

---

# 01™ SuperModified – Miniature controller for DC motors

## 1. General Description

The SuperModified™ combo of miniature PCBs is an all-in one motor control solution. Incorporating a 15-bit magnetic absolute encoder, an 8-bit, 20MHz AVR ATmega328p microcontroller and a 5-Amp MosFet H-bridge at an astonishing outline of 16mm x 16mm x 13.2 mm it is ideal for space constrained applications. The overall dimensions allow for this motion control system to be installed inside a standard RC servo, transforming the device to a full functionality servo motor. The 01™ SuperModified is a highly cost effective solution, delivering closed loop PID control at 9.765 KHz, with advanced motion profiling capabilities and many other features.

## 2. Features

- **Robust.** Fully protected power stage (by hardware). Firmware is aware of motor stalls, overcurrent, encoder health etc.
- **High-End profiled movements.** The 01™ Supermodified delivers smooth and accurate velocity profiled motion. With every new setpoint the profiles and transition from / to new setpoint are recalculated.
- **Extreme accuracy.** 15bit absolute encoder delivers 32768 discrete positions per revolution. Control loop runs at 9.765 KHz.
- **Cost effective.** The 01™ Supermodified includes DC motor controller and sensor in a complete motion control system.
- **Quick installation.** The 01™ Supermodified eliminates the need for sensor cabling. The only connections needed are power supply and a communication interface.
- **Easy to use.** After installation the 01™ Supermodified can be operated immediately by using our stand-alone PC application.
- **Multiple interfaces.** Electrical layer interfaces include I2C 5V, I2C 3.3V, RS-485, UART 5V, UART 3.3V, Position Pulse Modulation (standard RC servo compatibility mode).
- **Feature rich protocol (proprietary).** Synchronize the movements of all controllers on a bus, set up two or more motors to operate as one, configure how the controller reacts to errors etc.
- **Easy S/W Integration.** Programming Interfaces available for free for popular hardware and software platforms like Arduino and MatLab.
- **Modular.** Divided into three stand-alone modules:
  - 01™ MagEnc: Absolute 15-bit magnetic encoder.
  - 01™ PicoMcu: the world's smallest AVR development board, based on ATmega328P microcontroller, running at 20MHz.
  - 01™ MotDrv: Freescale MC34931, 5 Amp MOSFET H-Bridge, breakout PCB
- Includes 3 configurable digital IOs and 2 analog inputs.
- Includes bootloader (uart/RS-485) for firmware upgrades.



01™  
SuperModified

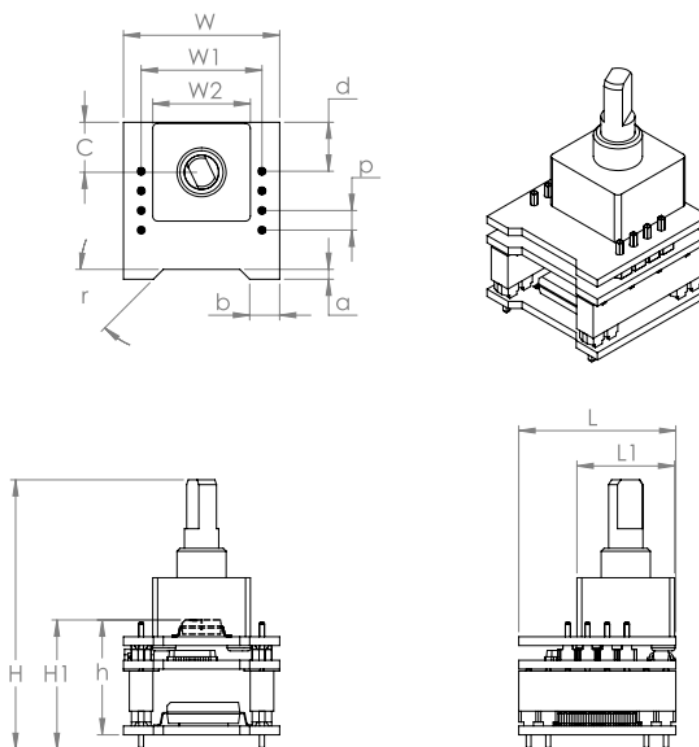
Miniature  
Controller for DC  
Motors

*"The robotic rebirth of  
the hobby servo"*

*Preliminary datasheet*

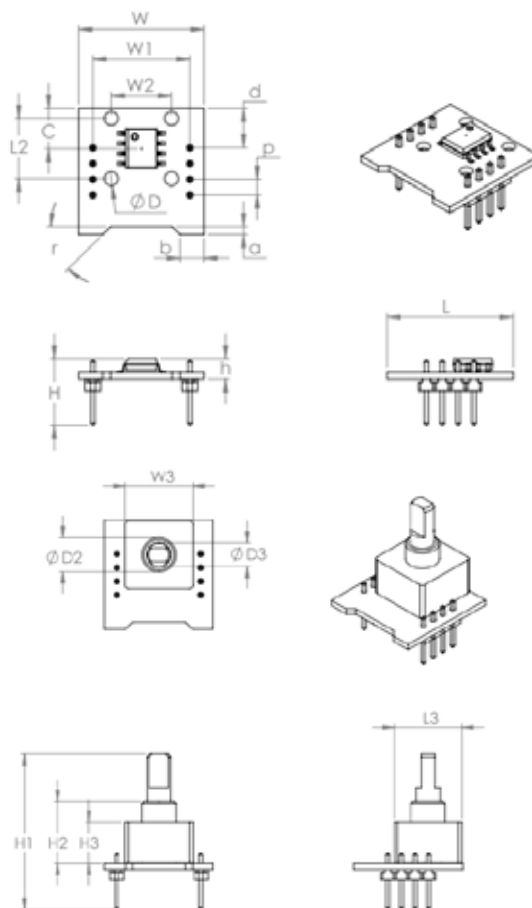


### 3. Mechanical layout



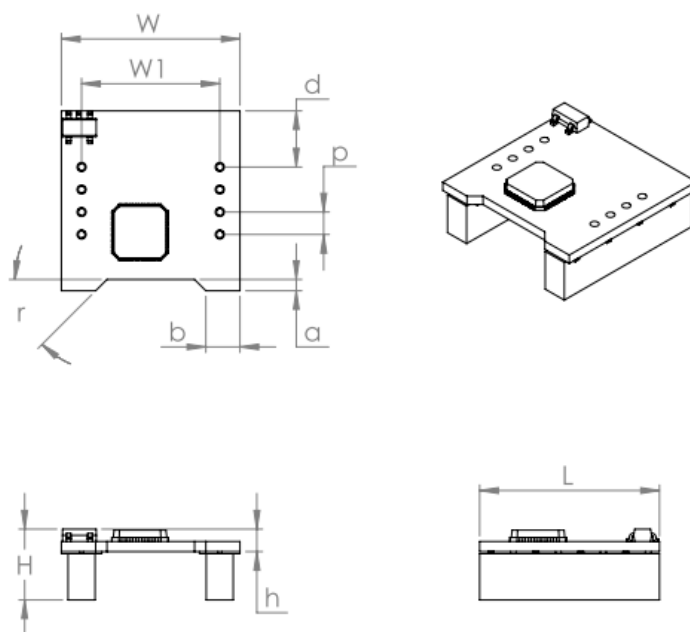
Description	Symbol	Value	Units
Number of Pins	n	<b>16</b>	Pins
Pitch	p	<b>2</b>	mm
Board Width	W	<b>16</b>	mm
Overall Row Spacing	W1	<b>12.3</b>	mm
Sensor Interface Width	W2	<b>9.95</b>	mm
Sensor Centre from Board Edge	C	<b>5.1</b>	mm
First Top Pin from Board Edge	d	<b>5</b>	mm
Leap Height	a	<b>1</b>	mm
Leap Width	b	<b>3.05</b>	mm
Leap Chamfer Angle	r	<b>45</b>	°
Overall Height (sensor interface included)	H	<b>27.5</b>	mm
Overall Height (sensor included)	H1	<b>13.2</b>	mm
Minimum Achievable Height	h	<b>11.7</b>	mm
Overall Length	L	<b>16</b>	mm
Sensor Interface Length	L1	<b>9.95</b>	mm

### 3.1. 01™ MagEnc



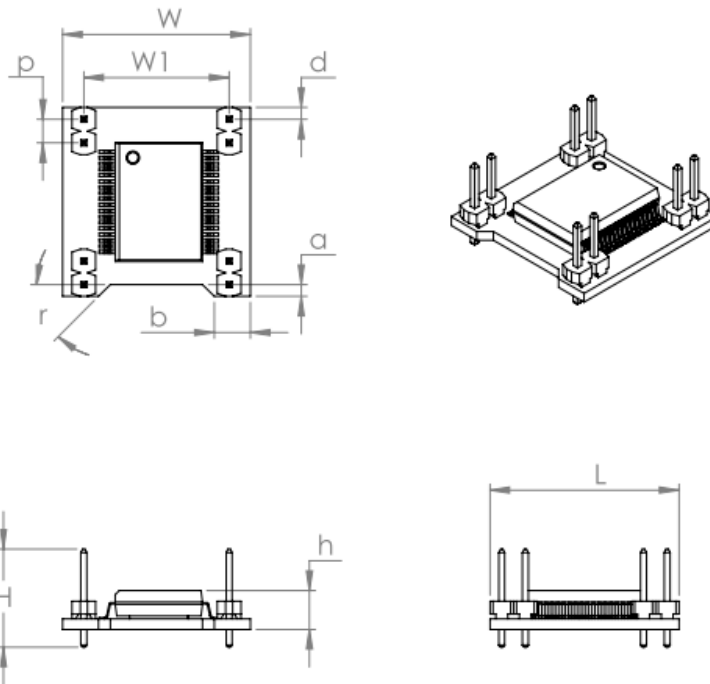
Description	Symbol	Value	Units
Number of Pins	n	8	Pins
Pitch	p	2	mm
Board Width	W	16	mm
Overall Row Spacing	W1	12.3	mm
Sensor Interface Mounting Holes Distance	W2	7.64	mm
Sensor Interface Width	W3	9.95	mm
Sensor Interface Ex. Bushing Diameter	D2	5	mm
Sensor Interface Axis Diameter	D3	3.5	mm
Sensor Centre from Board Edge	C	5.1	mm
First Top Pin from Board Edge	d	5	mm
Leap Height	a	1	mm
Leap Width	b	3.05	mm
Leap Chamfer Angle	r	45	°
Overall Height (w pin-headers, w/o interface)	H	8.5	mm
Overall Height (w/o pin-headers, w/o interface)	h	2.6	mm
Overall Height	H1	22.85	mm
Sensor Interface Base & Bushing Height	H2	9	mm
Sensor Interface Base Height		6	mm
Overall Length	L	16	mm
Sensor Interface Mounting Holes Distance	L2	7.64	mm

