

IDCA-10

Intelligent Brushed DC Motor Drive

Motor Control Technologies; LLC
www.mocontech.com

Motor Control Technologies, LLC (MCT) designs and manufactures motor drives used to power and control DC motors or similar resistive or inductive loads. MCT products include various inherent safety mechanisms. However, MCT products are not designed as fail-safe components and are not for use in critical equipment or life-support systems. Any use of MCT products in such applications is done solely at the risk of the user.

MCT products are not to be used in any and all activities related to the following fields: medical, military, aviation, aerospace, or government. Those using MCT products in the above-described fields do so at their own risk and agree to hold MCT harmless with respect to any possible bodily injury or property damage that may occur.

Customers are responsible for their applications when using MCT products. In order to decrease the likelihood of application and/or product failure, customers should supply proper design, automation, and operation safeguards. MCT shall not be liable for any personal injury or property loss that results from misuse, abuse, misapplication, or misconnection by the customer or damage that is attributable to acts of God.

The information contained within this manual is for informational purposes only and cannot be reproduced, in any form, without the written permission of Motor Control Technologies, LLC.

Table of Contents

Table of Contents	3
List of Figures	3
List of Tables	4
1. Overview	5
1.1. IDCA-10 Product Description	5
1.2. IDCA-10 Software and Firmware	7
2. IDCA-10 Top Plate	8
2.1. Pin Descriptions	8
2.2. Status Indicators	10
2.3. Electrical Connections.....	10
3. Operating Modes	16
3.1. Idle (default)	16
3.2. Open-loop Control.....	16
3.3. Closed-loop Speed Control	17
3.4. Closed-loop Position Control.....	17
3.5. Stand-alone Mode.....	21
4. Serial Communications	23
4.1. SPI Communications.....	23
4.2. SMBus (I ² C) Communications.....	23
4.3. Configuring Serial Bus Protocol	24
4.4. Data Transfer	25
4.5. Instruction Packet Definitions	28
4.6. MCT Op-codes Definitions	43
5. IDCA-10 Memory Registers	48
5.1. Read Only Registers	48
5.2. Read/Write Registers	49
Appendix A – PID Controller Basics	56
Appendix B – IDCA-10 Error Code Definitions	58
Appendix C – Electrical Characteristics	59
Appendix D - Mechanical Drawings	62
Revision History	63

List of Figures

Figure 1. IDCA-10 block diagram.....	5
Figure 2. PC-based system operation.....	6
Figure 3. Microcontroller system operation.....	6
Figure 4. Stand alone operation.....	6
Figure 5. Screen shot for the MCTUI.....	7
Figure 6. IDCA-10 screw terminals.....	8

Figure 7. Typical electrical connections for open-loop control.	11
Figure 8. Typical electrical connections for speed and position control.	12
Figure 9. Typical connection to the analog input line.	13
Figure 10. Typical limit switch implementation.	13
Figure 11. Open loop block diagram.	16
Figure 12. Closed-loop speed control with the IDCA-10.	17
Figure 13. Closed-loop position control block diagram.	18
Figure 14. Motor speed profile in control example 1.	19
Figure 15. Motor speed profile for two consecutive position packets	20
Figure 16. SPI bus block diagram.	23
Figure 17. SMBus/I2C block diagrams.	24
Figure 18. Command data packet structure.	26
Figure 19. Return data packet structure.	27
Figure 20. PID control block diagram.	56

List of Tables

Table 1. Data line definitions for pins X0 - X3.	9
Table 2. PWR indicator communication flash sequence	10
Table 3. PWR indicator configuration flash sequence.	10
Table 4. Analog input voltage ranges.	21
Table 5. Example CRC values.	28
Table 6. IDCA-10 hardware error codes.	53

1. Overview

1.1. IDCA-10 Product Description

The IDCA-10 is a fully integrated DC servo system comprised of communication, control logic, and power amplification systems. Figure 1 depicts a block diagram for the IDCA-10. The IDCA-10 possesses a central microcontroller dedicated to communication and control algorithms. The IDCA-10 uses 4-wire Serial Peripheral Interface (SPI) or 2-wire SMBus (I²C) serial bus to communicate with a master device. Other control inputs include analog input (stand-alone mode) dual encoder inputs and dual limit switch inputs. The IDCA-10 accepts power from a 9-28 VDC power supply. It also offers a +5 VDC regulated output for powering external components. An integrated H-bridge supplies Pulse Width Modulated (PWM) output to drive a DC brushed motor.

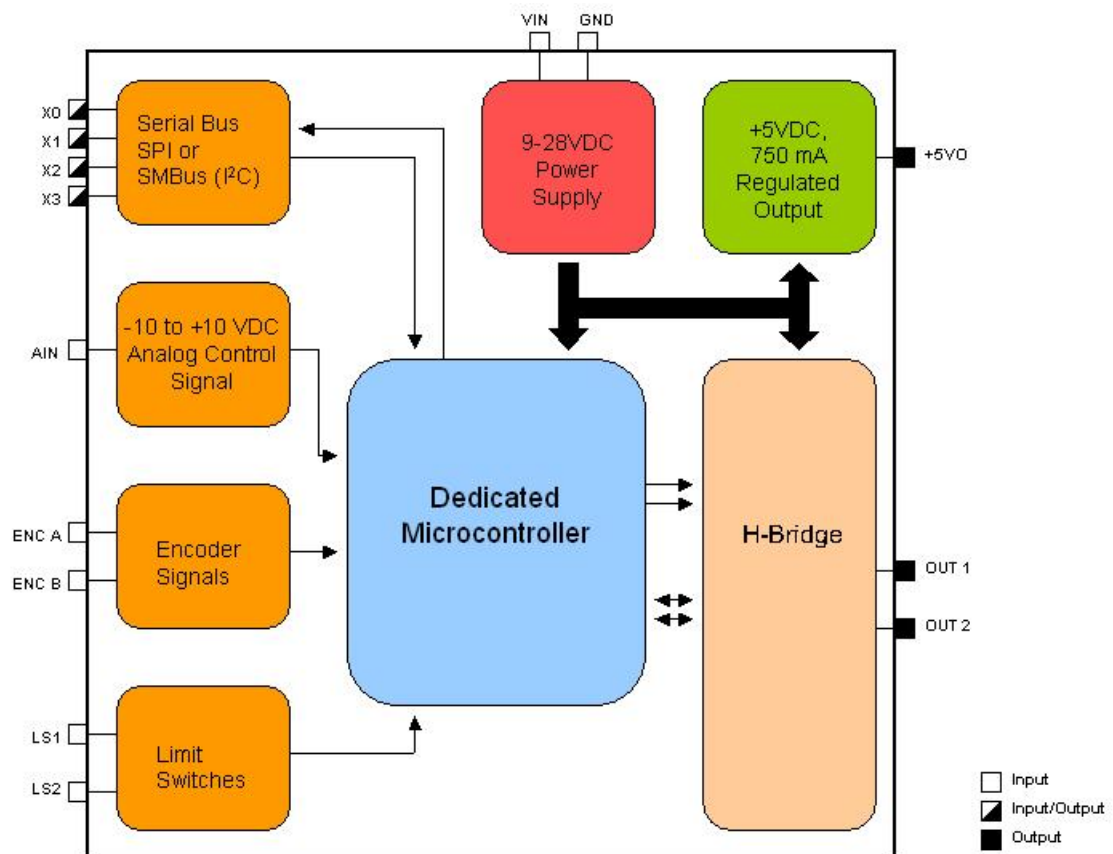


Figure 1. IDCA-10 block diagram.

The IDCA-10 is flexible enough to be used in a variety of applications and configurations. Figure 2 depicts the IDCA-10 being used with a PC-based

data acquisition system. In this scenario, an application running the PC uses the data acquisition hardware to relay data between the PC and the IDCA-10.

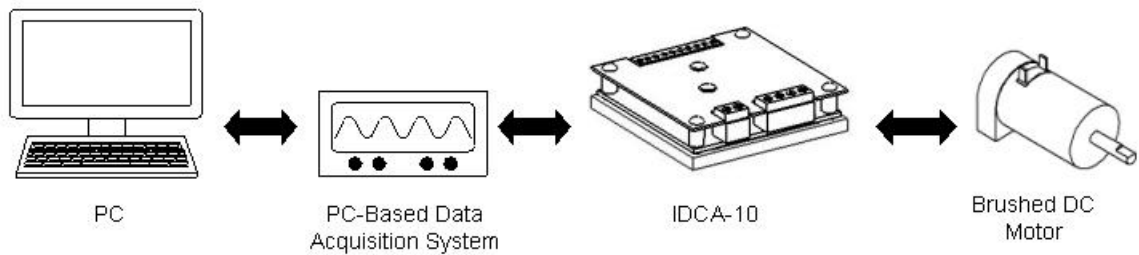


Figure 2. PC-based system operation.

Figure 3 depicts one scenario where the IDCA-10 communicates directly with a separate microcontroller. Data exchanged occurs directly between the microcontroller and the IDCA-10 on the serial bus. This is a common scenario where a micro controller system requires amplification and control hardware to drive a DC motor.

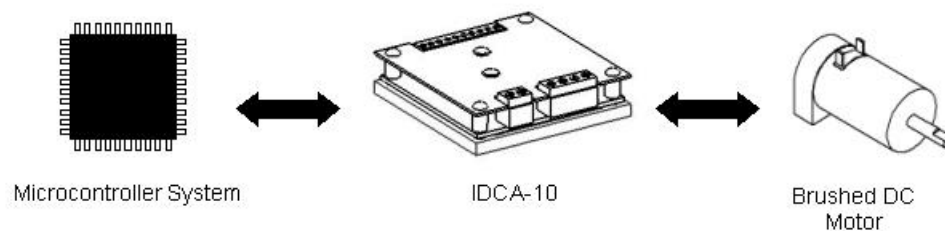


Figure 3. Microcontroller system operation.

Figure 4 illustrates the IDCA-10 being used in stand-alone mode. This unique feature of the IDCA-10 allows the user to configure the controller for a particular application, and control a motor with an analog control signal. Stand-alone mode is particularly useful in applications where a micro controller or PC systems are not practical.

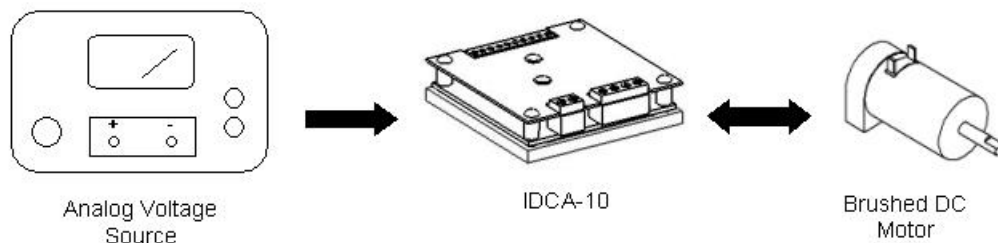


Figure 4. Stand alone operation.

1.2. IDCA-10 Software and Firmware

1.2.1. Software

MCT User Interface

The MCT User Interface (MCTUI) is a windows-based application used to interface to the IDCA-10 using a select group of data acquisition equipment. The MCTUI supports the following data acquisition hardware manufactured by [LabJack Corporation](#):

- U12
- U3 LV and U3 HV
- U6 and U6 Pro
- UE9 and UE9 Pro

The MCTUI provides all the tools necessary to properly tune and control an IDCA-10 from a PC. Please refer to the MCTUI documentation for further information.

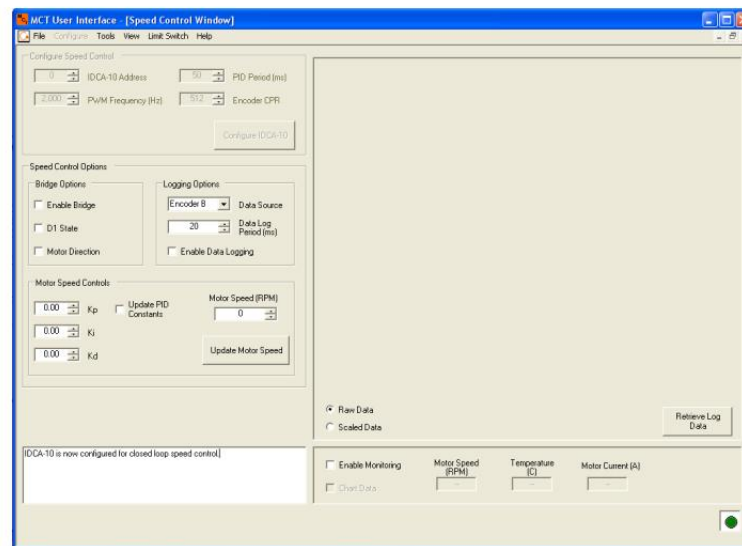


Figure 5. Screen shot for the MCTUI.

