LM628 Programming Guide

National Semiconductor
Application Note 693
Steven Hunt
January 1999

INTRODUCTION

The LM628/LM629 are dedicated motion control processors. Both devices control DC and brushless DC servo motors, as well as, other servomechanisms that provide a quadrature incremental feedback signal. Block diagrams of typical LM628/LM629-based motor control systems are shown in *Figures 1, 2.*

As indicated in the figures, the LM628/LM629 are bus peripherals; both devices must be programmed by a host processor. This application note is intended to present a concrete starting point for programmers of these precision motion controllers. It focuses on the development of short programs that test overall system functionality and lay the groundwork for more complex programs. It also presents a method for tuning the loop-compensation PID filter. (Note 1)

REFERENCE SYSTEM

Figure 15 is a detailed schematic of a closed-loop motor control system. All programs presented in this paper were developed using this system. For application of the programs in other LM628-based systems, changes in basic programming structure are not required, but modification of filter coefficients and trajectory parameters may be required.

I. PROGRAM MODULES

Breaking programs for the LM628 into sets of functional blocks simplifies the programming process; each block executes a specific task. This section contains examples of the principal building blocks (modules) of programs for the LM628.

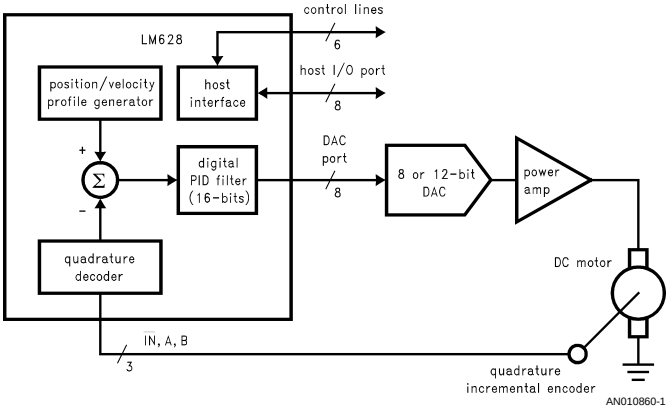


FIGURE 1. LM628-Based Motor Control System

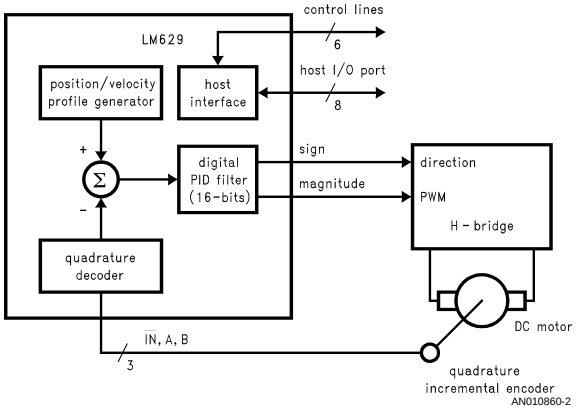


FIGURE 2. LM629-Based Motor Control System

Note 1: For the remainder of this paper, all statements about the LM628 also apply to the LM629 unless otherwise noted.

BUSY-BIT CHECK MODULE

The first module required for successful programming of the LM628 is a busy-bit check module.

The busy-bit, bit zero of the status byte, is set immediately after the host writes a command byte, or reads or writes the second byte of a data word. See *Figure 5*. While the busy-bit is set, the LM628 will ignore any commands or attempts to transfer data.

A busy-bit check module that polls the Status Byte and waits until the busy-bit is reset will ensure successful host/LM628 communications. ***It must be inserted after a command write, or a read or write of the second byte of a data word.*** *Figure 3* represents such a busy-bit check module. This module will be used throughout subsequent modules and programs.

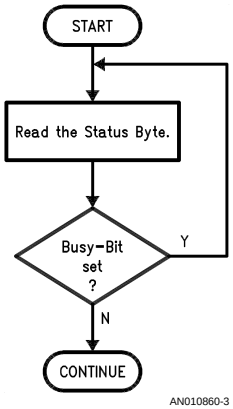


FIGURE 3. Busy-bit Check Module

Reading the Status Byte is accomplished by executing a RDSTAT command. RDSTAT is directly supported by LM628 hardware and is executed by pulling $\overline{CS}$, $\overline{PS}$, and $\overline{RD}$ logic low.

INITIALIZATION MODULE

In general, an initialization module contains a reset command and other initialization, interrupt control, and data reporting commands.

The example initialization module, detailed in *Table 1*, contains a hardware reset block and a PORT 12 command.

Hardware Reset Block

Immediately following power-up, a hardware reset ***must*** *be* executed. Hardware reset is initiated by strobing $\overline{RST}$ (pin 27) logic low for *a* ***minimum of eight LM628 clock periods.*** The reset routine begins after $\overline{RST}$ is returned to logic high. During the reset execution time, **1.5 ms** maximum, the LM628 will ignore any commands or attempts to transfer data.

A hardware reset forces the LM628 into the state described in what follows.

1. The derivative sampling coefficient, d_S, is set to one, and all other filter coefficients and filter coefficient input buffers are set to zero. With d_S set to one, the derivative sampling interval is set to $2048/f_{CLK}$.
2. All trajectory parameters and trajectory parameters input buffers are set to zero.
3. The current absolute position of the shaft is set to zero ("home").
4. The breakpoint interrupt is masked (disabled), and the remaining five interrupts are unmasked (enabled).
5. The position error threshold is set to its maximum value, 7FFF hex.
6. The DAC output port is set for an 8-bit DAC interface.

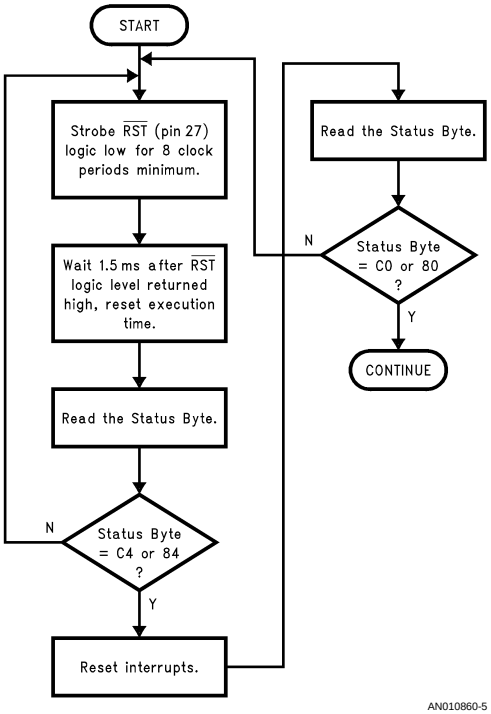


FIGURE 4. Hardware Reset Block

Figure 4 illustrates a hardware reset block that includes an LM628 functionality test. This test *should be* completed immediately following all hardware resets.

Reset Interrupts

An RSTI command sequence allows the user to reset the interrupt flag bits, bits one through six of the status byte. See *Figure 5*. It contains an RSTI command and one data word.

The RSTI command initiates resetting the interrupt flag bits. Command RSTI also resets the host interrupt output pin (pin 17).

TABLE 1. Initialization Module (with Hardware Reset)

Port	Bytes	Command	Comments
	(Note 5)	hardware reset	Strobe $\overline{RST}$, pin 27, logic low for eight clock periods minimum.
		wait	The maximum time to complete hardware reset tasks is 1.5 ms. During this reset execution time, the LM628 will ignore any commands or attempts to transfer data.
c (Note 2)	xx (Note 3)	RDSTAT	This command reads the status byte. It is directly supported by LM628 hardware and can be executed at any time by pulling $\overline{CS}$, $\overline{PS}$, and $\overline{RD}$ logic low. Status information remains valid as long as $\overline{RD}$ is logic low.
		decision	If the status byte is C4 hex or 84 hex, continue. Otherwise loop back to hardware reset.
c	1D	RSTI	This command resets *only* the interrupts indicated by zeros in bits one through six of the next data word. It also resets bit fifteen of the Signals Register and the host interrupt output pin (pin 17).
		Busy-bit Check Module	
d	xx	HB (Note 4)	don't care
d	00	LB	Zeros in bits one through six indicate *all* interrupts will be reset.
		Busy-bit Check Module	
c	xx	RDSTAT	This command reads the status byte.
		decision	If the status byte is C0 hex or 80 hex, continue. Otherwise loop back to hardware reset.
c	06	PORT12	The reset default size of the DAC port is eight bits. This command initializes the DAC port for a 12-bit DAC. It should not be issued in systems with an 8-bit DAC.
		Busy-bit Check Module	

Note 2: The 8-bit host I/O port is a dual-mode port; it operates in command or data mode. The logic level at $\overline{PS}$ (pin 16) selects the mode. Port c represents the LM628 command port-commands are written to the command port and the Status Byte is read from the command port. A logic level of "0" at $\overline{PS}$ selects the command port. Port d represents the LM628 data port — data is both written to and read from the data port. A logic level of "1" at $\overline{PS}$ selects the data port.

Note 3: x - don't care

Note 4: HB - high byte, LB - low byte

Note 5: All values represented in hex.

Immediately following the RSTI command, a single data word is written. The first byte is not used. Logical zeros in bits one through six of the second byte reset the corresponding interrupts. See *Figure 6*. Any combination of the interrupt flag bits can be reset within a single RSTI command sequence. This feature allows interrupts to be serviced according to a user-programmed priority.