### 3.2. 01™ PicoMcu



Description	Symbol	Value	Units
Number of Pins	n	<b>16</b>	Pins
Pitch	p	<b>2</b>	mm
Board Width	W	<b>16</b>	mm
Overall Row Spacing	W1	<b>12.3</b>	mm
First Top Pin from Board Edge	d	<b>5</b>	mm
Leap Height	a	<b>1</b>	mm
Leap Width	b	<b>3.05</b>	mm
Leap Chamfer Angle	r	<b>45</b>	°
Overall Height	H	<b>6.3</b>	mm
Height w/o connector	h	<b>2</b>	mm
Overall Length	L	<b>16</b>	mm

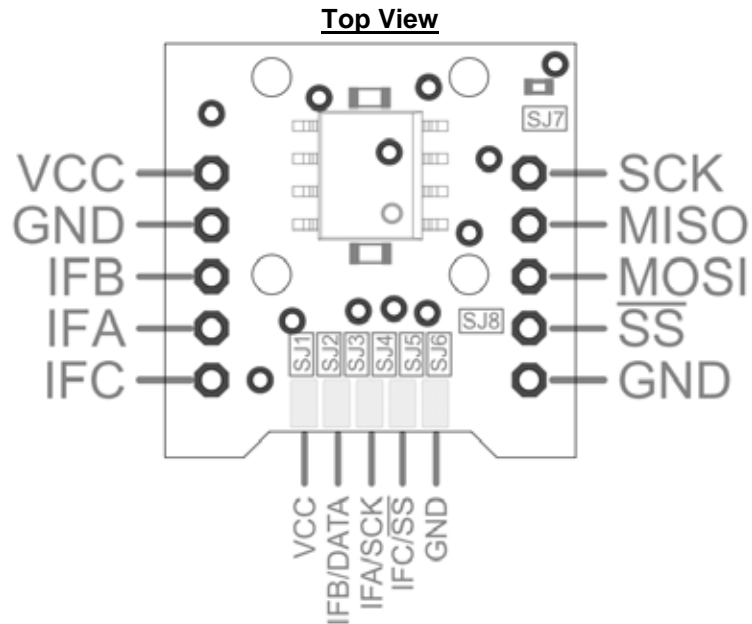
### 3.3. 01™ MotDrv



Description	Symbol	Value	Units
Number of Pins	n	8	Pins
Pitch	p	2	mm
Board Width	W	6	mm
Overall Row Spacing	W1	12.3	mm
Leap Height	a	1	mm
Leap Width	b	3.05	mm
Leap Chamfer Angle	r	45	°
Overall Height	H	8.3	mm
Height w/o connector	h	3.3	mm
Overall Length	L	6	mm

## 4. Pinout

### 4.1. 01™ MagEnc

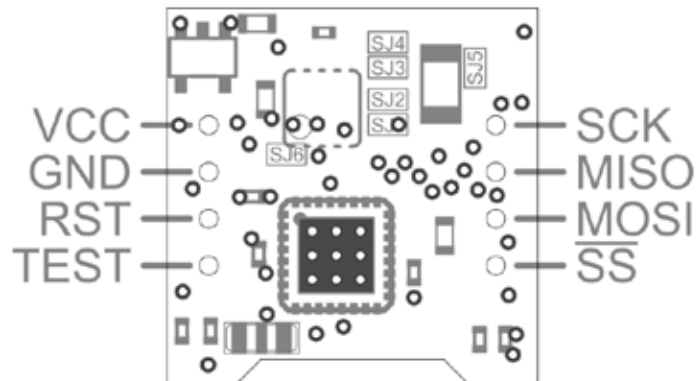


#### 4.1.1. Pin description

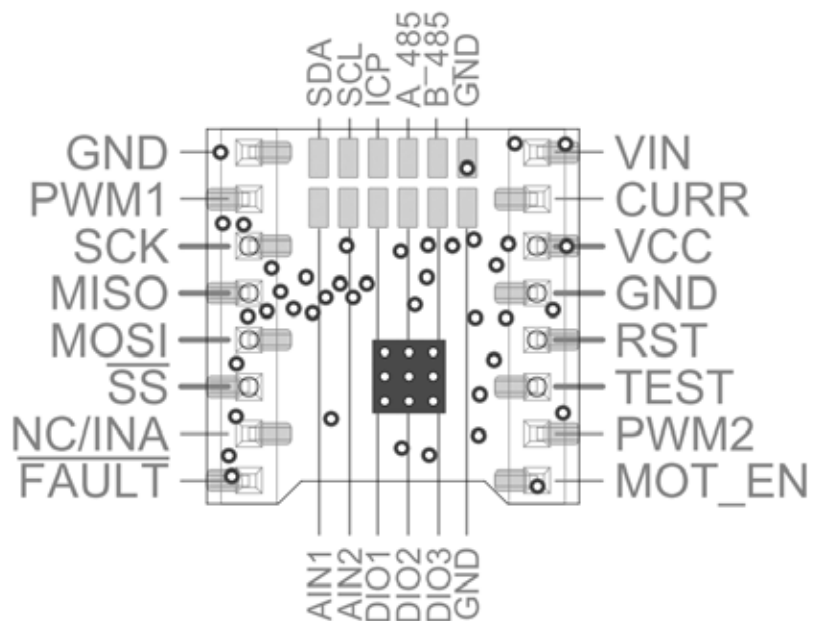
Name	Description
<b>Vcc</b>	Supply voltage. Regulated 5V, 14mA.
<b>GND</b>	Ground.
<b>IFA</b>	Interface A: IIF Phase A / Hall Switch Signal 1 / PWM / SPC output. Refer to <a href="#">TLE5012B datasheet</a> for more information.
<b>IFB</b>	Interface B: IIF Phase B / Hall Switch Signal 2. Refer to <a href="#">TLE5012B datasheet</a> for more information.
<b>IFC</b>	Interface C: External Clock / IIF Index / Hall Switch Signal 3. Refer to <a href="#">TLE5012B datasheet</a> for more information.
<b>SS</b>	Chip Select, active low.
<b>MISO</b>	Data of Synchronous Serial Interface (shorted to MOSI)
<b>MOSI</b>	Data of Synchronous Serial Interface (shorted to MISO)
<b>SCK</b>	Clock Input of Synchronous Serial Interface; Schmitt-Trigger input
<b>IFA / SCK</b>	Solder pad. Can be IFA or SCK as described above by means of the SJ3 and SJ4 solder jumpers. If SJ3 is shorted, the pad is configured as IFA. If SJ4 is shorted the pad is configured as SCK. Do not short both SJ3 and SJ4.
<b>IFB / DATA</b>	Solder pad. Can be IFB or DATA as described above by means of the SJ1 and SJ2 solder jumpers. If SJ1 is shorted, the pad is configured as IFB. If SJ2 is shorted the pad is configured as DATA. Do not short both SJ1 and SJ2.
<b>IFC / SS</b>	Solder pad. Can be IFC or SS as described above by means of the SJ5 and SJ6 solder jumpers. If SJ5 is shorted, the pad is configured as IFC. If SJ6 is shorted the pad is configured as SS. Do not short both SJ3 and SJ4.

## 4.2. 01™ PicoMcu

Top View



Bottom View



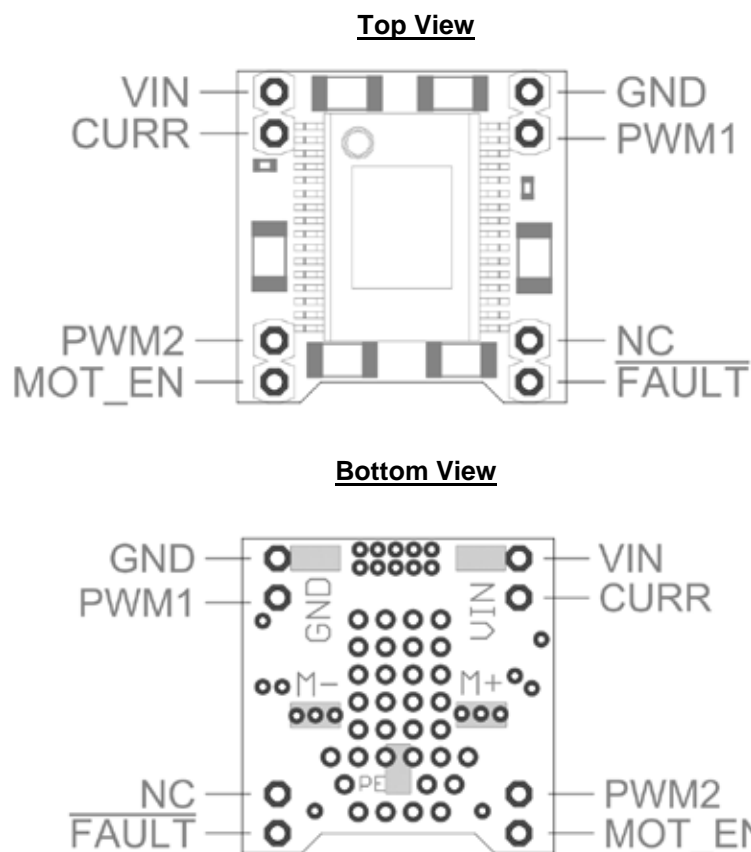
### 4.2.1. Pin description

Name	Description
<b>Vin</b>	Input voltage. Recommended 5V - 24 V.
<b>Vcc</b>	5V regulated voltage from onboard 200mA low drop-out regulator.
<b>GND</b>	Ground.
<b>RST</b>	Reset. Directly connected to ATmega328p Reset (PC6) pin.
<b>TEST</b>	Test pin. Reserved.
<b>SS</b>	Chip select. Directly connected to ATmega328p PB2 pin.
<b>MOSI</b>	SPI interface Master Out Slave In data line. Directly connected to ATmega328p PB3 pin.
<b>MISO</b>	SPI interface Master In Slave Out data line. Directly connected to ATmega328p PB5 pin.
<b>SCK</b>	SPI interface clock line. Directly connected to ATmega328p PB6 pin.
<b>PWM1</b>	PWM signal for 01™ MotDrv PCB.

<b>NC / INA</b>	Reserved. (Future use with other H bridge)	
<b>FAULT</b>	Active low fault input for 01 <sup>TM</sup> MotDrv PCB.	
<b>MOT_EN</b>	Motor enable for 01 <sup>TM</sup> MotDrv PCB.	
<b>PWM2</b>	PWM signal for 01 <sup>TM</sup> MotDrv PCB.	
<b>CURR</b>	Analog input for current measurement for 01 <sup>TM</sup> MotDrv PCB.	
<b>SDA</b>	I2C data. Directly connected to ATmega328p PC4 pin.	
<b>SCL</b>	I2C clock. Directly connected to ATmega328p PC5 pin.	
<b>ICP</b>	Input capture pin. Input for PPM (servo compatibility mode protocol). Directly connected to ATmega328p PB0 pin.	
<b>A_485</b>	<b>Interface</b>	<b>Functional description</b>
	RS-485	<b>A.</b> non-inverting RS-485 transceiver channel
	UART	<b>TXD.</b> UART transmit data
<b>B_485</b>	<b>Interface</b>	<b>Functional description</b>
	RS-485	<b>B.</b> inverting RS-485 transceiver channel
	UART	<b>RXD.</b> UART receive data
<b>AIN1</b>	Analog input 1. Directly connected to ATmega328 ADC6 pin. 0-5V input.	
<b>AIN2</b>	Analog input 2. Directly connected to ATmega328 ADC7 pin. 0-5V input.	
<b>DIO1</b>	Digital IO 1. Directly connected to ATmega328 PC0 pin. 0-5V output, 10mA maximum.	
<b>DIO2</b>	Digital IO 2. Directly connected to ATmega328 PC1 pin. 0-5V output, 10mA maximum.	
<b>DIO3</b>	Digital IO 3. Directly connected to ATmega328 PC2 pin. 0-5V output, 10mA maximum.	
<b>SJ1-SJ6</b>	<u>Solder jumpers. Configuration between RS485 or UART interface.</u>	
	<b>Interface</b>	<b>Solder jumper configuration</b>
	RS-485	SJ3, SJ4, SJ6. SJ5 for connection of included 120 Ohm terminating resistor.
	UART	SJ1, SJ2.