MCT UI DLL

TBD

1.2.2. Firmware

The IDCA-10 firmware is responsible for controlling the basic functionality and motor control operations. Firmware may be updated periodically by simply downloading the latest firmware version from the [MCT](#) website and using the firmware upgrade utility to reprogram the microcontroller. The firmware update utility is provided in the MCTUI. See the MCTUI documentation for further information.

2. IDCA-10 Top Plate

Figure 6 depicts the nameplate on the IDCA-10. All IO and power connections necessary for operation are made through the screw terminals located at the sides of the unit. All screw terminals are clearly marked for easy connection. The following section describes each screw terminals in detail.

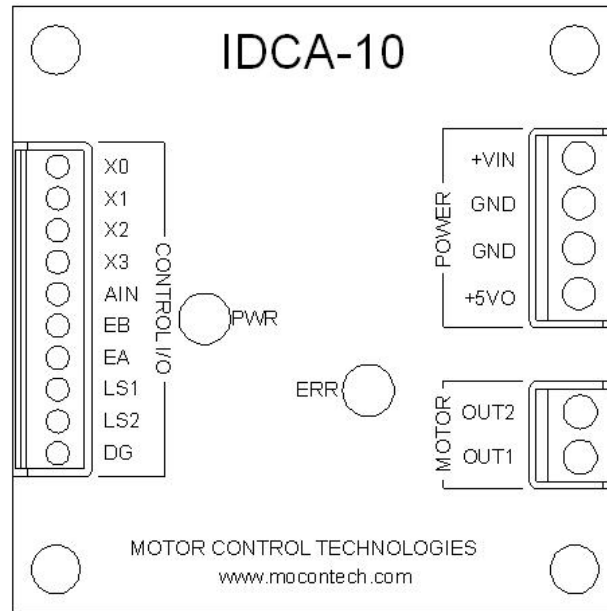


Figure 6. IDCA-10 screw terminals.

2.1. Pin Descriptions

2.1.1. Power

+VIN

Power input for the IDCA-10. Connect to the positive terminal on a 9-28 VDC power supply.

GND

Ground pin for the IDCA-10. Connect to the negative (GND) terminal on a 9-28 VDC power supply or chassis ground. The GND pins provide the primary ground to the IDCA-10 and motor. Always ensure that a good ground connection exists and proper grounding techniques are used. Always connect the GND directly to ground with as short a lead as possible. A 14-gauge ground lead is recommended for all applications.

+5VO

The +5VO pin is a regulated +5VDC output that can source up to 250 mA continuous and up to 500 mA intermittent (< 1 minute). The +5V pin may be

used to power external devices such as encoders, micro controllers, or other electronic hardware.

2.1.2. Motor

OUT1

Motor output terminal. Connect to power terminal on DC brushed motor.

OUT2

Motor output terminal. Connect to power terminal on DC brushed motor.

2.1.3. Control IO

X0 – X3

Pins X0 – X3 are used for transmitting data between a master device and the IDCA-10 (slave). The IDCA-10 uses either 4-wire SPI or 2-wire SMBus (I2C) protocols for communication. Table 1 lists how the serial lines are mapped to the X0 – X3 pins.

Table 1. Data line definitions for pins X0 - X3.

Pin	SPI Line	SMBus (I2C) Line
X0	MISO	SDA
X1	SCK	SCL
X2	NSS	Not used
X3	MOSI	Not used

2.1.4. AIN

AIN is the analog input line used in stand-alone mode to control the motor. AIN is only used in stand-alone mode.

EA and EB

Encoder input.

LS1 and LS1

Limit switch inputs.

DGND

DGND is the digital ground for use when interfacing with other electronic equipment. DGND is internally connected to GND through a fuse to protect connected devices from large ground currents. Always use the DGND as a ground reference when interfacing with other electronic equipment.

2.2. Status Indicators

2.2.1. PWR Indicator

The PWR indicator is a green LED that will turn on solid when power is applied to the unit. The PWR indicator also reports serial bus and mode configurations. Hardware configuration is reported by series of flashes at power-up or on a system reset. The flash sequence is divided into two short sequences separated by a 1 second pause (LED off). The first flash sequence indicates the serial bus being used. Two flashes indicate the SPI bus is configured. Three flashes indicate the SMBus/I²C serial bus is configured (Table 2). The second flash sequence indicates the IDCA-10 operating mode. Flash sequences can range from two to eight flashes, depending on the configured mode. Table 3 lists the flash count and associated operating modes.

Table 2. PWR indicator communication flash sequence

Flash Count	Operating Mode
2	SPI
3	SMBus/I ² C

Table 3. PWR indicator configuration flash sequence.

Flash Count	Operating Mode
2	Idle (default)
3	Open-loop
4	Speed control
5	Position control
6	SA, Open-loop
7	SA, Speed Control
8	SA, Position Control

Note:

SA = Stand-alone mode

2.2.2. ERR Indicator

The ERR indicator is used as a visual indicator to inform the user an error was encountered by hardware. Error codes defining encountered problems are stored in the ErrorReg memory register. See section 5.2 for error code definitions.

2.3. Electrical Connections

Figure 8 depicts typical wiring connections when operating a motor in open-loop mode. Encoder signals are not required in open-loop mode, but may be used to monitor motor speed. Figure 8 depicts typical wiring connections when interfacing with a motor and an integrated shaft encoder. Encoder feedback signals are required when operating in speed and position control modes.

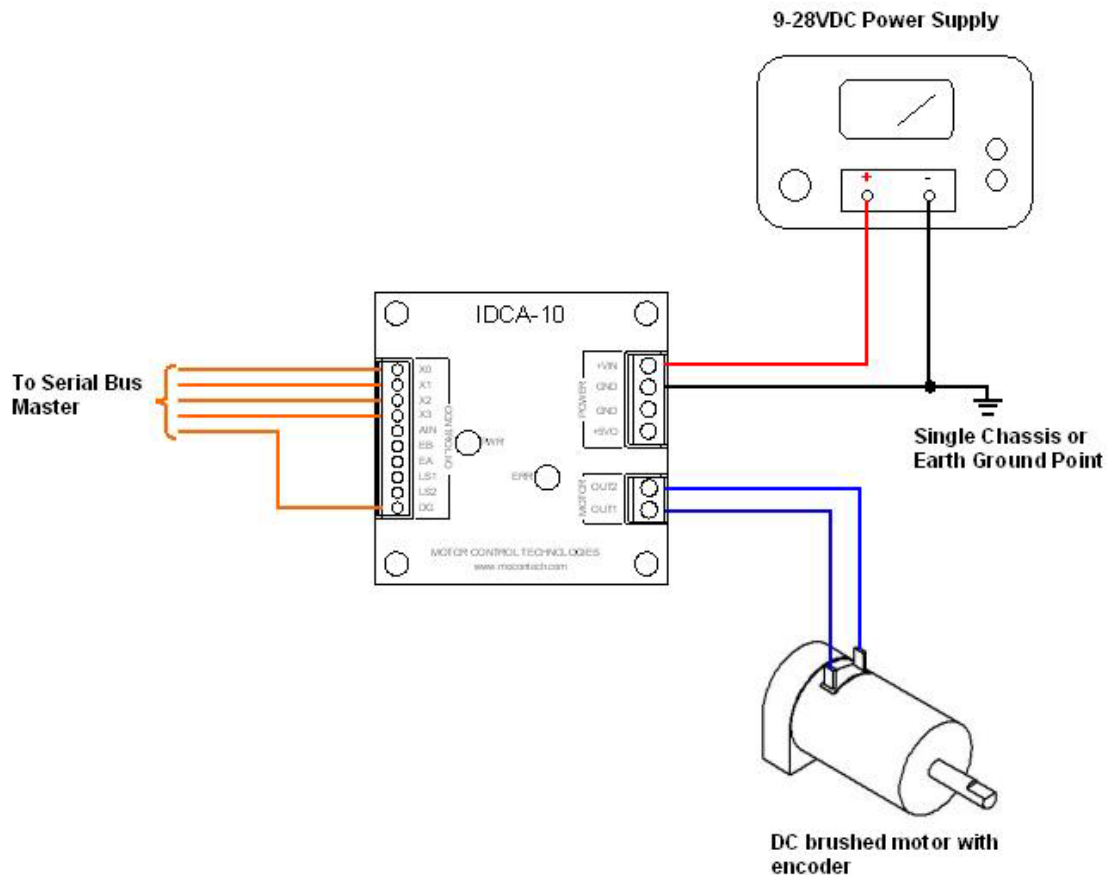


Figure 7. Typical electrical connections for open-loop control.

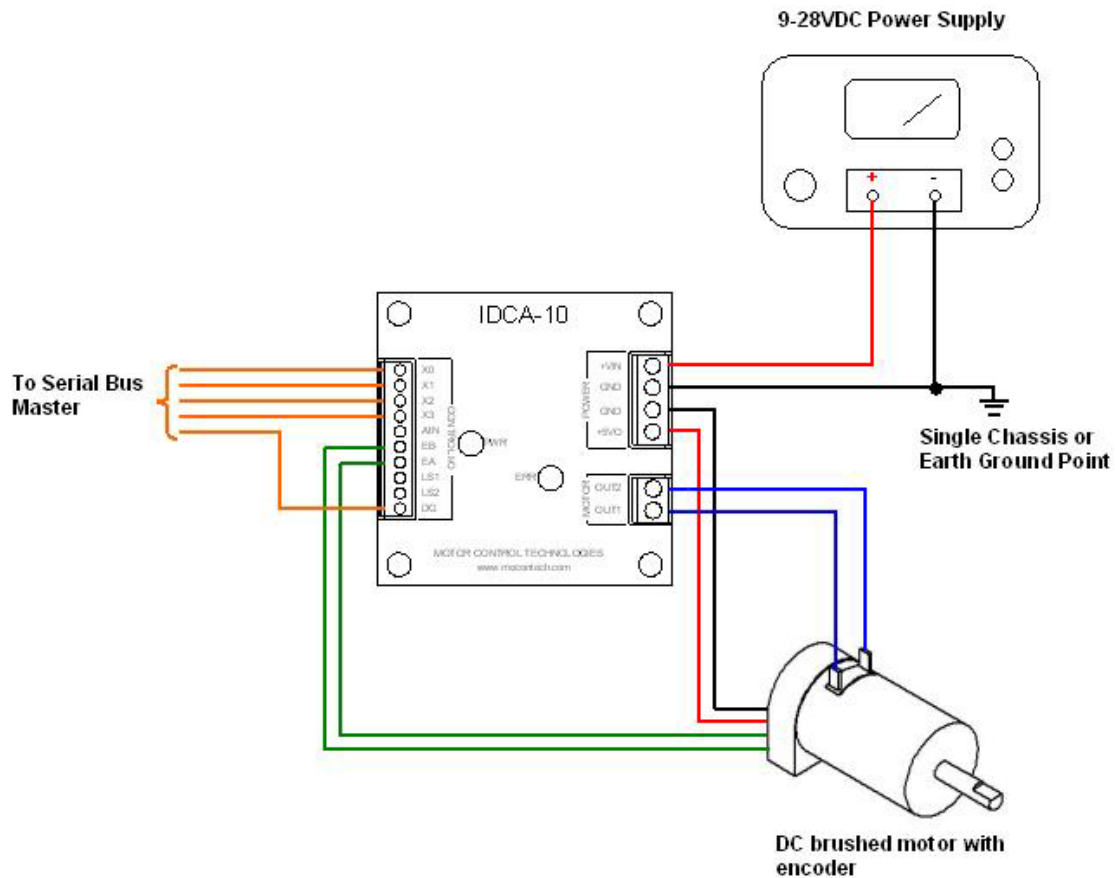


Figure 8. Typical electrical connections for speed and position control.

2.3.1. Analog Input (AIN)

The AIN pin is used for an analog control signal when operating in stand-alone mode. Stand-alone mode is special operating mode where command signals are interpreted from the analog input rather than command packets received via the serial bus. See section 3.5 for details about the AIN line. Figure 9 depicts a typical connection to the AIN line.

