Supermodifed™ – Miniature controller for DC motors

1. General Description

The Supermodifed™ DC motor controller is a combination of three miniature PCBs offering an all-in one motor control solution. Comprising a 12-bit magnetic absolute encoder, an 8-bit AVR ATMega328P microcontroller running at 20MHz and a 5-Amp MOSFET H-bridge, the Supermodified™ comes at an astounding form factor of 15,5 x 15,5 x 10.8 mm and is ideal for space constrained applications. Its miniature size makes it ideal for installation inside standard as well as, 1/4-scale RC servos, allowing for full servomotor functionality. The Supermodifed™ is a cost-effective solution, delivering closed-loop PID control at 1KHz, with advanced motion profiling capabilities and many other features found only in expensive, high-end equipment.

ZERO ONE
MECHATRONICS

Supermodifed™

Miniature Controller for DC Motors

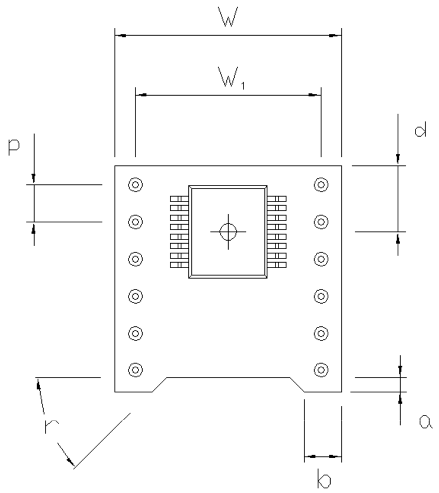
The robotic rebirth of the hobby servo

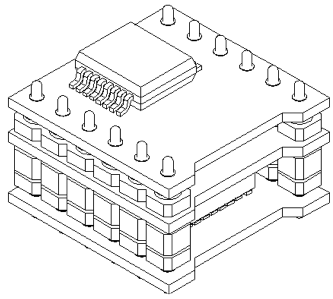
2. Features

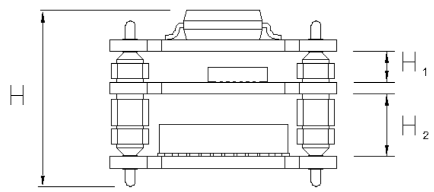
- **Miniature size. Ideal for space/weight critical applications such as robotics.**
- **Programmable. Based on the ATMega328P 8-bit RISK Atmel AVR microcontroller the Supermodifed™ controller can be programmed using standard development tools, commercially available.**
- **Open source. Download Supermodifed™ controller source code from our Google Code page. Schematics available from our Supermodifed™ product page.**
- **Modular. Divided into three stand-alone modules:**
  - **MagEnc™: Absolute 12-bit magnetic encoder.**
  - **PicoMcu™: the world's smallest AVR development board, based on ATMega328P microcontroller, running at 20MHz.**
  - **Motor Driver: MOSFET H-Bridge board based upon the Freescale MC33887 delivering 5 Amps continuous (with up to 7 Amps bursts).**
- **Multiple interfaces. Electrical layer interfaces include I2C 5V, I2C 3.3V, RS-485, UART 5V, UART 3.3V. Protocol interfaces include 01 Mechatronics open-source protocol, Modbus (under development), Position Pulse Modulation (standard RC-servo compatibility mode), Velocity Pulse Modulation.**
- **Freely available APIs (Application Programming Interfaces) for popular platforms like Arduino and Gumstix (under development).**
- **Includes four configurable digital IOs and four analog inputs.**
- **Includes on-board SPI as per ATMega168P datasheet.**

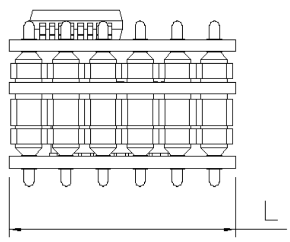
revision 1: 30/3/2010
Antonis Bouloubasis
Giannis Sissakis

3. Mechanical layout



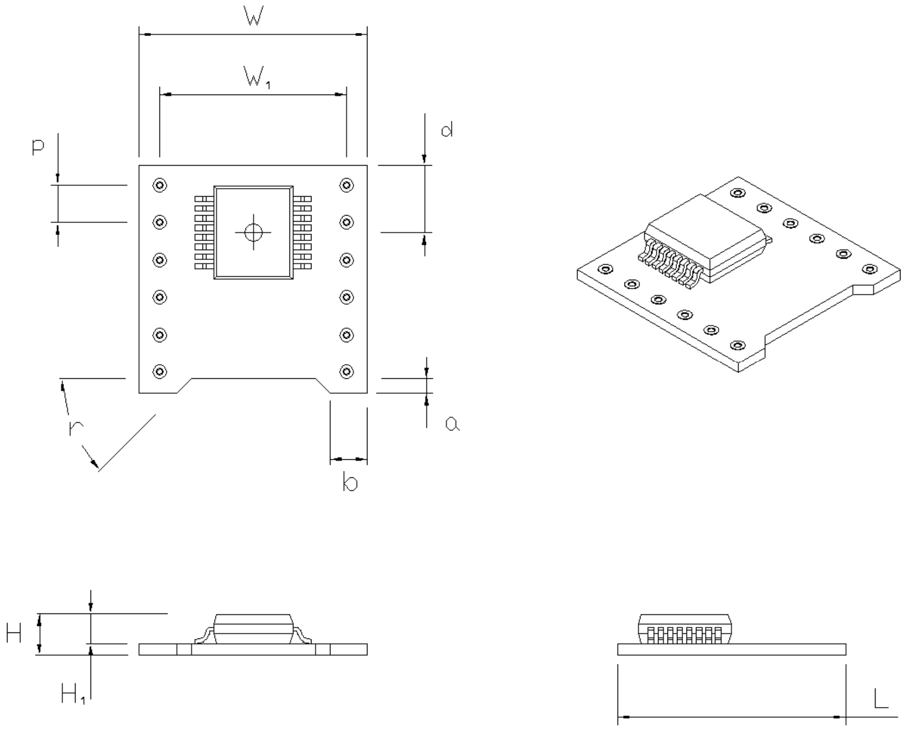






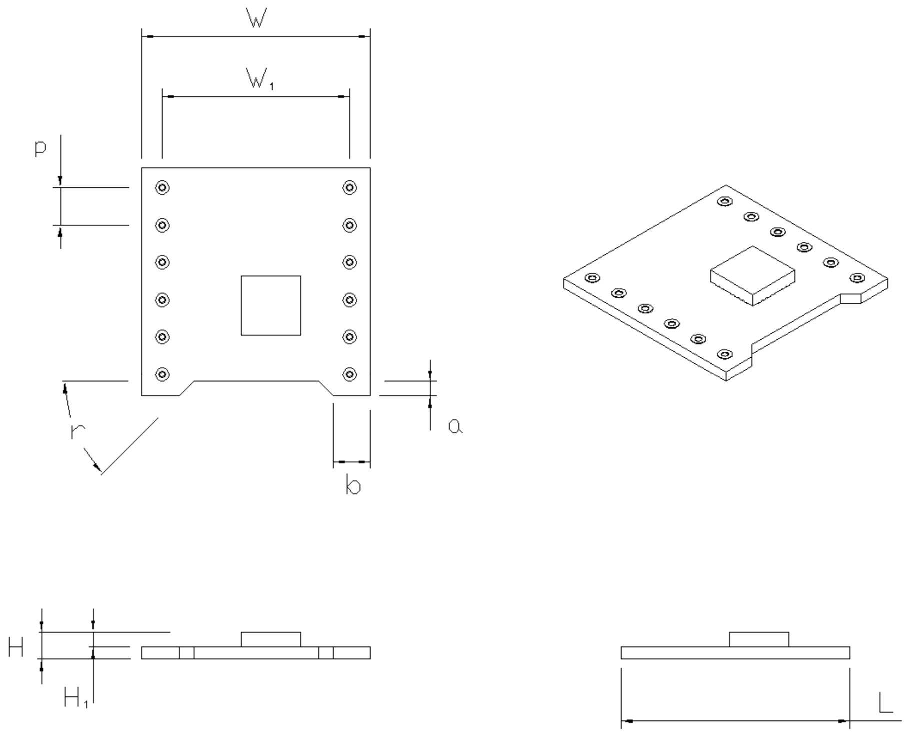
Description	Symbol	Value	Units
Number of Pins	n	**12**	Pins
Pitch	p	**2.54**	mm
Board Width	W	**15.5**	mm
Overall Row Spacing	W_1	**12.7**	mm
Leap Height	a	**1**	mm
Leap Width	b	**2.54**	mm
Leap Chamfer Angle	r	**45**	°
Overall Height	H	**10.8**	mm
Stacking Height MagEnc to MCU	H_1	**2.11**	mm
Stacking Height MCU to Motor Driver	H_2	**4.27**	mm
MagEnc IC Centre from Board Edge	d	**4.55**	mm
Overall Length	L	**15.5**	mm

3.1. MagEnc™



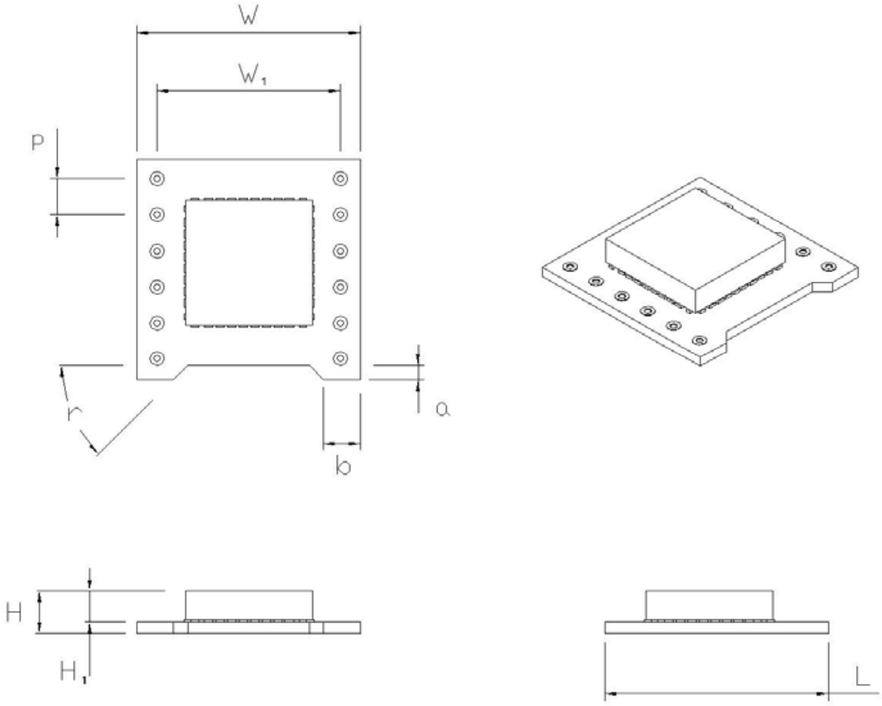
Description	Symbol	Value	Units
Number of Pins	n	**12**	Pins
Pitch	p	**2.54**	mm
Board Width	W	**15.5**	mm
Overall Row Spacing	W_1	**12.7**	mm
Leap Height	a	**1**	mm
Leap Width	b	**2.54**	mm
Leap Chamfer Angle	r	**45**	°
Top to Seating Plane	H	**2.8**	mm
IC Package Height	H_1	**2**	mm
MagEnc IC Centre from Board Edge	d	**4.55**	mm
Overall Length	L	**15.5**	mm

3.2. PicoMcu™



Description	Symbol	Value	Units
Number of Pins	n	**12**	Pins
Pitch	p	**2.54**	mm
Board Width	W	**15.5**	mm
Overall Row Spacing	W_1	**12.7**	mm
Leap Height	a	**1**	mm
Leap Width	b	**2.54**	mm
Leap Chamfer Angle	r	**45**	°
Top to Seating Plane	H	**1.8**	mm
IC Package Height	H_1	**1**	mm
Overall Length	L	**15.5**	mm

3.3. Motor Driver

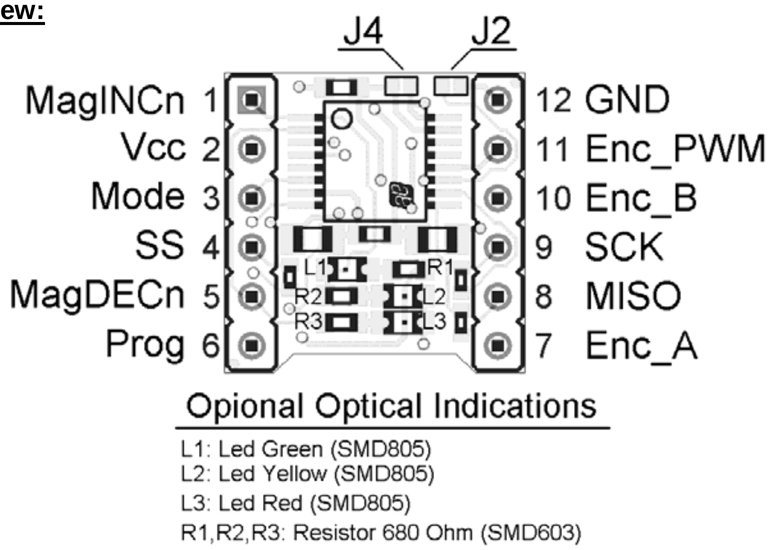


Description	Symbol	Value	Units
Number of Pins	n	**12**	Pins
Pitch	p	**2.54**	mm
Board Width	W	**15.5**	mm
Overall Row Spacing	W_1	**12.7**	mm
Leap Height	a	**1**	mm
Leap Width	b	**2.54**	mm
Leap Chamfer Angle	r	**45**	°
Top to Seating Plane	H	**3**	mm
IC Package Height	H_1	**2.2**	mm
Overall Length	L	**15.5**	mm

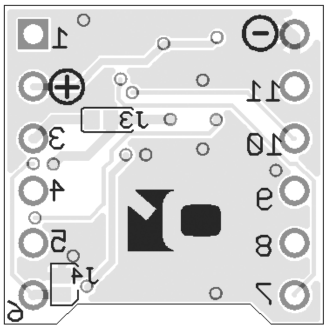
4. Pinout

4.1. MagEnc™

Top View:



Bottom View:

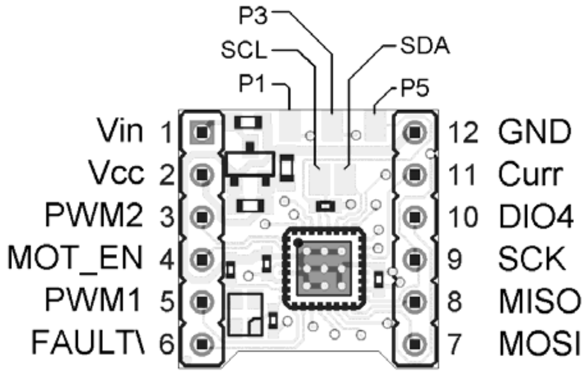


4.1.1. Pin description

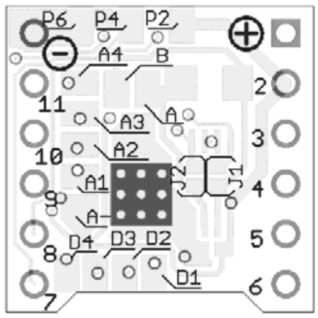
Name	Description
MagINCn	Magnet Field Magnitude INCrease; active low, indicates a distance reduction between the magnet and the device surface. See AS5145 datasheet for more details.
Vcc	Supply voltage. Regulated 5V, 16mA. For 3,3V operation connect solder jumper 4 (J4) prior to applying supply voltage
Mode	Select between slow (open, low: GND) and fast (high) mode. Internal pull-down resistor (~10kΩ). Fast mode can be permanently be selected by connecting solder jumper 3 (J3).
SS	Chip Select, active low; Schmitt-Trigger input, internal pull-up resistor (~50kΩ). Can be set to be permanently low by connecting solder jumper 2 (J2).
MagDECn	Magnet Field Magnitude DECrease; active low, indicates a distance increase between the device and the magnet. See AS5145 datasheet for more details.
Prog	OTP Programming Input and Data Input for Daisy Chain mode. Internal pull-down resistor (~74kΩ). See AS5145 datasheet for more details. Can be permanently low by connecting solder jumper 1 (J1).
Enc_A	Test output in default mode
MISO	Data Output of Synchronous Serial Interface
SCK	Clock Input of Synchronous Serial Interface; Schmitt-Trigger input
Enc_B	Test output in default mode
Enc_PWM	Pulse Width Modulation of approximately. 244Hz; 1µs/step (optional 122Hz; 2µs/step)
GND	Ground.

4.2. PicoMcu™

Top View



Bottom View



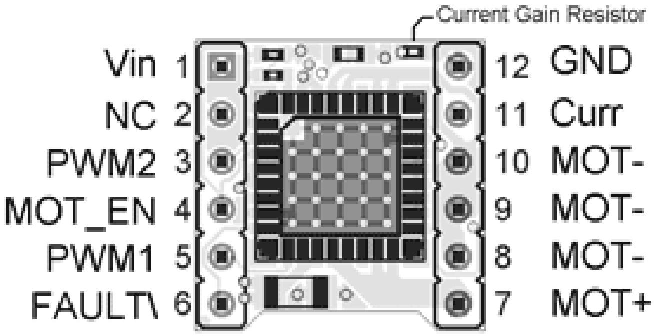
4.2.1. Pin description

Name	Description
Vin	Input voltage. Recommended 5V - 12 V.
Vcc	5V regulated voltage from onboard 200mA low drop-out regulator.
PWM2	PWM signal for Motor Driver PCB. Directly connected to ATMega328p PD3 pin.
MOT_EN	Motor enable for Motor Driver PCB. Directly connected to ATMega328p PD4 pin.
PWM1	PWM signal for Motor Driver PCB. Directly connected to ATMega328p PD5 pin.
FAULT	Active low fault input for Motor Driver PCB. Directly connected to ATMega328p PD6 pin.
MOSI	SPI interface Master Out Slave In data line. Directly connected to ATMega328p PB3 pin.
MISO	SPI interface Master In Slave Out data line. Directly connected to ATMega328p PB5 pin.
SCK	SPI interface clock line. Directly connected to ATMega328p PB6 pin.
DIO4	Digital IO 4. Connected directly to ATMega368P PB2 pin.
Curr	Analog input for current measurement for Motor Driver PCB. Directly connected to ATMega328p ADC7 pin
GND	Ground
SDA	**Interface** — **Functional description** I2C — **SDA**. I2C data

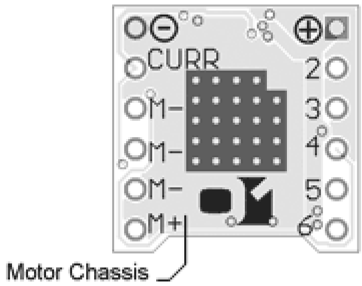
	RC Servo	**PMCMD**. Pulse modulated command, PPM (Position pulse modulation) or VPM (Velocity pulse modulation) according to PMSELECT state
	Analog	**ANCMD**. Analog command input. 2.5V → zero position / velocity.
SCL	**Interface**	**Functional description**
	I2C	**SCL**. I2C clock
	RC Servo	**PMSEL**. Select between PPM and PVM. PPM → Vcc (default), PVM → 0 V
	Analog	**ANSEL**. Select between analog control modes. Analog position control → Vcc (default) Analog velocity control → 0V
A	**Interface**	**Functional description**
	RS-485	**A**. non-inverting RS-485 transceiver channel
	UART	**TXD**. UART transmit data
B	**Interface**	**Functional description**
	RS-485	**B**. inverting RS-485 transceiver channel
	UART	**RXD**. UART receive data
P1	ISP (In System Programming) pin 1. Connected to ATMega368P PB4/MISO pin.	
P2	ISP pin 2. Connected to Vcc.	
P3	ISP pin 3. Connected to ATMega368P PB5/SCK pin.	
P4	ISP pin 4. Connected to ATMega368P PB3/MOSI pin.	
P5	ISP pin 5. Connected to ATMega368P reset pin.	
P6	ISP pin 2. Connected to ground.	
D1	Digital IO 1. Connected directly to ATMega368P PD7 pin.	
D2	Digital IO 2. Connected directly to ATMega368P PB0 pin.	
D3	Digital IO 3. Connected directly to ATMega368P PB1 pin.	
D4	Digital IO 4. Connected directly to ATMega368P PB2 pin.	
A-	Analog ground	
A1	Analog input 1. Connected directly to ATMega368P PC0/ADC0 pin.	
A2	Analog input 2. Connected directly to ATMega328P PC1/ADC1 pin.	
A3	Analog input 3. Connected directly to ATMega328P PC2/ADC2 pin.	
A4	Analog input 4. Connected directly to ATMega328P PC3/ADC3 pin.	

4.3. Motor Driver

Top View



Bottom View



4.3.1. Pin description

Name	Description
Vin	Input voltage. Recommended 5V - 12 V.
NC	No connection.
PWM2	PWM signal from PicoMcu™.
MOT_EN	Motor enable. Active high input. When low motor outputs are in high impedance state. On board pull-down of 4,7 KOhm
PWM1	PWM signal from PicoMcu™.
FAULT	Active low fault output. See MC33887 datasheet for more details.
M+	Motor connection.
M-	Motor connection.
Curr	Analog output proportional to motor current. See MC33887 datasheet for more details.
GND	Ground.
MOT Chassis	Motor chassis. Solder to motor chassis for improved immunity to noise created by the motor.

5. Electrical Characteristics

Absolute Maximum ratings (non operating)

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these or other conditions beyond those indicated in the “Operating Conditions” of this specification is not advised. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

5.1. Absolute Maximum ratings

Parameter	Symbol	Min	Max
Input Voltage	Vin	-0.5 V	14.5 V
Analog-in Voltage	Vain	-0.5 V	Vcc + 0.5 V
Digital-in Voltage	Vdin	-0.5 V	Vcc + 0.5 V
Logic supply voltage	Vcc	-0.5 V	6 V
I2C/RXD/PMSELECT pin voltage	V_{I2C}	-0.5V	Vcc + 0.5 V
RST pin Voltage	V_{RST}	-0.5 V	13 V
RS-485 voltage (transient)	V_{485}	-50 V	50 V
Logic supply current for off board use	Ivcc	0 mA	100 mA
I2C/TXD/PM output current	I_{I2C}	-20 mA	20 mA
Digital out current	Idout	-20 mA	20 mA
RS-485 output current - driver	I_{485D}	-250 mA	250 mA
RS-485 output current - receiver	I_{485R}	-24 mA	24 mA
Operating Temperature	T	-25 ^{o}C	125 ^{o}C
Humidity (non-condensing)	H	-	85.00%

5.2. Operating Conditions

Operating Conditions

Parameter	Symbol	Min	Max
Input Voltage	Vin	5V	12 V
Analog-in Voltage	Vain	0V	Vcc
Digital-in Voltage	Vdin	0V	Vcc
Logic supply voltage	Vcc	5 V	5 V
I2C/RXD/PMSELECT pin voltage	V_{I2C}	0 V	Vcc
RST pin Voltage	V_{RST}	0 V	Vcc
RS-485 voltage	V_{485}	-7 V	12 V
Logic supply current for off-board use	I_{vcc}	0 mA	50 mA
I2C/TXD/PM output current	I_{I2C}	-10 mA	10 mA
Digital out current	I_{dout}	-10 mA	10 mA
RS-485 output current - driver	I_{485}	-60 mA	60 mA
RS-485 output current - receiver	I_{485}	-8 mA	8 mA
Operating Temperature	T	0 ^{o}C	80 ^{o}C

5.3. DC and timing characteristics

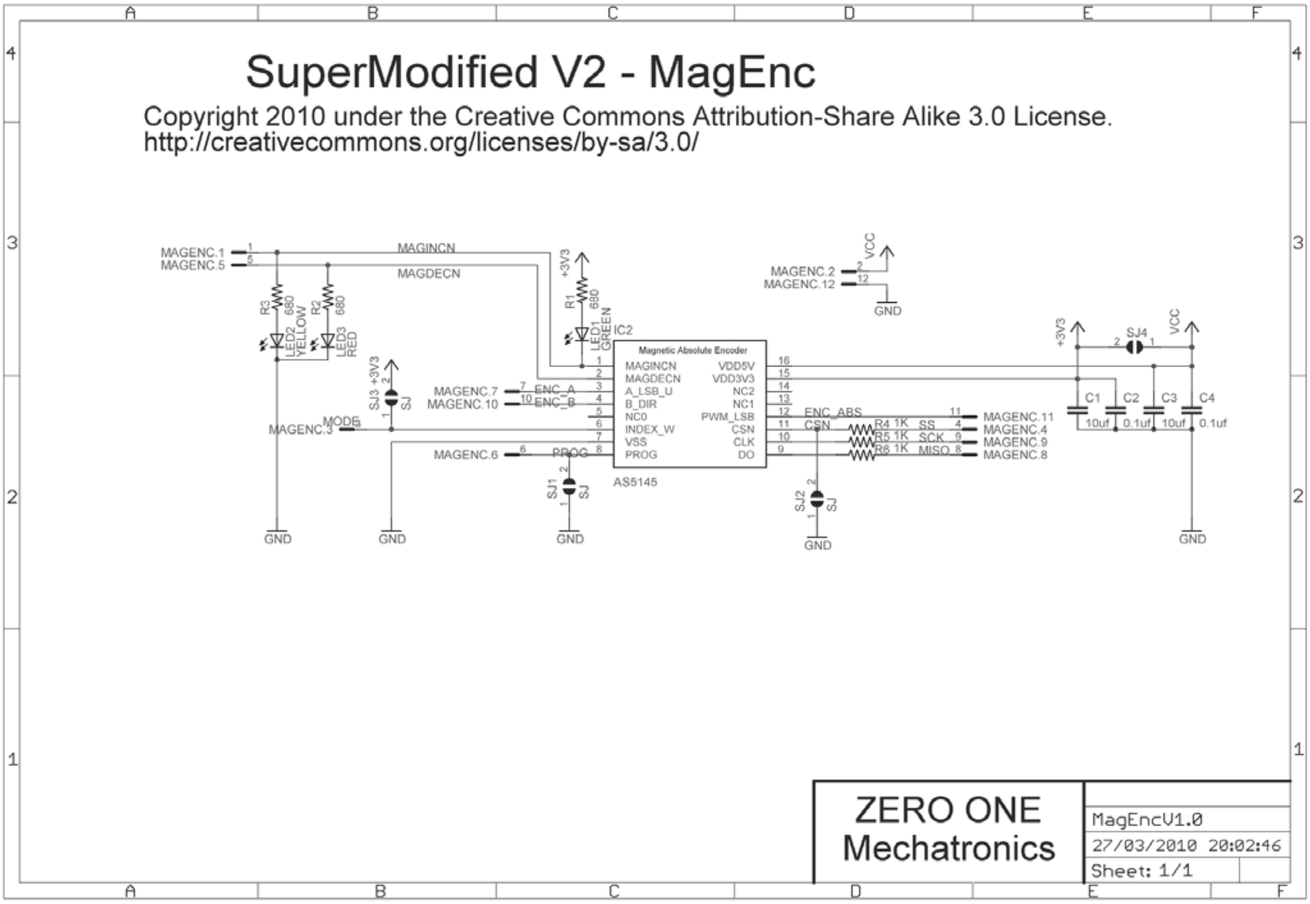
Parameter	Symbol	Typical
Input current, controller only	I_{CONTR}	20 mA
Motor current	I_{MOT}	5 A
Input current total	I_{TOT}	$I_{CONTR} + I_{MOT}$
DIO1-4 output source current	I_{DIOSRC}	5 mA
DIO1-4 output sink current	I_{DIOSNK}	20mA

For additional DC and timing characteristics refer to specific IC datasheet according to below table.