### 4.3. 01™ MotDrv



#### 4.3.1. Pin description

Name	Description
<b>Vin</b>	Input voltage. Recommended 5V - 12 V.
<b>NC</b>	No connection.
<b>PWM2</b>	PWM signal from 01™ PicoMcu.
<b>MOT_EN</b>	Motor enable. Active high input. When low motor outputs are in high impedance state. On board pull-down of 4,7 KOhm
<b>PWM1</b>	PWM signal from 01™ PicoMcu.
<b>FAULT</b>	Active low fault output. See <a href="#">MC34931 datasheet</a> for more details.
<b>M+</b>	Motor connection.
<b>M-</b>	Motor connection.
<b>CURR</b>	Analog output proportional to motor current. See <a href="#">MC34931 datasheet</a> for more details.
<b>GND</b>	Ground.
<b>PE</b>	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 and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### 5.1. Absolute Maximum ratings

Parameter	Symbol	Min	Max
Input Voltage	V <sub>in</sub>	0 V	28 V
Analog in Voltage	V <sub>ain</sub>	-0.5 V	V <sub>cc</sub> + 0.5 V
Digital in Voltage	V <sub>din</sub>	-0.5 V	V <sub>cc</sub> + 0.5 V
Logic supply voltage	V <sub>cc</sub>	-0.5 V	5.5 V
I2C/RXD/PMSELECT pin voltage	V <sub>I2C</sub>	-0.5V	V <sub>cc</sub> + 0.5 V
RST pin Voltage	V <sub>RST</sub>	-0.5 V	13 V
RS-485 voltage (transient)	V <sub>485</sub>	-50 V	50 V
Logic supply current for off board use	I <sub>vcc</sub>	0 mA	100 mA
I2C/TXD/PM output current	I <sub>I2C</sub>	-20 mA	20 mA
Digital out current	I <sub>dout</sub>	-20 mA	20 mA
RS-485 output current - driver	I <sub>485D</sub>	-250 mA	250 mA
RS-485 output current - receiver	I <sub>485R</sub>	-24 mA	24 mA
Operating Temperature	T	-25 °C	125 °C
Humidity (non condensing)	H	-	85.00%

### 5.2. Operating Conditions

### Operating Conditions

Parameter	Symbol	Min	Max
Input Voltage	V <sub>in</sub>	5V	24 V
Analog in Voltage	V <sub>ain</sub>	0V	V <sub>cc</sub>
Digital in Voltage	V <sub>din</sub>	0V	V <sub>cc</sub>
Logic supply voltage	V <sub>cc</sub>	5 V	5 V
I2C/RXD/PMSELECT pin voltage	V <sub>I2C</sub>	0 V	V <sub>cc</sub>
RST pin Voltage	V <sub>RST</sub>	0 V	V <sub>cc</sub>
RS-485 voltage	V <sub>485</sub>	-7 V	12 V
Logic supply current for off board use	I <sub>vcc</sub>	0 mA	50 mA
I2C/TXD/PM output current	I <sub>I2C</sub>	-10 mA	10 mA
Digital out current	I <sub>dout</sub>	-10 mA	10 mA
RS-485 output current - driver	I <sub>485</sub>	-60 mA	60 mA
RS-485 output current - receiver	I <sub>485</sub>	-8 mA	8 mA
Operating Temperature	T	0 °C	80 °C

### 5.3. DC and timing characteristics

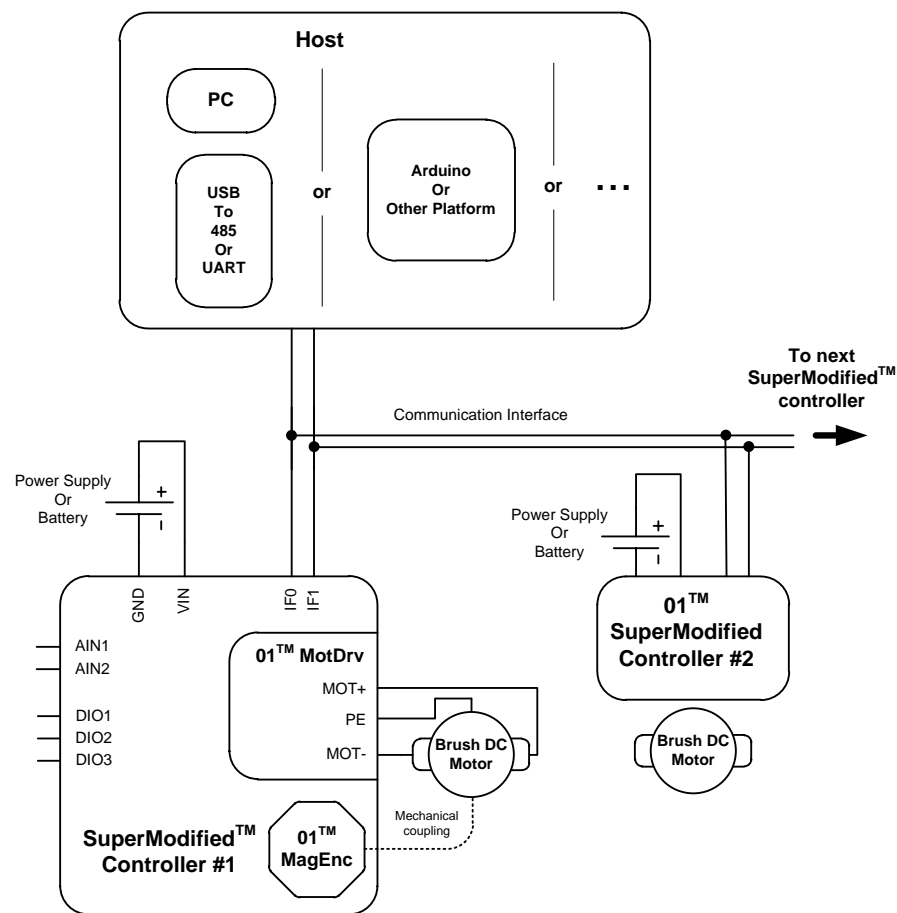
Parameter	Symbol	Typical
Input current, controller only	$I_{CONTR}$	20 mA
Motor current	$I_{MOT}$	5 A
Input current total	$I_{TOT}$	$I_{CONTR} + I_{MOT}$
DIO1-3 output source current	$I_{DIOSRC}$	5 mA
DIO1-3 output sink current	$I_{DIOSNK}$	20mA

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

IC	Function	Related pins
<a href="#">ATMega328P</a>	Microcontroller  Note: 01 <sup>TM</sup> SuperModified on-board crystal at 20MHz	Digital IOs (DIO1-3 ) Analog Ins (AIN1-2) RST I2C pins (SDA,SCL)
<a href="#">MIC5235</a>	Voltage regulator	VCC, GND
<a href="#">MC34931</a>	01 <sup>TM</sup> MotDrv	MOT+, MOT-, VIN, GND,
<a href="#">ISL3152E</a>	RS-485 transceiver	A_485, B_485

# 6. Connections

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



Pins		Connect to			
VIN		Regulated DC power supply capable of delivering 5 A. Or connect to a battery.			
GND					
SDA SCL A B ICP		Interface	SuperModified controller	Host Controller	Note
	I2C		SDA = SDA	SDA	Appropriate pull up resistors on host
			SCL = SCL	SCL	
	UART		A=TXD	RXD	120 Ohm terminal resistors at host and last controller.
			B=RXD	TXD	
RS-485		A=A	A (non inv.)	Custom configurations and features upon request	
		B=B	B (inverting)		
RC-Servo	ICP		Pulse modulated Input		
MOT+		Brush DC motor power leads. If connected with wrong polarity motor control cannot be achieved.			
MOT-					

---

<b>PE</b>	Motor chassis.
<b>DIO1-3</b>	External digital IOs. Output 0-5V, 20mA sink current, 5mA source current. Input 0-5V, impedance >10KOhm.
<b>AIN1-2</b>	External analog voltages 0-5V from sensors etc.

---

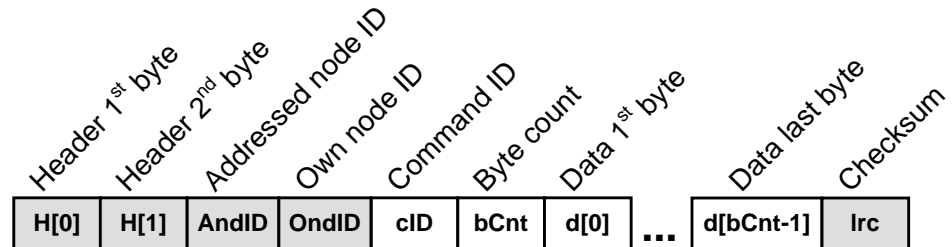
---




## 7. 01Mech Protocol

01 Mechatronics specifies a proprietary software layer protocol for use with the 01™ SuperModified controller and other 01 Mechatronics products. With 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 and implementation specifics according to data link layers defined by standards are analyzed at the specific data link layers.

### 7.1. Frame format

The general 01Mech protocol frame format is presented below



-  Optional according to if provided by data link layer
-  Mandatory regardless of data link layer
-  One box equals 1 byte

The 01Mech protocol specifies two types of data transfers.

- **Send command data transfer:** A bus master commands a bus slave (e.g. a 01™ SuperModified 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.

#### 7.1.1. 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.

#### 7.1.2. 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 01™ SuperModified controller with node ID = 0x04 to do something then: OndID = 0x01 and AndID = 0x04. When the 01™ SuperModified controller responds with a command response then OndID = 0x04 and AndID = 0x01.

#### 7.1.3. 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 01™ SuperModified 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 01™ SuperModified controller.

- **Broadcast set commands:** A bus master commands all slaves on the bus to do something: e.g. a master commands all 01™ SuperModified controllers attached to the bus to halt.

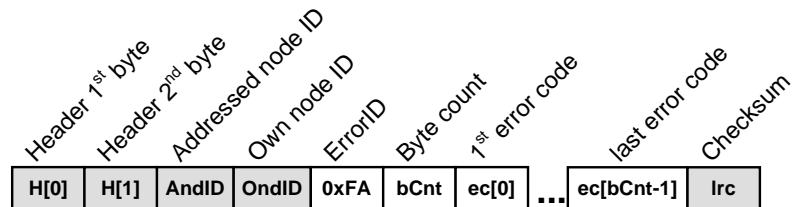
According to command classification, the 01Mech protocol specifies command ID ranges. The table below summarizes the latter statement.

