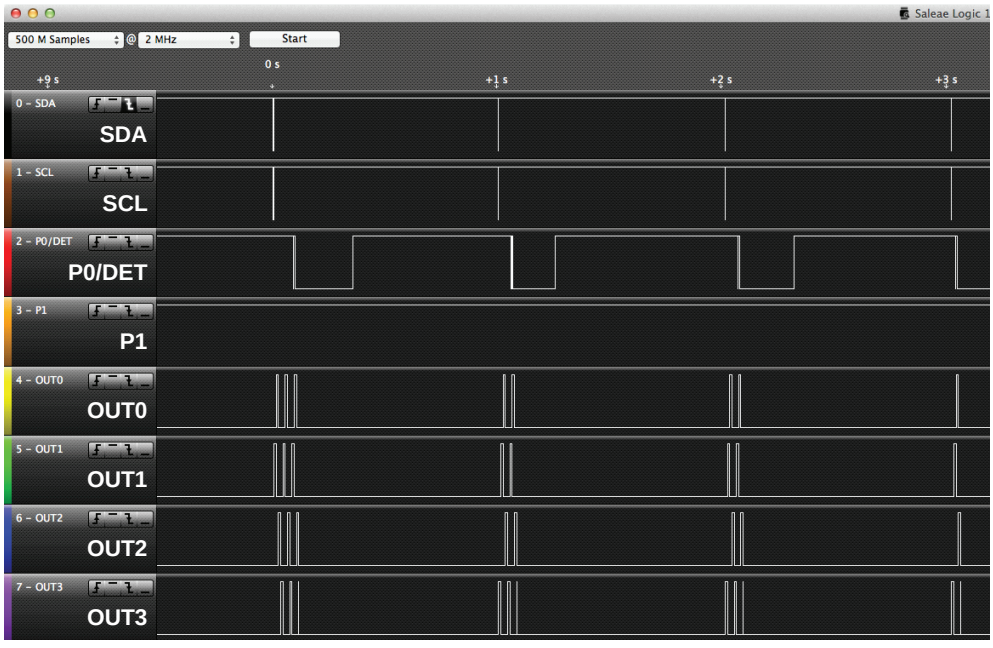


Fig 26. Motor home-position control (option 1) signal waveforms

5.3.2 Option 2: Connects sensor output to both P0/DET (pin 1) and P1 (pin 2) for implementing the home position control

Option 2 connects sensor output to both P0/DET (pin1) and P1 (pin2) for implementing the home position control. There is no need to write INTSTAT register for clearing interrupt status flags because the INTP0 auto clears INTP1 (flag) and INTP1 auto clears INTP0 (flag).

- **Write MCNTL = 0x90h** — to conditional-start motor if P0 input state is HIGH, then motor is started until the P0 input state is detected as LOW (motor is moved to right position) as shown in Figure 27 and Figure 28 for control signals and waveforms.

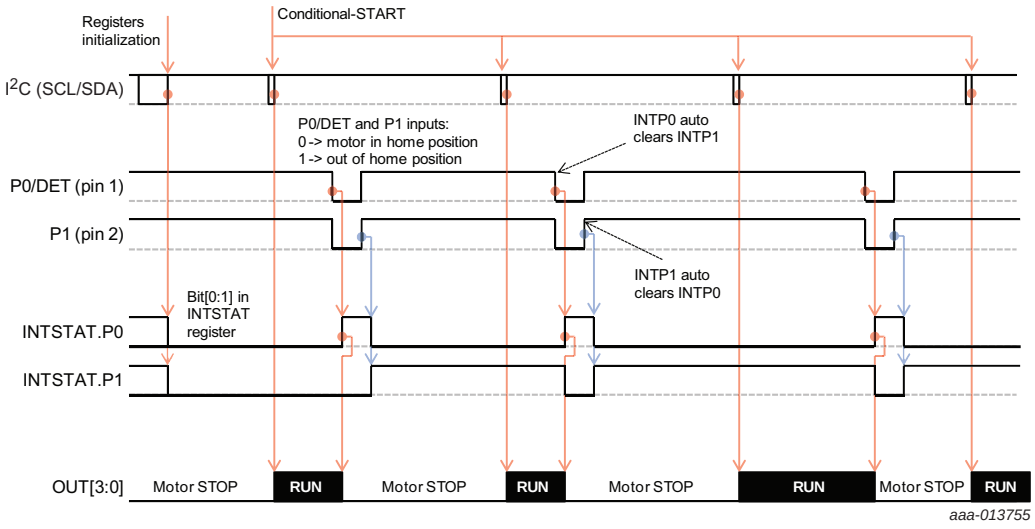


Fig 27. Motor home-position control (option 2) signals with internal interrupt status bits