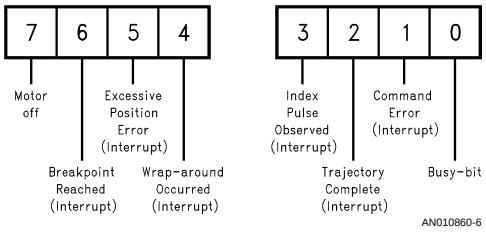


FIGURE 5. Status Byte Bit Allocation

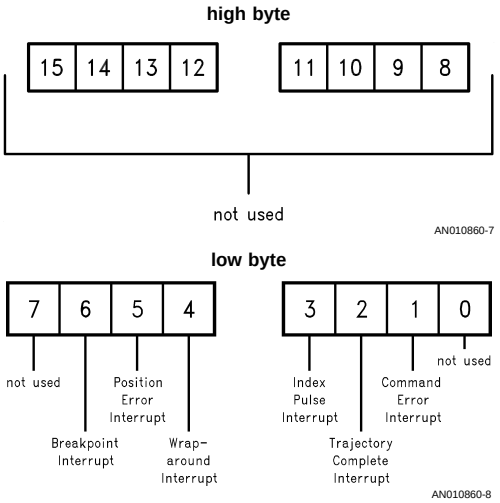


FIGURE 6. Interrupt Mask/Reset Bit Allocations

In the case of the example module, the second byte of the RSTI data word, 00 hex, resets *all* interrupt flag bits. See *Table 1*.

DAC Port Size

During both hardware and software resets, the DAC output port defaults to 8-bit mode. If an LM628 control loop utilizes a 12-bit DAC, command PORT12 should be issued immediately following the hardware reset block and all subsequent resets. Failure to issue command PORT12 will result in erratic, unpredictable motor behavior.

If the control loop utilizes an 8-bit DAC, command PORT12 must not be executed; this too will result in erratic, unpredictable motor behavior.

An LM629 will ignore command PORT8 (as it provides an 8-bit sign/magnitude PWM output). Command PORT12 *should not* be issued in LM629-based systems.

Software Reset Considerations

After the initial hardware reset, resets can be accomplished with either a hardware reset or command RESET (software reset). Software and hardware resets execute the same tasks (Note 6) and require the same execution time, 1.5 ms maximum. During software reset execution, the LM628 will ignore any commands or attempts to transfer data.

The hardware reset module includes an LM628 functionality test. This test is ***not*** required after a software reset.

Table 2 details an initialization module that uses a software reset.

Note 6: In the case of a software reset, the position error threshold remains at its pre-reset value.

TABLE 2. Initialization Module (with Software Reset)

Port	Bytes	Command	Comments
c	00	RESET	See Initialization Module text.
		wait	The maximum time to complete RESET tasks is 1.5 ms.
c	06	PORT12	The RESET default size of the DAC port is eight bits. This command initializes the DAC port for a 12-bit DAC. It should not be issued in a system with an 8-bit DAC.
		Busy-bit Check Module	
c	1D	RSTI	This command resets *only* the interrupts indicated by zeros in bits one through six of the next data word. It also resets bit fifteen of the Signals Register and (pin 17) the host interrupt output pin.
		Busy-bit Check Module	
d	xx	HB	Don't care
d	00	LB	Zeros in bits one through six indicate *all* interrupts will be reset.
		Busy-bit Check Module	

Comments

Figure 7 illustrates, in simplified block diagram form, the LM628. The profile generator provides the control loop input, desired shaft position. The quadrature decoder provides the control loop feedback signal, actual shaft position. At the first summing junction, actual position is subtracted from desired position to generate the control loop error signal, position error. This error signal is filtered by the PID filter to provide the motor drive signal.

After executing the example initialization module, the following observations are made. With the integration limit term (i_L) and the filter gain coefficients (k_p, k_i, and k_d) initialized to zero, the filter gain is zero. Moreover, after a reset, desired shaft position tracks actual shaft position. Under these conditions, the motor drive signal is zero. The control system can not affect shaft position. The shaft should be stationary

and “free wheeling”. If there is significant drive amplifier offset, the shaft may rotate slowly, but with minimal torque capability.

Note: Regardless of the free wheeling state of the shaft, the LM628 continuously tracks shaft absolute position.

FILTER PROGRAMMING MODULE

The example filter programming module is shown in *Table 4*.

Load Filter Parameters (Coefficients)

An LFIL (Load FILter) command sequence includes command LFIL, a filter control word, and a variable number of data words.

The LFIL command initiates loading filter coefficients into input buffers.

The two data bytes, written immediately after LFIL, comprise the filter control word. The first byte programs the derivative sampling coefficient, d_s (i.e. selects the derivative sampling interval). The second byte indicates, with logical ones in respective bit positions, which of the remaining four filter coefficients will be loaded. See *Figure 8*,*Table 3*. Any combination of the four coefficients can be loaded within a single LFIL command sequence.

Immediately following the filter control word, the filter coefficients are written. Each coefficient is written as a pair of data bytes, a data word. Because any combination of the four coefficients can be loaded within a single LFIL command sequence, the number of data words following the filter control word can vary in the range from zero to four.

In the case of the example module, the first byte of the filter control word, 00 hex, programs a derivative sampling coefficient of one. The second byte, x8 hex, indicates only the proportional gain coefficient will be loaded.

Immediately following the filter control word, the proportional gain coefficent is written. In this example, k_p is set to ten with the data word 000A hex. The other three filter coefficients remain at zero, their reset value.

Update Filter

The update filter command, UDF, transfers new filter coefficients from input buffers to working registers. Until UDF is executed, the new filter coefficients do not affect the transfer characteristic of the filter.

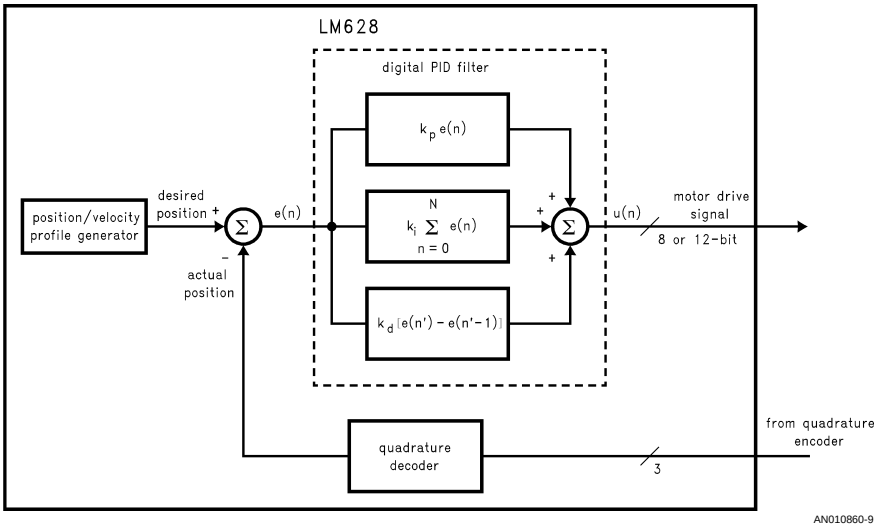


FIGURE 7. LM628 — Simplified Block Diagram Form

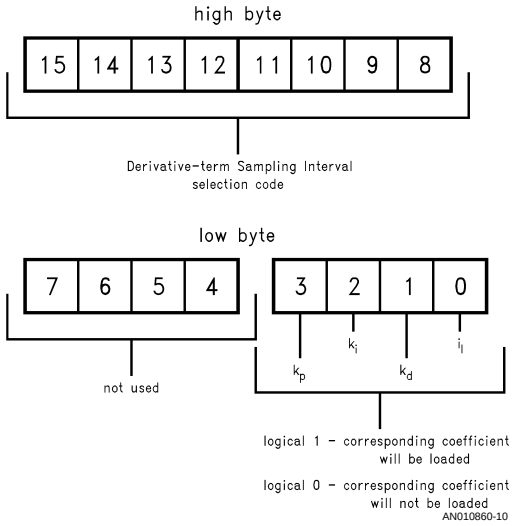


FIGURE 8. Filter Control Word Bit Allocation

TABLE 3. Derivative — Term Sampling Interval Selection Codes

Filter Control Word Bit Position								d_s	Selected Derivative-Term Sampling Interval — T_d
15	14	13	12	11	10	9	8		
0	0	0	0	0	0	0	0	1	T_s
0	0	0	0	0	0	0	1	2	$2T_s$
0	0	0	0	0	0	1	0	3	$3T_s$
0	0	0	0	0	0	1	1	4	$4T_s$
			•					•	•
			•					•	•
			•					•	•
1	1	1	1	1	1	1	1	256	$256T_s$

$$T_s = (2048) \times \left(\frac{1}{f_{CLK}}\right) \quad \text{System Sample Period}$$

$$T_d = d_s \times T_s \quad \text{Derivative-term Sampling Interval}$$

TABLE 4. Filter Programming Module

Port	Bytes	Command	Comments
c	1E	LFIL	This command initiates loading the filter coefficients input buffers.
		Busy-bit Check	Module
d	00	HB	These two bytes are the
d	x8	LB	filter control word. A 00 hex HB sets the derivative sampling interval to $2048/f_{CLK}$ by setting d_s to one. A x8 hex LB indicates only k_p will be loaded. The other filter parameters will remain at zero, their reset default value.
		Busy-bit Check	Module
d	00	HB	These two bytes set k_p
d	0A	LB	to ten.
		Busy-bit Check	Module
c	04	UDF	This command transfers new filter coefficients from input buffers to working registers. Until UDF is executed, coefficients loaded via the LFIL command do not affect the filter transfer characteristic.
		Busy-bit Check	Module

Comments

After executing both the example initialization and example filter programming modules, the following observations are made. Filter gain is nonzero, but desired shaft position continues to track actual shaft position. Under these conditions, the motor drive signal remains at zero. The shaft should be stationary and "free wheeling". If there is significant drive amplifier offset, the shaft may rotate slowly, but with minimal torque capability.