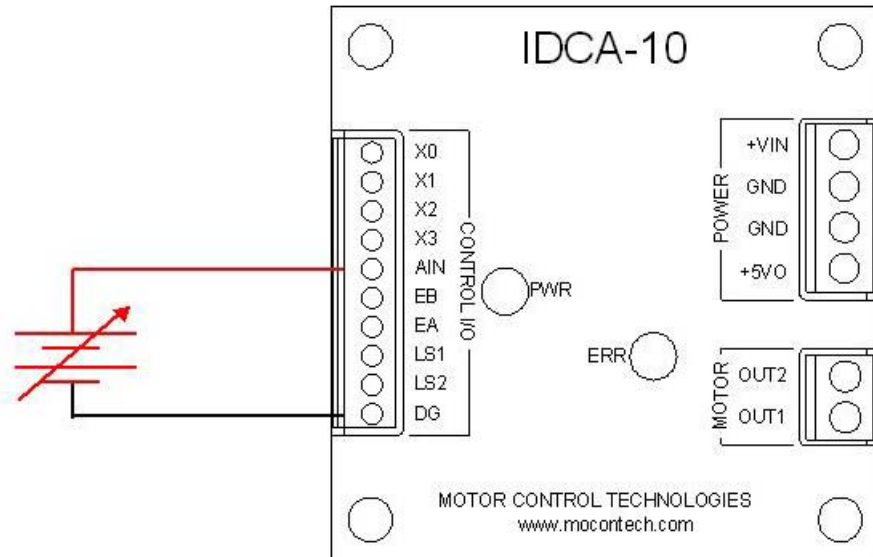


Figure 9. Typical connection to the analog input line.

2.3.2. Limit Switch (LS1 and LS2)

The limit switch inputs (LS1 and LS2) provide an interface for up to two limit switches. Limit switches are useful for static motor stops or for implementing hardware-controlled deceleration when the motor is at a particular location. Figure 10 depicts a typical limit switch implementation on the IDCA-10.

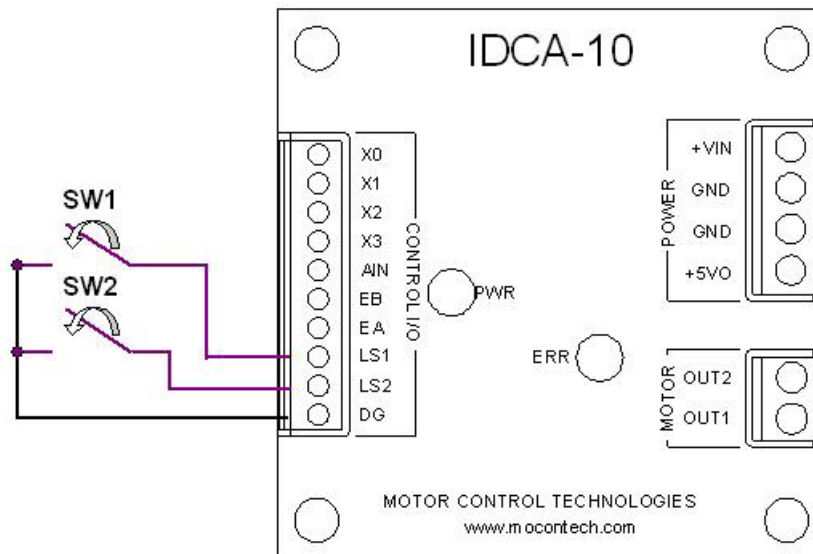


Figure 10. Typical limit switch implementation.

Limit Switch Configuration

LS1 and LS2 are disabled by default and must be configured before they are available for use. Both limit switch inputs are configured using the LIMSTAT, LS1Rate, and LS2Rate registers. All three registers are ignored by hardware until the either limit switches are enabled in the LIMSTAT register. Limit switch configuration allows either switch or both to be enabled at any time by setting the appropriate bits in the LIMSTAT register. LS1 can also be configured as a latch for the AIN line for stand-alone mode. See section 5.2 for further details about the LIMSTAT memory register.

LS1Rate and LS2Rate registers must be configured for the respective limit switch if enabled. The LSxRate register stores the deceleration rate the controller will use to stop the motor when a limit switch event occurs. Valid deceleration rates are 0-255. The LS1Rate register is ignored if LS1 is configured as a latch for the AIN line.

Limit Switch Functionality

LS1 and LS2 provide hardware controlled motor deceleration when a limit switch event is detected. A limit switch event occurs when an enabled limit switch line transitions from logic HIGH logic LOW. Internal pull-up resistors hold the limit switch lines high if nothing is connected. When using limit switches, connect the LSx line through a switch to ground so the line is pulled low when the switch is activated.

Limit switch lines are polled by firmware once every 50ms to detect a HIGH to LOW transition. Hardware will begin decelerating the motor as soon as the limit switch event is detected, regardless of what function is being performed. The controller will decelerate the motor at a rate as defined appropriate LSxRate register until it comes to a stop.

The value stored in LSxRate defines the amount the controller will decrease the motor speed until the motor comes to a rest. Motor speed is updated every 50 ms until the motor stops, and cannot be interrupted once the process is started.

The deceleration rate defined in LSxRate is implemented differently in open-loop control than in closed-loop control. In open-loop mode the LSxRate represents a change in PWM duty cycle. In closed-loop control the LSxRate represents a change in the motor command speed, represented in RPM. Therefore, the same LSxRate value will behave different depending on the mode. For example, an LSxRate value equal to 10 will result in a deceleration of 10% every 50ms in open-loop, but will result in a deceleration of 10 RPM every 50 ms in closed-loop control. This subtle difference can lead to drastically different deceleration rates. The user must adjust accordingly in different operating modes.

Motor Stop and Limit Switch Reset

Once the controller brings the motor to rest it will post a limit switch message to the Error queue and begin flashing the Error LED. Determining which limit switch was activated is determined by reading the value posted in the ERROR register. A value equal to 236 (0xEC) indicates LS1 was activated and a value equal to 237 (0xED) indicates LS2 was activated.

The end-behavior of a limit switch event mimics the same behavior as a level-3 error generated in hardware. That is, the hardware will disable all motor control and place the motor into a safe state. Communication is not interrupted when the motor is placed into safe state.

The limit switch must be manually reset once a limit switch event is detected and the motor is stopped. The following procedure outlines how to reset the limit switch lines:

1. Verify the error LED is blinking, indicating limit switch activation.
2. Read the value stored in the error queue to determine which limit switch was activated.
3. Return the activated limit switch line to logic HIGH by opening the limit switch or allowing the LSx line to float. The limit switch line must be cleared to prevent the controller from immediately processing the limit switch again after resetting.
4. Pull the opposite limit switch to logic LOW and hold at that state for approximately five seconds and then return the line state to logic HIGH. If LS1 is activated use LS2 to reset. If LS2 is activated use LS1 to reset. The activated limit switch is reset when error LED stops blinking and lights solid. The error LED will stay lit for two seconds and then turn off, indicating a return to normal operations.

Note:

The limit switch line used to generate a reset signal must be returned to logic HIGH while the error LED is lit solid. Failure to do so will result in the controller detecting a limit switch event on the reset signal line when it returns to normal operation.

3. Operating Modes

3.1. Idle (default)

Idle mode is the default mode the IDCA-10 will power-up in if it is not configured to start in one of the other operating modes. Idle mode is essentially a start/safe mode. The H-bridge and outputs are disabled, and communication bus is active, awaiting data packets from the master device. The IDCA-10 must be in idle mode when configuring the hardware to operate in any one of the other modes. Hardware must be reset into idle mode if the user desires to reconfigure from one mode to another, i.e. open-loop to speed control modes.

3.2. Open-loop Control

In open-loop mode the motor's speed is driven without respect to any feedback signals from the motor. In this mode the voltage across the motor's terminals and the applied load at the shaft determines the speed. Figure 11 depicts a block diagram representation of open-loop motor control with the IDCA-10.

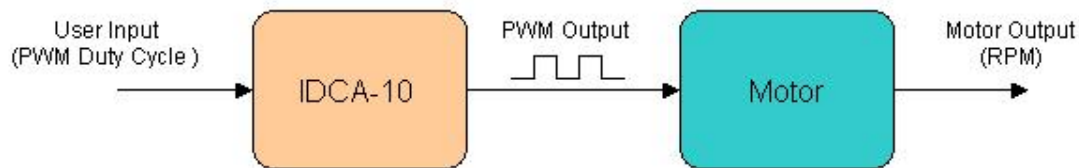


Figure 11. Open loop block diagram.

The voltage seen at the motor's terminals is controlled through Pulse Width Modulation (PWM) of the supply voltage V_{IN} . The result is scaled version of V_{IN} , where the scale factor is the duty cycle of the PWM output. For example, a 50% duty cycle will result in a terminal voltage equal to 50% of V_{IN} .

Equation 1.

$$V_{TERM} = PWM \text{ DUTY} * V_{IN}$$

Motor speed is proportional to the voltage across its terminals when operating at no-load conditions. At full voltage (V_{IN}) the motor will run at rated full speed. When the voltage is at 50% the motor will run at 50% of full speed, and so on.

Physical loading at the shaft will also determine the motor's speed at a given terminal voltage. The exact speed the motor settles at depends on the power band of the particular motor.

3.3. Closed-loop Speed Control

Closed-loop speed control expands on the open-loop concepts discussed above by incorporating a feedback signal from the motor. The feedback signal reports the motor speed to the controller where it is compared to the commanded speed. The difference between the command speed and the actual motor speed (error) is passed on to an internal PID controller. The PID controller adjusts the PWM duty cycle used to drive the motor. By closing the control loop with the encode signal, the IDCA-10 will force the motor to operate at a specified speed. See Appendix A for more details on the PID controller basic concepts. Figure 12 depicts a block diagram representation of closed-loop speed control with the IDCA-10.

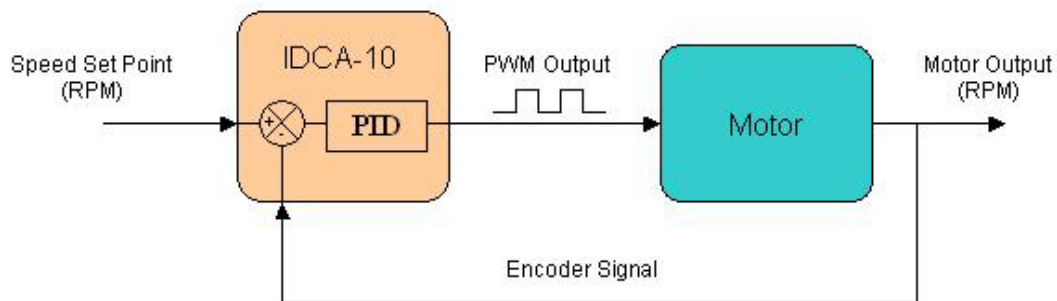


Figure 12. Closed-loop speed control with the IDCA-10.

3.4. Closed-loop Position Control

Closed loop position control builds off the concepts covered in section 3.3 (speed control). The user should review the speed control section if they are not familiar with the concepts. The PID constants used in the closed-loop controller may also be used in the closed-loop position controller. Figure 13 depicts the control block diagram that represents the closed-loop position controller implemented on the IDCA-10. As shown in the figure, closed-loop position requires three inputs:

1. Position
2. Max Speed
3. Approach Speed

The value sent to the *Position* input represents the target motor position represented in encoder counts. The value used for the *Max Speed* input represents the initial speed the motor will start running at represented in RPM. The *Approach* speed is the speed the motor will run at as it approaches its target position, represented in RPM.

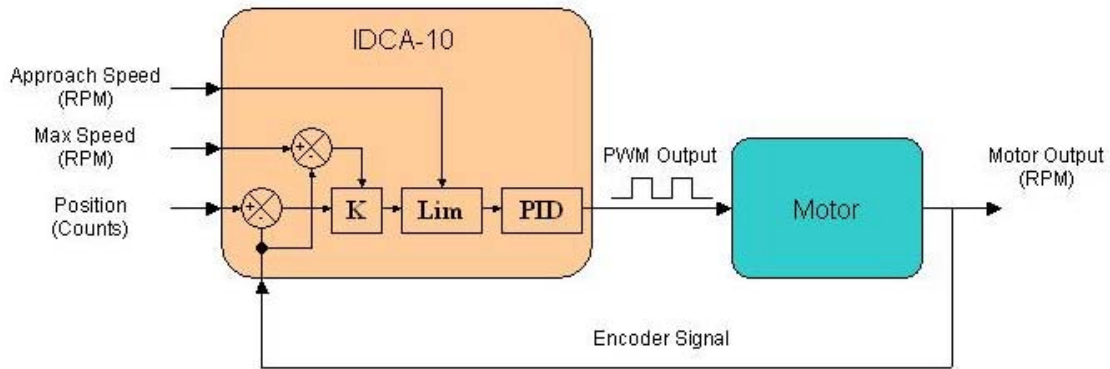


Figure 13. Closed-loop position control block diagram.

The three inputs may seem strange at first, but the use of these three inputs provides great flexibility. The following example best explains how each input affects control behavior.