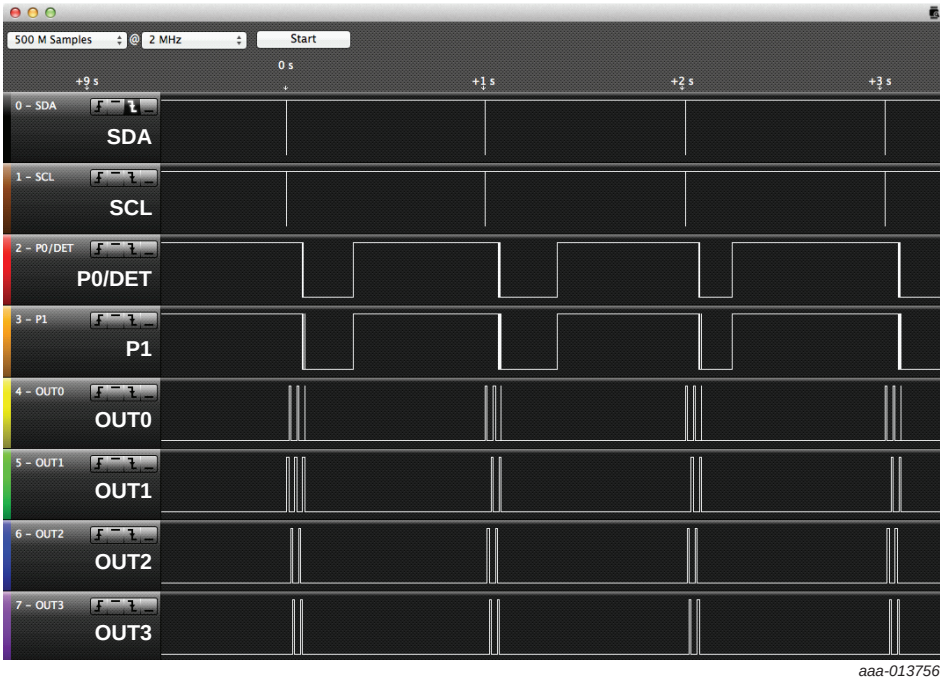


Fig 28. Home-position control (option 2) signal waveforms

6. PCA9629A versus PCA9629

Table 2 shows the comparison between the new, advanced PCA9629A device and the former PCA9629 (non-A) device.

Table 2. Comparison between PCA9629A and PCA9629

Feature/function	PCA9629A	PCA9629
Motor speed adjustment	• Motor speed change with ramp control any time while it is running • Asymmetrical ramp-up and ramp-down • 'Emergency Stop' to stop motor immediately without waiting for ramp-up or ramp-down completion	• Motor speed change with ramp control only when motor is STOP • Symmetrical ramp-up and ramp-down • 'Hard Stop' to stop motor after ramp-up or ramp-down sequence complete
Motor rotation action control	Programmable rotate action from 1 to 255 times or continuously	Rotate action only once or continuously
Motor STOP interrupt	Interrupt (maskable) is generated when motor is STOP	No interrupt when motor is STOP
Motor drive outputs	Programmable output either drive motor or General Purpose Output (GPO)	Output signals for driving motor use only
Motor home-position	Single command to bring motor to home-position without polling the P0 input	None
Motor brake function	Motor output states either '0', '1', or retaining the last state when motor is STOP with a programmable time-out timer (up to 1 second) to reset all outputs to '0' state	Motor output states either '0' or retaining the last state when motor is STOP without time-out timer
GPIO P0 and P1 input filters	Programmable noise filter on P0 and P1 inputs to suppress noise (up to 10 ms) and ensure that the correct interrupt event is generated	No input filter, hence any noise events on P0 and P1 inputs could prematurely trigger and generate interrupt
Loop delay timer	Dual loop delay timers (up to 1.02 second) for each reversing direction (CW to/from CCW)	One loop delay timer (up to 255 seconds) for both reversing direction (CW to/from CCW)
Output step counter	32-bit step counter for monitoring motor position or counting number of rotations in current run	None
Register map	• No need to have steps per rotation and number of rotations count for clockwise or counter-clockwise registers • Subcall and Allcall registers are placed to the bottom of the register map for easy software control • Reduced to 35 registers	• Need to set steps per rotation and number of rotations count for clockwise or counter-clockwise registers • Subcall and Allcall registers are at top of register map • Total 39 registers
Pin configuration	Hardware pin-to-pin compatible; software is not compatible because register map is different	

7. Summary

This application note outlined how to use PCA9629A to design and control unipolar stepper motors.

The PCA9629A is powerful and sophisticated stepper motor controller to support:

- Motor coil drive phase sequence signals with four outputs for use with external high current drivers to off-load CPU
- Motor ramp-up and ramp-down with Pulse Width Modulation (PWM) technique
- Changing speed and ramp rate while motor is running
- All three drive modes – wave drive, full-step drive and half-step drive
- Interrupt-based motor control from GPIO pins without CPU attention
- Watchdog timer to recover the system from unknown state
- Up to16 motors within one I^2C-bus interface

Table 3 shows the new and improved features and benefits of PCA9629A over the PCA9629.