Initially, k_p should be set below twenty, d_s should be set to one, and k_i, k_d, and i_l should remain at zero. These values will not provide optimum system performance, but they will be sufficient to test system functionality. See Tuning the PID Filter.

TRAJECTORY PROGRAMMING MODULE

Table 5 details the example trajectory programming module.

Load Trajectory Parameters.

An LTRJ (Load TRaJectory) command sequence includes command LTRJ, a trajectory control word, and a variable number of data words.

The LTRJ command initiates loading trajectory parameters into input buffers.

The two data bytes, written immediately after LTRJ, comprise the trajectory control word. The first byte programs, with logical ones in respective bit positions, the trajectory mode (velocity or position), velocity mode direction, and stopping mode. See Stop Module. The second byte indicates, with logical ones in respective bit positions, which of the three trajectory parameters will be loaded. It also indicates whether the parameters are absolute or relative. See *Figure 9.* Any combination of the three parameters can be loaded within a single LTRJ command sequence.

Immediately following the trajectory control word, the trajectory parameters are written. Each parameter is written as a pair of data words (four data bytes). Because any combination of the three parameters can be loaded within a single LTRJ command sequence, the number of data words following the trajectory control word can vary in the range from zero to six.

In the case of the example module, the first byte of the trajectory control word, 00 hex, programs the LM628 to operate in position mode. The second byte, 0A hex, indicates velocity and position will be loaded and both parameters are absolute. Four data words, two for each parameter loaded, follow the trajectory control word.

Start Motion Control

The start motion control command, STT (STarT), transfers new trajectory parameters from input buffers to working registers and begins execution of the new trajectory. Until STT is executed, the new trajectory parameters do not affect shaft motion.

Note: At this point no actual trajectory parameters are loaded. Calculation of trajectory parameters and execution of example moves is left for a later section.

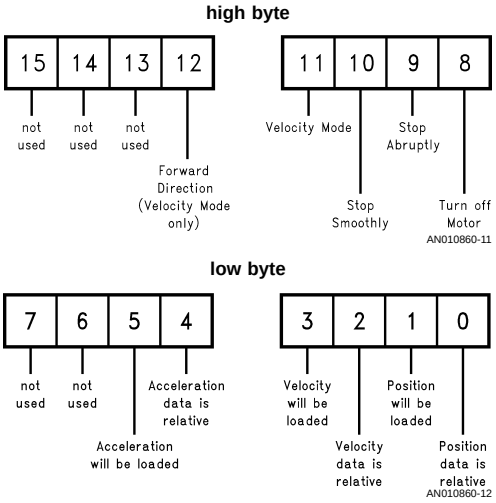


FIGURE 9. Trajectory Control Word Bit Allocation

TABLE 5. Trajectory Programming Module

Port	Bytes	Command	Comments
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	00	HB	These two bytes are the
d	0A	LB	trajectory control word. A 0A hex LB indicates velocity and position will be loaded and both parameters are absolute.
		Busy-bit Check Module	
d	xx	HB	Velocity is loaded in two
d	xx	LB	data words. These two bytes are the high data word.
		Busy-bit Check Module	
d	xx	HB	velocity data word (low)
d	xx	LB	
		Busy-bit Check Module	
d	xx	HB	Position is loaded in two
d	xx	LB	data words. These two bytes are the high data word.
		Busy-bit Check Module	
d	xx	HB	position data word (low)
d	xx	LB	
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.
		Busy-bit Check Module	

STOP MODULE

This module demonstrates the programming flow required to stop shaft motion.

While the LM628 operates in position mode, normal stopping is always smooth and occurs automatically at the end of a specified trajectory (i.e. no stop module is required). Under exceptional conditions, however, a stop module can be used to affect a premature stop.

While the LM628 operates in velocity mode, stopping is always accomplished via a stop module.

The example stop module, shown in *Table 5*, utilizes an LTRJ command sequence and an STT command.

Load Trajectory Parameters

Bits eight through ten of the trajectory control word select the stopping mode. See *Figure 9*.

In the case of the example module, the first byte of the trajectory control word, x1 hex, selects motor-off as the desired stopping mode. This mode stops shaft motion by setting the motor drive signal to zero (the appropriate offset-binary code to apply zero drive to the motor).

Setting bit nine of the trajectory control word selects stop abruptly as the desired stopping mode. This mode stops shaft motion (at maximum deceleration) by setting the target position equal to the current position.

Setting bit ten of the trajectory control word selects stop smoothly as the desired stopping mode. This mode stops shaft motion by decelerating at the current user-programmed acceleration rate.

Note: Bits eight through ten of the trajectory control word must be used exclusively; only one of them should be logic one at any time.

Start Motion Control

The start motion control command, STT, must be executed to stop shaft motion.

Comments

After shaft motion is stopped with either an “abrupt” or a “smooth” stop module, the control system will attempt to hold the shaft at its current position. If forced away from this desired resting position and released, the shaft will move back to the desired position. Unless new trajectory parameters are loaded, execution of another STT command will restart the specified move.

After shaft motion is stopped with a “motor-off” stop module, desired shaft position tracks actual shaft position. Consequently, the motor drive signal remains at zero and the control system can not affect shaft position; the shaft should be stationary and free wheeling. If there is significant drive amplifier offset, the shaft may rotate slowly, but with minimal torque capability. Unless new trajectory parameters are loaded, execution of another STT command will restart the specified move.

TABLE 6. Stop Module (Motor-Off)

Port	Bytes	Command	Comments
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	x1	HB	These two bytes are the
d	00	LB	trajectory control word. A x1 hex HB selects motor-off as the desired stopping mode. A 00 hex LB indicates no trajectory parameters will be loaded.
		Busy-bit Check Module	
c	01	STT	The start motion control command, STT, must be executed to stop shaft motion.
		Busy-bit Check Module	

II. PROGRAMS

This section focuses on the development of four brief LM628 programs.

LOOP PHASING PROGRAM

Following initial power-up, the correct polarity of the motor drive signal must be determined. If the polarity is incorrect (loop inversion), the drive signal will push the shaft away from its desired position rather than towards it. This results in "motor runaway", a condition characterized by the motor running continuously at high speed.

The loop phasing program, detailed in *Table 7*, contains both the example initialization and filter programming modules. It also contains an LTRJ command sequence and an STT command.

Note: Execution of this simple program is only required the *first* time a new system is used.

Load Trajectory Parameters

An LTRJ (Load TRaJectory) command sequence includes command LTRJ, a trajectory control word, and a variable number of data words.

In the case of the Loop Phasing Program, the first byte of the trajectory control word, 00 hex, programs the LM628 to operate in position mode. The second byte, 00 hex, indicates no trajectory parameters will be loaded (i.e. in this program, zero data words follow the trajectory control word). The three trajectory parameters will remain at zero, their reset value.

Start Motion Control

The start motion control command, STT (STarT), transfers new trajectory parameters from input buffers to working registers and begins execution of the new trajectory. Until STT is executed, the new trajectory parameters do not affect shaft motion.

TABLE 7. Loop Phasing Program

Port	Bytes	Command	Comments
		Initialization Module	
		Filter Programming Module	
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	00	HB	These two bytes are the trajectory control word. A 00 hex LB indicates no trajectory parameters will be loaded.
d	00	LB	
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.

Comments

Execution of command STT results in execution of the desired trajectory. With the acceleration set at zero, the profile generator generates a desired shaft position that is both constant and equal to the current absolute position. See *Figure 7*. Under these conditions, the control system will attempt to hold the shaft at its current absolute postion. The shaft will feel lightly "spring loaded". If forced (CAREFULLY) away from its desired position and released, the shaft will spring back to the desired position.

If the polarity of the motor drive signal is incorrect (loop inversion), motor runaway will occur immediately after execution of command STT, or after the shaft is forced (CAREFULLY) from its resting position.

Loop inversion can be corrected with one of three methods: interchanging the shaft position encoder signals (channel A and channel B), interchanging the motor power leads, or inverting the motor command signal before application to the motor drive amplifier. For LM629 based systems, loop inversion can be corrected by interchanging the motor power leads, interchanging the shaft position encoder signals, or logically inverting the PWM sign signal.

SIMPLE ABSOLUTE POSITION MOVE

The Simple Absolute Position Move Program, detailed in *Table 8*, utilizes both the initialization and filter programming modules, as well as, an LTRJ command sequence and an STT command.

Factors that influenced the development of this program included the following: the program must demonstrate simple trajectory parameters calculations, the program must demonstrate the programming flow required to load and execute an absolute position move, and correct completion of the move must be verifiable through simple observation.

Move: The shaft will accelerate at 0.1 rev/sec^2 until it reaches a maximum velocity of 0.2 rev/sec, and then decelerate to a stop exactly two revolutions from the starting position. See *Figure 10*.