Position Control Example 1

Givens: Shaft encoder resolution = 512 counts/rev
 Target Position = 10.5 Revolutions (shaft)
 Initial speed = 0 RPM
 Max Speed = 500 RPM
 Approach speed 250 RPM

The first step is to calculate the target position in terms of encoder counts. The value for the *Position* input is calculated with Equation 2:

Equation 2.

$$\text{Position} = 10.5 \text{ rev} * 512 \text{ counts/rev} = 5,376 \text{ counts}$$

The value calculated in Equation 2, Max Speed, and Approach speed are next sent to the IDCA-10 as command values. Upon execution the motor will start rotating at 500 RPM. The IDCA-10 will adjust the motor speed so the motor will be running at the commanded approach speed when it reaches its destination. The speed is adjusted proportionally to the distance traveled. For example, the motor's speed will be at the midpoint between *Max Speed* and *Approach Speed* when the motor has traveled half the commanded distance. Figure 14 depicts the motor speed profile that results from the values used in this example.

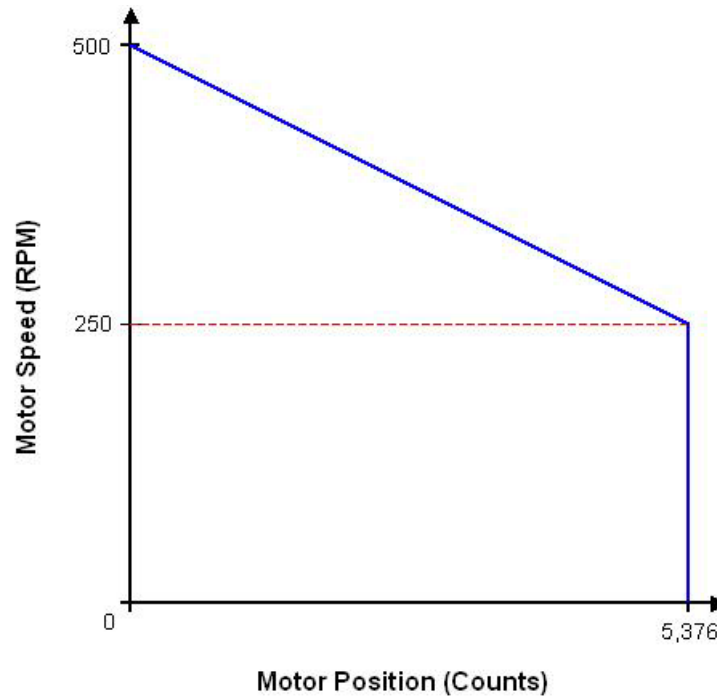


Figure 14. Motor speed profile in control example 1.

As Figure 14 shows, the motor starts rotating at 500 RPM and decreases proportionally to the distance traveled until it reaches its target position. Once the motor reaches its target the motor will stop and wait to execute another position command.

□

The example discussed above illustrates how the IDCA-10 processes a single position packet. There are some subtleties not discussed in the example that are important when using this mode:

1. The approach speed does not have to be less than the max speed. Approach Speed may be set equal to Max Speed or greater than Max Speed to generate different profiles. Setting Max Speed equal to Approach speed results in a constant speed. Setting Approach Speed greater than Max speed causes the motor to ramp up in speed rather than decrease.
2. The speed profile in Figure 14 shows how the IDCA-10 processes a single position packet. The motor will be commanded to 0RPM if no other command packets are stored in the instruction buffer.

The motor speed will transition from Approach Speed to the next command's Max Speed without stopping when more than one command is stored in the instruction buffer. This provides a smooth transition between data packets and allows the user to send instructions to execute complex position sequences. Figure

15 depicts what the motor speed profile would look like when two position command packets stored in the instruction buffer are executed in sequence.

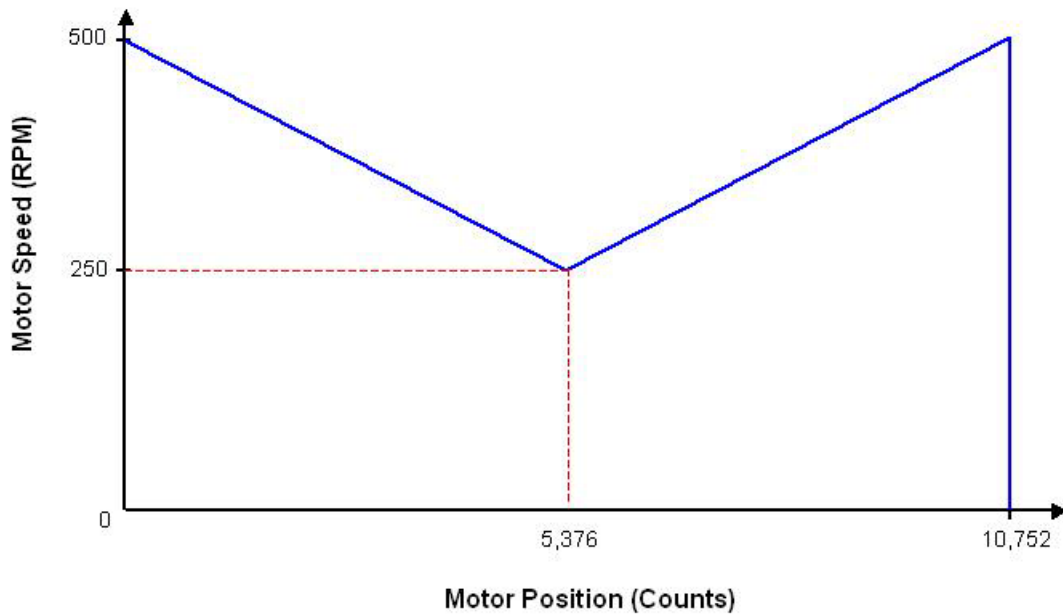


Figure 15. Motor speed profile for two consecutive position packets

3. Figure 14 and Figure 15 depict ideal speed profiles for a motor. The actual speed profiles achieved will ultimately depend on how well the PID controller is tuned for a given system. The user should spend ample time tuning their system in speed control mode before attempting to carry out position control operations.

3.4.1. Error Accumulation

Systems using dead reckoning for precise location must account for positioning error that accumulates over multiple position commands. Positioning errors will occur due to natural system latencies and system inertia. To account for these errors, the IDCA-10 has two data registers that store error that accumulates during use (ACCERR0 and ACCERR1). The ACCERR0 register accumulates the error when the motor direction is set to 0. ACCERR1 accumulates the error the motor direction is set to 1. Both error registers report the accumulated error in terms of encoder counts. ACCERR0 and ACCERR1 are located at memory addresses 0x002A and 0x002B. See section 5.1 for further information about these registers.

3.5. Stand-alone Mode

Stand-alone mode is a special operating mode that uses an analog input signal to command the motor's speed and/or position instead of command packets received on the serial bus. The use of an analog input is useful in remote applications where digital communications may not be practical. Stand-alone mode may be used with any of the three modes discussed above.

3.5.1. Open-loop – Sub Mode

Open-loop stand-alone operation uses the voltage sensed at the AIN pin to vary the PWM duty cycle to the motor. The AIN input range is mapped over the desired PWM range. See section 3.5.4 for further details on AIN configuration.

3.5.2. Speed Control – Sub Mode

Speed control stand-alone operation uses the voltage sensed at the AIN pin to vary the motor speed set point. The AIN input range is mapped over the desired speed range. See section 3.5.4 for further details on AIN configuration.

3.5.3. Position Control – Sub Mode

Position control stand-alone operation uses the voltage sensed at the AIN pin as a reference to an absolute motor position. The AIN input range is mapped to motor position such that the maximum and minimum AIN values correspond to the maximum position bounds in the respective direction. See section 3.5.4 for further details on AIN configuration.

3.5.4. Stand-alone Configuration Constants

AIN Range

The AIN Range value determines analog input range that gets mapped over the defined operating parameters. Table 4 lists the available input ranges for the AIN pin.

Table 4. Analog input voltage ranges.

AIN Mode	AIN Range
0	0 to +5 VDC
1	-5 VDC to + 5VDC
2	0 to +10 VDC
3	-10VDC to +10VDC

The midpoint of the AIN Range sets the zero-point for motor operations, and is used to determine motor direction. Values less than the zero point correspond to motor direction = 0 and values greater than the zero point correspond to motor direction = 1. AIN Range is configured in the ModeConfig memory register (See section 5.2)

Dead Band

The Dead Band value sets the minimum value AIN must change before the controller will take action. The Dead Band option will prevent noise on the AIN line from translating to erratic motor behavior. Dead Band is configured in the DeadBand memory register (See section 5.2).

Maximum Speed

The Maximum Speed value is used to map the AIN voltage range over a specified operating range. In open-loop mode Maximum Speed represents a maximum PWM duty cycle. In speed and position modes Maximum Speed represents the maximum motor speed represented in RPM. Maximum Speed is configured in the MaxSpeed memory register (See section 5.2).

Encoder Scale - Position Control Only

Encoder scale is used to set the maximum distance the motor is allowed to travel. Encoder Scale is configured in the EncScale memory register (See section 5.2).

Approach Speed - Position Control Only

Approach speed is used to set the speed of the motor when it reaches its target position. In some cases it may be desirable to set the approach speed to something other than the maximum speed to minimize overshoot, or achieve a particular velocity profile. Approach Speed value is configured in the AppSpeed memory register (See section 5.2).

4. Serial Communications

4.1. SPI Communications

The IDCA-10 is compatible with 4-wire Serial Peripheral Interface (SPI) protocol (CPOL = 0, CPHA = 1). Data transmission occurs synchronously between the master and slave via the MISO and MOSI lines. A clock signal for transmission timing is provided on the SCK line and a chip enable line is provided on the NSS line. Multiple slave devices can exist on a single SPI bus. A dedicated NSS line must be provided to each slave device to prevent communication interference between devices on the bus. Figure 16 depicts a typical multi slave implementation on the SPI bus.

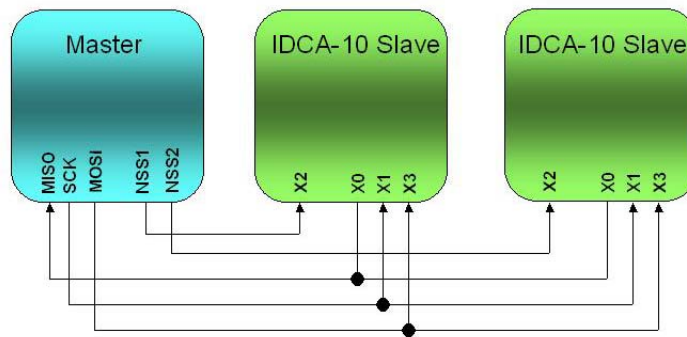


Figure 16. SPI bus block diagram.

4.2. SMBus (I²C) Communications

The IDCA-10 is compatible with the System Management Bus Specification (SMBus) and the Inter Integrated Circuit serial bus (I²C). Data transmission occurs on a two-wire bi-directional bus. A clock signal is supplied on the SCL line and data is transmitted on the SDA line. Data transmissions occur as a write or read operation. In both cases the master must initiate the data transfer operations. Multiple devices may exist on the SMBus/I²C serial bus. Slave devices will not respond to read or write request until it detects its address in the address byte of a data packet. Figure 17 depicts a typical multi slave implementation on the SMBus/I²C serial bus.

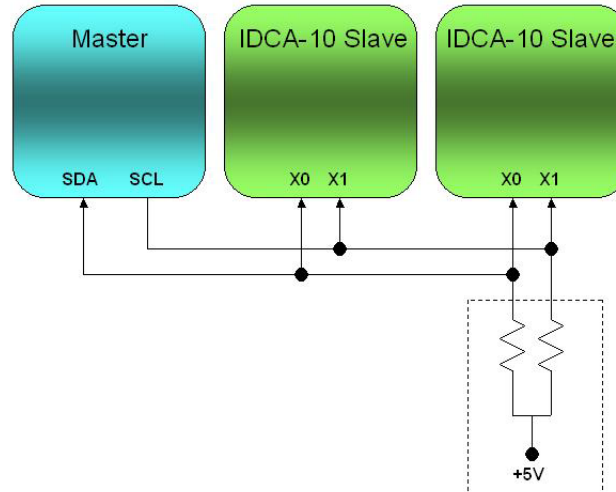


Figure 17. SMBus/I²C block diagrams.

A common SMBus/I²C bus configuration utilizes two pull-up resistors to pull the SCL and SDA lines up to +5VDC (Figure 17). The IDCA-10 has internal weak pull-up resistors on its SCL and SDA lines. External pull-ups are not necessary if one or more IDCA-10s are the only devices on the bus. However, external pull-ups may be necessary if the IDCA-10 is sharing the bus with other devices. The exact resistance value will depend on device requirements, bus impedance, and transmission rates. It is left to the user to determine the correct resistor value to meet application requirements.

