Digilent Adept Electro-Mechanical Control Interface (DEMC) Programmer's Reference Manual

Revision: August 16, 2010

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the programming interface to the Digilent Adept Electro-Mechanical Control Interface (DEMC) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DEMC subsystem and the API functions used to access its features.

The DEMC subsystem is used to control various types of electrical motor/actuator devices, such as: RC hobby servos, brushed DC motors, brushless DC motors, and unipolar and bipolar stepper motors.

DEMC Port Properties

The port properties are used to indicate which optional parts of the DEMC interface are supported by a given port. The property bits for DEMC ports are divided into two sets: The first set of property bits define a device class for the kind of electro-mechanical device the port is capable of controlling. The second set of property bits define device class specific properties. There may be overlap between the class specific bits for different device classes, so it is important to determine the device class before interpreting the remaining class specific property bits.

The values for all property bits are defined in the header file *demc.h*.

The following port properties bits define EMC device classes:

dprpEmcRcServo	This bit indicates that the port supports controlling RC hobby servos.
dprpEmcBrMotor	This bit indicates that the port supports controlling a brushed DC motor.
dprpEmcBlMotor	This bit indicates that the port support controlling a brushless DC motor.
dprpEmcUniStep	This bit indicates that the port supports controlling a four-phase uni-polar stepper motor.
dprpEmcBiStep	This bit indicates that the port supports controlling a two-phase bi-polar stepper motor.

Note: A given port will never support more than one of the device class properties: dprpEmcServo, dprpEmcBrdc, dprpEmcBldc, dprpEmcUstp, or dprpEmcBstp. If the device itself supports more than one of these device classes, the device will provide multiple ports with a different port for each supported device class.

The following are the class specific property bits for the dprpEmcServo class:

dprpEmcPos	This bit indicates that the port supports setting the position relative to a channel center or to the current channel position.
dprpEmcVel	This bit indicates that the port supports setting a velocity value to be used when changing the position.

dprpEmcAccel	This bit indicates that the port supports setting an acceleration value to be used when changing position.
dprpEmcDecel	This bit indicates that the port supports setting a deceleration value that is different than the acceleration value to use when changing position. This bit will only be set when the dprpEmcAccel bit is also set.
dprpEmcInvert	This bit indicates that the port supports inverting channel position values. This will not be supported unless *dprpEmcPos* is supported.
dprpEmcPersist	This bit indicates that the port supports saving and restoring port settings in non-volatile storage.
dprpEmcScript	This bit indicates that the port supports scripting. A script is a buffer containing a sequence of movement operations to be performed automatically. The exact operations and their encoding is device class specific. (Reserved for future use)

The following are class specific property bits for the dprpUniStep and dprpBiStep stepper motor classes:

dprpEmcAccel	This bit indicates that the port supports setting an acceleration value to be used when changing velocity or position.
dprpEmcDecel	This bit indicates that the port supports setting a deceleration value that is different than the acceleration value to use when changing velocity or position. This bit will only be set when the dprpEmcAccel bit is also set.
dprpEmcStepAbs	This bit indicates that the port supports stepping to an absolute position location.
dprpEmcStepRel	This bit indicates that the port supports stepping to a position relative to the current position.
dprpEmcStepSubstep	This bit indicates that the port supports step subdivisions.
dprpEmcScript	This bit indicates that the port supports scripting. A script is a buffer containing a sequence of movement operations to be performed automatically. The exact operations and their encoding is device class specific. (Reserved for future use)

The following are the class specific property bits for the dprpEmcBrdc class:

dprpEmcAccel	This bit indicates that the port supports setting an acceleration value to be used when changing velocity.
dprpEmcDecel	This bit indicates that the port supports setting a deceleration value that is different than the acceleration value to use when changing velocity. This bit will only be set when the dprpEmcAccel bit is also set.
dprpEmcLoopControl	This bit indicates that the port supports closed loop feedback control for controlling motor speed.

DEMC Basic API Functions

The following API functions make up the basic DEMC interface.