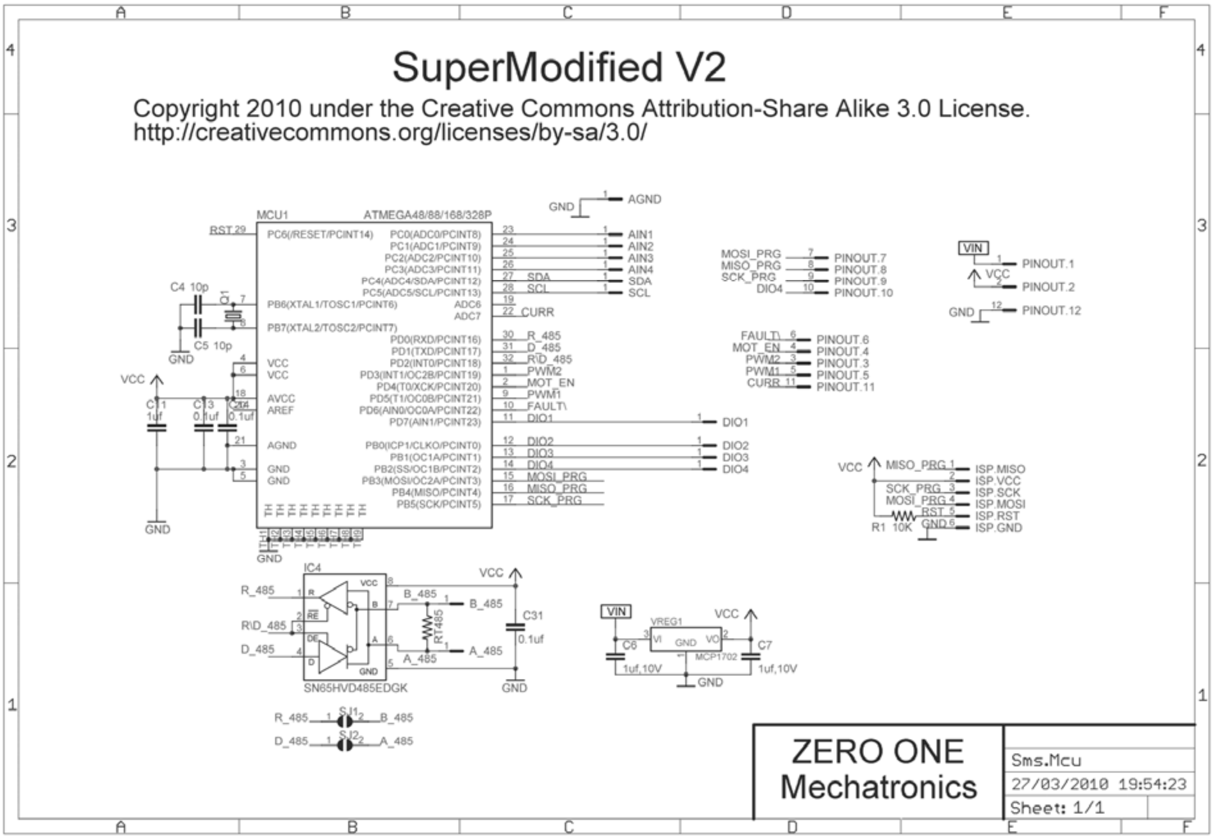
IC	Function	Related pins
ATMega168P	Microcontroller Note: Supermodifed™ on-board crystal at 20MHz	Digital IOs (DIO1-4) Analog IOs (AIN1-4) RST I2C pins (SDA,SCL)
MCP1702	Voltage regulator	VCC, GND
MC33887	Motor driver	MOT+, MOT-, VIN, GND,
SN65HVD485E	RS-485 transceiver	A_485, B_485

6. Electrical Layout

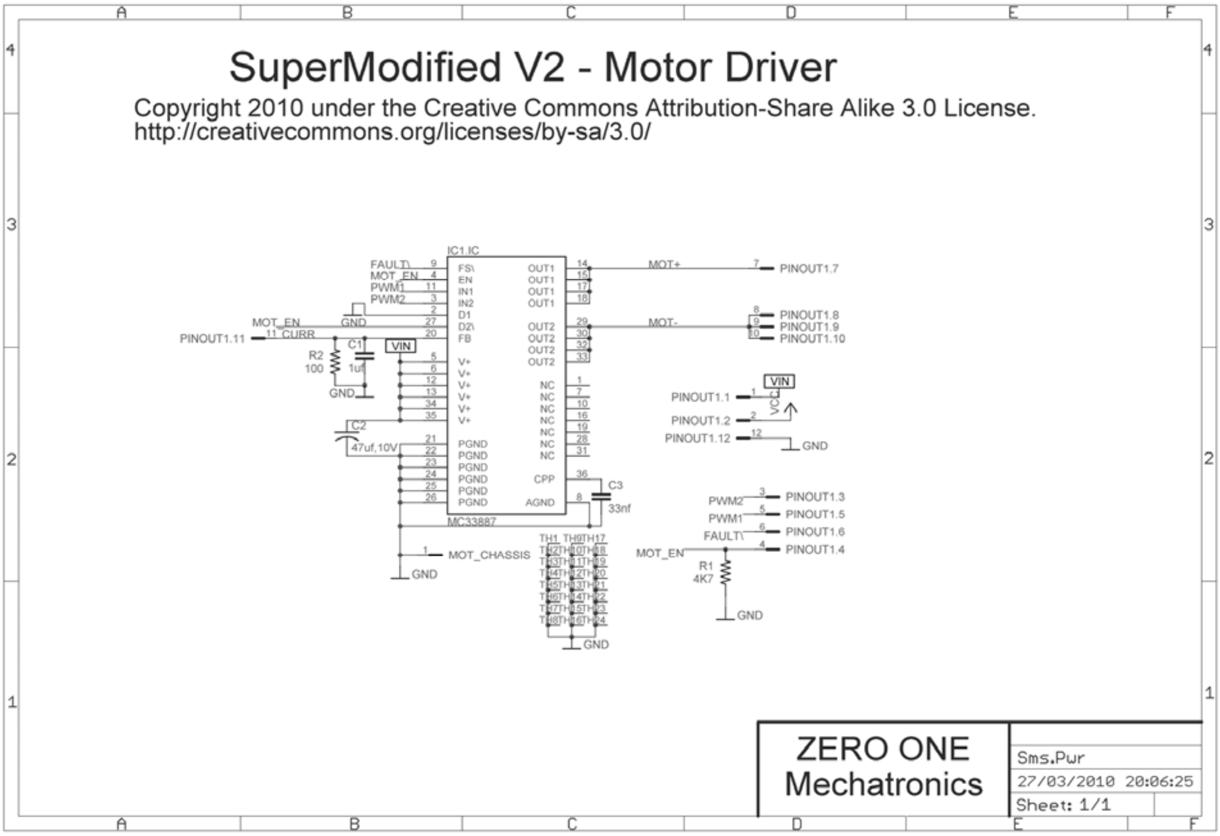
6.1. MagEnc™ Schematic



6.2. PicoMcu™ Schematic

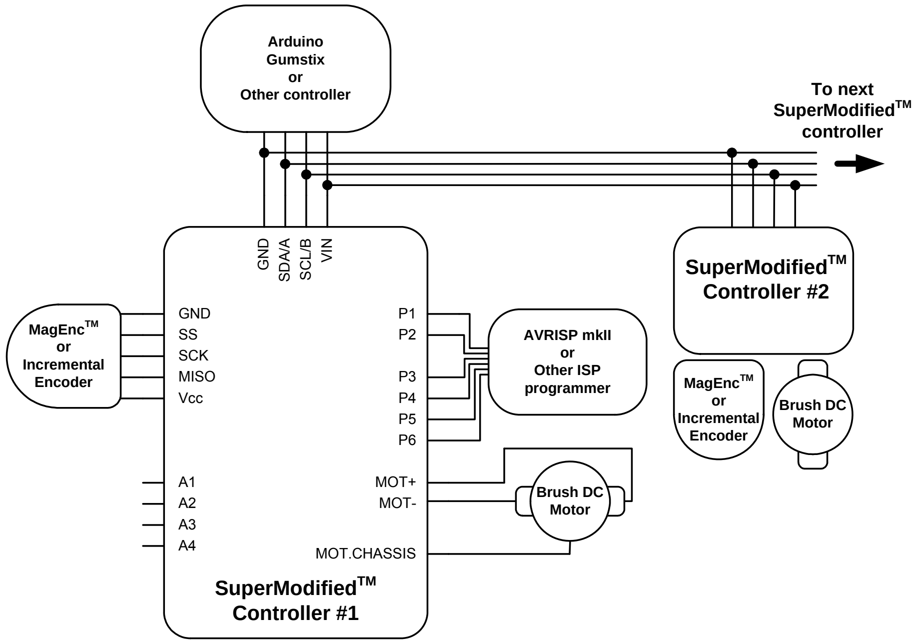


6.3. Motor Driver Schematic



7. Connections

In the following diagram the connections of a fully assembled motor controller are displayed.



Pins	Connect to
VIN **GND**	Regulated DC power supply capable of delivering at least I_{TOT}: 5.02 A.
SDA **SCL** **A** **B**	(see interface table below)
A1-4	External analog voltages 0-5V from sensors etc.
MOT+ **MOT-**	Brush DC motor power leads. If connected with wrong polarity motor control cannot be achieved.

Interface	Supermodifed controller	Host Controller	Note
I2C	SDA = SDA	SDA	Appropriate pull up resistors on host
	SCL = SCL	SCL	
UART	A=TXD	RXD	
	B=RXD	TXD	
RS-485	A=A	A (non inv.)	120 Ohm terminal resistors at host and last controller.
	B=B	B (inverting)	
RC-Servo	SDA=PM	Pulse modulated output	Select between PPM / PVM through the PMSEL pin
	SCL=PMSEL	Digital out	
Analog	SDA=ANCMD	DAC output	DAC output can also be a PWM output with a low-pass filter.
	SCL=ANSEL	Digital out	

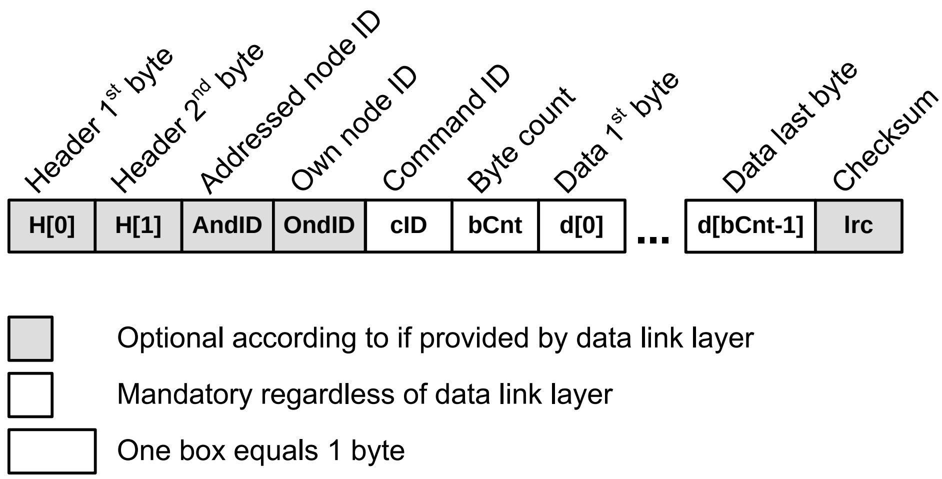
MOT CHASSIS	Motor chassis.
D1-4	External digital IOs. Output 0-5V, 20mA sink current, 5mA source current. Input 0-5V, impedance >10KOhm.
P2	Vcc pin of external ISP (In-System Programmer).
P6	Gnd pin of external ISP.
P4	MOSI pin of external ISP.
P1	MISO pin of external ISP
P3	SCK pin of external ISP.
P5	Reset pin of external ISP.

8. 01Mech Protocol

01 Mechatronics specifies an open-source software layer protocol for use with the Supermodifed™ controller as well as, other 01 Mechatronics products. Subject to the provision of appropriate H/W and minor modifications this protocol can support a multitude of electrical and data-link layers. The general form of the 01Mech protocol is analyzed in this section. Implementation specifics according to data link layers defined by standards are analyzed at the specific data link layers.

8.1. Frame format

The general 01Mech protocol frame format is presented below



The 01Mech protocol specifies two types of data transfers.

- **Send command data transfer:** A bus master commands a bus slave (e.g. a Supermodifed™ controller) to do something.
- **Command response data transfer:** A bus slave responds to the request initiated by a bus master. This data transfer can only be initiated as a result of a previous send command data transfer.

8.1.1. Header

The two header bytes are specified as: **H[0] = 0x55, H[1] = 0xAA.**
The header is used only for data link standards that do not support bus architectures inherently.

8.1.2. Addressed and own node ID

The AndID byte always contains the node ID of the device addressed on the bus. There is an exception when ndID = 0 (reserved for broadcasting).
The OndID byte always contains the node ID of the device that issues the message. There are specific data-link layers (like I2C) which incorporate part or all of the node ID bytes functionality.

For example if a master with ID = 0x01 wants to command a Supermodifed™ controller with node ID = 0x04 to do something then: OndID = 0x01 and AndID = 0x04. When the Supermodifed™ controller responds with a command response then OndID = 0x04 and AndID = 0x01.

8.1.3. Command ID & classification

The 01Mech protocol specifies three types of commands according to their bus master related functionality.

- **Set commands:** A bus master commands a bus slave to do something: e.g. move the motor connected to a Supermodifed™ controller with a constant velocity. These types of commands involve sending data to the slave.
- **Get commands:** A bus master requests data from a bus slave: e.g. a master requests the current motor velocity from a Supermodifed™ controller.
- **Broadcast set commands:** A bus master commands all slaves on the bus to do something: e.g. a master commands all Supermodifed™ controllers attached to the bus to halt.

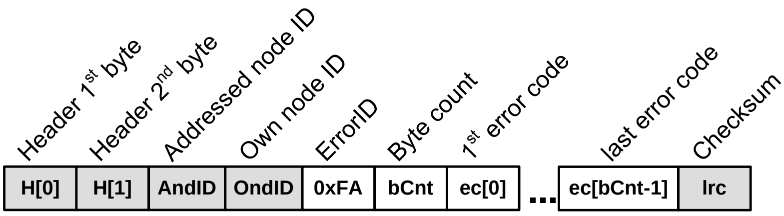
According to command classification, the 01Mech protocol specifies command ID ranges. The table below summarizes the latter statement.