4.3. Configuring Serial Bus Protocol

Serial bus configuration is stored in bit 0 of the ModeConfig memory register (See section 5.2). At power-up, hardware checks the ModeConfig register to determine how to configure the serial bus. A zero in the first bit indicates SPI mode and a one indicates SMBus/I²C. The default mode at power-up is SPI (bit 0 cleared). The user may toggle between modes by simply placing a jumper between LS1 and X2 and cycling power. Hardware will toggle the appropriate bit in ModeConfig to switch communication protocols. The new configuration will remain as long as power is applied to the IDCA-10. Cycling power will reset the bus to its original configuration, unless the jumper is installed. The current bus configuration must be saved to nonvolatile memory to make the change permanent.

Note:

Bus configuration is indicated by the first flash sequence at the PWR indicator when power is first applied. Two flashes indicate SPI. Three flashes indicate SMBus/I²C.

4.4. Data Transfer

The IDCA-10 is the slave for any data transfer regardless of operating mode or serial protocol. Data exchange must be initiated by the master, which sends a command packet to one of the slaves on the serial bus. In turn, the appropriate slave device processes the command packet and posts the results to its internal transmit buffer. Return data remains in the slave's transmit buffer until the master requests data from the slave.

SPI Data Transfer

SPI is a synchronous serial protocol where data between the master and slave is transmitted synchronously on the MISO and MOSI lines. The synchronous capabilities are not used in IDCA-10 transmissions because of its command/response operations. Instead, data transfer between the master and slave occur as two separate transmissions. The following steps illustrate a typical data exchange between master and an IDCA-10 on the SPI bus:

1. Command packet is constructed and the master sends data packet to the appropriate slave device. The master device ignores data returned from the slave during command packet transmission.
2. The master device waits for a minimum of 10 ms to allow the IDCA-10 to process the request and post data to its transmit buffer.
3. The master device transmits a NULL data packet to retrieve data from the slave's transmit buffer.

Return data from the slave is processed in 16-byte packets. The data packet used to retrieve data in SPI mode must also be 16 bytes long because data retrieval occurs synchronously with data transmission from the master to slave. A NULL data transmission is the preferred method to retrieve slave data in SPI mode. See section 4.5.1 for NULL packet definition.

SMBus (I²C)

The SMBus is a bi-directional bus where data is transmitted between the slave and the master on the SDA line. Data transfers occur asynchronously and therefore read requests are processed using a second transmission. The following steps illustrate a typical data exchange between master and an IDCA-10, using SMBus /I²C:

1. The master device constructs a command packet. The device address is left-shifted so it exists in the bits seven through one in the address byte.
2. Clear the least-significant bit in the address byte to indicate a write operation by the master.
3. Transmit data packet and wait a minimum of 10 ms to allow the IDCA-10 to process the request and post data to its transmit buffer.
4. The master constructs a NULL data packet and setting the least significant bit to intricate a read request to the slave.

5. The master reads 16 bytes from the slave's transmit buffer.

4.4.1. Command Packet Format

Command data packets are made up of two components, the PDU and the ADU. The PDU is comprised of function code and configuration data. The PDU format is always the same regardless of communication protocol being used. The ADU represents the complete data packet prepended with the device address and appended with the 16-bit CRC value. Command packet length varies depending on the instruction being sent. Instruction packet byte definitions and lengths are provided in section 4.5.

ADU format is essentially the same for SPI and I²C protocols with one minor exception. The I²C protocol uses the least significant bit of the address byte to inform the slave of a read or write request. The device address is left-shifted one bit in I²C communications, but is not in SPI communications. For this reason valid device addresses are limited to seven bits (0 – 127). The user must be aware of this to ensure the correct device address is transmitted with data packet. Figure 18 depicts the structures of a complete command packet.

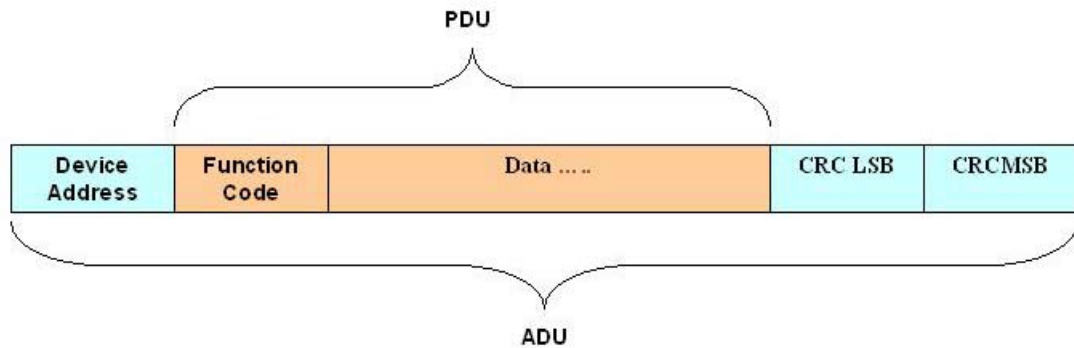


Figure 18. Command data packet structure.

4.4.2. Return Data Packet

Data packets returned by the slave are very similar in structure to command packets. The one major difference between the two is that all return data is constrained to 16 byte packets. Data returned to the master is contained in the 12 data bytes in the PDU. Unused data bytes are filled with zeroes. Return data longer than 12 bytes must be divided into multiple packets. Figure 19 depicts the structure for a complete return data packet.

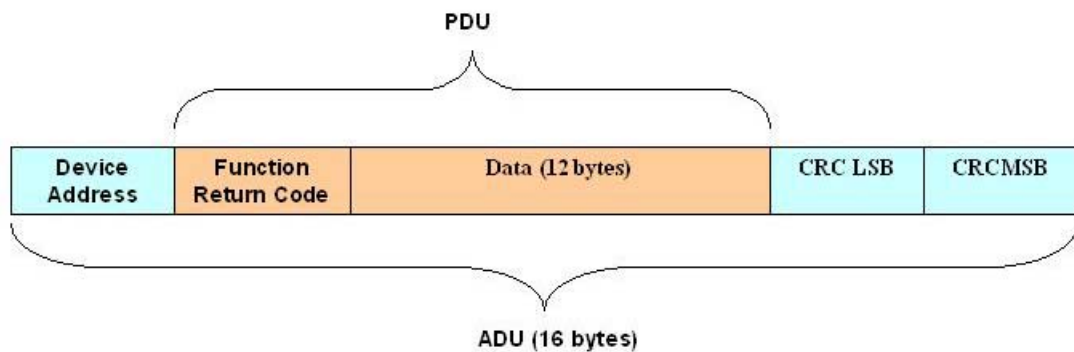


Figure 19. Return data packet structure.

Under normal error-free conditions, the IDCA-10 will echo the device address and function code to the master in a return data packet. A function code matching the original command function sent by the master indicates the message was processed without error and PDU contains valid data. Function codes offset by 128 indicate the hardware detected an error. Additional error information is reported in the PDU data bytes. Common return errors are defined in Appendix C.

4.4.3. CRC Algorithm

Every data packet contains two Cyclic Redundancy Check (CRC) bytes to ensure data integrity during transmission. The CRC calculation uses the device address and the data packet PDU to generate a 16-bit CRC value. The CRC value is appended as the last two bytes in the data packet, where the least-significant byte of the CRC value is the last byte in the transmission. Once data is received, the master or slave computes a CRC value for the device address and PDU, which is then compared to the CRC value transmitted with the packet. Matching transmitted and calculated CRC values indicates the data packet is valid. Mismatching values indicates the data was corrupted during transmission and should be disregarded.

The CRC calculation algorithm is as follows:

- Step 1: Initialize CRC result to 0xFFFF.
- Step 2: XOR the data byte with the most-significant byte of the current CRC result.
- Step 3: Left-shift the CRC result one bit.
- Step 4: XOR the CRC result with the polynomial (0x1021) if the most-significant bit is 1. Do not XOR the CRC result with the polynomial if the most-significant bit is 0.

Step 5: Repeat steps 3 and 4 for the remaining data bits in the data byte.

Step 6: Repeat steps 2 through 4 for the remaining data bytes in the data packet. The CRC bytes transmitted with the data packet are not included in the CRC calculation.

Table 5. Example CRC values.

Data Input (n bytes)	CRC Output
0x63	0xBD35
0x8C	0xB1F4
0x7D	0x4ECA
0xAA, 0xBB, 0xCC	0x6CFA
0x00, 0x00, 0xAA, 0xBB, 0xCC	0xB166

4.5. Instruction Packet Definitions

4.5.1. NULL Packet (0x41)

NULL data packet. The null data packet contains zeroes in all data bytes and is used to retrieve data from the slave device. The IDCA-10 does not execute any commands when it receives a NULL data packet. Therefore, any data constrained in the internal transmit buffer will remain unchanged. The NULL data transmission is also useful for to establish a heartbeat with the IDCA-10 without causing any actions to take place or loss of data in the transmit buffer.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x41
2-13	Data Bytes 0-11	0x00
14	CRC LSB	0x00 – 0xFF
15	CRC MSB	0x00 – 0xFF
Size = 16 bytes		

Transmitted Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code

Send 0x41 for a NULL data transfer

Bytes 2-12 NULL Data

Send zeroes.

Byte 13 - CRC MSB

CRC value most-significant byte.

Byte 14 - CRC LSB

CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x41 if no error occurred or a 0xC1 if an error occurred.

Byte 2 – 13 Data

Contains data stored in the IDCA-10 transmit buffer if no error is encountered. Byte 2 will contain an exception error and bytes 3-13 are set to zero if an error was encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.2. Software Reset (0x48)

Forces the IDCA-10 to perform a software reset, which is equivalent to cycling power on the unit. On reset the IDCA-10 will reboot into Idle mode or a pre configured mode. All RAM contents are reset to zero.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x48
2	Option	0x00 - 0xFF
3	CRC LSB	0x00 - 0xFF
4	CRC MSB	0x00 - 0xFF
Size = 5 bytes		

Transmitted Data

Device Address – Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Function Code – Send 0x48 for software reset.

Option – Set to zero to perform a normal reset. Set to a value greater than zero to force the IDCA-10 to reboot into idle mode. This option is useful if the unit is set to boot into a pre-configured mode.

CRC MSB – CRC value most-significant byte.

CRC LSB – CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x48 if no error occurred or a 0xC8 if an error occurred.

Byte 2 – Data

Returns zeroes on success. Byte 2 will contain an exception error and bytes 3-13 are set to zero if an error was encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.3. Retrieve Log Data (0x64)

The retrieve log command is used to retrieve log data stored in the 256-byte log buffer in RAM. The log buffer is separated into 22 subsets for easy retrieval. Data is stored chronologically, starting at subset 0. Subsets 0-20 are 12 bytes long and subset 21 is 4 bytes long.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x64
2	Log Data Subset	0x00 – 0x15
3	CRC LSB	0x00 – 0xFF
4	CRC MSB	0x00 – 0xFF
Size = 5 bytes		

Transmitted Data

Byte 0 - Device Address – Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code – Send 0x64 for to retrieve data.

Byte 2 - Log Data Subset – Send 0x00 to 0x15

Byte 3 - CRC MSB – CRC value most-significant byte.

Byte 0 - CRC LSB – CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x64 if no error occurred or a 0xE4 if an error was detected.

Bytes 2 to 13 – Log Data

Log data is returned as 6, 12-byte values. The MSB and the LSB are alternated for each word, starting with the first byte's MSB at byte 2 and ending with the sixth bytes LSB at byte 13.

Log data subset 21 only contains 4 bytes (2 words) of log data. Log data is returned in bytes 2 – 5. Bytes 6-12 are unused and return zeroes.

Byte 2 will contain an exception error and bytes 3-13 are set to zero if an error was encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.4. Access FLASH (0x65)

The *Access FLASH* function is used to perform rudimentary operations on FLASH memory contents.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x65
2	FLASH Option	0x00, 0x01, 0x04
3	CRC LSB	0x00 – 0xFF
4	CRC MSB	0x00 – 0xFF
Size = 5 bytes		

Transmitted Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code

Send 0x65 for to access FLASH contents

Byte 3 - FLASH Option

0x00 - Save calibration constants to non-volatile memory.

0x01 - Save the current configuration values to non-volatile memory.
Saving configuration constants to memory forces the IDCA-10 to boot into the current mode at power-up.

0x04 - Reset configuration constants so the IDCA-10 boots into its idle mode (factory default) at power-up.

All other values not listed are ignored.

Byte 1 - CRC MSB

CRC value most-significant byte.