DemcGetVersion(char * szVersion)

Parameters

szVersion	- pointer to buffer to receive version string

This function returns a version number string identifying the version number of the DEMC DLL. The symbol cchVersionMax declared in dpcdecl.h defines the longest string that can be returned in *szVersion*.

DemcGetPortCount(HIF hif, INT32 * pcprt)

Parameters

hif	- open interface handle on the device
pcprt	- pointer to variable to receive count of ports

This function returns the number of DEMC ports supported by the device specified by *hif*.

DemcGetPortProperties(HIF hif, INT32 prtReq, DWORD * pdprp)

Parameters

hif	- open interface handle on the device
prtReq	- port number to query
pdprp	- pointer to variable to return port property bits

This function returns the port properties bits for the specified DEMC port. The port properties bits indicate the specific features of the DEMC port functionality implemented by the specified port.

DemcEnable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to enable the default DEMC port (port 0) on the specified device. This function must be called before any functions that operate on the port may be called for the specified device.

DemcEnableEx(HIF hif, INT32 prtReq)

Parameters

hif	- open interface handle on the device
prtReq	- DEMC port number

This function is used to enable a specific port on devices that support multiple DEMC ports. This function must be called before any functions that operate on the port may be called. The *prtReq* parameter specifies the port number of the DEMC port to enable.

DemcDisable(HIF hif)

Parameters:

hif	- open interface handle on the device

This function is used to disable and end access to the currently enabled EMC port on the specified interface handle.

DEMC Servo Device Class

The following are the API functions used to access DEMC ports that support the RC hobby servo class, *dprpEmcRcServo*:

DemcServoStoreSettings(HIF hif)

Parameters:

hif	- open interface handle on the device

This function will save the current servo settings for the port currently enabled on interface handle *hif* in persistent storage. This function should not be called unless the port supports the *dprpEmcPersist* property. The saved settings include the following items:

Enabled/disabled state for all channels
Minimum and maximum pulse width values for all channels
Channel pulse width values for all channels
Channel center position values for all channels (if *dprpEmcPos* is supported)
Channel velocity values for all channels (if *dprpEmcVelocity* is supported)
Channel acceleration values for all channels (if *dprpEmcAccel* is supported)
Channel deceleration values for all channels (if *dprpEmcDecel* is supported)

DemcServoLoadSettings(HIF hif)

Parameters:

hif	- open interface handle on the device

Restore the previously saved servo settings for the port currently enabled on the specified handle. This function should not be called unless the port supports the *dprpEmcPersist* property.

DemcServoGetChnCount(HIF hif, INT32 * pcchn)

Parameters:

hif	- open interface handle on the device
pcchn	- variable to receive count of channels

Return the number of servo channels supported by the port currently enabled on the specified interface handle.

DemcServoChnEnable(HIF hif, INT32 chn)

Parameters:

hif	- open interface handle on the device
chn	- channel number of channel to enable

Enable the specified channel. Enabling a channel allows for control pulses to be sent to the corresponding servo connector and also allows for updates to occur on the channel.

DemcServoChnDisable(HIF hif, INT32 chn)

Parameters:

hif	- open interface handle on the device
chn	- channel number of channel to disable

Disables the specified channel. Servo control pulses will stop being sent to the corresponding servo connector. If any motion is currently in progress, it is halted.

DemcServoHalt(HIF hif, INT32 chn)

Parameters:

hif	- open interface handle on the device
chn	- channel number

Halt motion on the specified channel. If velocity control is enabled, and the specified channel is currently in motion, the target position will be changed to the current position and motion of the servo will be halted. If velocity control is not enabled, this function has no effect.

DemcServoHaltAll(HIF hif)

Parameters:

hif	- open interface handle on the device

Halt motion on all channels. For each channel: If velocity control is enabled, and the channel is currently in motion, the target position will be changed to the current position and motion of the servo will be halted. If velocity control is not enabled, this function has no effect.