Note: Absolute position is position measured relative to zero (home). An absolute position move is a move that ends at a specified absolute position. For example, independent of the current absolute position of the shaft, if an absolute position of 30,000 counts is specified, upon completion of the move the absolute position of the shaft will be 30,000 counts (i.e. 30,000 counts relative to zero). The example program calls for a position move of two revolutions. Because the starting absolute position is 0 counts, the move is accomplished by specifying an absolute position of 8000 counts. See *Table 8*.

The Quadrature Incremental Encoder

As a supplement to the trajectory parameters calculations, a brief discussion is provided here to differentiate between encoder ***lines*** and encoder ***counts***.

A quadrature incremental shaft encoder encodes shaft rotation as electrical pulses. *Figure 11* details the signals generated by a 3-channel quadrature incremental encoder. The LM628 decodes (or "counts") a quadrature incremental signal to determine the absolute position of the shaft.

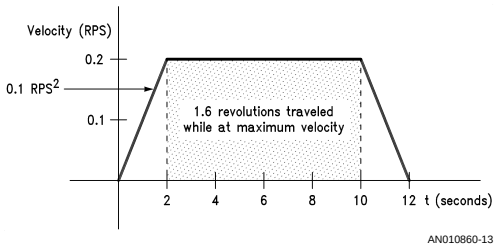


FIGURE 10. Velocity Profile for Simple Absolute Position Move Program

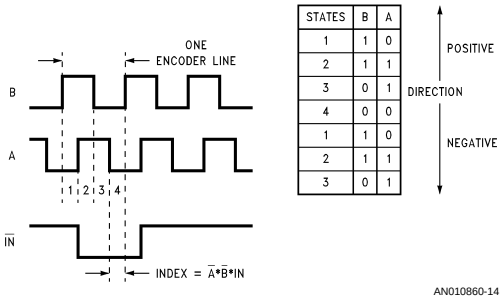


FIGURE 11. 3-Channel Quadrature Encoder Signals

The resolution of a quadrature incremental encoder is usually specified as a number of ***lines***. This number indicates the number of cycles of the output signals for each complete shaft revolution. For example, an N-line encoder generates N cycles of its output signals during each complete shaft revolution.

By definition, two signals that are in quadrature are 90° out of phase. When considered together, channels A and B (*Figure 11*) traverse four distinct digital states during each full cycle of either channel. Each state transition represents one ***count*** of shaft motion. The leading channel indicates the direction of shaft rotation.

Each line, therefore, represents one cycle of the output signals, and each cycle represents four encoder counts.

$$\left(N\frac{\text{CYCLES}}{\text{REVOLUTION}}\right)\times\left(4\frac{\text{COUNTS}}{\text{CYCLE}}\right)=4N\frac{\text{COUNTS}}{\text{REVOLUTION}}$$

The reference system uses a one thousand line encoder.

$$\left(1000\frac{\text{CYCLES}}{\text{REVOLUTION}}\right)\times\left(4\frac{\text{COUNTS}}{\text{CYCLE}}\right)=4000\frac{\text{COUNTS}}{\text{REVOLUTION}}$$

Sample Period

Sampling of actual shaft position occurs at a fixed frequency, the reciprocal of which is the system sample period. The system sample period is the unit of time upon which shaft acceleration and velocity are based.

$$T_S=(2048)\times\left(\frac{1}{f_{CLOCK}}\right)\ \text{System Sample Period}$$

The reference system uses an 8 MHz clock. The sample period of the reference system follows directly from the definition.

$$T_S=(2048)\times\left(\frac{1}{8\times10^6\ \text{Hz}}\right)=256\times10^{-6}\frac{\text{SECONDS}}{\text{SAMPLE}}$$

Trajectory Parameters Calculations

The shaft will accelerate at 0.1 rev/sec^2 until it reaches a maximum velocity of 0.2 rev/sec, and then decelerate to a stop exactly two revolutions from the starting position.

Trajectory parameters calculations for this move are detailed in *Figure 12*.

Comments

After completing the move, the control system will attempt to hold the shaft at its current absolute position. The shaft will feel lightly “spring loaded”. If forced away from its desired resting position and released, the shaft will move back to the desired position.

$$A = \left(4000 \frac{\text{COUNTS}}{\text{REVOLUTION}}\right) \times \left(256 \times 10^{-6} \frac{\text{SECONDS}}{\text{SAMPLE}}\right)^2 \times \left(0.1 \frac{\text{REVOLUTIONS}}{\text{SECOND}^2}\right) = 2.62 \times 10^{-5} \frac{\text{COUNTS}}{\text{SAMPLE}^2}$$

$$A = \left(2.62 \times 10^{-5} \frac{\text{COUNTS}}{\text{SAMPLE}^2}\right) \times (65{,}536) = 1.718 \frac{\text{COUNTS}}{\text{SAMPLE}^2} \quad \text{Acceleration Scaled}$$

$$A = 2 \frac{\text{COUNTS}}{\text{SAMPLE}^2} \quad \text{Acceleration Rounded}$$

$$A = 00\ 00\ 00\ 02 \text{ hex} \frac{\text{COUNTS}}{\text{SAMPLE}^2}$$

$$V = \left(4000 \frac{\text{COUNTS}}{\text{REVOLUTION}}\right) \times \left(256 \times 10^{-6} \frac{\text{SECONDS}}{\text{SAMPLE}}\right) \times \left(0.2 \frac{\text{REVOLUTIONS}}{\text{SECOND}}\right) = 0.2048 \frac{\text{COUNTS}}{\text{SAMPLE}}$$

$$V = \left(0.2048 \frac{\text{COUNTS}}{\text{SAMPLE}}\right) \times (65{,}536) = 13{,}421.77 \frac{\text{COUNTS}}{\text{SAMPLE}} \quad \text{Velocity Scaled}$$

$$V = 13{,}422 \frac{\text{COUNTS}}{\text{SAMPLE}} \quad \text{Velocity Rounded}$$

$$V = 00\ 00\ 34\ 6E \text{ hex} \frac{\text{COUNTS}}{\text{SAMPLE}}$$

$$P = \left(4000 \frac{\text{COUNTS}}{\text{REVOLUTION}}\right) \times (2.0 \text{ REVOLUTIONS}) = 8000 \text{ COUNTS}$$

$$P = 00\ 00\ 1F\ 40 \text{ hex COUNTS}$$

AN010860-29

FIGURE 12. Calculations of Trajectory Parameters for Simple Absolute Position Move

TABLE 8. Simple Absolute Position Move Program

Port	Bytes	Command	Comments
		Initialization Module	
		Filter Programming Module	
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers
		Busy-bit Check Module	
d	00	HB	These two bytes are the
d	2A	LB	trajectory control word. A 2A hex LB indicates acceleration, velocity, and position will be loaded and all three parameters are absolute.
d	00	HB	Acceleration is loaded in
d	00	LB	two data words. These two bytes are the high data word. In this case, the acceleration is 0.1 rev/sec^2.
		Busy-bit Check Module	
d	00	HB	acceleration data word
d	02	LB	(low)
d	00	HB	velocity is loaded in two
d	00	LB	data words. These two bytes are the high data word. In this case, the velocity is 0.2 rev/sec.
		Busy-bit Check Module	
d	34	HB	velocity data word (low)
d	6E	LB	
		Busy-bit Check Module	
d	00	HB	Position is loaded in two
d	00	LB	data words. These two bytes are the high data word. In this case, the position loaded is eight thousand counts. This results in a move of two revolutions in the forward direction.
		Busy-bit Check Module	
d	1F	HB	position data word (low)
d	40	LB	
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.

Note: Target position is the final requested position. If the shaft is stationary, and motion has not been stopped with a "motor-off" stop module, the current absolute position of the shaft is the target position. If motion has been stopped with a "motor-off" stop module, or a position move has begun, the absolute position that corresponds to the endpoint of the current trajectory is the target position. Relative position is position measured relative to the current target position of the shaft. A relative position move is a move that ends the specified "relative" number of counts away from the current target position of the shaft. For example, if the current target position of the shaft is 10 counts, and a relative position of 30,000 counts is specified, upon completion of the move the absolute position of the shaft will be 30,010 counts (i.e. 30,000 counts relative to 10 counts).

SIMPLE RELATIVE POSITION MOVE

This program demonstrates the programming flow required to load and execute a relative position move. See *Table 9*.

Move: Independent of the current resting position of the shaft, the shaft will complete thirty revolutions in the reverse direction. Total time to complete the move is fifteen seconds. Total time for acceleration and deceleration is five seconds.

Load Trajectory Parameters

The first byte of the trajectory control word, 00 hex, programs position mode operation. The second byte, 2B hex, indicates all three trajectory parameters will be loaded. It also indicates both acceleration and velocity will be absolute values while position will be a relative value.

Trajectory Parameters Calculations

Independent of the current resting position of the shaft, the shaft will complete thirty revolutions in the reverse direction. Total time to complete the move is fifteen seconds. Total time for acceleration and deceleration is five seconds.