Byte 1 - CRC LSB

CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x65 if no error occurred or a 0xE5 if an error was detected.

Bytes 2 to13 – Data

Byte 2 will echo the corresponding byte from the transmit buffer and bytes 3-13 will be filled with zeroes on success.

Byte 2 will contain an exception error and bytes 3-13 are set to zero if an error was encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.5. Read Registers (0x66)

The Read Registers function reads up to five of the Read-Only or Read/Write memory registers. Registers addresses are defined in section 5.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x66
2	Register Count	0x00 – 0x05
3	Reg. Add. 1 - MSB	0x00 – 0xFF
4	Reg. Add. 1 - LSB	0x00 – 0xFF
5	Reg. Add. 2 - MSB	0x00 – 0xFF
6	Reg. Add. 2 - LSB	0x00 – 0xFF
7	Reg. Add. 3 - MSB	0x00 – 0xFF
8	Reg. Add. 3 - LSB	0x00 – 0xFF
9	Reg. Add. 4 - MSB	0x00 – 0xFF
10	Reg. Add. 4 - LSB	0x00 – 0xFF
11	Reg. Add. 5 - MSB	0x00 – 0xFF
12	Reg. Add. 5 - LSB	0x00 – 0xFF
13	CRC LSB	0x00 – 0xFF
14	CRC MSB	0x00 – 0xFF
Size = 15 bytes		

Transmitted Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code

Send 0x66 to write to memory registers.

Byte 2 - Register Count

Send a value equal to the number of registers to read. Up to five registers may be read in a single transmission.

Bytes 3-12 Register Addresses

Register addresses to read. The number of addresses must correspond to the value passed in byte 2 in order to read all registers contained in the list. Byte 3 starts the address list with the MSB of the first address. The remaining bytes alternate between LSB and MSB for each address in the list, ending with the last register's LSB. Send zeroes in unused address bytes.

Byte 13 - CRC MSB

CRC value most-significant byte.

Byte 14 - CRC LSB

CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x66 if no error or a 0xE6 if an error is encountered.

Byte 2-13 – Data

Data bytes from memory registers from the address list transmitted in the command packet. Memory register contents are returned as two consecutive bytes starting with the MSB. Data pairs correspond to the order in which they were passed in the original command packet.

Byte 2 will contain an exception error value and the remaining data bytes will be set to zero if an error is encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

Write Registers (0x67)

Use the Write Registers function to write data to up to five read/write memory registers.

Byte	Contents	Valid Values (hex)
0	Device Address	0x00-0xEF
1	Function Code	0x67
2	Register Count	0x00 – 0x05
3	Reg. Add. 1 - MSB	0x00 – 0xFF
4	Reg. Add. 1 - LSB	0x00 – 0xFF
5	Data Byte - MSB	0x00 – 0xFF
6	Data Byte - LSB	0x00 – 0xFF
7	Reg. Add. 2 - MSB	0x00 – 0xFF
8	Reg. Add. 2 - LSB	0x00 – 0xFF
9	Data Byte - MSB	0x00 – 0xFF
10	Data Byte - LSB	0x00 – 0xFF
11	Reg. Add. 3 - MSB	0x00 – 0xFF
12	Reg. Add. 3 - LSB	0x00 – 0xFF
13	Data Byte - MSB	0x00 – 0xFF
14	Data Byte - LSB	0x00 – 0xFF
15	Reg. Add. 4 - MSB	0x00 – 0xFF

16	Reg. Add. 4 - LSB	0x00 – 0xFF
17	Data Byte - MSB	0x00 – 0xFF
18	Data Byte - LSB	0x00 – 0xFF
19	Reg. Add. 5 - MSB	0x00 – 0xFF
20	Reg. Add. 5 - LSB	0x00 – 0xFF
21	Data Byte - MSB	0x00 – 0xFF
22	Data Byte - LSB	0x00 – 0xFF
23	CRC LSB	0x00 – 0xFF
24	CRC MSB	0x00 – 0xFF
Size = 25 bytes		

Transmitted Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code

Send 0x67 to write to memory registers.

Byte 2 - Register Count

Send a value equal to the number of registers to write. Up to five registers may be written in a single packet transmission.

Bytes 3-22 Register Address and Data Byte

The register address and data to write are contained in three-byte triples starting at byte 3. The first byte in each triple is the register address MSB and the third byte is the data byte to write to memory

Byte 23 - CRC MSB

CRC value most-significant byte.

Byte 24 - CRC LSB

CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x67 if no error or a 0xE7 if an error is encountered.

Bytes 2 -13

Not used. Returns zeroes on success.

Byte 2 will contain an exception error value and the remaining data bytes will be set to zero if an error is encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.6. Configure IDCA-10 (0x68)

The *Configure IDCA-10* function is used to configure the IDCA-10 for a particular operation mode. *Configure IDCA-10* is a general configuration function and all fields are not used for all modes. Pass zeroes for data not required for configuration.

Byte	Contents	Valid Values (hex)	Modes
0	Device Address	0x00-0xEF	All
1	Function Code	0x68	All
2	Reserved	0x00	All
3	ModeConfig	0x00 – 0xFF	All
4	Reserved	0x00	All
5	BrdgConfig	0x00 – 0xFF	All
6	Reserved	0x00	All
7	Reserved	0x00	All
8	FreqSF – MSB	0x00 – 0xFF	All
9	FreqSF - LSB	0x00 – 0xFF	All
10	PIDUpdate - MSB	0x00 – 0xFF	SC, PC
11	PIDUpdate - LSB	0x00 – 0xFF	All
12	CPR – MSB	0x00 – 0xFF	All
13	CPR – LSB	0x00 – 0xFF	SC, PC
14	Kp Byte 1 – MSB	0x00 – 0xFF	SC, PC
15	KP Byte 2	0x00 – 0xFF	SC, PC
16	KP Byte 3	0x00 – 0xFF	SC, PC
17	Kp Byte 4 – LSB	0x00 – 0xFF	SC, PC
18	Ki Byte 1 – MSB	0x00 – 0xFF	SC, PC
19	KI Byte 2	0x00 – 0xFF	SC, PC
20	Ki Byte 3	0x00 – 0xFF	SC, PC
21	Ki Byte 4 – LSB	0x00 – 0xFF	SC, PC
22	Kd Byte 1 – MSB	0x00 – 0xFF	SC, PC
23	KD Byte 2	0x00 – 0xFF	SC, PC
24	KD Byte 3	0x00 – 0xFF	SC, PC
25	Kd Byte 4 - LSB	0x00 – 0xFF	PC
26	DeadBand - MSB	0x00 – 0xFF	SA
27	DeadBand - LSB	0x00 – 0xFF	SA
28	MaxSpeed - MSB	0x00 – 0xFF	PC, SAPC
29	MaxSpeed - LSB	0x00 – 0xFF	PC
30	AppSpeed - MSB	0x00 – 0xFF	PC
31	AppSpeed - MSB	0x00 – 0xFF	PC
32	EncScale - MSB	0x00 – 0xFF	SAPC

33	EncScale - LSB	0x00 – 0xFF	All
34	CRC LSB	0x00 – 0xFF	All
35	CRC MSB	0x00 – 0xFF	All
Size = 36 bytes			

Note:

OL = Open-loop

SC = Speed-control

PC = Position Control

SA = Stand-alone (all modes)

SAPC = Stand-alone position control

Transmitted Data

Byte 0 - Device Address

A unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode.

Byte 1 - Function Code

Send 0x68 to configure the IDCA-10 for operations.

Byte 2 - Reserved

Send 0x00.

Bytes 3 - ModeConfig

The mode configuration value sets the communication and operating modes for the IDCA-10.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	AINRNG1	AINRNG0	SAMODE	MODE1	MODE0	Reserved	SERCFG

AINRNG(1,0) – Analog input voltage range:

00 = 0-5 VDC

01 = ± 5 VDC

10 = 0-10 VDC

11 = ± 10 VDC

SAMODE – Stand-alone mode flag:

0 = IDCA-10 not in stand-alone mode

1 = IDCA-10 operating in stand-alone mode.

Mode(1,0) – Operating mode

00 = Idle (default)

01 = Open loop mode

10 = Speed control mode

11 = Position control mode

SERCFG – Serial bus configuration bit
0 = SPI
1 = I²C

Byte 4 - Reserved
Send 0x00

Byte 5 - BrdgConfig

The mode configuration value sets operating parameters for the IDCA-10s H-bridge. Used in all modes.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BRDGEN	Reserved	Reserved	Reserved	Reserved	D1	Reserved	DIR

BRDGEN – Bridge Enable State:
0 = Bridge Disabled – Sleep mode.
1 = Bridge Enabled

D1 – Disable Motor Outputs:
0 = Motor outputs enabled
1 = Motor outputs disabled and in high impedance state

DIR – Motor direction
0 = Positive terminal at OUT 1
1 = Positive terminal at OUT 2

Bytes 6 and 7 - Reserved
Send zeroes.

Bytes 8 and 9 - FreqSF

FreqSF contains the scale value used to calculate PWM frequency. Valid PWM frequencies are between 1000 Hz and 10 kHz. The PWM scale value is calculated using the following equation:

Equation 3

$$\text{FreqSF} = 24,500,000 \text{ Hz} \div \text{PWM Frequency}$$

Note:

FreqSF is represented at a 16-bit integer. Round the result from Equation 3 to the nearest whole number.

Byte 10 and 11 – PIDUpdate

This value sets the update period used by the PID controller in milliseconds. The minimum update period is 10 ms and the maximum is 65,535 milliseconds. PIDUpdate is only required in modes using the PID control loop. Send zeroes if not used.

Byte 12 and 13 – CPR

This value represents the number of counts per revolution generated by the motor encoder.

Byte 14 to 17 – Kp

This value sets the proportional gain constant used by PID controller. Kp is stored as a floating-point number, using the IEEE-754 standard.

Byte 18 to 21 – Ki

This value sets the integral gain constant used by PID controller. Ki is stored as a floating-point number, using the IEEE-754 standard.

Byte 22 to 25 – Kd

This value sets the derivative gain constant used by PID controller. Kd is stored as a floating-point number, using the IEEE-754 standard.

Byte 26 and 27 - DeadBand

DeadBand sets the dead band region for AIN when in stand-alone mode. Use of a dead band at AIN prevents the controller from changing due to noise at the analog input. The DeadBand value is calculated according to the following equation:

Equation 4

$$\text{DeadBand} = \text{Dead band Voltage} \div 0.005476$$

For example, to calculate DeadBand value equal to +/-0.1V:

Equation 5

$$\text{DeadBand} = 0.1V \div 0.005476 = 18.26 \rightarrow 18$$

Note:

DeadBand is represented at a 16-bit integer. Round the result from Equation 5 to the nearest whole number.

Byte 28 and 29 - MaxSpeed

MaxSpeed sets the starting speed, in RPM, for the motor when a new position command is initiated

Byte 30 and 31 - AppSpeed

AppSpeed sets the approach speed, in RPM, for the motor as it approaches a target position.

Byte 32 and 33 – EncScale

EncScale sets the encoder scale value used in stand-alone position control mode. The encoder scale value maps the analog input voltage to specified

number of motor revolutions. For example, the EncScale value required to map a 0-5VDC analog input range to 3 revolutions of 512-count encoder is:

Equation 6

$$(512 \text{ counts/rev} * 3\text{rev}) \div 2.5\text{Volts} = 614.4 \text{ counts/volt} \rightarrow 614$$

Note:

DeadBand is represented at a 16-bit integer. Round the result from Equation 6 to the nearest whole number.

Byte 34 - CRC MSB (All modes)

CRC value most-significant byte.

Byte 35 - CRC LSB (All modes)

CRC value least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x68 if no error occurred or a 0xE8 if an error is encountered.

Bytes 2 -13

The IDCA-10 will echo the corresponding bytes from the command packet on success.

Byte 2 will contain an exception error value and the remaining data bytes will be set to zero if an error is encountered by hardware.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.5.7. Write To Instruction Buffer (0x69)

Writes MCT Op-code instruction packets to the 256-byte instruction buffer. MCT Op-code packets are defined in section 4.6.

Byte	Contents	Valid Values (hex)
------	----------	--------------------

0	Device Address	0x00-0xEF
1	Function Code	0x69
2 – 11	MCT Op-code Sub Packet (10 bytes)	0x00 – 0xFF
12	CRC LSB	0x00 – 0xFF
13	CRC MSB	0x00 – 0xFF
Size = 14 bytes		

Transmitted Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The device address is ignored in SPI Mode

Byte 1 - Function Code

Send 0x69 to write to instruction buffer.

Bytes 2 to 11 – MCT Op-code sub packet

Send 10-byte operation code sub packets. MCT op-codes are further defined in section 4.6.