DemcServoSetPosAbs(HIF hif, INT32 chn, SHORT pos)

Parameters:

hif	- open interface handle on the device
chn	- channel number
pos	- position to set

Set the specified channel's target position. The *pos* value is a signed integer that specifies an offset from the channel's defined center. The resulting target position is clamped to the channel's minimum and maximum position values. If velocity control is enabled, the servo will start to move toward the target position at the current velocity value. If velocity control is not enabled, the servo will move as quickly as possible to the target position.

The units of channel position are in microseconds of pulse width. For example: the value 10 specifies that the pulse width of the servo control pulse will be the current channel center value plus 10 microseconds.

DemcServoSetPosRel(HIF hif, INT32 chn, SHORT rel)

Parameters:

hif	- open interface handle on the device
chn	- channel number
rel	- delta from current position

Set the specified channel's target position. The *rel* value is a signed integer that specifies an offset relative to the channel's current position. The resulting target position is clamped to the channel's minimum and maximum position values. If velocity control is enabled, the servo will start to move toward the target position at the current velocity value. If velocity control is not enabled, the servo will move as quickly as possible to the target position.

The units of channel position are in microseconds of pulse width. For example: the value 10 specifies that the pulse width for the servo control pulse will be set to the current channel pulse width plus 10 microseconds.

DemcServoGetPos(HIF hif, INT32 chn, SHORT * ppos)

Parameters:

hif	- open interface handle on the device
chn	- channel number
ppos	- pointer to variable to receive the position value

Returns the specified channel's current position. The position returned is the offset from the channel's defined center position. If velocity control is not enabled or the channel is not in motion, the value returned is the current target position. If velocity control is enabled and the channel is in motion, the value returned is the current actual position, not the target position.

The units of channel position are in microseconds of pulse width for the servo control pulse.

DemcServoGetMotion(HIF hif, DWORD * pdwMov)

Parameters:

hif	- open interface handle on the device
pdwMov	- variable to receive channel motion flags

Returns the current movement state of the servo channels. The return value is a bit mask that will have a bit set for each servo channel that is currently in motion. A servo channel is considered to be in motion if it has a non-zero channel velocity, has been given a new target position via the DemcServoSetPosAbs or DemcServoSetPosRel functions, and the servo has not yet reached the target position.

DemcServoSetVel(HIF hif, INT32 chn, USHORT vel)

Parameters:

hif	- open interface handle on the device
chn	- channel number
vel	- velocity to set

Sets the rotational velocity for the specified channel. Channel velocity can be used to control the rate at which a servo will rotate when given a new target position. This affects the DemcServoSetPosAbs and DemcServoSetPosRel functions. It has no effect on the DemcServoSetWidth function.

If the enabled port doesn't support velocity control, this function has no effect. If a channel velocity has been previously set, velocity control can be disabled by setting the velocity to 0.

The velocity is specified as an unsigned, fixed-point, fractional integer value, with the upper byte representing how many units the position will change per tick, and the lower byte representing a fraction of how many units to change per tick (a value of 1 in the lower byte corresponds to a change of 1/256 = 0.0039 units per tick). The tick rate is 10ms per tick.

The units associated with the velocity value are microseconds of pulse width per tick. The pulse width will be incremented or decremented by this value each 10ms tick. For example: a velocity of 256 will cause the pulse width to be adjusted by 256/256 (= 1) microsecond per tick. In 100 ticks (or 1 second) the pulse width will change by 100 microseconds. The default range of pulse width is 500usec to 1500usec. This velocity value will cause the servo to take 10 seconds to sweep from the minimum position value to the maximum position value.

To determine what value to pass to the function, multiply the desired velocity (microseconds per tick) by 256. For example, if you want a velocity of 4.73 microseconds per tick, you multiply this by 256 to yield 1211 (rounded up). So, to set a velocity of approximately 4.73, you would pass 1211 (or 0x04BB in hex) to this command.

To convert back, divide the fixed-point value by 256. For example, if the value returned by DemcServoGetVel is 7327 (or 0x1C9F in hex), you divide this by 256 to yield an actual velocity of 28.621 microseconds per tick.