The reference system utilizes a one thousand line encoder. The number of counts for each complete shaft revolution and the total counts for this position move are determined.

$$\left(1000\ \frac{\text{CYCLES}}{\text{REVOLUTION}}\right)\times\left(4\ \frac{\text{COUNTS}}{\text{CYCLE}}\right)=4000\ \frac{\text{COUNTS}}{\text{REVOLUTION}}$$

$$\left(4000\ \frac{\text{COUNTS}}{\text{REVOLUTION}}\right)\times(30\ \text{REVOLUTIONS})=120{,}000\ \text{COUNTS}$$

With respect to time, two-thirds of the move is made at maximum velocity and one-third is made at a velocity equal to one-half the maximum velocity (Note 7). Therefore, total counts traveled during acceleration and deceleration periods is one-fifth the total counts traveled. See *Figure 13*.

$$\frac{120{,}000\ \text{COUNTS}}{5}=24{,}000\ \text{COUNTS}\quad\text{total counts traveled during acceleration and deceleration}$$

$$\frac{24{,}000\ \text{COUNTS}}{2}=12{,}000\ \text{COUNTS}\quad\text{counts traveled during acceleration}$$

The reference system uses an 8 MHz clock. The sample period of the reference system is determined.

$$T_S=(2048)\times\left(\frac{1}{8\times10^6\text{Hz}}\right)=256\times10^{-6}\ \frac{\text{SECONDS}}{\text{SAMPLE}}$$

The number of samples during acceleration (and deceleration) is determined.

$$\frac{2.5\ \text{SECONDS}}{256\times10^{-6}\ \frac{\text{SECONDS}}{\text{SAMPLE}}}=9766\ \text{SAMPLES}\quad\text{number of samples during acceleration}$$

Using the number of counts traveled during acceleration and the number of samples during acceleration, acceleration is determined.

$$s=\frac{at^2}{2}\quad\text{distance traveled during time t at acceleration a}$$

$$a=\frac{2s}{t^2}=\frac{(2)\times(12{,}000\ \text{COUNTS})}{(9766\ \text{SAMPLES})^2}=0.000252\ \frac{\text{COUNTS}}{\text{SAMPLE}^2}$$

Total counts traveled while at maximum velocity is four-fifths the total counts traveled.

$$\frac{(4) \times (120{,}000\ \text{COUNTS})}{5} = 96{,}000\ \text{COUNTS}$$

Note 7: Average velocity during acceleration and deceleration periods is one-half the maximum velocity.

TABLE 9. Simple Relative Position Move Program

Port	Bytes	Command	Comments
		Initialization Module	
		Filter Programming Module	
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	00	HB	These two bytes are the trajectory control word. A 2B hex LB indicates all three parameters will be loaded and both acceleration and velocity will be absolute values while position will be a relative value.
d	2B	LB	
		Busy-bit Check Module	
d	00	HB	Acceleration is loaded in two data words. These two bytes are the high data word. In this case, the acceleration is 17 counts/sample^2.
d	00	LB	
		Busy-bit Check Module	
d	00	HB	acceleration data word (low)
d	11	LB	
		Busy-bit Check Module	
d	00	HB	Velocity is loaded in two data words. These two bytes are the high data word. In this case, velocity is 161,087 counts/sample.
d	02	LB	
		Busy-bit Check Module	
d	75	HB	velocity data word (low)
d	3F	LB	
		Busy-bit Check Module	
d	FF	HB	Position is loaded in two data words. These two bytes are the high data word. In this case, the position loaded is −120,000 counts. This results in a move of thirty revolutions in the reverse direction.
d	FE	LB	
		Busy-bit Check Module	
d	2B	HB	position data word (low)
d	40	LB	
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.

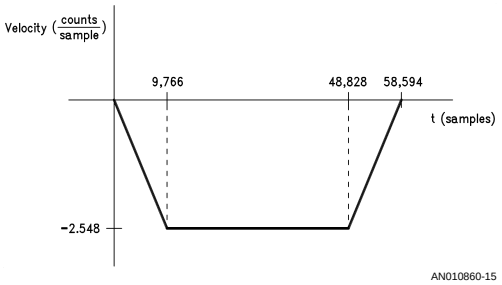


FIGURE 13. Velocity Profile for Simple Relative Position Move Program

The number of samples while at maximum velocity is determined.

$$\frac{10\ \text{SECONDS}}{256 \times 10^{-6}\ \frac{\text{SECONDS}}{\text{SAMPLE}}} = 39{,}062\ \text{SAMPLES}$$ number of samples while at maximum velocity

Using the total counts traveled while at maximum velocity and the number of samples while at maximum velocity, velocity is determined.

$$\frac{96{,}000\ \text{COUNTS}}{39{,}062\ \text{SAMPLES}} = 2.458\ \frac{\text{COUNTS}}{\text{SAMPLE}}$$

Both acceleration and velocity values are scaled.

$$\left(0.000252\ \frac{\text{COUNTS}}{\text{SAMPLE}^2}\right) \times (65{,}536) = 16.515\ \frac{\text{COUNTS}}{\text{SAMPLE}^2}$$

$$\left(2.458\ \frac{\text{COUNTS}}{\text{SAMPLE}}\right) \times (65{,}536) = 161{,}087.488\ \frac{\text{COUNTS}}{\text{SAMPLE}}$$

Acceleration and velocity are rounded to the nearest integer and all three trajectory parameters are converted to hexadecimal.

$$A = 17 = 00\ 00\ 00\ 11\ \text{hex}\ \frac{\text{COUNTS}}{\text{SAMPLE}^2}$$

$$V = 161{,}087 = 00\ 02\ 75\ 3F\ \text{hex}\ \frac{\text{COUNTS}}{\text{SAMPLE}}$$

$$P = -120{,}000 = \text{FF FE 2B 40 hex COUNTS}$$

BASIC VELOCITY MODE MOVE WITH BREAKPOINTS

This program demonstrates basic velocity mode programming and the (typical) programming flow required to set both absolute and relative breakpoints. See *Table 10*.

Move: The shaft will accelerate at 1.0 rev/sec^2 until it reaches a maximum velocity of 2.0 rev/sec. After completing twenty forward direction revolutions (including revolutions during acceleration), the shaft will accelerate at 1.0 rev/sec^2 until it reaches a maximum velocity of 4.0 rev/sec. After completing twenty forward direction revolutions (including revolutions during acceleration), the shaft will decelerate (at 1.0 rev/sec^2) to a stop. See *Figure 14*.

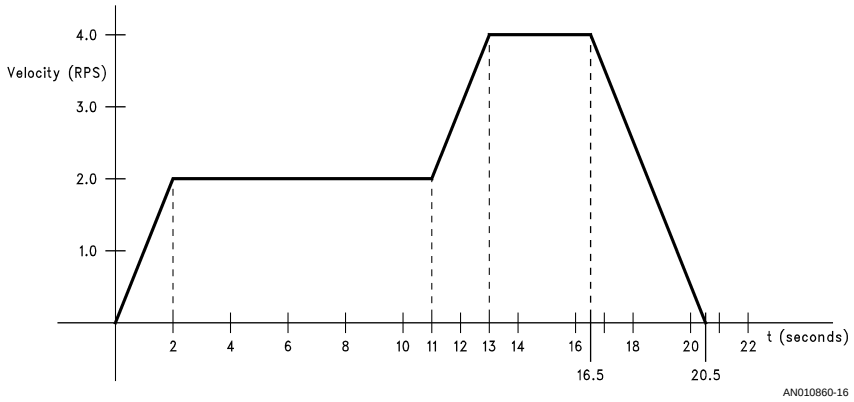


FIGURE 14. Velocity Profile for Basic Velocity Mode with Breakpoints Program

Mask Interrupts

An MSKI command sequence allows the user to determine which interrupt conditions result in host interrupts; interrupting the host via the host interrupt output (pin 17). It contains an MSKI command and one data word.

The MSKI command initiates interrupt masking.

Immediately following the MSKI command, a single data word is written. The first byte is not used. Bits one through six of the second byte determine the masked/unmasked status of each interrupt. See *Figure 6*. Any zeros in this 6-bit field mask (disable) the corresponding interrupts while any ones unmask (enable) the corresponding interrupts.

In the case of the examlple program, the second byte of the MSKI data word, 40 hex, enables the breakpoint interrupt. All other interrupts are disabled (masked).

When interrupted, the host processor can read the Status Byte to determine which interrupt condition(s) occurred. See *Figure 5*.

Note: Command MSKI controls only the host interrupt process. Bits one through six of the Status Byte reflect actual conditions independent of the masked/unmasked status of individual interrupts. This feature allows interrupts to be serviced with a polling scheme.

Set Breakpoints (Absolute and Relative)

An SBPA command sequence enables the user to set breakpoints in terms of absolute shaft position. An SBPR command sequence enables setting breakpoints relative to the current target position. When a breakpoint position is reached, bit six of the status byte, the breakpoint interrupt flag, is set to logic high. If this interrupt is enabled (unmasked), the host will be interrupted via the host interrupt output (pin 17).

An SBPA (or SBPR) command initiates loading/setting a breakpoint. The two data words, written immediately following the SBPA (or SBPR) command, represent the breakpoint position.

The example program contains a relative breakpoint set at 80,000 counts relative to position zero (the current target position). This represents a move of twenty forward direction revolutions. When this position is reached, the LM628 interrupts the host processor, and the host executes a sequence of commands that increases the maximum velocity, resets the breakpoint interrupt flag, and loads an absolute breakpoint.

The example program contains an absolute breakpoint set at 160,000 counts. When this absolute position is reached, the LM628 interrupts the host processor, and the host executes a Smooth Stop Module.