Command Type	ID Range low limit	ID Range high limit
Set commands	0	99
Get commands	100	199
Broadcast set commands	200	249
reserved	250	255

Thus a set command ID can exist in the decimal range of 0-99, a get command ID in the range of 100-199, and a broadcast set command ID in the range of 200-249. Commands are device-specific and an accompanying document specifying the command set of the device must be provided by the manufacturer.

8.1.4. Error frames

One of the reserved command IDs, 250 = 0xFA is used for a special command specified by the 01Mech protocol: the error command. If one or more errors are present in the Supermodifed™ controller they will be reported using this special command ID as illustrated in the below error frame schematic.
For the error list codes please refer to section 16.



8.1.5. Byte Count

The bCnt is specified as the number of bytes the data argument of a command consists of. Note that *only* data bytes are counted.

8.1.6. Data

A command can contain from zero and up-to 255 bytes of data.

8.1.7. Checksum

The checksum is an LRC (Longitudinal Redundancy Check) checksum. It is calculated for the mandatory bytes of the data frame i.e. command ID, byte count, data[0] to data[byte count -1]. The calculation of an LRC checksum are simple and straightforward: An exclusive or is applied on the packet bytes consecutively. A C implementation is given below:

```c
typedef unsigned char u08;

u08 zoProtocolLRC(u08 *lrcBytes, u08 lrcByteCount)
{
	u08 i;
	u08 lrc = 0;

	for( i=0; i<lrcByteCount; i++)
		lrc ^= lrcBytes[i];

   return lrc;
}
```

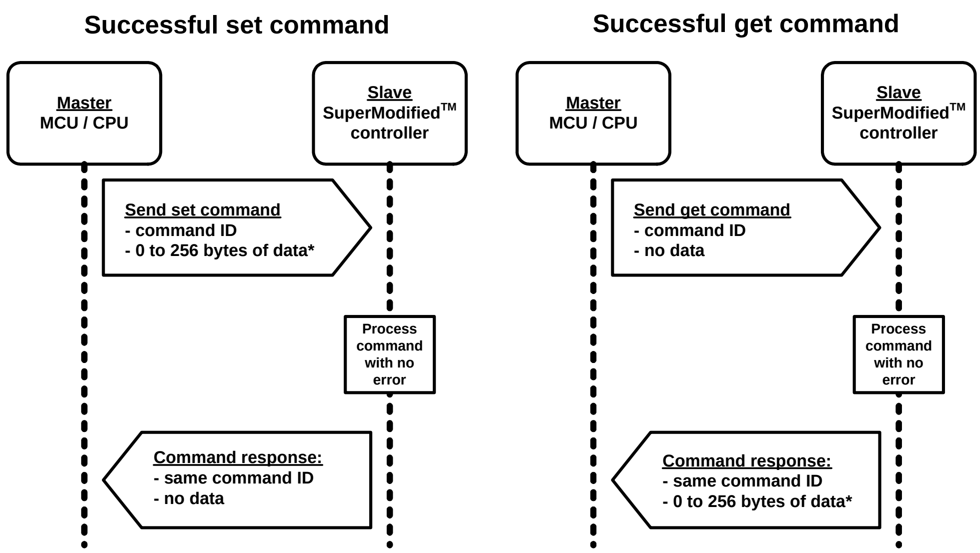
In some data-link standards the checksum functionality is included (e.g. CAN bus). When the 01Mech protocol is implemented over such data link standards the checksum bytes are omitted.

8.2. Transmission byte order

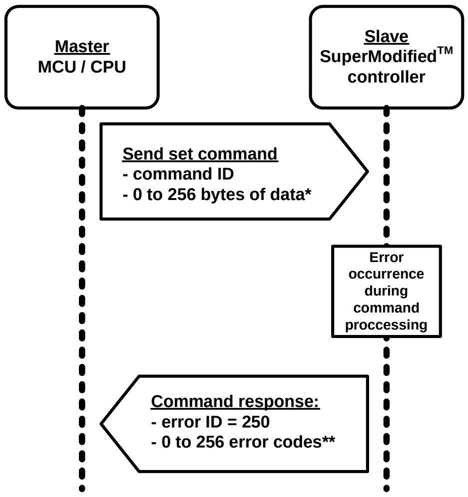
Data are transferred lowest byte first. Refer to general frame format at section 7.1

8.3. Control flow & errors

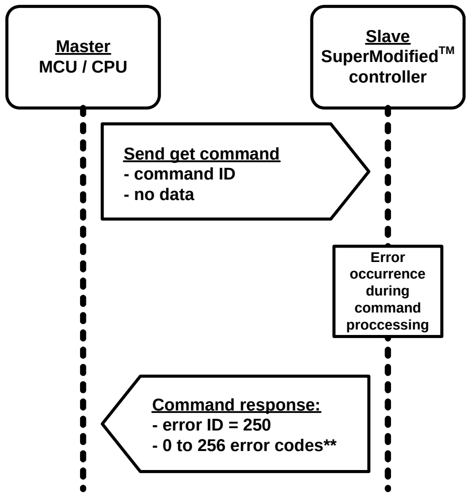
According to the 01Mech protocol specification the following scenarios of communication flow control can exist.



Error during set command

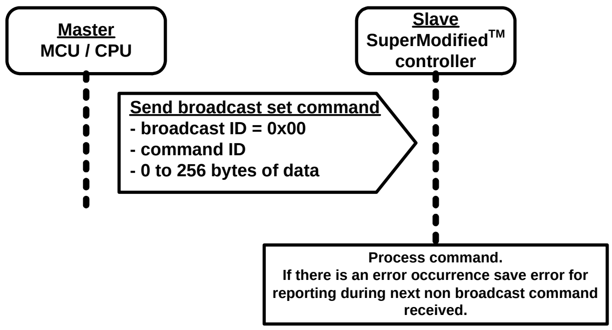


Error during get command



*According to device command set. ** According to device error set.

Broadcast command



The slave error reporting mechanism is introduced in the above flow charts. Errors can occur on a bus slave in a synchronous or an asynchronous manner. In the case of the Supermodifed™ controller a synchronous error can be the reception of an erroneous checksum where an asynchronous error can be the detection of a motor stall condition.

8.3.1. Communication flow examples

The 01Mech protocol specifies that the error set of a slave device, similarly to the command set is device specific and thus provided by the device manufacturer. The error set of the Supermodifed™ controller is specified in section 15 of this document.

Examples of communication scenarios according to the 01Mech protocol:

Example 1: *Master (own ID = 0x01) sends a ‘set PID gain P’ (set command with 2 bytes of data → 16-bit value) command to the Supermodifed controller with ID=0x04, through UART interface and gets a valid command response:*

1.The bus master issues the command on UART:
H[0] = 0x55, H[1] = 0xAA , AndID = 0x04, OndID = 0x01, cID = set PID gain P command ID, bCnt = 2, d[0] = gain P low byte, d[1] = gain P high byte, lrc

2. The Supermodifed™ controller on the bus with ID 0x04 accepts the command, executes it and issues a command response (other slaves realize that they are not addresses by the node ID value):
H[0]=0x55, H[1] = 0xAA, AndID = 0x01, OndID = 0x04, cID = set PID gain P command ID, bCnt = 0, lrc

Example 2: *Master sends a '*set PID gain P' *command to the Supermodifed controller with ID #4, through I2C interface and gets a valid command response:*

1. The bus master issues the command on I2C:
OndID = 0x01, cID = set PID gain P ID, bCnt = 2, d[0] = gain P low byte, d[1] = gain P high byte, lrc
Note that the header is omitted altogether. The addressed node ID is incorporated in the I2C slave address

2. The Supermodifed™ controller on the bus with ID 0x04 (i.e. slave address) accepts the command and issues a command response:
OndID = 0x04, cID = set PID gain P, bCnt = 0, lrc

Example 3: *Master sends a 'start' command (set command with 0 bytes of data) to the Supermodifed controller with ID #5, through UART interface and gets a valid command response:*

1.The bus master issues the command on UART:
H[0] = 0x55, H[1] = 0xAA , AndID = 0x05, OndID = 0x01, cID = start command ID, bCnt = 0, lrc

2. The Supermodifed™ controller on the bus with ID 0x05 accepts the command and issues a command response:
H[0]=0x55, H[1] = 0xAA, AndID = 0x01, OndID = 0x05, cID = start command ID, bCnt = 0, lrc

Example 4: *Master sends a 'get position' command (get command expecting a 32 bit signed answer → 4 bytes) to the Supermodifed controller with ID #4, through UART interface and gets a valid command response:*

1.The bus master issues the command on UART:
H[0] = 0x55, H[1] = 0xAA , AndID = 0x04, OndID = 0x01, cID = get position command ID, bCnt = 0, lrc

2. The Supermodifed™ controller on the bus with ID 0x04 accepts the command and issues a command response :
H[0]=0x55, H[1] = 0xAA, AndID = 0x01, OndID = 0x04, cID = get position command ID, bCnt = 4, d[0] = position[0] (LSB), d[1] = position[1], d[2] = position[2], d[3] = position[3] (MSB), lrc

Example 5: *Master sends a '*set PID gain P' *command (with wrong lrc) to the Supermodifed controller with ID #4, through I2C interface and gets an error command response:*

1. The bus master issues the command on I2C:
OndID = 0x01, cID = set PID gain P ID, bCnt = 2, d[0] = gain P low byte, d[1] = gain P high byte, lrc (but the master sends the wrong lrc !)

2. The Supermodifed™ controller on the bus with ID 0x04 identifies the wrong lrc, does not execute the command and issues an error response:
OndID = 0x04, cID = 250 (error ID), bCnt = 1,d[0] = wrong lrc error code, lrc

Example 6: *Master sends a '*set PID gain P' *command to the Supermodifed controller with ID #4, through I2C interface and gets an error command response:*

1. The bus master issues the command on I2C:
OndID = 0x01, cID = set PID gain P ID, bCnt = 2, d[0] = gain P low byte, d[1] = gain P high byte, lrc

2. The Supermodifed™ controller on the bus with ID 0x04 in the meantime has detected a motor stalled condition and an over-current condition, so it does not execute the command and issues an error response:
OndID = 0x04, cID = 250 (error ID), bCnt = 2, d[0] = motor stalled error code, d[0]

= over-current error code, lrc

Example 7: *Master sends a 'broadcast stop' command (set command with 0 bytes of data) to the all Supermodifed controllers on the bus, through UART interface:*

1.The bus master issues the command on UART:
H[0] = 0x55, H[1] = 0xAA , AndID = 0x00 (broadcast ID), OndID = 0x01, cID = broadcast stop command ID, bCnt = 0, lrc

2. The Supermodifed™ controllers on the bus with any ID accept the command and do not issue a command response:

8.3.2. Recommended master operation

The bus master should wait on the slave response with a timeout – in case the slave does not respond. Source code for the implementation of a master according to the 01Mech protocol can be found on the 01 Mechatronics Supermodifed™ Google Code page. Check this resource regularly as code for various platforms is continuously under development.

9. Configuring mode of interface

The Supermodifed™ controller comes pre-programmed from 01 Mechatronics according to ordered interface mode. Change of the mode of interface can currently be done only by reprogramming the Supermodifed™ controller, according to desired interface. The source code of the Supermodifed™ is distributed under the GNU General public license V3.0 and is available for download from our Supermodifed™ Google Code page: http://code.google.com/p/zoSupermodifed/

9.1. Programming the controller

Detailed instructions on how to program the controller through the In-System Programming interface can be found in the "Atmel AVR programming guide" document, written by 01 Mechatronics and available for download from our zoAvrLib and zoSupermodifed Google Code pages.

10. I2C Interface

The Supermodifed™ controller I2C interface uses the I2C specification standard to exchange data with other devices connected on the same I2C bus.

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006. I2C is not only used on single boards, but also to connect components which are linked via cable. Simplicity and flexibility are key characteristics that make this bus attractive to many applications. The latest I2C specification is available directly from NXP.

For specific information on the Supermodifed™ I2C hardware please consult the ATMega328p datasheet.

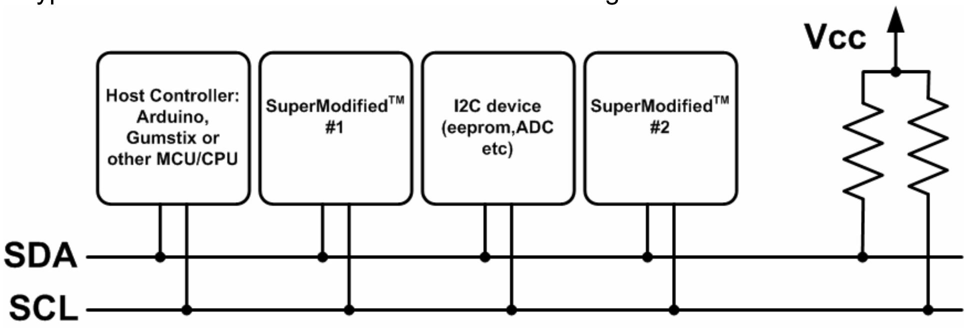
10.1. Electrical layer

The I2C bus allows communication between devices residing on the same I2C bus by the means of two bi-directional lines: SDA for data and SCL for clock. The standard utilizes bipolar logic. A logic 1 is represented as Vcc Volts and logic 0 is represented as 0 Volts on the SDA line. The SCL line is used to clock the data transfer.

An I2C bus needs pull up resistors on the SDA and SCL lines for correct operation. For a logic high an I2C device tri-states the corresponding I2C pin, thus allowing the line to be pulled to Vcc by the pull up resistor. For a logic low an I2C device pulls down the line to 0 Volts using internal open-collector circuitry.

10.1.1. Connections

A typical electrical interconnection of an I2C bus is given below.



10.1.2. Bus speeds

The following two bus speed modes are supported by the Supermodifed™ controller.