Byte 12 - CRC MSB

CRC value most-significant byte.

Byte 13 - CRC LSB

CRC value Least-significant byte.

Return Data

Byte 0 - Device Address

Unique address used to define the IDCA-10 on the serial bus. The IDCA-10 will return its current address regardless of which serial protocol is being used.

Byte 1 – Function Code

The IDCA-10 returns a 0x69 if no error occurred or a 0xE9 if an error is encountered.

Byte 2 – Bytes Written

Number of bytes written to the 256-byte memory buffer. The number of bytes written equals the number of applicable bytes in the op-code sub packet.

Byte 3 – Bytes Available

Number of unused bytes in 256-byte memory buffer.

Bytes 4-13

Returns zeroes.

Note:

Byte 2 will contain an exception code if an error was encountered during transmission, and remaining data bytes are set to zero.

Byte 14 - CRC MSB

Return packet CRC value most-significant byte.

Byte 15 - CRC LSB

Return packet CRC value Least-significant byte.

4.6. MCT Op-codes Definitions

The MCT operational codes (op-codes) are a set of instructions used to command the IDCA-10 to complete different tasks. All op-codes are placed into the 256-byte instruction buffer where they await execution by hardware. Multiple op-codes may be queued in the buffer to sequentially execute a complicated set of tasks.

4.6.1. Pause (0x00)

Pauses the internal instruction handler a specified number of milliseconds. The Pause command is used when a short delay is desired between the execution of two consecutive op-codes.

Byte	Contents	Valid Values (hex)
0	MCT Op-code	0x00
1	Period (ms)	0x00 – 0xFF
2	Reserved	0x00
3	Reserved	0x00
4	Reserved	0x00
5	Reserved	0x00
6	Reserved	0x00
7	Reserved	0x00
8	Reserved	0x00
9	Reserved	0x00
Size = 10 bytes		

4.6.2. Set Motor Direction (0x01)

Changes motor direction. Motor direction is set according to the value of the

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-Code	1	0x01
Motor Direction	0 – 255	0x00 – 0xFF
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

Motor Direction byte:

0 = Positive terminal at the OUT1 pin.

>0 = Positive terminal at the OUT2 pin.

4.6.3. Enable/Disable Bridge (0x02)

Enables or disables the IDCA-10 H-bridge. The H-bridge goes into sleep mode and outputs are placed into a high-impedance state when it is disabled.

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-code	2	0x02
Enable	0 – 255	0x00 – 0xFF
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

Enable:

0 = H-bridge disabled

>0 = H-bridge enabled

4.6.4. Disable Motor Outputs (0x03)

Disables H-bridge outputs without placing bridge into sleep mode. Passing a value greater than zero in the Disable data byte disables the outputs.

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-code	3	0x03

Disable	0 – 255	0x00 – 0xFF
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

Disable:

0 = H-bridge outputs enabled

>0 = H-bridge enabled placed into high impedance state

4.6.5. Update Motor Speed (0x04)

Use to update motor speed.

Contents	Valid Values (dec)	Valid Values (hex)
Pause Op-code	4	0x04
Motor Speed – MSB	0 – 255	0x00 – 0xFF
Motor Speed - LSB	0 – 255	0x00 – 0xFF
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

Motor Speed:

The value passed for motor speed in open-loop mode is a 12-bit representation of the PWM duty cycle where 0 corresponds to 0% and 4095 corresponds to 100%. The motor speed value is the motor speed in RPM for all other modes.

4.6.6. Log Data (0x05)

Configures the internal data logger to record data from one of three sources at a specified rate.

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-code	5	0x05
Data Source	0 – 2	0x00 – 0x02
Sample Period	0-255	0x00 – 0xFF
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00

Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

Data Source:

- 0 = Encoder A (Only available in open-loop mode)
- 1 = Encoder B
- 2 = Motor current

Values greater than two are ignored by hardware

Period:

Data logger period measured in milliseconds (10ms to 255 ms).

4.6.7. Clear Instruction Buffer (0x06)

Clears all pending instructions queued in the 256-byte instruction buffer. Instructions being processed when this op-code is called are not affected by this op-code.

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-code	6	0x06
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Reserved	0x00	0x00
Size = 10 bytes		

4.6.8. Update Motor Position (0x10)

Used to command a motor to a desired position (position mode only). See section 3.4 for more details on position control mode.

Contents	Valid Values (dec)	Valid Values (hex)
MCT Op-Code	16	0x10
Max Speed - MSB	0 – 255	0x00 – 0xFF
Max Speed – LSB	0 – 255	0x00 – 0xFF
App. Speed – MSB	0 – 255	0x00 – 0xFF
App. Speed - LSB	0 – 255	0x00 – 0xFF
Distance Byte 1 (MSB)	0 – 255	0x00 – 0xFF
Distance Byte 2	0 – 255	0x00 – 0xFF
Distance Byte 3	0 – 255	0x00 – 0xFF

Distance Byte4 (LSB)	0 – 255	0x00 – 0xFF
Pos. Config	0 – 255	0x00 – 0xFF
Size = 10 bytes		

Max Speed:

Starting speed for motor in RPM.

App Speed:

Approach speed measured in RPM.

Distance:

Command distance to travel measured in encoder counts.

Position Config:

Position configuration byte. See description of PosCurCfg memory register in section 5.2 for more information.

5. IDCA-10 Memory Registers

The following section lists the IDCA-10 memory registers accessible using the Read Registers and Write Registers functions (See sections 4.5.5 and 4.5.7).

5.1. Read Only Registers

Reserved Registers (0x0000 – 0x0025)

Reserved for use by the manufacturer.

TempSense (0x0026)

This register holds the most recent ADC code read from the internal temperature sensor. Use the following reading to convert the stored value to a valid temperature:

Equation 7.

$$\text{Temp (deg C)} = 0.182 * \text{TempSense} - 305.08$$

Note:

$$C \rightarrow F = C * 9/5 + 32$$

CurSense (0x0027)

This register holds the most recent ADC code for the current reading for the current sensed at the H-bridge. Convert the register value to current using Equation 8.

Equation 8.

$$\text{Current (Amps)} = 0.00224 * \text{CurSense}$$

AnlgInp (0x0028)

This register holds the most recent ADC code measured at the AIN terminal. AIN measurements are only valid when operating in stand-alone mode. Use Equation 9 to convert the register value to a valid voltage.

Equation 9.

$$\text{AIN (Volts)} = 0.00548 * \text{AnlgInp} - 12.225$$

Reserved Register (0x0029)

Reserved for use by the manufacturer.

AccErr0 (0x002A)

AccErr0 stores the error accumulated when the motor dir bit in BdgConfig is set to 0. AccErr will only accumulate error when the AccErr bit is set in the PosCurCfg register. ACCERR0 is automatically reset to 0x00 when the AccErr bit is cleared in PosCurCfg.

AccErr1 (0x002B)

ACCERR1 stores the error accumulated when the motor dir bit in BdgConfig is set to 1. AccErr1 will only accumulate error when the AccErr bit is set in the PosCurCfg register. AccErr1 is automatically reset to 0x00 when the AccErr bit is cleared in PosCurCfg.

Reserved Registers (0x002C – 0x00FF)

Reserved for use by the manufacturer.

5.2. Read/Write Registers

DevAdd (0x0100)

Device address. Valid values are 0x00 – 0xEF. The most-significant bit is ignored.

ModeConfig (0x0101)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	AINRNG1	AINRNG0	SAMODE	MODE1	MODE0	Reserved	SERCFG

The ModeConfig byte defines operating mode and various operating parameters:

Bit 7 – Reserved

Bits 6,5 – Analog input range for AIN

00 = 0-5 VDC

01 = ± 5 VDC

10 = 0-10 VDC

11 = ± 10 VDC

Bit 4 – Stand-alone mode enable bit

1 = Stand-alone enabled

0 = Stand-alone disabled

Bits 3,2 – Operating mode bits

00 = Reset/Idle

01 = Open-loop

10 = Speed control
11 = Position control

Bit 1 – Reserved

Bit 0 – Defines the communication protocol for the serial bus.
1 = SMBus (I2C)
0 = SPI

BdgConfig (0x0102)

The BdgConfig byte defines configuration parameters for the IDCA-10's H-bridge.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BRDGEN	Reserved	Reserved	Reserved	Reserved	D1	Reserved	DIR

Bit 7 – Bridge Enable State:
0 = Bridge Disabled (Sleep mode)
1 = Bridge Enabled

Bit 6 – Reserved

Bit 5 – Reserved

Bit 4 – Reserved

Bit 3 – Reserved

Bit 2 – Disable Motor Outputs:
0 = Motor outputs enabled
1 = Motor outputs disabled and in high impedance state

Bit 1 – Reserved

Bit 0 – Motor direction
0 = Positive terminal at OUT 1
1 = Positive terminal at OUT 2

PosCurCfg (0x0103)

The PosCurCfg byte defines various operating parameters for position control mode.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	ACCERRR	Reserved	DIR

Bit 7 – Reserved

Bit 6 – Reserved

Bit 5 – Reserved

Bit 4 – Reserved

Bit 3 – Reserved

Bit 2 – Accumulate error enable bit

1 = Error accumulation on

0 = Error accumulation off

Bit 1 – Reserved

Bit 0 – Direction bit.

0 = Positive terminal at OUT 1

1 = Positive terminal at OUT 2

LogMode (0x0104)

The LogMode byte defines which values are logged when the internal data logger is started. One of three data sources may be set according to values stored in this register.

Register Values:

0x00 – Ignored by hardware

0x01 – Log Encoder A values

0x02 – Log Encoder B values

0x04 – Log motor current values

0x05 – 0xFF Ignored by hardware

LimStat (0x0105)

The limits switch status register defines and reports various operating parameters when using the limit switch inputs LS1 and LS2.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LSPROC1	LSPROC0	LSCMPLT	LSLTCH	Reserved	Reserved	ENLS2	ENLS1

Bit 7 – Indicates that L.S. 2 was activated. LSPROC1 is automatically set to 0 once the LS instructions are executed by firmware.

0 = Not processing limit switch 2 code

1 = Processing limit switch 2 code.

Bit 6 – Indicates that L.S. 1 was activated. LSPROC0 is automatically set to 0 once the LS instructions are executed by firmware.
0 = Not processing limit switch 1 code
1 = Processing limit switch 1 code.

Bit 5 – This bit gets set when the motor is commanded to zero velocity by the LS code.
0 = Processing LS code
1 = LS code complete

Bit 4 - Limit switch latch. LS1 is used as a latch for AIN when set. LSLTCH is only valid when controller is operating in stand-alone mode.
0 = LS not used as a latch
1 = LS used as AIN latch signal

Bit 3 – Reserved

Bit 2 – Reserved

Bit 1 – Enable Limit switch 1
0 = LS1 disabled
1 = LS0 enabled

Bit 0 – Enable Limit switch 0
0 = LS0 disabled
1 = LS0 enabled

ErrorReg (0x0106)

The error register reports internal errors generated by the IDCA-10. IDCA-10 internal errors are separated into three groups depending on severity:

1. Level 0 - General warnings and do not pose any problems to operation. Hardware can recover from level 0 errors without intervention.
2. Level 1 - Mid-level warnings informing the user that problems were encountered that will lead to undesirable operation if errors persist. Level 1 errors are indicated by a blinking error LED.
3. Level 2 – High-level errors and will cause hardware to stop the motor and go into a safe state. Level 2 errors indicate that a problem was encountered that and require some sort of intervention from the user. The error LED will turn on solid when a level 2 error is generated by hardware.

Table 6. IDCA-10 hardware error codes.

Source	Severity Level	Code	Description
SPI/I2C	0	0x01	SPI – Write collision
SPI/I2C	0	0x02	SPI – Receive buffer overrun
SPI/I2C	0	0x03	SPI – SPI timeout
Reserved	0	0x04 – 0x55	
Reserved	1	0x056 – 0xAA	
Bridge Fault	2	0xAB	Motor I.C. fault line set. Caused by over-temperature shutdown, short circuit or other fault mode.
Reset Source	2	0xAC – 0xEB	Hardware reset caused by source other than software reset.
LS1 Set	2	0xEC	
LS2 Set	2	0xED	
Instruct. Buff	2	0xEE	Invalid Op-code
Reserved	2	0xEF – 0xFF	

Reserved (0x0107 – 0x011F)

Reserved for use by the manufacturer.

FreqSF (0x0120)

This value defines the scale value used to calculate PWM frequency at the H-bridge.

DutyCnt (0x0121)

This registers stores the PWM duty cycle currently being used to drive the motor. The duty cycle is represented as a 12-bit value where 0 corresponds to 0% duty cycle and 4095 corresponds to 100% duty cycle. Values greater than 4095 are automatically set to 4095 by firmware.