Table 3. PCA9629A key features and benefits

***NEW** = new design for PCA9629A; **Improved** = improvement over PCA9629.*

	Features	Benefits
NEW	Restart motor with new speed and operation	Allows changing the motor speed and operation on the fly without stopping motor
NEW	Re-enable ramp-up or ramp-down during current ramp operation	Allows updating ramp rate curve on the fly without stopping motor
NEW	Motor outputs can be configured as general purpose outputs	Support bypass mode
NEW	Generate an interrupt when motor stop	Off-load CPU bandwidth — no interrupt polling is necessary
NEW	Motor home position control from P0 input	Single command to bring motor to home position
NEW	32-bit step counter to count output step pulses	Host can find current motor position and number of rotations by reading step counter value
NEW	Programmable filter on P0 or P1 input	Avoid false interrupt trigger on P0 or P1 input
Improved	Perform motor action settings from 1 to 255 or continuously	Allows performing multiple actions up to 255 or repeat without CPU reprogramming
Improved	Motor brake/stop with time-out control to set output state: all ‘0’, all ‘1’, or hold last state	Flexible brake feature to protect motor from overheating
Improved	Dual loop reversal mode timers	Allow asymmetrical delay in motor reverse operation
Improved	±3 % output step pulse accuracy	Comparable with best-in-class accuracy

Visit the NXP website at www.nxp.com for more information including design tool (ramp-up/ramp-down setting calculation spread sheet), PCA9629A demo kit as shown in Figure 29, PCA9629A demo quick start guide and PCA9629A demo board user manual.

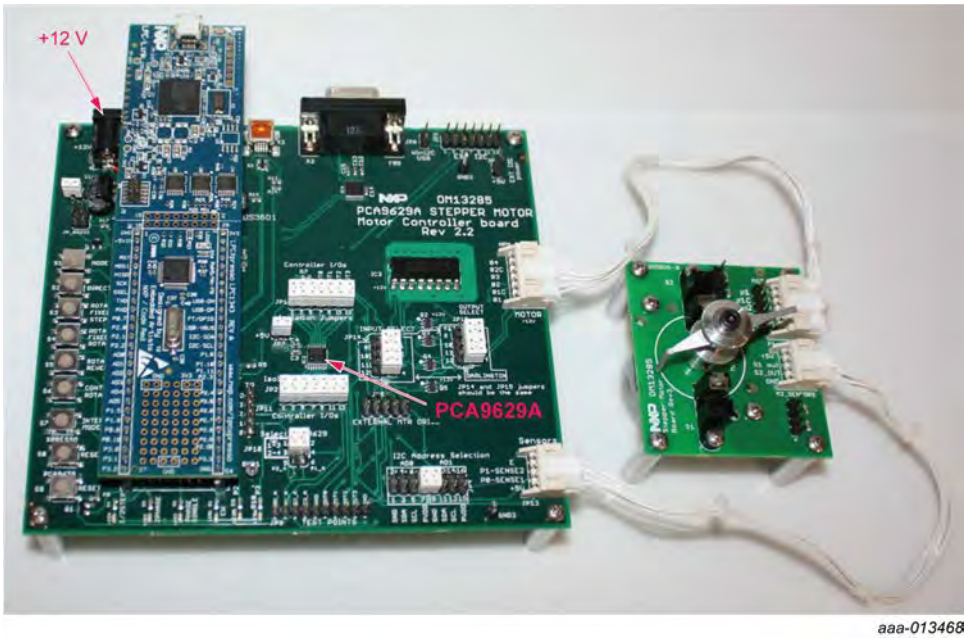


Fig 29. PCA9629A demo kit (OM13285)

8. Abbreviations

Table 4. Abbreviations

Acronym	Description
CCW	Counter-ClockWise
CDM	Charged-Device Model
CPU	Central Processing Unit
CW	ClockWise
ESD	ElectroStatic Discharge
Fm+	Fast-mode Plus
GPIO	General Purpose Input/Output
GPO	General Purpose Output
HBM	Human Body Model
I^2C-bus	Inter-Integrated Circuit bus
I/O	Input/Output
LED	Light Emitting Diode
MCU	MicroController Unit
POR	Power-On Reset
pps	pulses per second
PWM	Pulse Width Modulator

9. References

[1] **PCA9629A, Fm+ I^2C-bus advanced stepper motor controller —** Product data sheet; NXP Semiconductors; www.nxp.com/documents/data_sheet/PCA9629A.pdf

[2] The PCA9629A Stepper Motor training module link in Digikey; http://dkc1.digikey.com/us/en/tod/NXP/Stepper-Motor-Controller/Stepper-Motor-Controller.html

10. Legal information

10.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

10.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I²C-bus — logo is a trademark of NXP Semiconductors N.V.

11. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 27 June 2014
Document identifier: AN11483