- **Standard-mode (Sm)**, with a bit rate up to 100 kbit/s
- **Fast-mode (Fm)**, with a bit rate up to 400 kbit/s

10.1.3. Pull up resistor sizing

According to the system Vcc the pull up resistors can be calculated by the following formulae

100KHz Operation

Minimum pull-up: $Rp_{\min} = \frac{Vcc - 0.4V}{3mA}$, Maximum pull-up $Rp_{\max} = \frac{1000ns}{C_b}$

C_b : Capacitance of one bus line in pF.

400KHz Operation

Minimum pull-up: $Rp_{\min} = \frac{Vcc - 0.4V}{3mA}$, Maximum pull-up $Rp_{\max} = \frac{300ns}{C_b}$

Recommended values for 5V systems : 2.2 kOhm
Recommended values for 3.3V systems : 1,5 kOhm

10.1.4. Considerations

The I2C standard dictates that bus capacitance for each line should be kept below 10pF. Bus capacitance is very important for correct bus operation. For the wiring of the SDA and SCL lines a twisted-pair configuration is recommended to ensure minimum bus capacitance.
Note that bus capacitance is also influenced by the length of the wires, so they should be kept as short as possible. If the overall length of an I2C bus induces a higher bus capacitance than 10 pF correct operation cannot be ensured. In that case an I2C repeater is recommended.
The Supermodifed™ controller can overcome such limitations utilizing the RS-485 bus which allows for great distances between bus nodes. As a rule of thumb I2C is adequate for bus lengths of up to 1m and it should be avoided without the use of repeaters for longer buses.

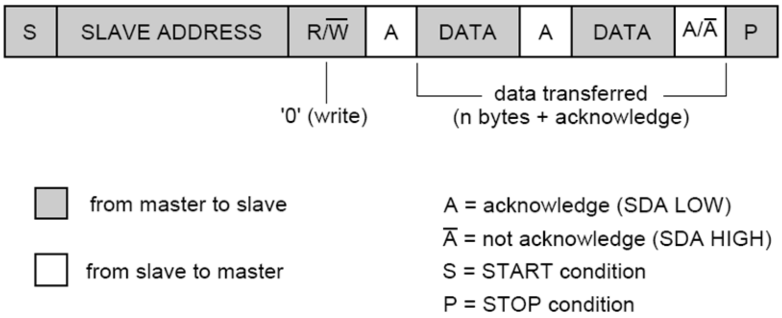
10.2. Data link layer

Devices on the I2C bus are distinguished according to their role on the bus as masters or slaves. The I2C standard specifies three basic types of messages:

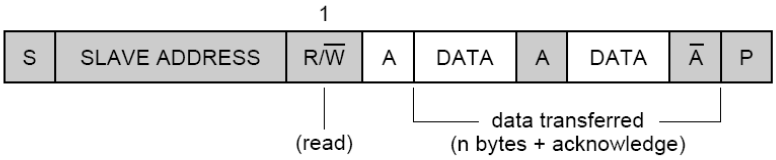
- Single message where a master writes data to a slave;
- Single message where a master reads data from a slave;
- Combined messages, where a master issues at least two reads and/or writes to one or more slaves.

Below the two most essential types of messages are presented.

Master transmit message



Master receive message



10.2.1. Supermodifed slave addresses

SupermodifedTM controllers come preprogrammed with a slave address of 0x04. If more than one SupermodifedTM controller is ordered slave addresses increase by 1 for every additional controller.
e.g.
SupermodifedTM controller #1 -> slave address 0x04
SupermodifedTM controller #2 -> slave address 0x05
And so on.
SupermodifedTM slave addresses can be also adjusted at customer request or even re-configured by the end-user by reprogramming the device. For details refer to our “Atmel AVR programming guide”.

10.2.2. General call address

The I2C bus specification includes some special addresses. One of them utilized by the SupermodifedTM controller is the general call address.

The general call address is for addressing every device connected to the I2C-bus at the same time. An I2C bus master can initiate a general call, transmitting the same data to all I2C devices on the bus.

General call address :

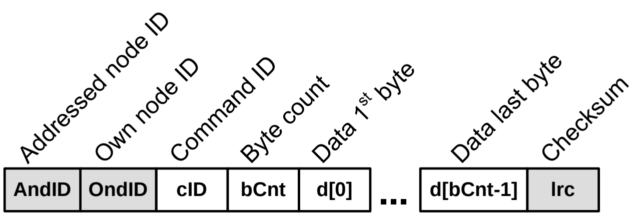
Slave address	R/$\overline{W}$ bit
0000 000	0

10.3. 01Mech Protocol over I2C

Implementation specifics for the 01Mech protocol over I2C are given below.

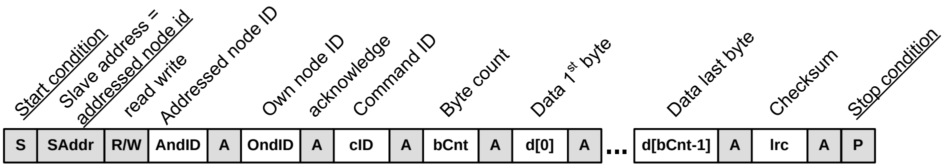
10.3.1. Frame structure

The frame structure of the 01Mech protocol over I2C does not include header bytes as their functionality is incorporated in the I2C data link layer. Thus the frame structure of the 01Mech protocol over I2C is the one shown below.



One box equals 1 byte

When examined in combination with the I2C standard the 01Mech protocol frame on the I2C bus is depicted below.



I2C standard specific

01Mech protocol specific

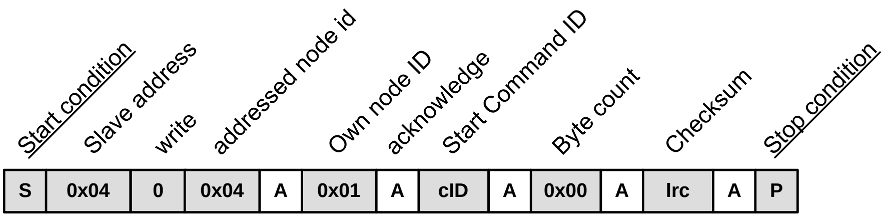
10.3.2. Communication flow control

For the implementation of the 01Mech protocol over I2C only the Master Transmit messages are used (as they are specified by the I2C standard). The host controller and the Supermodifed™ controller take turns at being masters on the I2C bus. A transaction is always initiated by the host controller.

10.3.3. Communication example over I2C

Example scenario: *The host controller with ID = 0x01 sends a 'start command' to the SupermodifedTM controller with node ID = 0x04. Note: The host controllers I2C address should be set according to its node ID thus 0x01.*

1. The host controller sends the following master transmit message.
(I2C bus representation)

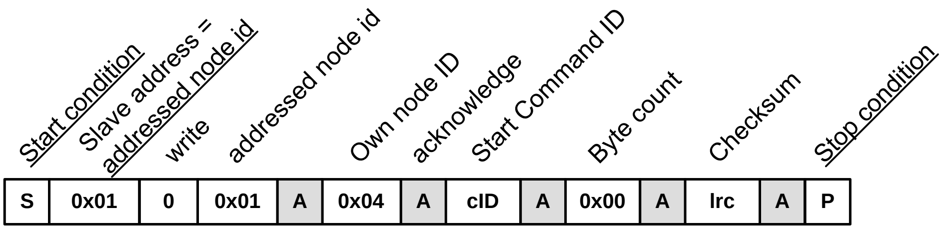


From host controller to SuperModified™ controller

From SuperModified™ controller to host controller

The host controller then waits (with a timeout) for the Supermodifed™ to respond on the I2C bus.

2. The Supermodifed controller accepts the command and issues a command response using again an I2C master transmit message. The latter is this:



From host controller to SuperModified™ controller

From SuperModified™ controller to host controller

In a similar manner all I2C transactions are carried out. Note that the Supermodifed™ controller will never initiate a master transmit unless a command is received.

11. UART, RS-485 interfaces

The UART (Universal Asynchronous Receiver Transmitter) is not a communication standard but rather a piece of hardware. However it is the main workhorse behind EIA-232, EIA-422 and EIA-485 electrical standards (previously named RS-232 and so on). The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART reassembles the bits into complete bytes. Three lines are used: TXD (used to transmit data), RXD (used to receive data) and GND (provides a common voltage reference between the two communicating systems). Note that naming of the UART pins/lines is usually host system related.

A UART chip (or a UART module integrated in a microcontroller) is usually externally connected to an electrical-standard-compliant transceiver IC. However it can also be used as it is. Of course utilizing the appropriate transceiver has many advantages (capability of serial communications over longer distances, improved noise immunity etc.).

For more information on the UART utilized by the SupermodifedTM controller please refer to the ATMega168p datasheet at the USART0 section.

11.1. Electrical Layer

Electrical layer specifics for the RS-485 interface and the UART interface as they are utilized for the Supermodifed™ controller:

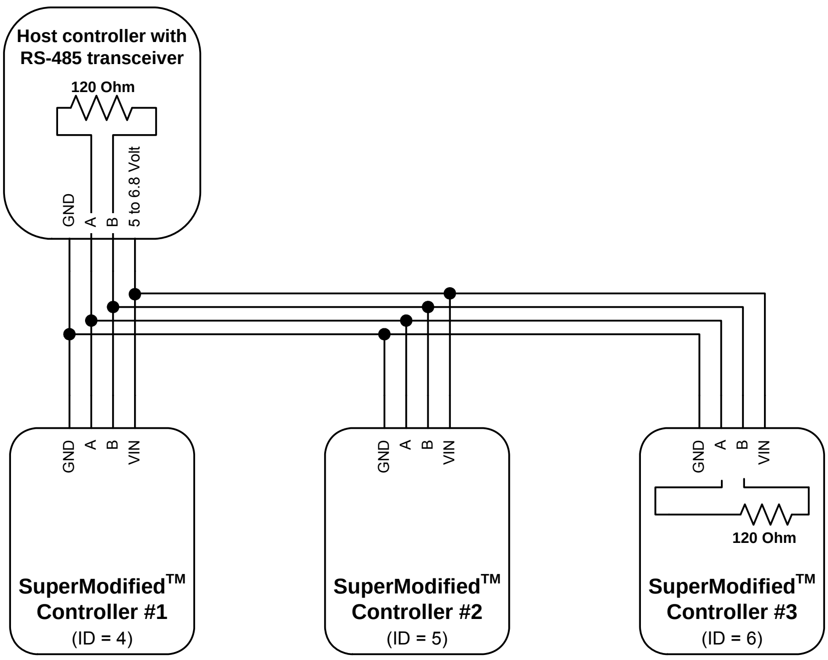
11.1.1. RS-485 specification

RS-485 electrical layer specifics

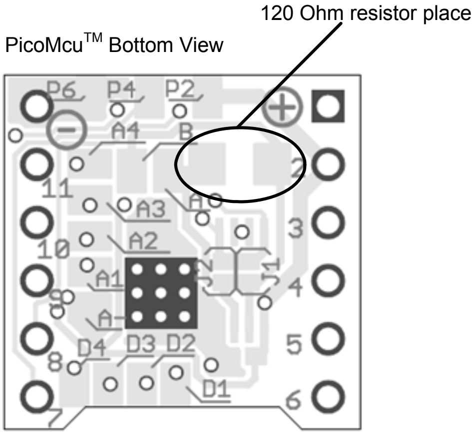
Physical Media :	Twisted Pair
Network Topology :	Point-to-point, Multi-dropped, Multi-point
Maximum Distance :	1200 meters
Mode of Operation :	Differential
Voltage Levels :	+/-6V (Commonly used)
Mark(1) :	Negative Voltages
Space(0) :	Positive voltages
Signals (half-duplex):	A = non inverting Tx/Rx B = inverting Tx/Rx

11.1.2. RS-485 Connections

When one or more SupermodifedTM controllers are connected on a RS-485 bus, connections should be as illustrated below.



The SupermodifedTM controller has a place for an 120 Ohm (SMD1206) RS-485 terminal resistor. However it is by default not soldered in place across the A and B RS-485 terminals. This resistor can be soldered according to customer requests by 01 Mechatronics. It can also be soldered in place by the user. The place of this resistor is illustrated below.



11.1.3. UART specification

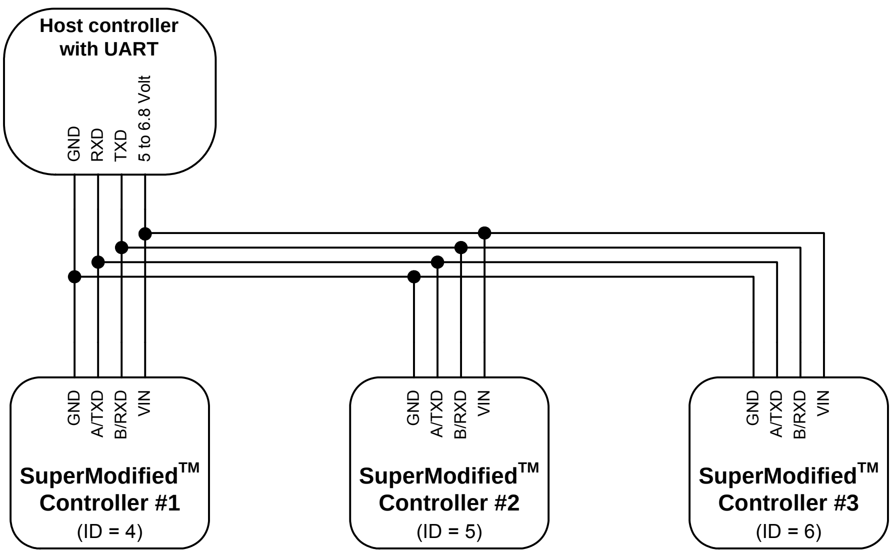
UART electrical layer specifics (not part of any standard)

Physical Media :	3 wires (twisting recommended)
Network Topology :	Point-to-point, Single master multi point.
Maximum Distance :	~1 meter
Mode of Operation :	Single ended
Voltage Levels :	0-Vcc (Vcc = 5Volts or 3.3 Volts)
Mark(1) :	0 Volts
Space(0) :	Vcc
Signals:	TXD = transmit RXD = receive GND = common voltage reference between connected systems

Note that the UART interface pins can be connected to an external RS-232 transceiver thus making the SupermodifedTM controller RS-232 compliant.