Distance (0x0122 - 0x0123)

The Distance register stores the command distance represented in encoder counts. The value stored in the Distance register is only valid when operating in position control mode. The distance value is represented as 32-bit unsigned integer. The most-significant word (16 bits) is stored at address 0x0122 and the lest-significant word (16-bytes) is stored at address 0x0123. Valid values for command distances range from 0 - 4,294,967,295.

DeadBand (0x0124)

The DeadBand register stores the value used to represent the dead band region for AIN in stand-alone mode.

MaxSpeed (0x0125)

The MaxSpeed register stores the motor speed (RPM) to use when the initiating a new move command.

EncScale (0x0126)

The encoder scale value stores the scale value used to map the AIN voltage to motor position.

AppSpeed (0x0127)

The MaxSpeed register stores the motor speed (RPM) to use when the approaching a target position.

VelSP (0x0128)

Stores the current motor velocity set point.

CPR (0x0129)

This register holds the number of counts per revolution specified for the motor encoder.

PIDUpdate (0x012A)

This register stores the update period, in milliseconds, used by the PID controller.

Reserved (0x012B - 0x012F)

Reserved for use by the manufacturer.

LS1Rate (0x0130)

Stores the deceleration rate used by limit switch 1.

LS2Rate (0x0131)

Stores the declaration rate used by limit switch 2.

FBy (0x0132)

FBy is the y-intercept for the correction equation used when measuring motor current.

AINy (0x0133)

AINy is the y-intercept for the correction equation used when measuring voltage at the AIN input.

Ty (0x0134)

Ty is the offset used to correct temperature offsets at the internal temperature sensor.

Reserved (0x0134 – 0x014F)

Reserved for use by the manufacturer.

Kp (0x0150 – 0x0151)

Kp stores the proportional gain value use by the PID controller. Kp is stored as a floating-point number, using the IEEE-754 standard.

Ki (0x0152 – 0x0153)

Ki stores the integral gain value use by the PID controller. Ki is stored as a floating-point number, using the IEEE-754 standard.

Kd (0x0154 – 0x0155)

Kd stores the derivative gain value use by the PID controller. Ki is stored as a floating-point number, using the IEEE-754 standard.

FBm (0x0156 – 0x0157)

FBm stores the slope for the correction formula used when measuring motor current. FBm is stored as a floating-point number, using the IEEE-754 standard.

ASCm (0x0158 – 0x0159)

ASCm stores the slope for the correction formula used when measuring voltage at AIN. ASCm is stored as a floating-point number, using the IEEE-754 standard.

Reserved (0x015A - 0xFFFF)

Reserved for use by the manufacturer.

Appendix A – PID Controller Basics

A PID controller is made up of three components that perform different mathematical operation on the input signal:

1. Proportional Gain – Multiplies the error signal by a proportional gain value.
2. Integral Gain – Multiplies the error signal by integral gain and integrates (sums) the result over time.
3. Derivative Gain – Multiplies the error signal by the derivative gain and differentiates the result with respect to time.

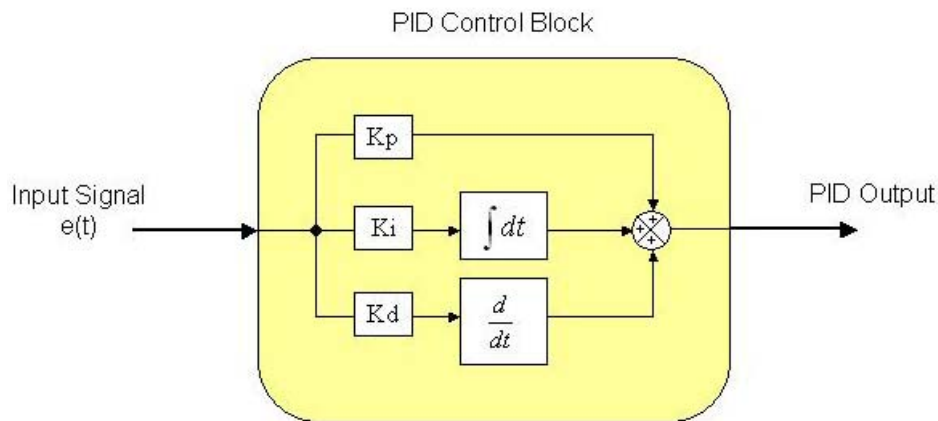


Figure 20. PID control block diagram.

As shown in Figure 20, the results from each component are summed together to form the output from the PID block. In the case of the IDCA-10, the output from the PID block is the PWM duty cycle used to drive the motor.

Proportional Component Effects

The proportional component affects motor rise time when commanded to a new speed. The proportional component has the largest affect on the PID output signal when the error is large. The influence on the output signal will decrease as the motor approaches its set point. Increasing the proportional gain (K_p) will decrease the system rise time. However, large values for K_p can cause excessive overshoot and lead to system instability. Start with a small value for K_p and adjust slowly until the desired rise time is attained.

Integral Component Effects

The integral component affects the steady state error the system. That is, it determines how close the motor's speed comes to the commanded set point. In general, increasing the integral gain (K_i) will decrease the steady state error. Excessive values at K_i can lead to integrator wind-up, which can lead to instability when a load is suddenly removed.

Derivative Component Effects

The derivative component adds damping to the motor system. Damping is useful when a system exhibits oscillations around a set point. Increasing the derivative gain (K_d) will damp out oscillations and force the motor to settle towards the commanded speed. The IDCA-10 inherently adds a significant amount of damping to the system. Therefore, the derivative gain is typically not used except in special situations. Set the derivative gain to zero for most applications.

Appendix B – IDCA-10 Error Code Definitions

The following table outlines error data contained in a return packet PDU when the IDCA-10 detects and posts an error. Error values generated by the IDCA-10 are contained in the second byte of the PDU.

Error	Value	Description
Illegal Function	0x01	Function code sent by the master is not valid.
Illegal Data Address	0x02	Memory register address being accessed is not valid or a write operation is being performed on a read-only register.
Illegal Data Value	0x03	Data sent in command packet was not valid.
Slave Device Failure	0x04	Communication failure or other internal error occurred.
Slave Device Busy	0x06	The IDCA-10 was busy processing another command and could not process the transmitted command packet.
Memory Parity Error	0x08	The CRC value generated by the slave does not match the transmitted CRC value. Data corrupted.

Appendix C – Electrical Characteristics

General

Characteristic	Symbol	Min	Typ.	Max	Unit
Supply Voltage	$+V_S$	8	--	28	V
H-bridge Output Voltage	OUT_n	--	$+V_S - 0.5$	--	V
Continuous Output Current ⁽¹⁾	I_{OUT}	0	--	5	A
Current Limiting Threshold	I_{LIM}	5.0	6.5	7.8	A
Over temperature Shutdown ⁽²⁾	T_{MAX}	175	--	225	C
Peripheral Voltage (+5V0) $I_{out} = 0$ A $I_{out} = 250$ mA $I_{out} = 500$ mA ⁽³⁾	+5V0	-- -- --	5.5 5.1 4.6	-- -- --	V V V
Peripheral Output Current ⁽³⁾	IP_{max}	--	--	500	mA
Quiescent Current Drive Enabled (EN = 1) +Vs = 9V +Vs = 24V Drive Disabled (EN = 0) +Vs = 9V +Vs = 24V	I_Q	-- -- -- --	31.5 18.5 26.8 13.5	-- -- -- --	mA mA mA mA
Operating Temperature Range	T_{OP}	-40	--	85	°C

NOTES:

1. Inability to adequately dissipate heat from the drive unit will result in lower continuous current limit due to over temperature shutdown limits.
2. H-bridge IC junction temperature.
3. 500 mA loads are not to exceed 30s in duration.

Control IO

Characteristic	Symbol	Min	Typ.	Max	Unit
Control I/O voltage limits	V_I	-10	--	10	V
Control Input logic levels HIGH input voltage LOW input voltage	V_{IH} V_{IL}	2.3 --	-- --	-- 1.0	V V
Control I/O Input Impedance	R_I	--	10	--	k Ω
Control I/O Output Impedance	R_O	--	10	--	k Ω
Serial Bus Clock Freq SPI SMBus (I ² C)	F_{SER}	300 40	-- --	10,000 10,000	Hz

Analog Input (AIN)

Characteristic	Symbol	Min	Typ.	Max	Unit
AIN voltage limits	V_{AI}	-12	--	12	V
AIN Input Impedance	R_{AI}	--	100	--	k Ω

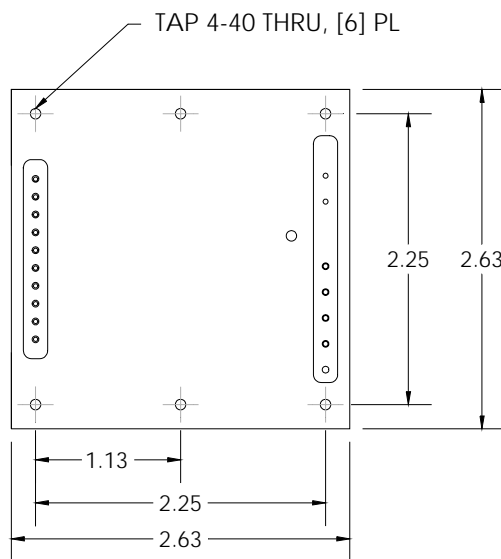
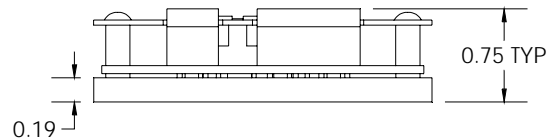
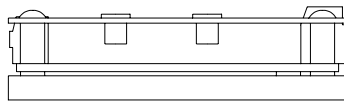
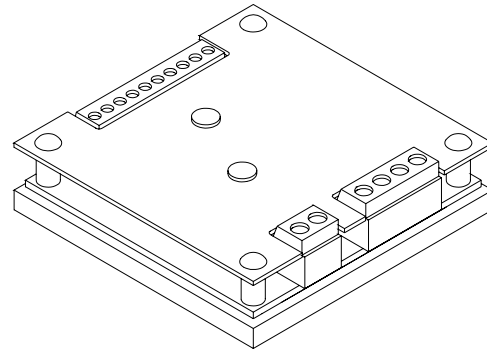
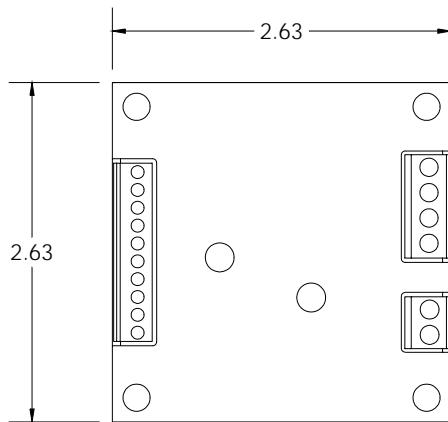
H-bridge

Characteristic	Symbol	Min	Typ.	Max	Unit
PWM Frequency	f_{PWM}	1	--	10	kHz
Bridge Resistance ⁽¹⁾	R_{BR}	--	0.240	--	Ω
Bridge Current Feedback Signal $I_{BRIDGE} = 0 \text{ A}$ $I_{BRIDGE} = 0.5 \text{ A}$ $I_{BRIDGE} = 1.5 \text{ A}$ $I_{BRIDGE} = 3.0 \text{ A}$ $I_{BRIDGE} = 6.0 \text{ A}$	V_{FB}	0 35 286 571 1.143	-- 77.5 357 714 1.429	5 156 428 857 1.715	mV mV mV mV V
Short Circuit Threshold (high-side)	I_{THS}	11	13	16	A
Short Circuit Threshold (low-side)	I_{TLS}	9	11	14	A
Recommended Minimum Motor Winding Resistance ⁽²⁾ $+V_s = 28\text{V}$ $+V_s = 18\text{V}$ $+V_s = 12\text{V}$ $+V_s = 8\text{V}$	R_{MOT}	2.3 1.5 1.0 0.75	-- -- -- --	-- -- -- --	Ω

NOTES:

1. R_{BR} value when the junction temperature at 25 C.
2. Motor winding resistances less than that noted for R_{MOT} can result in excessive bridge currents during braking, and can cause serious damage to the IDCA-10

Appendix D - Mechanical Drawings



(BOTTOM VIEW)

1. Dimensions shown are in inches.
2. All dimensions ± 0.010 "

NOTE:

The DCA-10 heat sink and enclosure are electrically isolated from +VIN and GND. Do not use the heat sink as a ground connection.

Revision History

- November, 2010 – Initial release