Note that although the velocity is unsigned, it is not allowed to exceed a maximum of 32767 (0x7FFF hex), because the velocity becomes a signed quantity internally.

DemcServoGetVel(HIF hif, INT32 chn, USHORT * pvel)

Parameters:

hif	- open interface handle on the device
chn	- channel number
pvel	- pointer to variable to receive velocity value

Returns the current velocity setting for the specified channel. If the enabled port doesn't support velocity control the returned value will be 0.

DemcServoSetWidth(HIF hif, INT32 chn, USHORT tusWdt)

Parameters:

hif	- open interface handle on the device
chn	- channel number
tusWdt	- channel pulse width to set

Sets the width of the servo control pulses for the specified channel. The channel pulse width is specified in microseconds. The value will be clamped to the channel's current minimum and maximum values. This function ignores relative positioning and velocity control, if currently enabled, and sets the servo pulse width directly to the specified value.

DemcServoGetWidth(HIF hif, INT32 chn, USHORT * ptusWdt)

Parameters:

hif	- open interface handle on the device
chn	- channel number
ptusWdt	- variable to receive channel pulse width

Returns the actual pulse width for the specified channel. The value returned is the current channel pulse width in microseconds.

DemcServoSetCenter(HIF hif, INT32 chn, USHORT tusCtr)

Parameters:

hif	- open interface handle on the device
chn	- channel number
tusCtr	- pulse width in microseconds of channel center.

Set the pulse width for the center position for the specified channel. This sets the pulse width that is used as the origin of rotation (i.e. the '0' position) for the servo position coordinate system. It doesn't necessarily correspond to the servo's actual center of rotation or the mid-point between the minimum and maximum position values. The origin can't be set lower than the channel's current minimum pulse width or higher than the channel's current maximum pulse width. If the origin pulse width specified is lower than the current minimum or greater than the current maximum, it will be clamped to the minimum or maximum as appropriate. The default initial value is 1000 microseconds.

DemcServoGetCenter(HIF hif, INT32 chn, USHORT * ptusCtr)

Parameters:

hif	- open interface handle on the device
chn	- channel number
ptusCtr	- variable to receive pulse width of channel center

Return the current pulse width for the center position of the specified channel.

DemcServoSetMin(HIF hif, INT32 chn, USHORT tusMin)

Parameters:

hif	- open interface handle on the device
chn	- channel number
tusMin	- minimum channel pulse width in microseconds

Set the minimum pulse width for the specified channel. This sets the limit of rotation for one direction of motion of the servo. It is not allowed to be set to a higher value than the channel's current maximum pulse width. If the pulse width specified is greater than the current maximum for the channel, it will be set equal to the channel's current maximum value. If it is set higher than the channel's current center pulse width, the center pulse width will be set equal to the new minimum. The default initial value is 500 microseconds.

The practical minimum value that can be used will vary from one model/manufacturer's servo to another. Setting this value too low can result in damage to the servo if it allows the servo position to be set to a value lower than the servo is physically capable of responding to. If this occurs, the servo will continually attempt to drive the servo motor through the mechanical stop at its limit of rotation.

DemcServoGetMin(HIF hif, INT32 chn, USHORT * ptusMin)

Parameters:

hif	- open interface handle on the device
chn	- channel number
ptusMin	- variable to receive minimum channel pulse width

Return the current setting for the channel minimum pulse width.

DemcServoSetMax(HIF hif, INT32 chn, USHORT tusMax)

Parameters:

hif	- open interface handle on the device
chn	- channel number
tusMax	- maximum channel pulse width in microseconds

Set the maximum pulse width for the specified channel. This sets the limit of rotation for one direction of motion of the servo. The maximum pulse width can't be set to a lower value than that channel's current minimum pulse width. If the pulse width specified is less than the current minimum for the channel, it will be set equal to the channel's current minimum value. If the value set is lower than the channel's current center pulse width, the center pulse width will be set to the new maximum. The default initial value is 1,500 microseconds.