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.

#### 7.1.4. 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 01™ SuperModified controller they will be reported using this special command ID as illustrated in the below error frame schematic. For a full list of error codes refer to section error code reference.



#### 7.1.5. 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.

#### 7.1.6. Data

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

#### 7.1.7. 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:

```
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;
}
```

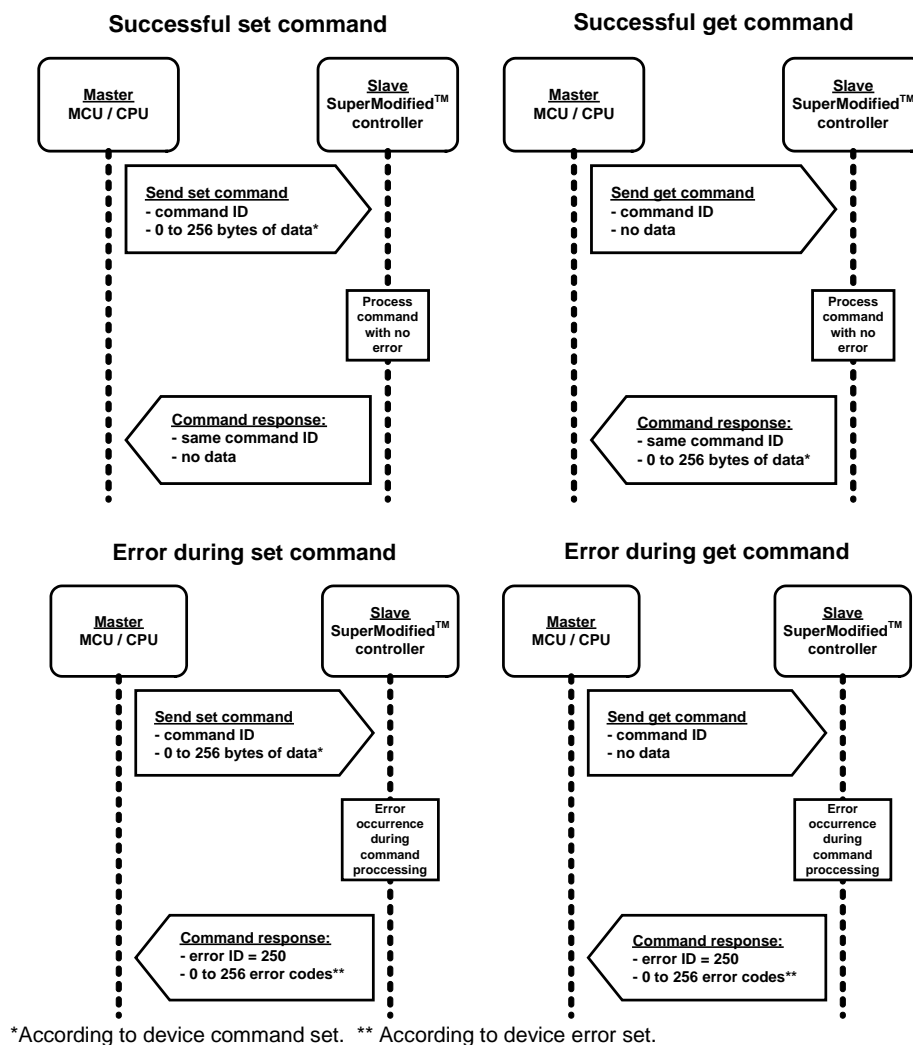
## 7.2. Transmission byte order

## 7.3. Control flow & errors

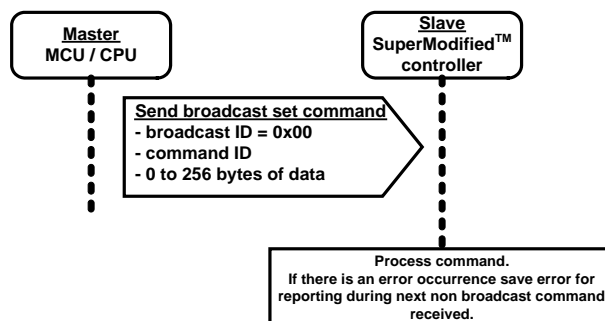
In some data link standards a checksum functionality is included (e.g. CAN bus). When the 01Mech protocol is implemented over such data link standards the checksum bytes are omitted.

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

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



### 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 01™ SuperModified controller a synchronous error can be e.g. the reception of an erroneous checksum where as an asynchronous error can be e.g. a detection of a motor stalled condition.

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 01™ SuperModified controller is specified in section 15 of this document.

### 7.3.1. Communication flow examples

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 SuperModified 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 01™ SuperModified 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 SuperModified 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 01™ SuperModified 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 SuperModified 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 01™ SuperModified 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 SuperModified 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 01<sup>TM</sup> SuperModified 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 SuperModified 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 01<sup>TM</sup> SuperModified 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 SuperModified 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 01<sup>TM</sup> SuperModified 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[1] = over-current error code, lrc

**Example 7:** Master sends a 'broadcast stop' command (set command with 0 bytes of data) to the all SuperModified 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 01<sup>TM</sup> SuperModified controllers on the bus with any ID accept the command and do not issue a command response:

### 7.3.2. Recommended master operation

The master of a bus should wait on the slave response with a timeout – in case the slave does not respond. Example source code can be found in 01<sup>TM</sup> Mechatronics Arduino library.

### 7.4. Error reset

Whenever an error occurs the 01<sup>TM</sup> device will maintain this error and continue reporting it with an error response until it is explicitly reset by the execution of an Error Reset command. Refer to the commands reference section for more information.

## 8. I2C Interface

The 01<sup>TM</sup> SuperModified 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 01<sup>TM</sup> SuperModified I2C hardware please consult the [ATMega328p datasheet](#).

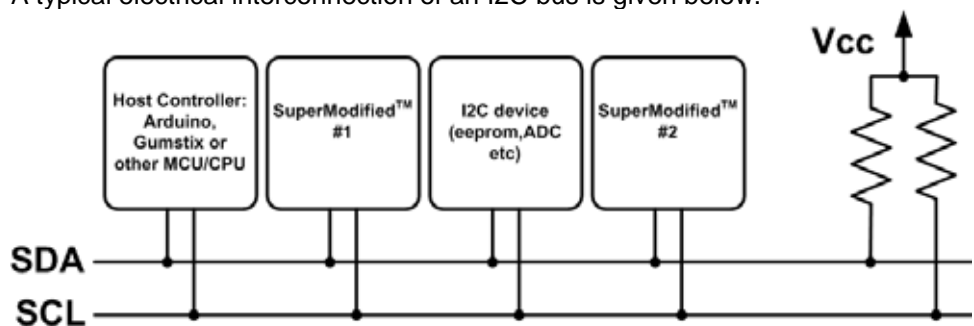
### 8.1. Electrical layer

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.

#### 8.1.1. Connections

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



#### 8.1.2. Bus speeds

The following two bus speed modes are supported by the 01<sup>TM</sup> SuperModified controller.

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

#### 8.1.3. Pull up resistor sizing

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

##### 100KHz Operation

$$\text{Minimum pull-up: } R_{p_{\min}} = \frac{V_{cc} - 0.4V}{3mA}, \text{ Maximum pull-up } R_{p_{\max}} = \frac{1000ns}{C_b}$$

$C_b$  : Capacitance of one bus line in pF.

##### 400KHz Operation

$$\text{Minimum pull-up: } R_{p_{\min}} = \frac{V_{cc} - 0.4V}{3mA}, \text{ Maximum pull-up } R_{p_{\max}} = \frac{300ns}{C_b}$$

Recommended values for 5V systems : 2.2 kOhm

Recommended values for 3.3V systems : 1,5 kOhm

#### 8.1.4. 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 cable 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 01<sup>TM</sup> SuperModified 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 where it should be avoided (without repeaters) for longer buses.

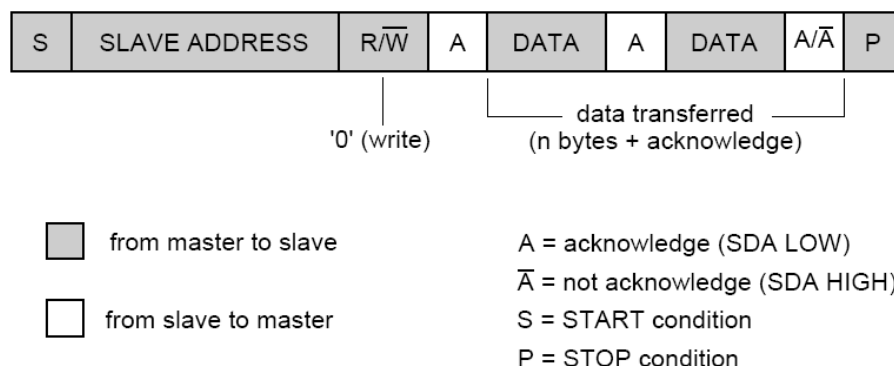
#### 8.2. Data link layer

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

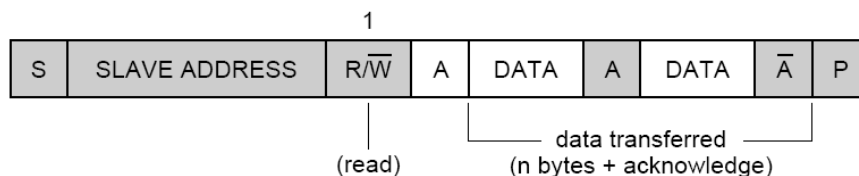
- Single message where a master writes data to a slave;
- Single message where a master reads data from a slave;
- Combined messages, where a master issues at least two reads and/or writes to one or more slaves.

Below the two most essential types of messages are presented.

##### Master transmit message



##### Master receive message



#### 8.2.1. SuperModified slave addresses

01<sup>TM</sup> SuperModified controllers come preprogrammed with a slave address of 0x04.

If more than one 01<sup>TM</sup> SuperModified controller is ordered slave addresses increase by 1 for every additional controller.

e.g.

01<sup>TM</sup> SuperModified controller #1 -> slave address 0x04

01<sup>TM</sup> SuperModified controller #2 -> slave address 0x05

And so on.