Breakpoint positions for this example program are determined.

$$\left(4000\ \frac{\text{COUNTS}}{\text{REVOLUTION}}\right) \times (20\ \text{REVOLUTIONS})$$

$$= 80{,}000\ \text{COUNTS}\quad \text{relative breakpoint}$$

$$\left(4000\ \frac{\text{COUNTS}}{\text{REVOLUTION}}\right) \times (40\ \text{REVOLUTIONS})$$

$$= 160{,}000\ \text{COUNTS}\quad \text{absolute breakpoint}$$

Load Trajectory Parameters

This example program contains two LTRJ command sequences. The trajectory control word of the first LTRJ command sequence, 1828 hex, programs forward direction velocity mode, and indicates an absolute acceleration and an absolute velocity will be loaded. The trajectory control word of the second LTRJ command sequence, 180C hex, programs forward direction velocity mode, and indicates a relative velocity will be loaded. See *Figure 9*.

Trajectory parameters calculations follow the same format as those detailed for the simple absolute position move. See *Figure 12*.

TABLE 10. Basic Velocity Mode Move with Breakpoints Program

Port	Bytes	Command	Comments
		Initialization Module	
		Filter Programming Module	
c	1C	MSKI	Mask interrupts.
		Busy-bit Check Module	
d	xx	HB	don't care
d	40	LB	A 40 hex LB enables (unmasks) the breakpoint interrupt. All other interrupts are disabled (masked).
		Busy-bit Check Module	
c	21	SPBR	This command initiates loading a relative breakpoint.
		Busy-bit Check Module	
d	00	HB	A breakpoint is loaded
d	01	LB	in two data words. These two bytes are the high data word. In this case, the breakpoint is 80,000 counts relative to the current commanded target position (zero).
		Busy-bit Check Module	
d	38	HB	breakpoint data word (low)
d	80	LB	
		Busy-bit Check Module	
c	1F	LTRJ	Load trajectory.
		Busy-bit Check Module	
d	18	HB	These two bytes are the
d	28	LB	trajectory control word. A 18 hex HB programs forward direction velocity mode operation. A 28 hex LB indicates acceleration and velocity will be loaded and both values are absolute.
		Busy-bit Check Module	
d	00	HB	Acceleration is loaded in
d	00	LB	two data words. These two bytes are the high data word. In this case, the acceleration is 1.0 rev/sec^2.
		Busy-bit Check Module	
d	00	HB	acceleration data word (low)
d	11	LB	
		Busy-bit Check Module	
d	00	HB	Velocity is loaded in two
d	02	LB	data words. These two bytes are the high data word. In this case, velocity is 2.0 rev/s.
		Busy-bit Check Module	
d	0C	HB	velocity data word (low)
d	4A	LB	
		Busy-bit Check Module	
c	01	STT	Start motion control.
		Busy-bit Check Module	
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	18	HB	These two bytes are the
d	0C	LB	trajectory control word. A 18 hex HB programs forward direction velocity mode operation. A 0C hex LB indicates only velocity will be loaded and it will be a relative value.
		Busy-bit Check Module	
d	00	HB	Velocity is loaded in two
d	02	LB	data words. These two bytes are the high data word. In this case, velocity is 2.0 rev/s. Because this is a relative value, the current velocity will be increased by 2.0 rev/s. The resultant velocity will be 4.0 rev/s.
		Busy-bit Check Module	
d	0C	HB	velocity data word (low)
d	4A	LB	
		wait	This wait represents the host processor waiting for an LM628 breakpoint interrupt.
c	01	STT	Start motion control.
		Busy-bit Check Module	
c	1D	RSTI	Reset interrupts.
		Busy-bit Check Module	
d	xx	HB	don't care
d	00	LB	Zeros in bits one through six reset **all** interrupts.

TABLE 10. Basic Velocity Mode Move with Breakpoints Program (Continued)

Port	Bytes	Command	Comments
		Busy-bit Check Module	
c	20	SPBA	This command initiates loading an absolute breakpoint.
		Busy-bit Check Module	
d	00	HB	A breakpoint is loaded
d	02	LB	in two data words. These two bytes are the high data word. In this case, the breakpoint is 160,000 counts absolute.
		Busy-bit Check Module	
d	71	HB	breakpoint data word (low)
d	00	LB	
		wait	This wait represents the host processor waiting for an LM628 breakpoint interrupt.
		"Smooth" Stop Module	

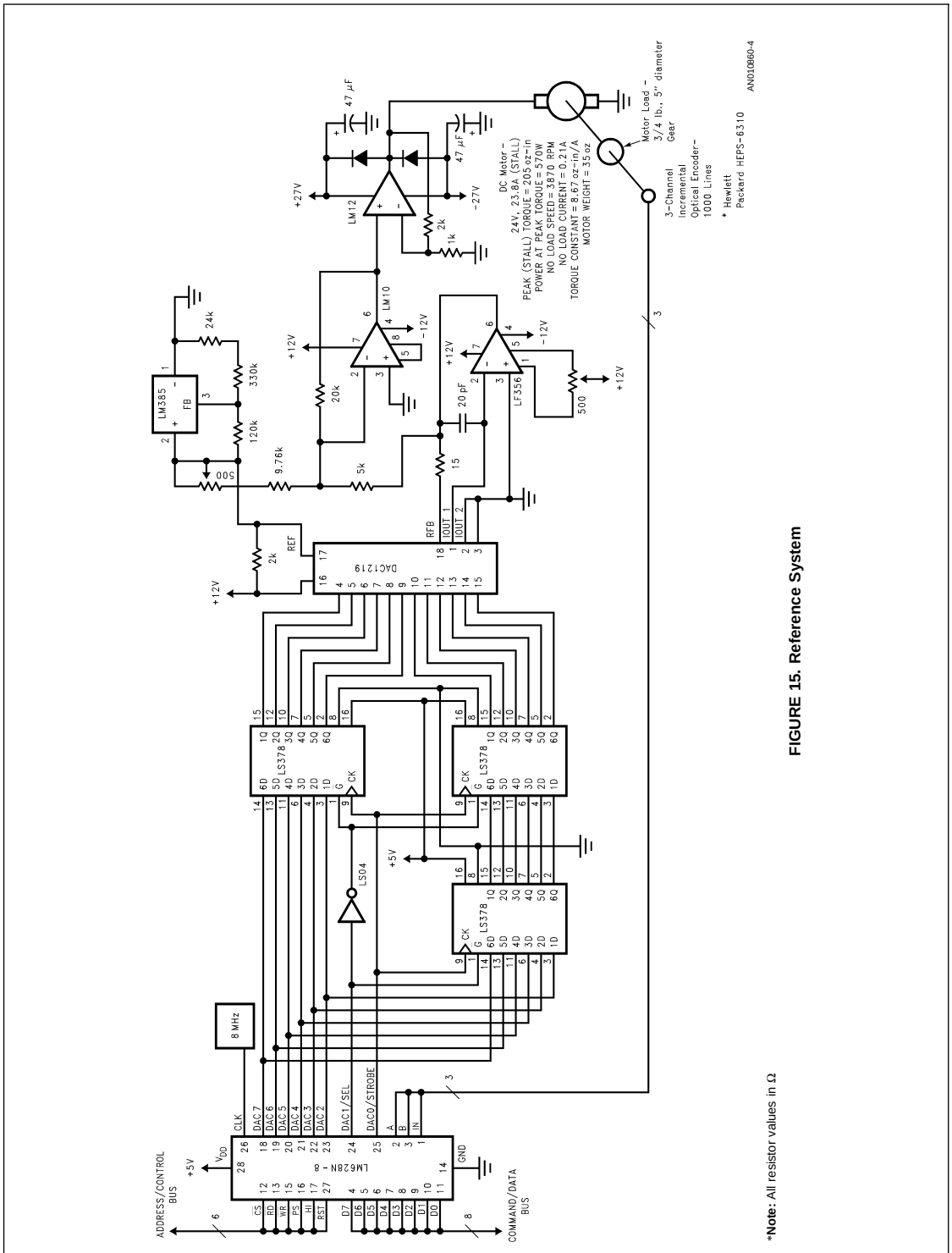


FIGURE 15. Reference System

***Note:** All resistor values in Ω

III. TUNING THE PID FILTER

BACKGROUND

The transient response of a control system reveals important information about the "quality" of control, and because a step input is easy to generate and sufficiently drastic, the transient response of a control system is often characterized by the response to a step input, the system step response.

In turn, the step response of a control system can be characterized by three attributes: maximum overshoot, rise time, and settling time. These step response attributes are defined in what follows and detailed graphically in *Figure 16*.

1. The maximum overshoot, Mp, is the maximum peak value of the response curve measured from unity. The amount of maximum overshoot directly indicates the relative stability of the system.
2. The rise time, t_r, is the time required for the response to rise from ten to ninety percent of the final value.
3. The settling time, t_s, is the time required for the response to reach and stay within two percent of the final value.

A critically damped control system provides optimum performance. The step response of a critically damped control system exhibits the minimum possible rise time that maintains zero overshoot and zero ringing (damped oscillations). *Figure 17* illustrates the step response of a critically damped control system.

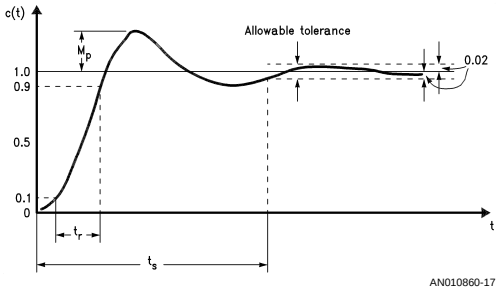


FIGURE 16. Unit Step Response Curve Showing Transient Response Attributes

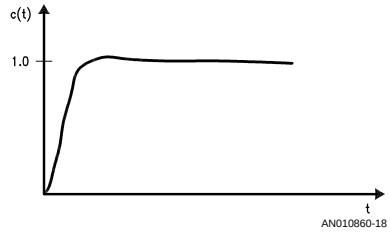


FIGURE 17. Unit Step Response of a Critically Damped System

INTRODUCTION

The LM628 is a digital PID controller. The loop-compensation filter of a PID controller is usually tuned experimentally, especially if the system dynamics are not well known or defined.