The practical maximum value that can be used will vary from one model/manufacturer's servo to another. Setting this value too high can result in damage to the servo if it allows the servo position to be set to a value higher than the servo is physically capable of responding to. If this occurs, the servo will continually attempt to drive the servo motor through the mechanical stop at its limit of rotation.

DemcServoGetMax(HIF hif, INT32 chn, USHORT * ptusMax)

Parameters:

hif	- open interface handle on the device
chn	- channel number
ptusMax	- variable to receive maximum channel pulse width

Return the current setting for the channel maximum pulse width.

DemcServoSetInvert(HIF hif, INT32 chn, BOOL fInvert)

Parameters:

hif	- open interface handle on the device
chn	- channel number
fInvert	- TRUE or FALSE to enable/disable channel inversion

Set or clear the channel inversion flag to enable or disable channel inversion for the specified channel. When inversion is enabled on a channel, the direction of rotation will be reversed when using the DemcServoSetPosAbs and DemcServoSetPosRel functions. There is no standard for the direction of rotation of a servo with respect to increasing or decreasing pulse width. Some manufacturer's servos rotate clockwise with increasing servo pulse width and some rotate counter-clockwise. When using a mixture of servos from different manufacturers, this function can be used to normalize the direction of rotation of the servos with respect to the position values sent. This function has no effect unless the port supports the *dprpEmcInvert* property.

DemcServoGetInvert(HIF hif, INT32 chn, BOOL * pfInvert)

Parameters:

hif	- open interface handle on the device
chn	- channel number
pfInvert	- variable to receive channel inversion setting

Returns the current setting of the channel inversion flag for the specified channel.

DEMC Brushed DC Motor Device Class

The following are the API functions used to access DEMC ports that support controlling brushed DC motors:

DemcBrdcSetVel(HIF hif, INT32 vel)

Parameters:

hif	- open interface handle on the device
vel	- velocity to set

Sets the rotational velocity for the motor attached to the port. The velocity value is a signed value in the range –32768 to 32767. Negative velocity values give rotation in one direction, while positive values give rotation in the opposite direction. A velocity value of 32767 sets the motor to full speed; a velocity value of 0 will stop the motor rotation.

DemcBrdcGetVelSet(HIF hif, INT32 * pvel)

Parameters:

hif	- open interface handle on the device
pvel	- pointer to variable to receive motor velocity

This function returns the currently set motor velocity.

DemcBrdcGetVelCur(HIF hif, INT32 * pvel)

Parameters:

hif	- open interface handle on the device
pvel	- pointer to variable to receive motor velocity

This function returns the current actual motor velocity. This function is only supported on ports that support the *dprpEmcLoopControl* property.

DemcBrdcSetLoopEnable(HIF hif, BOOL fEnabled)

Parameters:

hif	- open interface handle on the device
fEnabled	- TRUE to enable/FALSE to disable closed loop speed control

This function is used to enable/disable closed loop feedback control for the port. This call is ignored if the port doesn't support the *dprpEmcLoopControl* property.

DemcBrdcGetLoopEnable(HIF hif, BOOL * pfEnabled)

Parameters:

hif	- open interface handle on the device
pfEnabled	- variable to receive enabled state of closed loop control

This function is used to query whether or not closed loop control is currently enabled for the port.

DemcBrdcSetEncoderPeriod(HIF hif, double tsPrdMin, double tsPrdMax)

Parameters:

hif	- open interface handle on the device
tsPeriod	- motor encoder period

This function is used to set the parameters used to convert the motor shaft encoder signal into a motor angular velocity value. The motor shaft encoder is assumed to produce a square wave signal whose period is inversely proportional to motor angular velocity. The value provided for *tsPrdMin* should be the period in seconds of the encoder output at the slowest velocity the motor will turn reliably and *tsPrdMax* is the period in seconds for the maximum motor speed.