01<sup>TM</sup> SuperModified slave addresses be adjusted by connecting only one of the controllers on the bus each time and execute a SetNodeId Command. See commands reference for more information

### 8.2.2. General call address

The I2C bus specification includes some special addresses. One of them utilized by the 01™ SuperModified 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 :**

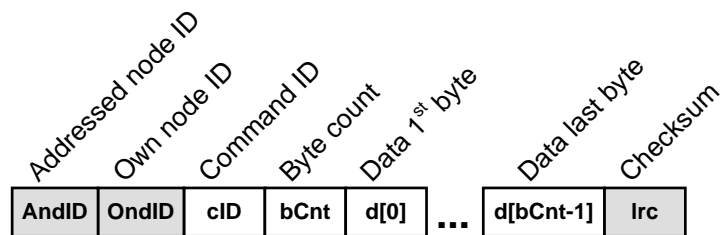
Slave address	R / $\overline{W}$ bit
0000 000	0

## 8.3. 01Mech Protocol over I2C

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

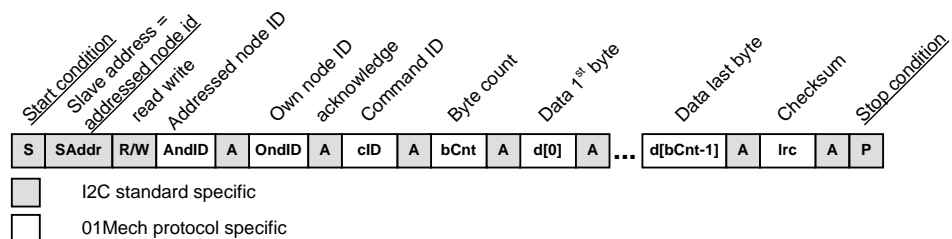
### 8.3.1. Frame structure

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.



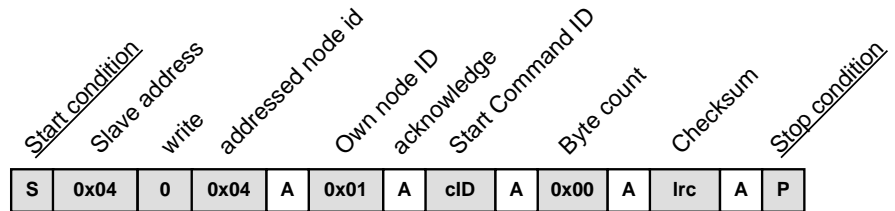
### 8.3.2. Communication flow control

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 01™ SuperModified controller take turns at being masters on the I2C bus. A transaction is always initiated by the host controller.

### 8.3.3. Communication example over I2C

**Example scenario:** The host controller with ID = 0x01 sends a 'start command' to the 01™ SuperModified 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)



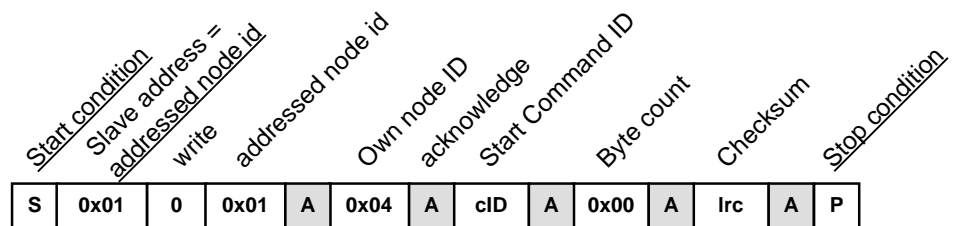
From host controller to SuperModified™ controller



From SuperModified™ controller to host controller

The host controller then waits (with a timeout) for the 01™ SuperModified to respond on the I2C bus.

2. The SuperModified 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 01™ SuperModified controller will never initiate a master transmit unless a command is received.

## 9. 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.

An 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 01™ SuperModified controller please refer to the [ATMega328p datasheet](#) at the USART0 section.

### 9.1. Electrical Layer

Electrical layer specifics for the RS-485 interface and the UART interface as they are utilized for the 01™ SuperModified controller:

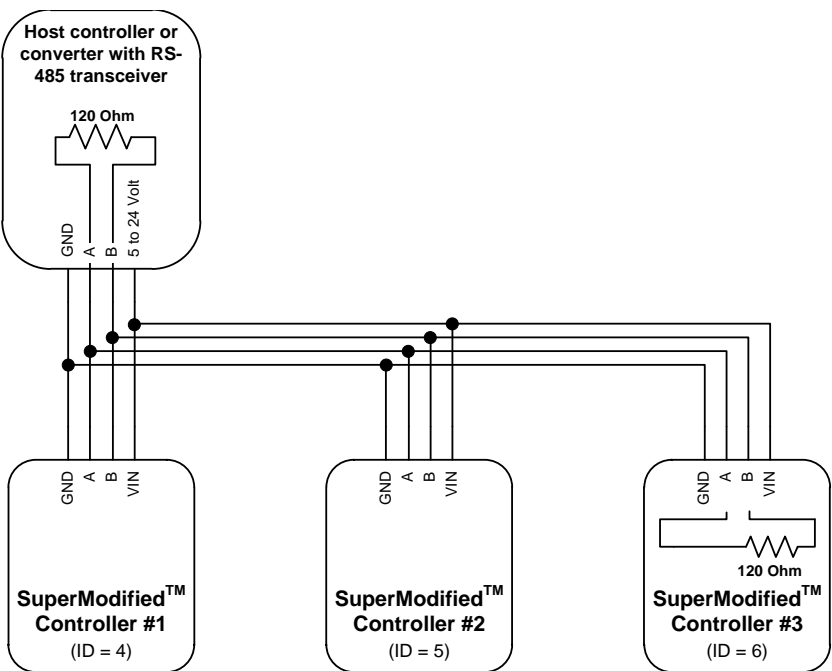
#### 9.1.1. RS-485 specification

##### RS-485 electrical layer specifics

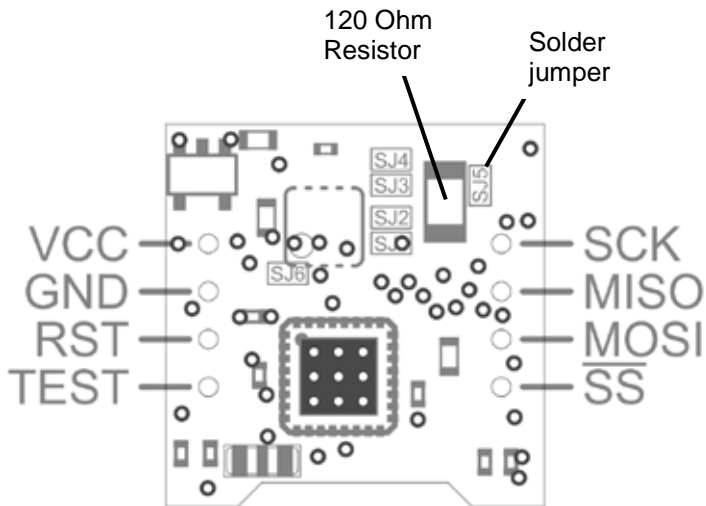
Physical Media :	Twisted Pair
Network Topology :	Point-to-point, Multi-dropped, Multi-point
Maximum Distance :	1200 meters
Mode of Operation :	Differential
Voltage Levels :	+/-6V (Commonly used)
Mark(1) :	Negative Voltages
Space(0) :	Positive voltages
Signals (half-duplex):	A = non inverting Tx/Rx B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

#### 9.1.2. RS-485 Connections

When one or more 01™ SuperModified controllers are connected on a RS-485 bus, connections should be as illustrated below.



The 01™ SuperModified controller includes an already soldered 120 Ohm (SMD1206) RS-485 terminal resistor. It can be easily connected by means of solder jumper SJ5.



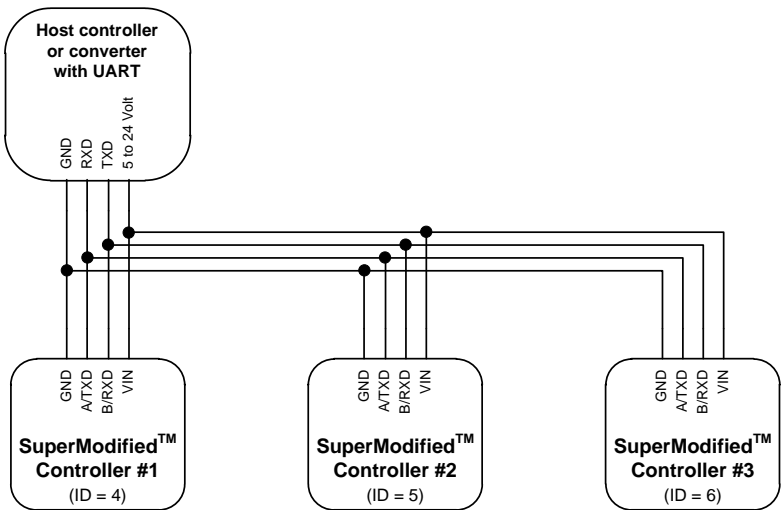
9.1.3. UART specification

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 01™ SuperModified controller RS-232 compliant.

9.1.4. UART connections

When using the UART interface of the 01™ SuperModified controller connections should be as illustrated below:





---

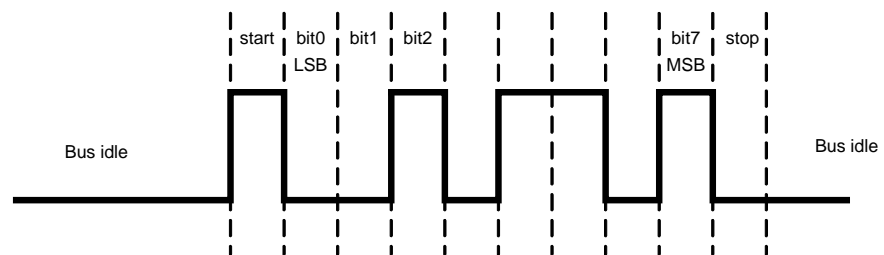
Note that the same connections can be used if an external RS-232 transceiver is used with every 01™ SuperModified controller.

### 9.1.5. 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 SuperModified controller.

## 9.2. 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:



## 9.3. 01Mech Protocol over UART

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

The host controller is the only master of the UART bus and the only device that can initiate a transaction.

## 10. RC Servo Interface

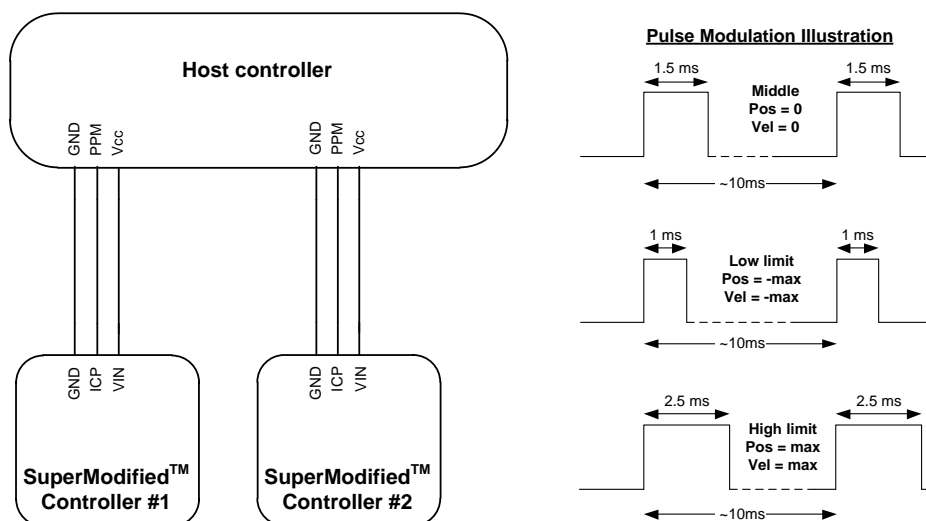
The 01™ SuperModified controller offers compatibility with the very popular RC Servo Position Pulse Modulation interface.