11.1.4. UART connections

When using the UART interface of the Supermodifed™ controller connections should be as illustrated below:



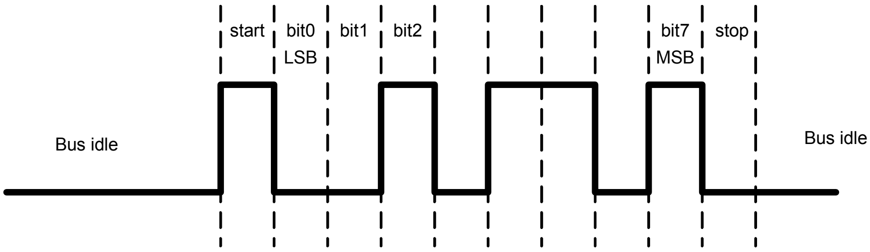
Note that the same connections can be used if an external RS-232 transceiver is used with every Supermodifed™ controller.

11.1.5. Considerations

The number of devices that can be connected as illustrated above depends on the host controller UART fan-out. The fan-out is simply the number of inputs that can be connected to an output before the current required by the inputs exceeds the current that can be delivered by the output while still maintaining correct logic levels. Refer to the UART manufacturer datasheet for current capability of the TXD pin and to the ATMega328p datasheet for the current input needs of pin RXD of the Supermodifed controller.

11.2. Data link layer

The data link layer used for both UART and RS-485 data transfers is identical. The same applies in the case of an external RS232 transceiver. A frame on any of these layers is illustrated below:



Henceforth by referring to UART a reference to RS485 (and RS232) interfaces are implied.

11.3. 01Mech Protocol over UART

The UART implementation of the 01Mech protocol complies fully to the general form presented in section 7. The host controller is the only master of the UART bus and the only device that can initiate a transaction.

12. RC Servo Interface

The Supermodifed™ controller offers compatibility with the very popular RC Servo Position Pulse Modulation interface.

Furthermore the SCL pin can be used to select between position control and velocity control.

SCL connected to Vcc or left floating → Position control.
SCL connected to ground → Velocity control.

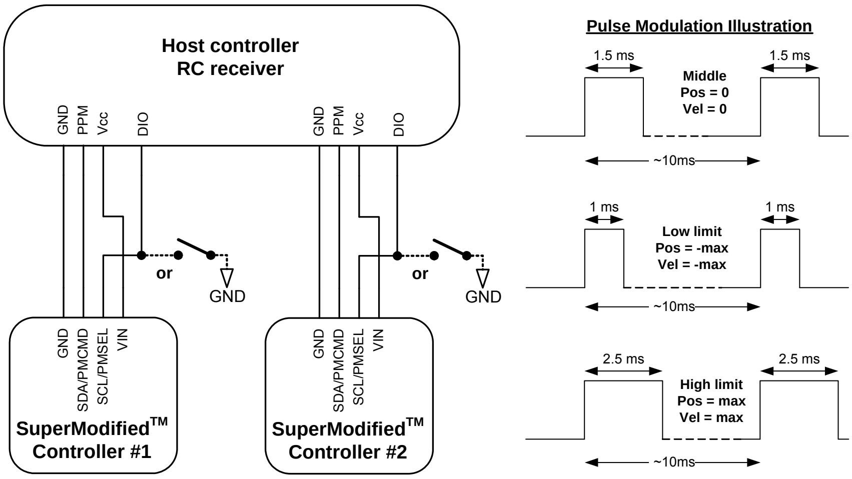
Limit positions that correspond to 1ms and 2ms pulses are configured by default to ± 180 degrees. However they can be configured by 01 Mechatronics upon customer request.

Also by default velocities are limited ± 1000 ticks/second. Again they can be changed according to customer's request.

Note that the above interface option implies the use of 01 Mechatronics MagEnc™ encoder together with the Supermodifed™ controller.

12.1. Connections

The Supermodifed™ controller, when configured for RC-Servo interface, should be connected following the principles illustrated below:



12.2. RC-Servo Interface limitations and capabilities

The RC-Servo interface does not support all available commands of the Supermodifed™ controller. i.e. no feedback on position or velocity etc.
However it provides full motion capabilities in a straight-forward way. Supermodifed™ servos can be pre-configured by 01 Mechatronics to have custom position or velocity limits, to execute profiled position with custom accelerations and velocities etc. Please contact us for more details. The analog programming scheme which is presented below can also be used to adjust these settings to some extend.

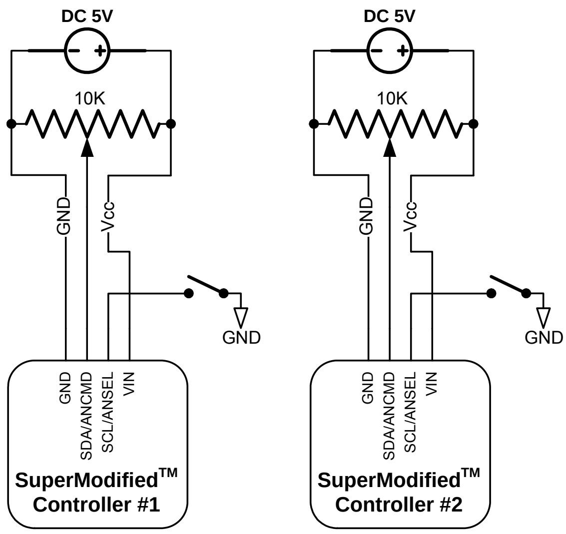
13. Analog Interface

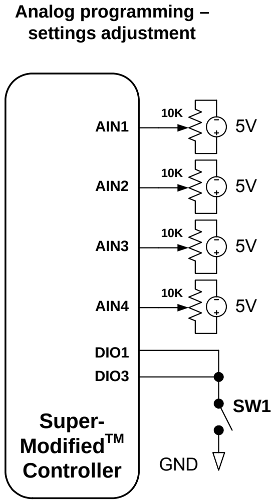
Similarly to the RC-Servo interface the analog interface is a simple and straight-forward way to control the Supermodifed™.
The SCL pad is again used as a digital input and selects between position or velocity control.
SCL = Vcc or left floating → position control
SCL = 0 Volts → velocity control

SDA pin is configured as an analog input and accepts the analog command.
0 Volts → -maximum position / -maximum velocity
2.5 Volts → center position / zero velocity
5 Volts → maximum position/ maximum velocity

Default limit positions and velocities, interface capabilities and limitations are identical with the RC servo interface.

13.1. Connections





Some key settings of the Supermodifed™ controller can be adjusted, to some extend, using the analog programming mode illustrated above.

The Supermodifed™ controller must be set up with the connections shown. Digital I/Os 1 and 3 must have the capability to be easily connected to GND. A means for providing 0-5Volts analog voltages to AIN1-4 is also mandatory. Multi-turn, 10KOhm potentiometers are recommended. The command interface (analog or RC) must also be connected and ready to command the controller.

13.2. Analog programming mode

The procedure for analog programming is the following:

- Power-up the Supermodifed™ controller
- To initiate the analog programming mode pull DIO1 and DIO3 to ground for more than 3 seconds.
- Release the ground connection and allow the pins to be pulled to Vcc by the internal pull-up resistors.
- Adjust settings by modifying the AIN1-4 voltages according to your preferences. Evaluate the modifications by commanding the controller to do position/velocity movements.
- When finished pull DIO1 and DIO3 to ground for more than 3 seconds.
- The settings are written to internal EEPROM.
- The controller now behaves as according to your spefication.

If you do not wish to complete the adjustment just disconnect the controller from the power supply before completing the process.

Default settings can always be restored by the following procedure:

- Power up the controller.
- Pull DIO2 and DIO4 to ground for more than 3 seconds.

For all settings this rule applies:
2.5V → default value, 0V→100% decrease, 5V→ 100% increase.

<u>Correspondence of AIN input to setting</u>

AN1: Max position/velocity
AN2: Min position/velocity
AN3: Acceleration/deceleration of profiled positioning
AN4: Velocity of profiled positioning

14. ModBus over RS-485 Interface

Currently under development.

15. Command set

Commands are presented in ascending order according to Command ID. Tx and Rx notations are host related ie Tx = transmitted by the host controller, Rx = received by the host controller. The Supermodifed™ controller is assumed to have a default node ID value = 0x04 and the host controller is assumed to have a node ID value = 0x01. The standard frame for UART interface contains two header bytes H[0]=0x55,H[1]=0xAA. These are not used for the I2C interface.

15.1. Position nomenclature

When controlling a motor, a controller must keep track of its rotary position. In order to achieve this with an incremental encoder the motor position is considered to be 0 at some point (e.g. at power up or after a homing sequence) and all consecutive positions are calculated relative to this zero using the incremental encoder. This is a 'kind' of absolute position. Henceforth this position will be called **Incremental Absolute Position**. The range of this position is software related according to the registers assigned to it. For the Supermodifed™ controller it can be -2147483648 to +2147483648 ticks (32 bit signed integer).

The Supermodifed™ controller when interfaced with the MagEnc™ absolute encoder can get incremental and absolute readings for rotary position. The position acquired by the Supermodifed™ controller through the MagEnc™ absolute interface will be referred to as **Absolute Position**. Note that this position can have values 0-4096 ticks (which is the MagEnc™ resolution).

Finally there is the Relative Incremental Position which is calculated in respect to current position with information provided by the Incremental Absolute Position. We will refer to this position simply as **Relative Position**.

15.2. Set commands

15.2.1. Set PID gain P

Command ID:	0x00	Name:	Set PID gain P
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-
Description:	Sets the Proportional gain of the internal PID control loop.		
Notes:	Stored in EEPROM. Default value: TBD		

Example: Set gain P = 2000 (= 0x07D0)

	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
Host frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x00**	**0x02**	**0xD0**	**0x07**	**0xD5**

	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
Slave frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x00**	**0x00**	**0x00**

15.2.2. Set PID gain I

Command ID:	0x01	Name:	Set PID gain I
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-
Description:	Sets the Integral gain of the internal PID control loop.		
Notes:	Stored in EEPROM. Default value: TBD		

Example: Set gain I = 1000 (= 0x03E8)

	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
Host frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x01**	**0x02**	**0xE8**	**0x03**	**0xE8**

	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
Slave frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x01**	**0x00**	**0x01**

15.2.3. Set PID gain D

Command ID:	**0x02**	Name:	**Set PID gain D**
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Sets the Derivative gain of the internal PID control loop.

Notes: Stored in EEPROM. Default value: TBD

Example: Set gain D = 10000 (= 0x2710)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x02**	**0x02**	**0x10**	**0x27**	**0x37**

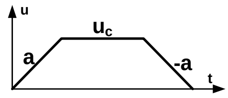
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x02**	**0x00**	**0x02**

15.2.4. Set profile acceleration

Command ID:	**0x03**	Name:	**Set profile acceleration**
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Set the desired acceleration (a) and deceleration (-a) for profiled position and velocity movements.

Notes: Stored in EEPROM.
Measured in ticks/sec^2.
Default value: TBD



Example: Set profile acceleration = 800 (0x0320)ticks/sec^2

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x03**	**0x02**	**0x20**	**0x03**	**0x22**

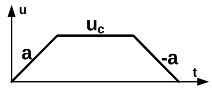
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x03**	**0x00**	**0x03**

15.2.5. Set profile constant velocity

Command ID:	**0x04**	Name:	**Set profile constant velocity**
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Set the desired constant velocity (u_c) of position or velocity profiled movements.

Notes: Stored in EEPROM.
Measured in ticks/sec.
Default value: TBD



Example: Set constant velocity = 800 (0x0320) ticks/sec

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x04**	**0x02**	**0x20**	**0x03**	**0x25**

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x04**	**0x00**	**0x04**

15.2.6. Set current limit

Command ID:	**0x05**	Name:	**Set current limit**
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Sets the maximum average current allowed to be supplied to the motor for a specific duration (DurationForCurrentLimit).

Notes: Stored in EEPROM. Measured in mA. Default value: 1000/5000 (1A/5A versions). Does not apply for 1A Supermodifed version.
⚠ Adjust this setting according to motor rating.

Example: Set current limit at = 900 (= 0x0384) mA

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x05**	**0x02**	**0x84**	**0x03**	**0x80**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x05**	**0x00**	**0x05**		

15.2.7. Set duration for current limit

Command ID:	**0x06**	Name:	**Set duration for current limit**
Tx data bytes:	**2**	*interpreted as:*	16 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: If the average current is above the current limit for the current limit duration an error is generated and the motor is stopped.

Notes: Stored in EEPROM. Measured in mSec. Default value: 5000
⚠ Adjust this setting according to motor ratings.

Example: Set current limit duration = 10000 (= 0x2710) mSec

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x06**	**0x02**	**0x10**	**0x27**	**0x37**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x06**	**0x00**	**0x06**		

15.2.8. Move with velocity

Command ID:	**0x07**	Name:	**Move with velocity**
Tx data bytes:	**2**	*interpreted as:*	16 bit signed integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Configures the velocity setpoint according to data and sets the motor in velocity control mode.

Notes: Measured in ticks/sec. Motor starts to move with commanded velocity immediately and without any motion profiles.

Example: Move with +500 (0x01F4) ticks/sec

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x07**	**0x02**	**0xF4**	**0x01**	**0xF0**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x07**	**0x00**	**0x07**		

15.2.9. Move to absolute position

Command ID:	**0x08**	Name:	**Move to absolute position**
Tx data bytes:	**4**	*interpreted as:*	32 bit signed integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Configures the absolute position setpoint and sets the motor in position control mode. Absolute Incremental Position is implied.

Notes: Measured in ticks. Motor starts to move to commanded position immediately and without any motion profiles. (max acceleration, then full stop at commanded position)

Example: Move to absolute position =10240 (0x00002800) ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x08**	**0x04**	**0x00**	**0x28**	**0x00**	**0x00**	**0x2B**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x08**	**0x00**	**0x08**				

15.2.10. Move to relative position

Command ID:	0x09	Name:	Move to relative position
Tx data bytes:	4	*interpreted as:*	32 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Configures the absolute position setpoint and sets the motor in position control mode.