DemcBrdcSetLoopParam(HIF hif, double kp, double ki, double kd)

Parameters:

hif	- open interface handle on the device
kp	- coefficient for proportional control term
ki	- coefficient for integral control term
kd	- coefficient for differential control term

This function is used to set the coefficients used in the PID closed loop control algorithm. The values for *kp*, *ki*, and *kd* should be in the range 0.0-1.0. This function call is ignored if the port doesn't support closed loop control.

DemcBrdcSetLoopPeriod(HIF hif, double tsPeriod)

Parameters:

hif	- open interface handle on the device
tsPeriod	- closed loop control algorithm sample period to use

This function is used to set how often the closed loop control algorithm is run. This essentially sets the sample rate of the closed loop controller. The value of *tsPeriod* is the time in seconds between executions of the closed loop control algorithm. Typical values will be in the range of 10ms to 50ms (0.010 – 0.050). See the product documentation for the specific product being used to determine the range and precision of allowed values and the default value.

DEMC Stepper Motor Device Classes

The following are the API functions used to access DEMC ports that support uni-polar or bi-polar stepper motors:

DemcStepSetRate(HIF hif, double stvReq, double * pstvSet)

Parameters:

hif	- open interface handle on the device
stvReq	- requested step rate
pstvSet	- variable to receive step rate obtained

This function is used to set the maximum angular velocity (step rate) to use when stepping the motor. The step rate is specified in units of steps per second. If the port supports sub-stepping and sub-stepping is in effect for the port, the velocity specifies the number of sub-steps per second.

If the port supports acceleration and deceleration, the specified step rate will be the maximum step rate to which the motor will accelerate. If the port doesn't support acceleration and deceleration, this is the step rate that will be used for all motor shaft movement.

DemcStepGetRate(HIF hif, double * pstvCur)

Parameters:

hif	- open interface handle on the device
pstvCur	- variable to receive current step rate

This function returns the current angular velocity (step rate) for the port. The return value is in steps per second.

DemcStepSetSubstep(HIF hif, DWORD sbstReq, DWORD * psbstSet)

Parameters:

hif	- open interface handle on the device
sbstReq	- requested number of step subdivisions
psbstSet	- variable to receive number of step subdivisions obtained

This function is used to set the number of step subdivisions to use when stepping the motor. This function is ignored if the port doesn't support the *dprpEmcStepSubstep* property.

The *sbstReq* value specifies the number of sub-steps into which to divide the motor's inherent step size. This must be a positive integer, power-of-2, between 1 and 128. The actual values supported depend on the specific device port.

Sub-stepping (also called micro-stepping) allows the motor to be stepped to intermediate positions between the natural step positions of the motor. This involves partially energizing adjacent coils in the motor. The most common form of sub-stepping is called half-stepping. Half stepping is obtained by specifying 2 for *sbstReq*.

DemcStepGetSubstep(HIF hif, DWORD * psbstCur)

Parameters:

hif	- open interface handle on the device
psbstCur	- variable to receive current number of step subdivisions

This function returns the current number of subdivisions being used to step the motor. The default value is 1.

DemcStepSetAccel(HIF hif, double staReq, double * pstaSet)

Parameters:

hif	- open interface handle on the device
staReq	- requested acceleration rate
pstaSet	- variable to receive acceleration rate obtained

This function is used to set the angular acceleration (rate of change of step rate) to use when stepping the motor. It is only meaningful to call this function is the port supports acceleration, which is indicated by the *dprpEmcAccel* bit being set in the port properties. The specified acceleration will also be used to decelerate the motor as well unless the ports supports a separate deceleration value.

DemcStepGetAccel(HIF hif, double * pstaCur)

Parameters:

hif	- open interface handle on the device
pstaCur	- variable to receive current acceleration rate

This function returns the current angular acceleration (rate of change of step rate) in use when stepping the motor.

DemcStepSetDecel(HIF hif, double staReq, double * pstaSet)

Parameters:

hif	- open interface handle on the device
staReq	- requested deceleration rate
pstaSet	- variable to receive acceleration rate obtained

This function is used to set the angular deceleration (rate of change of step rate) to use when stepping the motor. This value is used to decelerate the motor shaft when approaching the final position. It is only meaningful to call this function for ports that support a separate deceleration value that is different than the acceleration value. This is indicated by the *dprpEmcDecel* property bit being set.