The ultimate goal of tuning the PID filter is to critically damp the motor control system — provide optimum tracking and settling time.

As shown in *Figure 7*, the response of the PID filter is the sum of three terms, a proportional term, an integral term, and a derivative term. Five variables shape this response. These five variables include the three gain coefficients (k_p, k_i, and k_d), the integration limit coefficient (i_l), and the derivative sampling coefficient (d_s). *Tuning the filter equates to determining values for these variable coefficients, values that critically damp the control system* .

Filter coefficients are best determined with a two-step experimental approach. In the first step, the values of k_p, k_i, and k_d (along with i_l and d_s) are systematically varied until reasonably good response characteristics are obtained. Manual and visual methods are used to evaluate the effect of each coefficient on system behavior. In the second step, an oscilloscope trace of the system step response provides detailed information on system damping, and the filter coefficients, determined in step one, are modified to critically damp the system.

Note: In step one, adjustments to filter coefficient values are inherently coarse, while in step two, adjustments are inherently fine. Due to this coarse/fine nature, steps one and two complement each other, and the two-step approach is presented as the "best" tuning method. The PID filter can be tuned with either step one or step two alone.

STEP ONE — MANUAL VISUAL METHOD

Introduction

In the first step, the values of k_p, k_i, and k_d (along with i_l and d_s) are systematically varied until reasonably good response characteristics are obtained. Manual and visual methods are used to evaluate the effect of each coefficient on system behavior.

Note: The next four numbered sections are ordered steps to tuning the PID filter.

1. Prepare the System

The initialization section of the filter tuning program is executed to prepare the system for filter tuning. See *Table 11*. This section initializes the system, presets the filter parameters (k_p, k_i, il = 0, k_d = 2, d_s = 1), and commands the control loop to hold the shaft at the current position.

After executing the initialization section of the filter tuning program, both desired and actual shaft positions equal zero; the shaft should be stationary. Any displacement of the shaft constitutes a position error, but with both k_p and k_i set to zero, the control loop can not correct this error.

2. Determine the Derivative Gain Coefficient

The filter derivative term provides damping to eliminate oscillation and minimize overshoot and ringing, stabilize the system. Damping is provided as a force proportional to the rate of change of position error, and the constant of proportionality is k_d x d_s. See *Figure 18*.

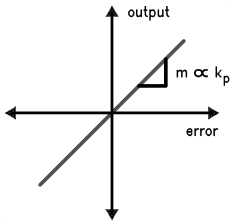
Coefficients k_d and d_s are determined with an iterative process. Coefficient k_d is systematically increased until the shaft begins high frequency oscillations. Coefficient d_s is then increased by one. The entire process is repeated until d_s reaches a value appropriate for the system.

The system sample period sets the time interval between updates of position error. The derivative sampling interval is an

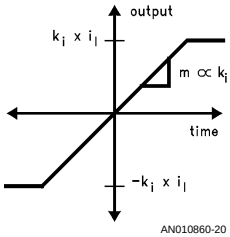
integer multiple of the system sample period. See *Table 3*. It sets the time interval between successive position error samples used in the derivative term, and, therefore, directly affects system damping. The derivative sampling interval should be five to ten times smaller than the system mechanical time constant — this means many systems will require low d_s. In general, however, k_d and d_s should be set to give the largest $k_d \times d_s$ product that maintains acceptably low motor vibrations.

Note: Starting k_d at two and doubling it is a good method of increasing k_d. Manually turning the shaft reveals that with each increase of k_d, the resistance of the shaft to turning increases. The shaft feels increasingly sluggish and, because k_d provides a force proportional to the rate of change of position error, the faster the shaft is turned the more sluggish it feels. For the reference system, the final values of k_d and d_s are 4000 and 4 respectively.

Proportional Term



Integral Term



Derivative Term

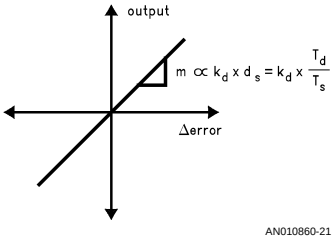


FIGURE 18. Proportional, Integral, and Derivative (PID) Force Components

TABLE 11. Initialization Section— Filter Tuning Program

Port	Bytes	Command	Comments
c	00	RESET	See Initialization Module Text
		wait	The maximum time to complete RESET tasks is 1.5 ms.
c	06	PORT12	The RESET default size of the DAC port is eight bits. This command initializes the DAC port for a 12-bit DAC. It should not be issued in systems with an 8-bit DAC.
		Busy-bit Check Module	
c	1D	RSTI	This command resets **only** the interrupts indicated by zeros in bits one through six of the next data word. It also resets bit fifteen of the Signals Register and the host interrupt pin (pin 17).
		Busy-bit Check Module	
d	xx	HB	don't care
d	00	LB	Zeros in bits one through six indicate **all** interrupts will be reset.
		Busy-bit Check Module	
c	1C	MSKI	This command masks the interrupts indicated by zeros in bits one through six of the next data word.
		Busy-bit Check Module	
d	xx	HB	don't care
d	04	LB	A 04 hex LB enables (unmasks) the trajectory complete interrupt. All other interrupts are disabled (masked). See *Figure 6*.
		Busy-bit Check Module	
c	1E	LFIL	This command initiates loading the filter coefficients input buffers.

TABLE 11. Initialization Section — Filter Tuning Program (Continued)

Port	Bytes	Command	Comments
		Busy-bit Check Module	
d	00	HB	These two bytes are the
d	x2	LB	filter control word. A 00 hex HB sets the derivative sampling interval to $2048/f_{CLK}$ by setting d_s to one. A x2 hex LB indicates only k_d will be loaded. The other filter parameters will remain at zero, their reset default value.
		Busy-bit Check Module	
d	00	HB	These two bytes set k_d
d	02	LB	to two.
		Busy-bit Check Module	
c	04	UDF	This command transfers new filter coefficients from input buffers to working registers. Until UDF is executed, coefficients loaded via the LFIL command do not affect the filter transfer characteristic.
		Busy-bit Check Module	
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	00	HB	These two bytes are the
d	00	LB	trajectory control word. A 00 hex LB indicates no trajectory parameters will be loaded.
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.

3. Determine the Proportional Gain Coefficient

Inertial loading causes following (or tracking) error, position error associated with a moving shaft. External disturbances and torque loading cause displacement error, position error associated with a stationary shaft. The filter proportional term provides a restoring force to minimize these position errors. The restoring force is proportional to the position error and increases linearly as the position error increases. See *Figure 18*. The proportional gain coefficient, k_p, is the constant of proportionality.

Coefficient k_p is determined with an iterative process — the value of k_p is increased, and the system damping is evaluated. This is repeated until the system is critically damped.

System damping is evaluated manually. Manually turning the shaft reveals each increase of k_p increases the shaft "stiffness". The shaft feels spring loaded, and if forced away from its desired holding position and released, the shaft "springs" back. If k_p is too low, the system is over damped, and the shaft recovers too slowly. If k_p is too large, the system is under damped, and the shaft recovers too quickly. This causes overshoot, ringing, and possibly oscillation. The proportional gain coefficient, k_p, is increased to the largest value that does not cause excessive overshoot or ringing. At this point the system is critically damped, and therefore provides optimum tracking and settling time.

Note: Starting k_p at two and doubling it at each iteration is a good method of increasing k_p. The final value of k_p for the reference system is 40.

4. Determine the Integral Gain Coefficient

The filter proportional term minimizes the errors due to inertial and torque loading. The integral term, however, provides a corrective force that can eliminate following error while the shaft is spinning and the deflection effects of a static torque load while the shaft is stationary. This corrective force is proportional to the position error and increases linearly with time. See *Figure 18*. The integral gain coefficient, k_i, is the constant of proportionality.

High values of k_i provide quick torque compensation, but increase overshoot and ringing. In general, k_i should be set to the smallest value that provides the appropriate compromise between three system characteristics: overshoot, settling time, and time to cancel the effects of a static torque load. In systems without significant static torque loading, a k_i of zero may be appropriate.

The corrective force provided by the integral term increases linearly with time. The integration limit coefficient, i_l, acts as a clamping value on this force to prevent integral wind-up, a backlash effect. As noted in *Figure 18*, i_l limits the summation of error (over time), not the product of k_i and this summation. In many systems i_l can be set to its maximum value, 7FFF hex, without any adverse effects. The integral term has no effect if i_l is set to zero.

For the test system, the final values of k_i and i_l are 5 and 1000 respectively.

STEP TWO — STEP RESPONSE METHOD

Introduction

The step response of a control system reveals important information about the "quality" of control — specifically, detailed information on system damping.

In the second step to tuning the PID filter, an oscilloscope trace of the control system step response is used to accurately evaluate system damping, and the filter coefficients, determined in step one, are fine tuned to critically damp the system.

Software Considerations

The step generation section of the filter tuning program provides the control loop with a repetitive small-signal step input. This is accomplished by repeatedly executing a small position move with high maximum velocity and high acceleration. See *Figure 19* and *Table 12*.