Notes: Measured in ticks. Motor starts to move to commanded position immediately and without any motion profiles. Position is calculated relative to current position.

Example: Move to -3000 (0xFFFFF448) ticks relative to current position

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x09**	**0x04**	**0x48**	**0xF4**	**0xFF**	**0xFF**	**0xBF**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x09**	**0x00**	**0x09**				

15.2.11. Profiled move with velocity

Command ID:	0x0A	Name:	Profiled move with velocity
Tx data bytes:	2	*interpreted as:*	16 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Configures the velocity position setpoint and sets the motor in profiled velocity control mode.

Notes: Measured in ticks/sec. Motor starts to move immediately with acceleration setting until the velocity setpoint is reached. Then it continues with commanded velocity.

Example: Move motor -500 (0x FE0C) ticks/sec

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0A**	**0x02**	**0x0C**	**0xFE**	**0xFA**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0A**	**0x00**	**0x0A**		

15.2.12. Profiled move to absolute position

Command ID:	0x0B	Name:	Profiled move to absolute position
Tx data bytes:	4	*interpreted as:*	32 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Configures the absolute position setpoint and sets the motor in profiled position control mode.

Notes: Measured in ticks. Motor starts to move to commanded absolute position following the given motion profile. Incremental Absolute Position is implied.

Example: Move to -3000 (0xFFFFF448) ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0B**	**0x04**	**0x48**	**0xF4**	**0xFF**	**0xFF**	**0xB3**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0B**	**0x00**	**0x0B**				

15.2.13. Profiled move to relative position

Command ID:	0x0C	Name:	Profiled move to relative position
Tx data bytes:	4	*interpreted as:*	32 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Configures the absolute position setpoint and sets the motor in position control mode.

Notes: Measured in ticks. Motor starts to move to commanded absolute position following the given motion profile.

Example: Move to 3000 (0x00000BB8) ticks relative to current position

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0C**	**0x04**	**0xB8**	**0x0B**	**0x00**	**0x00**	**0xBB**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0C**	**0x00**	**0x0C**				

15.2.14. Set velocity setpoint

Command ID:	**0x0D**	Name:	**Set velocity setpoint**
Tx data bytes:	**2**	*interpreted as:*	16 bit signed integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Stores a velocity setpoint. Used in conjunction with broadcast command “Do move” for synchronized motions of many SupermodifedTM controllers on the same bus.

Notes: Measured in ticks/sec. The received velocity setpoint is buffered. A following broadcast “Do move” command will initiate non profiled move with velocity according to buffered setpoint.

***Example: Store a velocity setpoint = 0** (can be used to simultaneously halt all motors together with a “Do move” broadcast command)*

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0D**	**0x02**	**0x00**	**0x00**	**0x0F**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0D**	**0x00**	**0x0D**		

15.2.15. Set absolute position setpoint

Command ID:	**0x0E**	Name:	**Set absolute position setpoint**
Tx data bytes:	**4**	*interpreted as:*	32 bit signed integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Stores an absolute position setpoint. Used in conjunction with broadcast command “Do move” for synchronized motions of many SupermodifedTM controllers on the same bus.

Notes: Measured in ticks. The received absolute position setpoint is buffered. A following broadcast “Do move” command will initiate non profiled absolute move to buffered setpoint.

Example: Store an absolute move to +64 (0x00000040) ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0E**	**0x04**	**0x40**	**0x00**	**0x00**	**0x00**	**0x4A**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0E**	**0x00**	**0x0E**				

15.2.16. Set relative position setpoint

Command ID:	**0x0F**	Name:	**Set relative position setpoint**
Tx data bytes:	**4**	*interpreted as:*	32 bit signed integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Stores relative position setpoint. Used in conjunction with broadcast command “Do move” for synchronized motions of many SupermodifedTM controllers on the same bus.

Notes: Measured in ticks. The received relative position setpoint is buffered. A following broadcast “Do move” command will initiate non profiled relative move to buffered setpoint.

Example: Store a relative move to 0 ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x0F**	**0x04**	**0x00**	**0x00**	**0x00**	**0x00**	**0x0B**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x0F**	**0x00**	**0x0F**				

15.2.17. Set profiled velocity setpoint

Command ID:	0x10	Name:	Set profiled velocity setpoint
Tx data bytes:	2	*interpreted as:*	16 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Stores relative velocity setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many Supermodifed™ controllers on the same bus.

Notes: Measured in ticks/sec. The received velocity setpoint is buffered. A following broadcast "Do move" command will initiate profiled move with velocity according to buffered setpoint.

Example: Store a profiled velocity setpoint = 783 (0x030F) ticks/sec

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x10**	**0x02**	**0x0F**	**0x03**	**0x1E**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x10**	**0x00**	**0x10**		

15.2.18. Set profiled absolute position setpoint

Command ID:	0x11	Name:	Set profiled absolute position setpoint
Tx data bytes:	4	*interpreted as:*	32 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Stores an absolute position setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many Supermodifed™ controllers on the same bus.

Notes: Measured in ticks. The received absolute position setpoint is buffered. A following broadcast "Do move" command will initiate profiled absolute move to buffered setpoint.

Example: Store a profiled absolute position setpoint = 0 ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x11**	**0x04**	**0x00**	**0x00**	**0x00**	**0x00**	**0x15**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x11**	**0x00**	**0x11**				

15.2.19. Set profiled relative position setpoint

Command ID:	0x12	Name:	Set profiled relative position setpoint
Tx data bytes:	4	*interpreted as:*	32 bit signed integer
Rx data bytes:	0	*interpreted as:*	-

Description: Stores a relative position setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many Supermodifed™ controllers on the same bus.

Notes: Measured in ticks. The received relative position setpoint is buffered. A following broadcast "Do move" command will initiate profiled absolute move to buffered setpoint.

Example: Store a profiled relative position setpoint = 0 ticks

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x12**	**0x04**	**0x00**	**0x00**	**0x00**	**0x00**	**0x16**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x12**	**0x00**	**0x12**				

15.2.20. Configure digital IOs

Command ID:	0x13	Name:	Configure digital IOs
Tx data bytes:	1	*interpreted as:*	8 bit unsigned integer
Rx data bytes:	0	*interpreted as:*	-

Description: Configures whether the four on-board digital IOs DIO1–DIO4 will be inputs or outputs.

Notes:	Stored in EEPROM. The four low bits of the data byte correspond to Digital IO configuration: Bit0 → DIO1, if set DIO1 is configured as output; if zero DIO1 configured as input. So on for rest bits.

Example: Set DIO1 and DIO4 as outputs, DIO2 and 3 as inputs

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x13**	**0x01**	**0x09**	**0x1B**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x13**	**0x00**	**0x13**	

15.2.21. Set digital outputs

Command ID:	**0x14**	Name:	**Set digital outputs**
Tx data bytes:	**1**	*interpreted as:*	8 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-
Description:	Configures the state of the on-board digital outputs.		
Notes:	The four low bits of the data byte correspond to Digital Outputs state: Bit0 → DIO1, if set and DIO is configured as output digital out →Vcc. If zero digital output state → 0V. If the corresponding DIO is not configured as an output the command will be discarded.		

Example: Set DIO4 high

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x14**	**0x01**	**0x08**	**0x14**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x14**	**0x00**	**0x14**	

15.2.22. Set node ID

Command ID:	**0x15**	Name:	**Set node ID**
Tx data bytes:	**1**	*interpreted as:*	8 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-
Description:	Configures the node ID of the Supermodifed™ Controller.		
Notes:	Stored to EEPROM. After execution of this command the controller no longer responds to its previous ID. Power cycling for a HW reset is advised after this command.		

Example: Change controller ID from 0x04 to 0x05

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x15**	**0x01**	**0x05**	**0x11**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x15**	**0x00**	**0x15**	

15.2.23. Set local acceptance mask

Command ID:	**0x16**	Name:	**Set Local Acceptance Mask**
Tx data bytes:	**1**	*interpreted as:*	8 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-
Description:	Configures the Supermodifed™ controller to be addressed by using more than one IDs. This command is actually used to allow for groups of Supermodifed™ controllers to be commanded all at the same time with a single command. The mechanism of the implementation is illustrated in pseudo-code below:		

if (controller_ID AND local_acceptance_mask) =
(received_ID AND local_acceptance_mask)
Then
Accept command as if correct ID was received

For the command response the local acceptance mask is not used. In this way from a group of controllers accepting the command, only one –the group leader- whose ID is equal to the AndID issued by the master will respond.

Local acceptance mask functionality is discarded for ‘get commands’.

Notes: Stored to EEPROM.

Example:

There are 10 Supermodifed controllers on a bus. Their IDs are set to be: 0x10, 0x11, 0x12, 0x13, 0x14, 0x20, 0x21, 0x22, 0x23, 0x24. Controllers with IDs 0x10 and 0x20 are the group leaders. All controllers have their local acceptance mask set to 0xF0. This means that only the four high bits of an AndID are examined in order to decide if the controller will execute the command or not. The lower four bits are don’t cares.

When the master issues a command to node 0x10, all controllers 0x10, 0x11, 0x12, 0x13, 0x14 will execute the command. Only controller 0x10 will issue a command response. Accordingly when the master issues a command to node 0x20 controllers 0x20, 0x21, 0x22, 0x23, 0x24 will execute the command. Only controller 0x20 will issue a command response.

Example: Set local acceptance mask of controller 0x10 to 0xF0

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x10**	**0x01**	**0x16**	**0x01**	**0xF0**	**0xE7**

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x10**	**0x16**	**0x00**	**0x16**

15.2.24. Set baud rate UART

Command ID:	**0x17**	Name:	**Set baud rate UART**
Tx data bytes:	**4**	*interpreted as:*	32 bit unsigned integer
Rx data bytes:	**0**	*interpreted as:*	-

Description: Configures the UART baud rate. Measured in bps (bits per sec)

Notes: Stored to EEPROM. Minimum value: 600 bps. Maximum value 115200 bps. Default value 9600 Bps
After execution of this command the controller no longer responds to previous baud rate.

Example: Change controller UART baud rate from 9600 to 19200 Bps

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x17**	**0x04**	**0x00**	**0x4B**	**0x00**	**0x00**	**0x58**

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x17**	**0x00**	**0x17**

15.2.25. Reset incremental position

Command ID:	0x18	Name:	Reset incremental position
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-

Description: Sets the current Incremental Absolute Position = 0.

Notes: On reception of this command the controller switches to position control with position setpoint = 0, thus maintaining its position. All buffered setpoints are disabled. This command should be used only during homing sequences implemented by the end user on the host controller. These homing sequences should involve low motor speeds.

Example: Reset Incremental position

	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
Host frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x18**	**0x00**	**0x18**
Slave frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x18**	**0x00**	**0x18**

15.2.26. Start

Command ID:	0x19	Name:	Start
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-

Description: Initializes the Supermodifed™ Ccntroller.

Notes: The SupermodifedTM controller upon power up is not executing any type of control, thus not applying any voltage force on the attached motor. When this command is received incremental position is reset, all internal circuitry and memory is initialized and the controller enters position control mode with position setpoint = 0.

Example: Start the Supermodifed™ controller

	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
Host frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x19**	**0x00**	**0x19**
Slave frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x19**	**0x00**	**0x19**

15.2.27. Halt

Command ID:	0x1A	Name:	Halt
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-

Description: Stops the motor.

Notes: The SupermodifedTM controller is switched to position control with position setpoint = current position.

Example: Halt the motor attached to the Supermodifed™ controller.

	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
Host frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x1A**	**0x00**	**0x1A**
Slave frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x1A**	**0x00**	**0x1A**

15.2.28. Stop

Command ID:	0x1B	Name:	Stop
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-

Description: Un - Initializes the Supermodifed™ Ccntroller.

Notes: The Supermodifed™ controller does not execute control loop.

Example: Start the Supermodifed™ controller

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x1B**	**0x00**	**0x1B**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x1B**	**0x00**	**0x1B**

15.2.29. Set error reporting level

Command ID: 0x1C Name: Set error reporting level

Tx data bytes: **1** *interpreted as:* 8bit unsigned integer

Rx data bytes: **0** *interpreted as:* -

Description: Sets the error reporting level.
0(default): Only serious errors are reported. e.g. motor over-current error (motor operation will be halted in order to prevent damage to the motor). If such an error exists normal command response frames will be overridden by the error frame.
1: All warnings are treated as errors. By default warnings do not cause the override of a command response frame with an error frame. A warning can be generated if e.g. the Supermodifed™ controller receives a move position but it has not received a Start command prior to that.

Notes: Stored to EEPROM.

Example: Set error reporting level to 1.

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x1C**	**0x01**	**0x01**	**0x1C**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x1C**	**0x00**	**0x1C**	

15.3. Get Commands

15.3.1. Get PID gain P

Command ID: 0x64 Name: Get PID gain P

Tx data bytes: **0** *interpreted as:* -

Rx data bytes: **2** *interpreted as:* 16-bit unsigned integer

Description: Gets the proportional gain from the Supermodifed™ controller.

Notes: This setting is read from on-board EEPROM.

Example: Get PID gain P (which is e.g. 2000 = 0x07D0)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x64**	**0x00**	**0x64**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x64**	**0x02**	**0xD0**	**0x07**	**0xB1**

15.3.2. Get PID gain I

Command ID: 0x65 Name: Get PID gain I

Tx data bytes: **0** *interpreted as:* -

Rx data bytes: **2** *interpreted as:* 16-bit unsigned integer

Description: Gets the integral gain from the Supermodifed™ controller.

Notes: This setting is read from on-board EEPROM.

Example: Get PID gain I (which is e.g. 1000 = 0x03E8)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x65**	**0x00**	**0x65**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x65**	**0x02**	**0xE8**	**0x03**	**0x8C**

15.3.3. Get PID gain D

Command ID:	0x66	Name:	Get PID gain D
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit unsigned integer
Description:	Gets the derivative gain from the Supermodifed™ controller.		
Notes:	This setting is read from on-board EEPROM.		

Example: Get PID gain D (which is e.g. 10000 = 0x2710)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x66**	**0x00**	**0x66**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x66**	**0x02**	**0x10**	**0x27**	**0x53**

15.3.4. Get profile acceleration

Command ID:	0x67	Name:	Get profile acceleration
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit unsigned integer
Description:	Gets the profile acceleration in ticks/sec^2.		
Notes:	This setting is read from on-board EEPROM.		