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

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

### 10.1. Connections

The 01™ SuperModified controller, when configured for RC-Servo interface, should be connected following the principles illustrated below:



The pulse modulation scheme is also illustrated.

### 10.2. RC-Servo Interface limitations and capabilities

The RC-Servo interface does not support all available commands of the 01™ SuperModified controller. i.e. no feedback on position or velocity etc. However it provides full motion capabilities in a straight-forward way. 01™ SuperModified servos can be pre-configured by 01 Mechatronics to have custom position limits, execute multiple turns for PPM commands, execute velocities or profiles etc. Please contact us at [info@01mechatronics.com](mailto:info@01mechatronics.com) for more details.

## 11. 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 01™ SuperModified 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.

### 11.1. 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 01™ SuperModified controller it can be -2147483648 to +2147483648 ticks (32 bit signed integer).

The 01™ SuperModified controller when interfaced with the 01™ MagEnc absolute encoder can get incremental and absolute readings for rotary position. The position acquired by the 01™ SuperModified controller through the 01™ MagEnc absolute interface will be referred to as **Absolute Position**. Note that this position can have values 0-32767 ticks (which is the 01™ 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. Henceforth we will refer to this position simply as **Relative Position**.

## 11.2. Set commands

### 11.2.1. 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)**

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

### 11.2.2. 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)**

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

### 11.2.3. 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	OnID	cID	bCnt	D[0]	D[1]	Irc
frame	0x55	0xAA	0x04	0x01	0x02	0x02	0x10	0x27	0x37
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc		
frame	0x55	0xAA	0x01	0x04	0x02	0x00	0x02		

### 11.2.4. Set profile acceleration

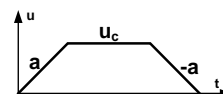
Command ID: **0x03** Name: **Set profile acceleration**

Tx data bytes: **4** *interpreted as:* 32 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<sup>2</sup>.  
Default value:  
10000 (for RC servos)



**Example: Set profile acceleration = 8000 (0x1F40) ticks/sec<sup>2</sup>**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x03	0x04	0x40	0x1F	0x00	0x00	0x58
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x03	0x00	0x03				

### 11.2.5. Set profile constant velocity

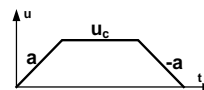
Command ID: **0x04** Name: **Set profile constant velocity**

Tx data bytes: **4** *interpreted as:* 32 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:  
30000 for RC servos



**Example: Set constant velocity = 15000 (0x03A98) ticks/sec**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x04	0x04	0x98	0x3A	0x00	0x00	0xA2
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x04	0x00	0x04				

### 11.2.6. 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: 5000.



⚠ 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]	Irc
frame	0x55	0xAA	0x04	0x01	0x05	0x02	0x84	0x03	0x80
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc		
frame	0x55	0xAA	0x01	0x04	0x05	0x00	0x05		

#### 11.2.7. 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 responds immediately to the error according to ErrorReaction.		
Notes:	Stored in EEPROM. Measured in mSec. Default value: 2000 ⚠ 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]	Irc
frame	0x55	0xAA	0x04	0x01	0x06	0x02	0x10	0x27	0x33
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc		
frame	0x55	0xAA	0x01	0x04	0x06	0x00	0x06		

#### 11.2.8. Move with velocity

Command ID:	0x07	Name:	Move with velocity
Tx data bytes:	4	interpreted as:	32 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 +5000 (0x1388) ticks/sec**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x07	0x04	0x88	0x13	0x00	0x00	0x98
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x07	0x00	0x07				

#### 11.2.9. Move to absolute position

Command ID:	0x08	Name:	Move to absolute position
Tx data bytes:	8	interpreted as:	64 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 (0x0002800) ticks**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x08	0x08	0x00	0x28	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	Irc							
	0x00	0x00	0x00	0x28							



### 11.2.10. Move to relative position

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x08	0x00	0x08

Command ID: **0x09** Name: **Move to relative position**

Tx data bytes: **8** interpreted as: 64 bit signed integer

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

Description: Configures the relative 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 (0x FFFFFFFF448) ticks relative to current position**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x09	0x08	0x48	0xF4	0xFF	0xFF	0xFF

Host	D[5]	D[6]	D[7]	Irc
	0xFF	0xFF	0xFF	0xBD

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x09	0x00	0x09

### 11.2.11. Profiled move with velocity

Command ID: **0x0A** Name: **Profiled move with velocity**

Tx data bytes: **4** interpreted as: 32 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]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x0A	0x04	0x0C	0xFE	0xFF	0xFF	0xFC

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x0A	0x00	0x0A

### 11.2.12. Profiled move to absolute position

Command ID: **0x0B** Name: **Profiled move to absolute position**

Tx data bytes: **8** interpreted as: 64 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 (0x FFFFFFFF448) ticks**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x0B	0x08	0x48	0xF4	0xFF	0xFF	0xFF

Host	D[5]	D[6]	D[7]	Irc
frame	0xFF	0xFF	0xFF	0xBF

Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x0B	0x00	0x0B

### 11.2.13. Profiled move to relative position

Command ID:	<b>0x0C</b>	Name:	<b>Profiled move to relative position</b>
Tx data bytes:	<b>8</b>	interpreted as:	64 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Configures the relative 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.		

**Example: Move to 3000 (0x0000BB8) ticks relative to current position**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x0C	0x08	0xB8	0x0B	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	Irc							
frame	0x00	0x00	0x00	0xB7							
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x0C	0x00	0x0C				

### 11.2.14. Set velocity setpoint

Command ID:	<b>0x0D</b>	Name:	<b>Set velocity setpoint</b>
Tx data bytes:	<b>4</b>	interpreted as:	32 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores a velocity setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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 = 10000 (0x2710)**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x0D	0x04	0x10	0x27	0x00	0x00	0x3E
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x0D	0x00	0x0D				

### 11.2.15. Set absolute position setpoint

Command ID:	<b>0x0E</b>	Name:	<b>Set absolute position setpoint</b>
Tx data bytes:	<b>8</b>	interpreted as:	64 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores an absolute position setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x0E	0x08	0x40	0x00	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	Irc							
frame	0x00	0x00	0x00	0x46							
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x0E	0x00	0x0E				

### 11.2.16. Set relative position setpoint

Command ID:	<b>0x0F</b>	Name:	<b>Set relative position setpoint</b>
Tx data bytes:	<b>8</b>	interpreted as:	64 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores a position setpoint relative to the current command. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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 an absolute move to +64 (0x00000040) ticks**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x0F	0x08	0x40	0x00	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	Irc							
frame	0x00	0x00	0x00	0x47							
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x0F	0x00	0x0F				

### 11.2.17. Set profiled velocity setpoint

Command ID:	<b>0x10</b>	Name:	<b>Set profiled velocity setpoint</b>
Tx data bytes:	<b>4</b>	interpreted as:	32 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores relative velocity setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x10	0x04	0x0F	0x03	0x00	0x00	0x18
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x10	0x00	0x10				

### 11.2.18. Set profiled absolute position setpoint

Command ID:	<b>0x11</b>	Name:	<b>Set profiled absolute position setpoint</b>
Tx data bytes:	<b>8</b>	interpreted as:	64 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores an absolute position setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x11	0x08	0x00	0x00	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	Irc							
frame	0x00	0x00	0x00	0x19							
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x11	0x00	0x11				



### 11.2.19. Set profiled relative position setpoint

Command ID:	<b>0x12</b>	Name:	<b>Set profiled relative position setpoint</b>
Tx data bytes:	<b>8</b>	interpreted as:	64 bit signed integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Stores a relative position setpoint. Used in conjunction with broadcast command "Do move" for synchronized motions of many 01™ SuperModified 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 = 100000 ticks (0x186A0)**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x04</b>	<b>0x01</b>	<b>0x12</b>	<b>0x08</b>	<b>0xA0</b>	<b>0x86</b>	<b>0x01</b>	<b>0x00</b>	<b>0x00</b>
Host	D[5]	D[6]	D[7]	lrc							
frame	<b>0x00</b>	<b>0x00</b>	<b>0x00</b>	<b>0x3D</b>							
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	lrc				
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x01</b>	<b>0x04</b>	<b>0x12</b>	<b>0x00</b>	<b>0x12</b>				

### 11.2.20. Configure digital IOs

Command ID:	<b>0x13</b>	Name:	<b>Configure digital IOs</b>
Tx data bytes:	<b>1</b>	interpreted as:	8 bit unsigned integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Configures whether the three on-board digital IOs DIO1–DIO3 will be inputs or outputs.		
Notes:	Stored in EEPROM. The three 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 DIO3 as outputs, DIO2 as input**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	lrc
frame	0x55	0xAA	0x04	0x01	0x13	0x01	0x05	0x17
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	lrc	
frame	0x55	0xAA	0x01	0x04	0x13	0x00	0x13	

### 11.2.21. Set digital outputs

Command ID:	<b>0x14</b>	Name:	<b>Set digital outputs</b>
Tx data bytes:	<b>1</b>	interpreted as:	8 bit unsigned integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Configures the state of the on-board digital outputs.		
Notes:	The three 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 DIO3 high**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	lrc
frame	0x55	0xAA	0x04	0x01	0x14	0x01	0x04	0x11
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	lrc	
frame	0x55	0xAA	0x01	0x04	0x14	0x00	0x14	

### 11.2.22. 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 01™ SuperModified Controller.

Notes: Stored to EEPROM. After execution of this command the controller no longer responds to its previous ID. Turn power off and then on (HW reset) is required after this command. Recommended to be executed with only one node on the bus.

**Example: Change controller ID from 0x04 to 0x05**

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

### 11.2.23. 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 01™ SuperModified controller to be addressed by using more than one IDs. This command is actually used to allow for groups of 01™ SuperModified 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 SuperModified 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	OnID	cID	bCnt	D[0]	Irc
frame	0x55	0xAA	0x10	0x01	0x16	0x01	0xF0	0xE7
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc	
frame	0x55	0xAA	0x01	0x10	0x16	0x00	0x16	

#### 11.2.24. 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 115200 Bps After execution of this command and a HW reset the controller no longer responds to previous baud rate.		

**Example: Change controller UART baud rate to 19200 Bps**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x04	0x01	0x17	0x04	0x00	0x4B	0x00	0x00	0x58
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x17	0x00	0x17				

#### 11.2.25. 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 (if it was started) 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.		

**Example: Reset Incremental position**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x04	0x01	0x18	0x00	0x18
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x18	0x00	0x18

#### 11.2.26. Start

Command ID:	0x19	Name:	Start
Tx data bytes:	0	interpreted as:	-
Rx data bytes:	0	interpreted as:	-
Description:	Initializes the 01 <sup>TM</sup> SuperModified Ccontroller.		
Notes:	The 01 <sup>TM</sup> SuperModified upon power up is not executing any type of control, thus not applying any voltage to the attached motor. When this command is received incremental position is reset, memory is initialized and the controller enters position control mode with position setpoint = 0.		