DemcStepGetDecel(HIF hif, double * pstaCur)

Parameters:

hif	- open interface handle on the device
pstaCur	- variable to receive current deceleration rate

This function returns the current angular deceleration (rate of change of step rate) in use when stepping the motor. If the port doesn't support a separate deceleration value (indicated by the *dprpEmcDecel* bit being set in the port properties), this will be the same value as the acceleration value.

DemcStepMoveAbs(HIF hif, INT32 stpReq)

Parameters:

hif	- open interface handle on the device
stpReq	- requested new position value

This function is used to move the stepper motor shaft to a new position. The value passed for *stpReq* indicates the new angular position for the motor shaft (in units of steps or sub-steps if sub-stepping is in effect). The motor shaft will be moved using the current velocity and acceleration values to the indicated position.

The range of allowed values for *stpReq* is the full range of a signed 32-bit integer value. However, the range is constrained to fit within the limits specified by the DemcStepSetLimits function.

DemcStepMoveRel(HIF hif, INT32 stpReq)

Parameters:

hif	- open interface handle on the device
stpReq	- requested new position value

This function is used to move the stepper motor shaft to a new position. The value passed for *stpReq* indicates the delta to the current position of the motor shaft (in units of steps, or sub-steps if sub-stepping is in effect). The motor shaft will be moved using the current velocity and acceleration values by the indicated amount.

The range of allowed values for *stpReq* is the full range of a signed 32-bit integer value. However, the range is constrained to fit within the limits specified by the DemcStepSetLimits function.

DemcStepSetPos(HIF hif, INT32 stpReq)

Parameters:

hif	- open interface handle on the device
stpReq	- requested new position value

This function is used to set the position value that corresponds to the current motor shaft position. The current position is set to 0 automatically when the port is enabled. The current position is automatically updated whenever the motor is stepped by calling either the *DemcStepMoveAbs* or *DemcStepMovRel* functions.

DemcStepGetPos(HIF hif, INT32 * pstpCur)

Parameters:

hif	- open interface handle on the device
pstpCur	- variable to receive current position value

This function returns the current position of the stepper motor shaft.

DemcStepSetLimits(HIF hif, INT32 stpMax, INT32 stpMin)

Parameters:

hif	- open interface handle on the device
stpMax	- maximum position value allowed
stpMin	- minimum position value allowed

This function is used to set the limits on the allowed positions for the motor shaft.

Requests to move the motor position to a value outside the specified limits will result in the motor stopping at the limit value. Ensure that the current motor position is within the limits being set when calling this function. If the current motor position is outside the specified limits, the result is undefined.

The limit position values are specified in units of steps, or sub-steps if sub-stepping is in effect for the port. Note that if sub-stepping is enabled after the limits have been set, the limits will not be recomputed and may be incorrect. If sub-stepping is being used, the sub-step value should be set before setting the limit values.

The default values when the port is initialized are +/- 2,147,483,647 (i.e. the symmetrical range of values allowed for a signed 32 bit integer).

DemcStepGetLimits(HIF hif, INT32 * pstpMaxCur, INT32 pstpMinCur)

Parameters:

hif	- open interface handle on the device
pstpMax	- variable to receive max position limit value
pstpMin	- variable to receive min position limit value

This function returns the current limit values for motor position.

DemcStepGetMotion(HIF hif, BOOL * pfMov)

Parameters:

hif	- open interface handle on the device
pfMov	- variable to receive channel motion flag

This function returns the current movement state of the port. This will return TRUE if the motor shaft is currently in motion, and FALSE if not.

DemcStepHalt(HIF hif)

Parameters:

hif	- open interface handle on the device

Halt motion on the stepper motor. This causes the controller to immediately stop stepping the motor. No deceleration will be done (if the port supports deceleration).