Example: Get profile acceleration (which is e.g. 800 = 0x0320)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x67**	**0x00**	**0x67**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x67**	**0x02**	**0x20**	**0x03**	**0x46**

15.3.5. Get profile constant velocity

Command ID:	0x68	Name:	Get profile constant velocity
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit unsigned integer
Description:	Gets the constant velocity for profiled motion in ticks/sec.		
Notes:	This setting is read from on-board EEPROM.		

Example: Get profile constant velocity (which is e.g. 800 = 0x0320)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x68**	**0x00**	**0x68**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x68**	**0x02**	**0x20**	**0x03**	**0x49**

15.3.6. Get current limit

Command ID:	0x69	Name:	Get current limit
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit unsigned integer
Description:	Gets the motor current limit in mA.		
Notes:	This setting is read from on-board EEPROM. This setting does not apply for the 1A version.		

Example: Get current limit (which is e.g. 5000 = 0x1388)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x69**	**0x00**	**0x69**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x69**	**0x02**	**0x88**	**0x13**	**0xF0**

15.3.7. Get current limit duration

Command ID:	0x6A	Name:	Get current limit duration
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit unsigned integer

Description: Gets the motor current limit duration in mS. If a current over the current_limit appears at the motor for more than current_limit_duration then the over-current error is generated and the motor operation is halted.

Notes: This setting is read from on-board EEPROM.
This setting does not apply for the 1A version.

Example: Get current limit duration (which is e.g. 5000 = 0x1388)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6A**	**0x00**	**0x6A**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6A**	**0x02**	**0x88**	**0x13**	**0xF3**

15.3.8. Get digital IO configuration

Command ID:	0x6B	Name:	Get digital IO configuration
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	1	*interpreted as:*	8-bit unsigned integer

Description: Gets the digital IO configuration. Lower nibble (4 lower bits) are set or reset according to IO configuration

Notes: This setting is read from on-board EEPROM.

Example: Get digital IO configuration (which is e.g. all outputs)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6B**	**0x00**	**0x6B**	
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6B**	**0x01**	**0x0F**	**0x65**

15.3.9. Get local acceptance mask

Command ID:	0x6C	Name:	Get local acceptance mask
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	1	*interpreted as:*	8-bit unsigned integer

Description: Gets the local acceptance mask of the addressed controller. Local acceptance mask functionality is explined in the set local acceptance mask command.

Notes: This setting is read from on-board EEPROM.

Example: Get local acceptance mask (which is e.g. no acceptance mask)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6C**	**0x00**	**0x6C**	
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6C**	**0x01**	**0xFF**	**0x92**

15.3.10. Get digital inputs

Command ID:	0x6D	Name:	Get digital inputs
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	1	*interpreted as:*	8-bit unsigned integer

Description: Gets the state of the on-board digital inputs. Stored in lower nibble of received byte. DIO1 is represented by bit 0, DIO2 by bit 1 and so on.

Notes: If a DIO is configured as output reads will return the digital output state.

Example: Get digital inputs (which are e.g. all zeros)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6D**	**0x00**	**0x6D**
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6D**	**0x01**	**0x00**	**0x6C**

15.3.11. Get analog inputs

Command ID:	**0x6E**	Name:	**Get analog inputs**
Tx data bytes:	**0**	*interpreted as:*	-
Rx data bytes:	**8**	*interpreted as:*	4 x 16-bit unsigned integers

Description: Gets the analog voltages read by the on-board ADC on the four analog inputs AIN1-4. D[0] and D[1] contain the value corresponding to AIN1, D[2] and D[3] the value corresponding to AIN2 and so on.

Notes: On board ADC resolution is 10 bits. A 50% weighted running average filter is applied on readings.

Example: Get analog inputs (which e.g. all read 2.5Volts)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6E**	**0x00**	**0x6E**				
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6E**	**0x08**	**0x00**	**0x02**	**0x00**	**0x02**	**0x00**
	D[5]	D[6]	D[7]	lrc							
	0x02	**0x00**	**0x02**	**0x66**							

15.3.12. Get position

Command ID:	**0x6F**	Name:	**Get position**
Tx data bytes:	**0**	*interpreted as:*	-
Rx data bytes:	**4**	*interpreted as:*	32-bit signed integer

Description: Gets the current Incremental Absolute position in ticks.

Notes: -

Example: Get position (which is e.g. 5683 = 0x1633 ticks)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc				
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x6F**	**0x00**	**0x6F**				
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x6F**	**0x04**	**0x33**	**0x16**	**0x00**	**0x00**	**0x4E**

15.3.13. Get absolute position

Command ID:	**0x70**	Name:	**Get absolute position**
Tx data bytes:	**0**	*interpreted as:*	-
Rx data bytes:	**2**	*interpreted as:*	16-bit unsigned integer

Description: Gets the current Absolute position in ticks. Resolution is 10bits. This is the raw reading from the absolute encoder interface.

Notes: Only available if a MagEnc™ magnetic absolute encoder is interfaced to the Supermodifed™ controller.

Example: Get absolute position (which is e.g. 512 = 0x0200 ticks)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x70**	**0x00**	**0x70**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x70**	**0x02**	**0x00**	**0x02**	**0x70**

15.3.14. Get velocity

Command ID:	0x71	Name:	Get velocity
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit signed integer

Description: Gets the current velocity in ticks/sec.

Notes: Only available if a MagEnc™ magnetic absolute encoder is interfaced to the Supermodifed™ controller.

Example: Get velocity (which is e.g. 400 ticks/sec = 0x0190 ticks/sec)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x71**	**0x00**	**0x71**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x71**	**0x02**	**0x90**	**0x01**	**0xE2**

15.3.15. Get current

Command ID:	0x72	Name:	Get current
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	2	*interpreted as:*	16-bit signed integer

Description: Gets the current in mA.

Notes: Only available with the 5A version.

Example: Get current (which is e.g. 187mA 0x00BB)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc		
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x72**	**0x00**	**0x72**		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x72**	**0x02**	**0xBB**	**0x00**	**0xCB**

15.3.16. Get warning

Command ID:	0x73	Name:	Get warning
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	1	*interpreted as:*	8-bit unsigned integer

Description: Gets the most recent warning.
If the error reporting level is set to 0 warnings are stored inside the Supermodifed™ in a circular buffer. The last warning is popped from the buffer and sent through the communication interface.

Notes: If no warnings exist the no_error error code will be returned

Example: Get warning (e.g. there are no warnings)

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc	
frame	**0x55**	**0xAA**	**0x04**	**0x01**	**0x73**	**0x00**	**0x73**	
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	lrc
frame	**0x55**	**0xAA**	**0x01**	**0x04**	**0x73**	**0x01**	**0x00**	**0x72**

15.4. Broadcast commands

15.4.1. Do move

Command ID:	0xC9	Name:	Do move
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-

Description: All controllers execute their pre-buffered sepoints.

Notes: If there is no pre-buffered setpoint (or it has already been executed by a previous Do Move) command is discarded.

Example: Move all motors attached to controllers on the bus to respective buffered setpoint.

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x00**	**0x01**	**0xC8**	**0x00**	**0xC8**

15.4.2. Global start

Command ID:	0xC9	Name:	Global start
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-
Description:	Starts all Supermodifed™ controllers on the bus.		
Notes:	-		

Example: Start all Supermodifed™ controllers on the bus.

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x00**	**0x01**	**0xC9**	**0x00**	**0xC9**

15.4.3. Global halt

Command ID:	0xCA	Name:	Global halt
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-
Description:	Halts all Supermodifed™ controllers on the bus by applying position control and position setpoint = current position.		
Notes:	-		

Example: Halt all Supermodifed™ controllers on the bus.

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x00**	**0x01**	**0xCA**	**0x00**	**0xCA**

15.4.4. Global stop

Command ID:	0xCB	Name:	Global stop
Tx data bytes:	0	*interpreted as:*	-
Rx data bytes:	0	*interpreted as:*	-
Description:	De-activates all Supermodifed™ controllers on the bus.		
Notes:	-		

Example: Stop all Supermodifed™ controllers on the bus.

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	**0x55**	**0xAA**	**0x00**	**0x01**	**0xCB**	**0x00**	**0xCB**

16. Error code reference

During operation, various errors and warnings may appear. Errors are assigned a severity level: 0 → serious errors and 1→ warnings.

Whenever an error is detected by the Supermodifed™ controller it is stored in a queue in memory and is reported according to error reporting level on the next command response issued by the controller: i.e. on the next received command the controller will not issue a regular command response but rather transmit an error frame like in section 8.1.4

According to error code action may or may not be taken by the controller's software. The default error reporting level is set to 0 i.e. only 0 level errors are reported through the command response override mechanism.

A list of the available error codes, their assigned error level and their probable cause is presented below.

Code	Name	Level	Description:/cause/notes
			Motor/Encoder related errors
0x01	Motor Stalled	0	*Description:* A motor stalled condition has occurred. *Cause:* 1) The motor is actually stalled. 2) The motor has been instructed to achieve a non-achievable velocity. *Action taken:* When this error occurs the Supermodifed™ controller shuts off power to the motor. *Resolution:* Check your mechanical implementation. Make sure the motor is adequately sized for the intended purpose.
0x02	Encoder Overflow	0	*Description:* The incremental position is about to overflow. *Cause:* The motor has traveled extremely long towards the same direction. (2147482624 encoder ticks) *Action taken:* When this error occurs the Supermodifed™ controller shuts off power to the motor. *Resolution:* None. This error is almost impossible to occur.
0x03	Encoder Underflow	0	*Description:* The incremental position is about to underflow. *Cause:* The motor has traveled extremely long towards the same direction. (-2147482624 encoder ticks) *Action taken:* When this error occurs the Supermodifed™ controller shuts-off power to the motor. An error response is issued during the next communication cycle. *Resolution:* None. This error is almost impossible to occur.
			Command Related Errors
0x11	Invalid command ID	0	*Description:* A communication packet with a non existent command ID has been received. *Cause:* The host controller issued an invalid command ID *Action taken:* None. The command is not executed. An error response is issued immediately. *Resolution:* Check your communication software for possible errors.
0x12	Invalid set command byte-count	0	*Description:* A set command with a wrong bytecount has been received. *Cause:* The host controller issued an invalid set command bytecount *Action taken:* None. The command is not executed. An error response is issued immediately. *Resolution:* Check your communication software for possible errors.
0x13	Invalid argument	0	*Description:* A set command with an invalid data argument has been received. For example attempting to set the Supermodifed node ID to 0. *Cause:* The host controller issued an invalid data argument. *Action taken:* None. The command is not executed. An error response is issued immediately. *Resolution:* Check your communication software for possible errors.
0x14	Invalid command for motor state	0	*Description:* The received command is invalid for the given motor state. ie the Supermodifed™ controller is instructed to move the motor with a specific velocity prior to receiving a Start command (initialization and PID activation). *Cause:* described above. *Action taken:* None. The command is not executed. An error response is issued immediately. *Resolution:* Issue a Start command before attempting to issue movement commands.
0x15	Invalid byte-count	1	*Description:* A get or broadcast command has been received with invalid bytecount. *Cause:* described above. *Action taken:* None. The command *is* executed. An error response is issued immediately only if the error reporting level is set to 1.

			Resolution: Check your communication software for possible errors
			I2C related errors
0x23	Arbitration lost	1	*Description:* The Supermodifed^{TM} controller lost arbitration when trying to issue a command response. *Cause:* described above. *Action taken:* The Supermodifed^{TM} controller tries another 5 times to issue the command response. If all tries fail an error response is issued. *Resolution:* Check your communication software for possible errors. The only reason for arbitration loss is a second device trying to be I2C master at the same time.
0x31	Packet override	0	*Description:* A communication packet override condition was detected. *Cause:* Communication at this rate cannot be handled by the Supermodifed^{TM} controller. *Action taken:* An error response is issued. *Resolution:* Introduce some delay between communication cycles with the Supermodifed^{TM} controller.
0x32	Invalid receive byte-count	0	*Description:* The received I2C packet had a different byte length than it should according to packet byte-count. *Cause:* described above. *Action taken:* The command is not executed. An error response is issued immediately. *Resolution:* Check your communication software for possible errors
			UART related errors
0x41	Memory allocation error	0	*Description:* UART initialization detected not enough memory for UART buffers. *Cause:* User programming of the controller has taken up all memory. *Action taken:* None. *Note:* The Supermodifed^{TM} controller cannot initiate UART transactions or may not function at all. *Resolution:* Reduce memory needed for your program inside the Supermodifed^{TM} controller.
0x46	Frame error	0	*Description:* UART detected a frame error. Ie invalid number of data bits, invalid number of stop bits etc *Cause:* 1) Incompatible host UART settings. 2) Extreme noise 3) Bad/damaged cabling *Action taken:* None. The command involving the frame error is discarded. An error response is issued if possible during next transaction. *Resolution:* Check host controller UART settings. Check cabling.
0x47	Parity error	0	*Description:* UART detected a parity error. *Cause:* 1) Incompatible host UART settings. 2) Extreme noise 3) Bad/damaged cabling *Action taken:* None. The command involving the parity error is discarded. An error response is issued during next transaction. *Resolution:* Check host controller UART settings. Check cabling.
0x48	Receive buffer overflow	0	*Description:* A receive buffer overflow was detected. *Cause:* Communication at this rate cannot be handled by the Supermodifed^{TM} controller. *Action taken:* An error response is issued. *Resolution:* Introduce some delay between communication cycles with the Supermodifed^{TM} controller.
0x49	Receive data override	0	*Description:* A receive data override condition was detected. *Cause:* Communication at this rate cannot be handled by the Supermodifed^{TM} controller. *Action taken:* An error response is issued. *Resolution:* Introduce some delay between communication cycles with

			the Supermodifed™ controller.
			Protocol related errors
0x52	Wrong LRC	0	*Description:* The received packet had a wrong lrc. *Cause:* described above. *Action taken:* The command is not executed. An error response is issued immediately. *Resolution:* Check your communication software for possible errors