**Example: Start the 01<sup>TM</sup> SuperModified controller**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x04	0x01	0x19	0x00	0x19
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x01	0x04	0x19	0x00	0x19



### 11.2.27. Halt

Command ID: **0x1A** Name: **Halt**

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

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

Description: Stops the motor.

Notes: The 01™ SuperModified controller is switched to position control with position setpoint = current position. The Halt command is valid only after a start command has been issued.

**Example: Halt the motor attached to the 01™ SuperModified controller.**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x04</b>	<b>0x01</b>	<b>0x1A</b>	<b>0x00</b>	<b>0x1A</b>
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x01</b>	<b>0x04</b>	<b>0x1A</b>	<b>0x00</b>	<b>0x1A</b>

### 11.2.28. Stop

Command ID: **0x1B** Name: **Stop**

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

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

Description: Un - Initializes the 01™ SuperModified Controller.

Notes: The 01™ SuperModified controller does not execute any control loop.

**Example: Start the 01™ SuperModified controller**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x04</b>	<b>0x01</b>	<b>0x1B</b>	<b>0x00</b>	<b>0x1B</b>
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	lrc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x01</b>	<b>0x04</b>	<b>0x1B</b>	<b>0x00</b>	<b>0x1B</b>

### 11.2.29. Set error reaction

Command ID: **0x1C** Name: **Set error reaction**

Tx data bytes: **20** *interpreted as:* 20 x 8bit unsigned integers

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

Description: The 01™ Supermodified controller recognizes a plethora of errors that might occur during operation. They are described extensively in the error reference section. Below is a list of these errors ordered by error ID.

- Motor Stalled
- Encoder Overflow
- Encoder Underflow
- Motor Overcurrent
- Encoder Health
- Invalid command ID
- Invalid set command bytecount
- Invalid command argument
- Invalid command for motor state
- Invalid get command bytecount
- I2C arbitration lost
- I2C packet override
- I2C invalid received bytecount
- UART memory allocation error
- UART received frame error
- UART received parity error
- UART receive buffer overflow
- UART data overwritten
- UART packet receive timeout
- Wrong LRC



The 01™ Supermodified controller can react to each of these errors immediately and without any interaction with the host controller.

This automatic, immediate reaction to errors is only active only if a start command has already been issued. Otherwise no action is taken upon an error occurrence.

There are 3 ways that the 01™ Supermodified controller can react to an error:

- Do nothing. This is represented by 0.
- Issue a STOP command, thus release any force applied by the motor. This is represented by 1.
- Issue a HALT command, thus stop moving but continue controlling the motor and maintain the position at which the error occurred. This is represented by 2.

Each byte of the setErrorReaction command configures how the 01™ Supermodified controller will react to each error. The order of the bytes in the packet is the opposite from the above error list, thus D[0] configures "UART received packet timeout", D[1] configures "UART data overwritten" and so on.

Notes: Stored to EEPROM.

Defaults:

Motor Stalled	: STOP
Encoder Overflow	: HALT
Encoder Underflow	: HALT
Motor Overcurrent	: STOP
Encoder Health	: STOP
Invalid command ID	: DO NOTHING
Invalid set command bytecount	: DO NOTHING
Invalid command argument	: DO NOTHING
Invalid command for motor state	: DO NOTHING
Invalid get command bytecount	: DO NOTHING
I2C arbitration lost	: HALT
I2C packet override	: HALT
I2C invalid received bytecount	: DO NOTHING
UART memory allocation error	: STOP
UART received frame error	: HALT
UART received parity error	: HALT
UART receive buffer overflow	: HALT
UART data overwritten	: HALT
UART packet receive timeout	: HALT
Wrong LRC	: DO NOTHING

**Example: Set reaction of encoder overflow and underflow error to STOP and everything else to DO NOTHING.**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]
frame	0x55	0xAA	0x04	0x01	0x1C	0x14	0x00	0x00	0x00	0x00	0x00
Host	D[5]	D[6]	D[7]	D[8]	D[9]	D[10]	D[11]	D[12]	D[13]	D[14]	D[15]
frame	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Host	D[16]	D[17]	D[18]	D[19]	Irc						
frame	0x00	0x01	0x01	0x00	0x08						
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc				
frame	0x55	0xAA	0x01	0x04	0x1C	0x00	0x1C				



### 11.2.30. Set anti-windup

Command ID:	<b>0x1D</b>	Name:	<b>Set Anti-windup</b>
Tx data bytes:	<b>4</b>	interpreted as:	32 bit unsigned integer
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Configures the PID anti-windup limit.		
Notes:	Stored to EEPROM. Default value: 80000 for RC-servos After execution of this command the setting is immediately updated. After a HW reset the setting is retained.		

**Example: Change anti-windup to 64000 (0xFA00)**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x04</b>	<b>0x01</b>	<b>0x1D</b>	<b>0x04</b>	<b>0x00</b>	<b>0xFA</b>	<b>0x00</b>	<b>0x00</b>	<b>0xE3</b>
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x01</b>	<b>0x04</b>	<b>0x1D</b>	<b>0x00</b>	<b>0x1D</b>				

### 11.2.31. Reset errors

Command ID:	<b>0x1E</b>	Name:	<b>Reset Errors</b>
Tx data bytes:	<b>0</b>	interpreted as:	-
Rx data bytes:	<b>0</b>	interpreted as:	-
Description:	Resets all controller errors.		
Notes:	The 01™ SuperModified controller resets all errors that have occurred up to that point in time. If an error occurs at any given time all responses after that will be error responses, that report the error, as well as any errors that have occurred afterwards. The only way to reset the error is by executing this command. This is done to ensure that the host includes error handling and is aware of the error occurrence.		

**Example: Reset errors**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x04</b>	<b>0x01</b>	<b>0x1E</b>	<b>0x00</b>	<b>0x1E</b>
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	<b>0x55</b>	<b>0xAA</b>	<b>0x01</b>	<b>0x04</b>	<b>0x1E</b>	<b>0x00</b>	<b>0x1E</b>

## 11.3. Get Commands

### 11.3.1. Get PID gain P

Command ID:	<b>0x64</b>	Name:	<b>Get PID gain P</b>
Tx data bytes:	<b>0</b>	interpreted as:	-
Rx data bytes:	<b>2</b>	interpreted as:	16-bit unsigned integer
Description:	Gets the proportional gain from the 01™ SuperModified 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	OnID	cID	bCnt	Irc		
frame	0x55	0xAA	0x04	0x01	0x64	0x00	0x64		
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	Irc
frame	0x55	0xAA	0x01	0x04	0x64	0x02	0xD0	0x07	0xB1

### 11.3.2. Get PID gain I

Command ID:	<b>0x65</b>	Name:	<b>Get PID gain I</b>
Tx data bytes:	<b>0</b>	interpreted as:	-
Rx data bytes:	<b>2</b>	interpreted as:	16-bit unsigned integer
Description:	Gets the integral gain from the 01™ SuperModified 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	Irc		
frame	0x55	0xAA	0x04	0x01	0x65	0x00	0x65		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	Irc
frame	0x55	0xAA	0x01	0x04	0x65	0x02	0xE8	0x03	0x8C

### 11.3.3. 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 01<sup>TM</sup> SuperModified 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	Irc		
frame	0x55	0xAA	0x04	0x01	0x66	0x00	0x66		
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	Irc
frame	0x55	0xAA	0x01	0x04	0x66	0x02	0x10	0x27	0x53

### 11.3.4. Get profile acceleration

Command ID: **0x67** Name: **Get profile acceleration**

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

Rx data bytes: **4** interpreted as: 32-bit unsigned integer

Description: Gets the profile acceleration in ticks/sec<sup>2</sup>.

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	Irc				
frame	0x55	0xAA	0x04	0x01	0x67	0x00	0x67				
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x01	0x04	0x67	0x04	0x20	0x03	0x00	0x00	0x40

### 11.3.5. Get profile constant velocity

Command ID: **0x68** Name: **Get profile constant velocity**

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

Rx data bytes: **4** interpreted as: 32-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	Irc				
frame	0x55	0xAA	0x04	0x01	0x68	0x00	0x68				
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x01	0x04	0x68	0x04	0x20	0x03	0x00	0x00	0x4F

### 11.3.6. 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	Irc
------	------	------	-------	-------	-----	------	-----



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**

### 11.3.7. 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.  
**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**

### 11.3.8. 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. The 3 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 0x07 0x6D**

### 11.3.9. 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 explained 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**

### 11.3.10. 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 the three lower bits of received byte. DIO1 is represented by bit 0, DIO2 by bit 1 and DIO3 is represented by bit 2.





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	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x04	0x01	0x6D	0x00	0x6D
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0] Irc
frame	0x55	0xAA	0x01	0x04	0x6D	0x01	0x00 0x6C

### 11.3.11. 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 two analog inputs AIN1-2. Also reads the voltage present on DIO1 and DIO2, thus allowing them to be used as analog inputs as well. D[0] and D[1] contain the value corresponding to AIN1, D[2] and D[3] the value corresponding to AIN2, D[4] and D[5] the value corresponding to the voltage DIO1 and so on.

Notes: On board ADC resolution is 10 bits. Voltage reference is 5V and corresponds to 1024 ADC reading. A 50% weighted running average filter is applied on readings. Sampling rate is approximately 2KHz.

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

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x04	0x01	0x6E	0x00	0x6E
Slave	H[0]	H[1]	AndID	OnID	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] Irc
							0x02 0x00 0x02 0x66

### 11.3.12. Get position

Command ID: **0x6F** Name: **Get position**

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

Rx data bytes: **8** interpreted as: 64-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	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x04	0x01	0x6F	0x00	0x6F
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0] D[1] D[2] D[3] D[4]
frame	0x55	0xAA	0x01	0x04	0x6F	0x08	0x33 0x16 0x00 0x00 0x00
Slave							D[5] D[6] D[7] Irc
frame	0x00	0x00	0x00	0x42			

### 11.3.13. 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 15bits. This is the raw reading from the absolute encoder interface.

Notes: Only available if a 01™ MagEnc magnetic absolute encoder is interfaced to the 01™ SuperModified controller.



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

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

#### 11.3.14. Get velocity

Command ID:	0x71	Name:	Get velocity
Tx data bytes:	0	interpreted as:	-
Rx data bytes:	4	interpreted as:	32-bit signed integer
Description:	Gets the current velocity in ticks/sec.		
Notes:	Only available if a 01 <sup>IM</sup> MagEnc magnetic absolute encoder is interfaced to the 01 <sup>TM</sup> SuperModified controller.		

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

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc				
frame	0x55	0xAA	0x04	0x01	0x71	0x00	0x71				
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x01	0x04	0x71	0x04	0x90	0x01	0x00	0x00	0xE4

#### 11.3.15. 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:	-		

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

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

#### 11.3.16. Get Error Reaction

Command ID:	0x73	Name:	Get error reaction
Tx data bytes:	0	interpreted as:	-
Rx data bytes:	20	interpreted as:	20 x 8-bit unsigned integer
Description:	Gets the configuration of the error reaction settings.		
Notes:	See the Set Error Reaction command for a full description of the configuration details. Each byte represents the setting for the error reaction to the corresponding error.		