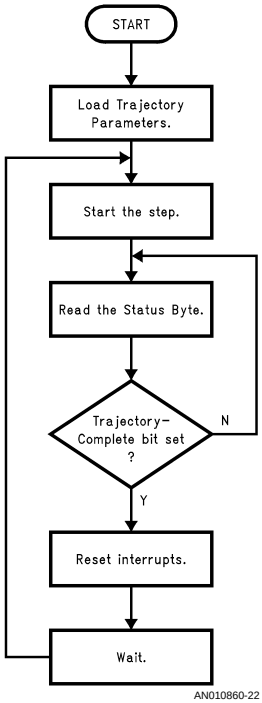


FIGURE 19. Step Generation Section of Filter Tuning Program

TABLE 12. Step Generation Section — Filter Tuning Program

Port	Bytes	Command	Comments
c	1F	LTRJ	This command initiates loading the trajectory parameters input buffers.
		Busy-bit Check Module	
d	00	HB	These two bytes are the
d	2B	LB	trajectory control word. A 2B hex LB indicates acceleration, velocity, and position will be loaded and both acceleration and velocity are absolute while position is relative.
		Busy-bit Check Module	
d	00	HB	Acceleration is loaded in
d	04	LB	two data words. These two bytes are the high data word.
		Busy-bit Check Module	
d	93	HB	acceleration data word (low)
d	E0	LB	
		Busy-bit Check Module	
d	00	HB	Velocity is loaded in two
d	07	LB	data words. These two bytes are the high data word.
		Busy-bit Check Module	
d	A1	HB	velocity data word (low)
d	20	LB	
		Busy-bit Check Module	
d	00	HB	Position is loaded in two
d	00	LB	data words. These two bytes are the high data word.
		Busy-bit Check Module	
d	00	HB	position data word (low)
d	C8	LB	
		Busy-bit Check Module	
c	01	STT	STT must be issued to execute the desired trajectory.
		Busy-bit Check Module	
c	xx	RDSTAT	This command reads the Status Byte. It is directly supported by LM628 hardware and can be executed at any time by pulling $\overline{CS}$, $\overline{PS}$, and $\overline{RD}$ logic low. Status information remains valid as long as $\overline{RD}$ is logic low.
		decision	If the Trajectory Complete interrupt bit is set, continue. Otherwise loop back to RDSTAT.
c	1D	RSTI	This command resets **only** the interrupts indicated by zeros in bits one through six of the next data word. It also resets bit fifteen of the Signals Register and the host interrupt pin (pin 17).
d	xx	HB	don't care
d	00	LB	Zeros in bits one through six indicate **all** interrupts will be reset.
		wait	This wait block inserts a delay between repetitions of the step input. The delay is application specific, but a good range of values for the delay is 5 ms to 5000 ms.
		loop	Loop back to STT.

Hardware Considerations

For a motor control system, an oscilloscope trace of the system step response is a graph of the real position of the shaft versus time after a small and instantaneous change in desired position.

For an LM628-based system, no extra hardware is needed to view the system step response. During a step, the voltage across the motor represents the system step response, and an oscilloscope is used to generate a graph of this response (voltage).

For an LM629-based system, extra hardware is needed to view the system step response. *Figure 20* illustrates a circuit for this purpose. During a step, the voltage output of this circuit represents the system step response, and an oscilloscope is used to generate a graph of this response.

The oscilloscope trigger signal, a rectangular pulse train, is taken from the host interrupt output pin (pin 17) of the LM628/LM629. This signal is generated by the combination of a trajectory complete interrupt and a reset interrupts (RSTI) command. See *Figure 19*.

Note: The circuit of *Figure 20* can be used to view the step response of an LM628-based system.

Observations

What follows are example oscilloscope traces of the step response of the reference system.

Note 8: All traces were generated using the circuit of *Figure 20*.

Note 9: All traces were generated using the following "step" trajectory parameters: relative position, 200 counts; absolute velocity, 500,000 counts/sample; acceleration, 300,000 counts/sample/sample. These values generated a good small-signal step input for the reference system; other systems will require different trajectory parameters. In general, step trajectory parameters consist of a small relative position, a high velocity, and a high acceleration.

The position parameter must be relative. Otherwise, a define home command (DFH) must be added to the main loop of the step generation section — filter tuning program. See *Figure 19*.

The circuit for viewing the system step response uses an 8-bit analog-to-digital converter. See *Figure 20*. To prevent converter overflow, the step position parameter must not be set higher than 200 counts.

Note 10: The circuit of *Figure 20* produces an "inverted" step response graph. The oscilloscope input was inverted to produce a positive-going (more familiar) step response graph.

Figure 21 represents the step response of an under damped control system; this response exhibits excessive overshoot and long settling time. The filter parameters used to generate this response were as follows: k_p, 35; k_i, 5; k_d, 600; d_s, 4; i_l, 1000. *Figure 21* indicates the need to increase k_d, the derivative gain coefficient.

Figure 22 represents the step response of an over damped control system; this response exhibits excessive rise time which indicates a sluggish system. The filter parameters used to generate this response were as follows: k_p, 35; k_i, 5; k_d, 10,000; d_s, 7; i_l, 1000. *Figure 22* indicates the need to decrease k_d and d_s.

Figure 23 represents the step response of a critically damped control system; this response exhibits virtually zero overshoot and short rise time. The filter parameters used to generate this response were as follows: k_p, 40; k_i, 5; k_d, 4000; d_s, 4; i_l, 1000.

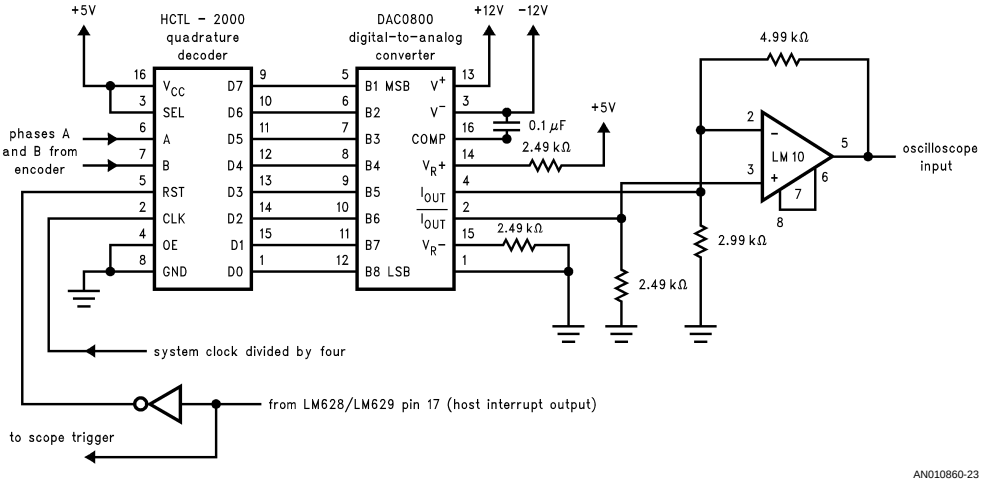


FIGURE 20. Circuit for Viewing the System Step Response with an Oscilloscope

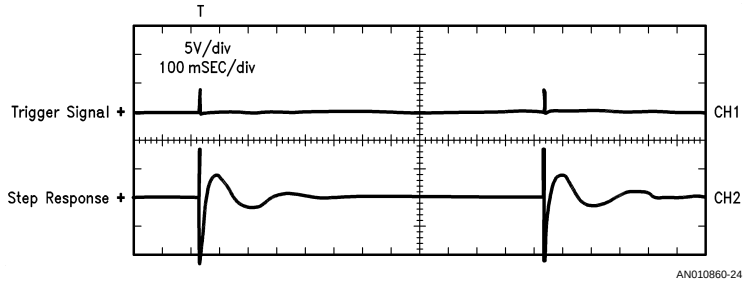


FIGURE 21. The Step Response of an Under Damped Control System

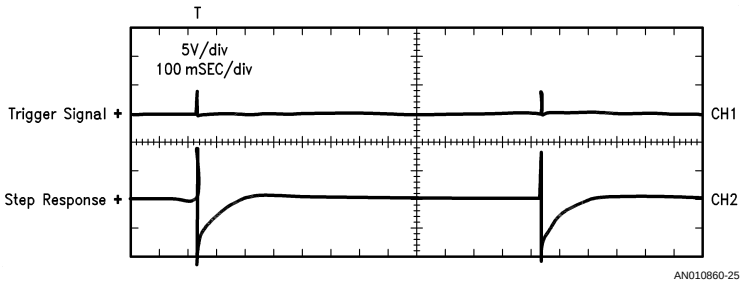


FIGURE 22. The Step Response of an Over Damped Control System

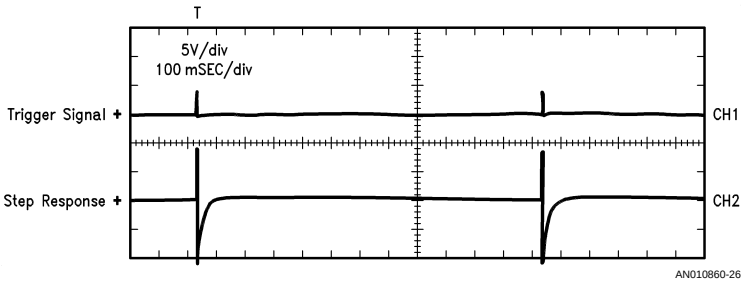


FIGURE 23. The Step Response of a Critically Damped Control System

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507