**Example: Get Error Reaction (e.g. defaults are received)**

Host	H[0]	H[1]	AndID	OndID	cID	bCnt	Irc					
frame	0x55	0xAA	0x04	0x01	0x73	0x00	0x73					
Slave	H[0]	H[1]	AndID	OndID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	D[4]	
frame	0x55	0xAA	0x01	0x04	0x73	0x14	0x00	0x02	0x02	0x02	0x02	
Slave	D[5]	D[6]	D[7]	D[8]	D[9]	D[10]	D[11]	D[12]	D[13]	D[14]	D[15]	
frame	0x02	0x01	0x00	0x02	0x02	0x00	0x00	0x00	0x00	0x00	0x01	
Slave	D[16]	D[17]	D[18]	D[19]	Irc							
frame	0x01	0x02	0x02	0x01	0x67							

### 11.3.17. Get Antiwindup

Command ID: **0x74** Name: **Get antiwindup**

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

Rx data bytes: **4** interpreted as: 32-bit unsigned integer

Description: Gets the PID anti-windup limit.

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

**Example: Get antiwindup (which is e.g. 80000 = 0x013880)**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc				
frame	0x55	0xAA	0x04	0x01	0x74	0x00	0x74				
Slave	H[0]	H[1]	AndID	OnID	cID	bCnt	D[0]	D[1]	D[2]	D[3]	Irc
frame	0x55	0xAA	0x01	0x04	0x74	0x04	0x80	0x38	0x01	0x00	0xC9

## 11.4. Broadcast commands

### 11.4.1. Do move

Command ID: **0xC8** Name: **Do move**

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

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

Description: All controllers execute their pre-buffered setpoints.

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	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x00	0x01	0xC8	0x00	0xC8

### 11.4.2. Global start

Command ID: **0xC9** Name: **Global start**

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

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

Description: Starts all 01<sup>TM</sup> SuperModified controllers on the bus.

Notes: -

**Example: Start all 01<sup>TM</sup> SuperModified controllers on the bus.**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x00	0x01	0xC9	0x00	0xC9

### 11.4.3. Global halt

Command ID: **0xCA** Name: **Global halt**

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

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

Description: Halts all 01<sup>TM</sup> SuperModified controllers on the bus by applying position control and position setpoint = current position.

Notes: -

**Example: Halt all 01<sup>TM</sup> SuperModified controllers on the bus.**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x00	0x01	0xCA	0x00	0xCA

### 11.4.4. Global stop

Command ID: **0xCB** Name: **Global stop**

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

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

Description: De-activates all 01<sup>TM</sup> SuperModified controllers on the bus.

Notes: -

**Example: Stop all 01<sup>TM</sup> SuperModified controllers on the bus.**

Host	H[0]	H[1]	AndID	OnID	cID	bCnt	Irc
frame	0x55	0xAA	0x00	0x01	0xCB	0x00	0xCB



## 12. Error code reference

During operation, various errors may appear. Hardware related errors manifest in an asynchronous way in respect to communication frames.

Whenever an error is detected by the 01™ SuperModified controller it is reported 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 (refer to section 8.1.4).

According to the error code and the error reaction configuration (as described in the corresponding set command) action may or may not be taken by the controller's firmware.

Once an error has been detected it remains in memory until it is explicitly reset by executing a Reset Errors command. If the host controller does not issue a Reset Errors command after an error has occurred, the 01™ Supermodified controller will effectively stop executing new commands, as the last error together with any new ones will always force an error response. This is done to ensure that the host controller processes errors and resets them only if it is safe for the given application.

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

Code	Name	Default Reaction	Description:/cause/notes
<b>Motor/Encoder related errors</b>			
0x01	Motor Stalled	STOP	<u>Description:</u> A motor stalled condition has occurred. <u>Cause:</u> 1) The motor is actually stalled. 2) The motor has been instructed to achieve a non-achievable velocity. 3) Motor is connected with wrong polarity <u>Resolution:</u> Check your mechanical implementation. Make sure the motor is adequately sized for the intended purpose. Check the polarity of the motor connections
0x02	Encoder Overflow	HALT	<u>Description:</u> The incremental position is about to overflow. <u>Cause:</u> The motor has traveled extremely long towards the same direction. The encoder overflow error happens at $9,22 \times 10^{18}$ ticks and before the position registers have actually overflowed. <u>Resolution:</u> None. This error is almost impossible to occur.
0x03	Encoder Underflow	HALT	<u>Description:</u> The incremental position is about to underflow. <u>Cause:</u> The motor has traveled extremely long towards the same direction. The encoder underflow error happens at $-9,22 \times 10^{18}$ ticks and before the position registers have actually overflowed. <u>Resolution:</u> None. This error is almost impossible to occur.
0x04	Motor Overcurrent	STOP	<u>Description:</u> The motor has been subjected to current greater than the configured current limit for a duration that exceeds current limit duration. <u>Cause:</u> 1)The current limit has not been configured correctly. 2) The motor is stressed beyond its current limits <u>Resolution:</u> 1) Configure the current limit to a higher value. 2) Check your mechanical implementation and make sure the motor is adequately sized for the intended purpose.
0x05	Encoder Health	STOP	<u>Description:</u> The encoder cannot be read or communication to the encoder indicated an error. <u>Cause:</u> 1) Bad connections to the MagEnc absolute encoder. 2) Damaged encoder. 3) Damaged encoder parts. <u>Resolution:</u> Check the MagEnc plastic mounting parts for damage. Make sure that the MagEnc module is connected to the PicoMCU module.

Command Related Errors			
<b>0x11</b>	Invalid command ID	DO NOTHING	<p><u>Description:</u> A communication packet with a non-existent command ID has been received. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors.</p>
<b>0x12</b>	Invalid set command byte-count	DO NOTHING	<p><u>Description:</u> A set command with a wrong bytecount has been received. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors.</p>
<b>0x13</b>	Invalid argument	DO NOTHING	<p><u>Description:</u> A set command with an invalid data argument has been received. For example attempting to set the SuperModified node ID to 0. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors.</p>
<b>0x14</b>	Invalid command for motor state	DO NOTHING	<p><u>Description:</u> The received command is invalid for the given motor state. ie the 01<sup>TM</sup> SuperModified controller is instructed to move the motor with a specific velocity prior to receiving a Start command (initialization and PID activation). The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Issue a Start command before attempting to issue movement commands.</p>
<b>0x15</b>	Invalid get command byte-count	DO NOTHING	<p><u>Description:</u> A get or broadcast command has been received with invalid bytecount. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors</p>
I2C related errors			
<b>0x21</b>	I2C Arbitration lost	HALT	<p><u>Description:</u> The 01<sup>TM</sup> SuperModified controller lost arbitration when trying to issue a command response. The 01<sup>TM</sup> SuperModified controller tries another 5 times to issue the command response. If all tries fail an error response is issued.</p> <p><u>Resolution:</u> 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.</p>
<b>0x22</b>	I2C Packet override	HALT	<p><u>Description:</u> A communication packet override condition was detected.</p> <p><u>Cause:</u> Communication at this rate cannot be handled by the 01<sup>TM</sup> SuperModified controller.</p> <p><u>Resolution:</u> Introduce some delay between communication cycles with the 01<sup>TM</sup> SuperModified controller.</p>
<b>0x23</b>	I2C Invalid receive byte-count	DO NOTHING	<p><u>Description:</u> The received I2C packet had a different byte length than it should according to packet byte-count. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors</p>
UART related errors			
<b>0x31</b>	Memory allocation error	STOP	<p><u>Description:</u> Uart initialization detected not enough memory for uart buffers.</p> <p><u>Cause:</u> User programming (future implementation) of the controller has taken up all memory.</p> <p><u>Note:</u> The 01<sup>TM</sup> SuperModified controller cannot initiate Uart transactions or may not function at all.</p> <p><u>Resolution:</u> Reduce memory needed for your program inside the 01<sup>TM</sup> SuperModified controller.</p>
<b>0x32</b>	Frame error	HALT	<p><u>Description:</u> Uart detected a frame error. ie invalid number of data bits, invalid number of stop bits etc</p> <p><u>Cause:</u> 1) Incompatible host uart settings. 2) Extreme noise 3) Bad/damaged cabling</p>

			<p><u>Note:</u> The command involving the frame error is discarded. An error response is issued if possible during next transaction.</p> <p><u>Resolution:</u> Check host controller Uart settings. Check cabling.</p>
<b>0x33</b>	Parity error	HALT	<p><u>Description:</u> Uart detected a parity error.</p> <p><u>Cause:</u> 1) Incompatible host uart settings. 2) Extreme noise 3) Bad/damaged cabling</p> <p><u>Note:</u> The command involving the parity error is discarded. An error response is issued during next transaction.</p> <p><u>Resolution:</u> Check host controller Uart settings. Check cabling.</p>
<b>0x34</b>	Receive buffer overflow	HALT	<p><u>Description:</u> A receive buffer overflow was detected.</p> <p><u>Cause:</u> Communication at this rate cannot be handled by the 01™ SuperModified controller.</p> <p><u>Note:</u> An error response is issued.</p> <p><u>Resolution:</u> Introduce some delay between communication cycles with the 01™ SuperModified controller.</p>
<b>0x35</b>	Receive data override	HALT	<p><u>Description:</u> A receive data override condition was detected.</p> <p><u>Cause:</u> Communication at this rate cannot be handled by the 01™ SuperModified controller.</p> <p><u>Note:</u> An error response is issued.</p> <p><u>Resolution:</u> Introduce some delay between communication cycles with the 01™ SuperModified controller.</p>
<b>0x36</b>	Receive packet timeout	HALT	<p><u>Description:</u> Data reception stopped before a whole command was received and the receive timeout of 200ms has expired.</p> <p><u>Cause:</u> 1) Host controller error. 2) Damaged cabling/disconnection.</p> <p><u>Note:</u> An error response is issued.</p> <p><u>Resolution:</u> Check the host controller software for errors. Check cabling to the 01™ Supermodified controller.</p>
<b>Protocol related errors</b>			
<b>0x41</b>	Wrong LRC	DO NOTHING	<p><u>Description:</u> The received packet had a wrong lrc. The command is not executed. An error response is issued immediately.</p> <p><u>Resolution:</u> Check your communication software for possible errors.</p>

---

## 13. Contents

1. General Description .....	1
2. Features.....	1
3. Mechanical layout .....	2
3.1. 01™ MagEnc.....	3
3.2. 01™ PicoMcu .....	4
3.3. 01™ MotDrv .....	5
4. Pinout.....	6
4.1. 01™ MagEnc.....	6
4.1.1. Pin description .....	6
4.2. 01™ PicoMcu .....	7
4.2.1. Pin description .....	7
4.3. 01™ MotDrv .....	9
4.3.1. Pin description .....	9
5. Electrical Characteristics .....	10
5.1. Absolute Maximum ratings.....	10
5.2. Operating Conditions .....	10
5.3. DC and timing characteristics .....	11
6. Connections .....	12
7. 01Mech Protocol .....	14
7.1. Frame format.....	14
7.1.1. Header .....	14
7.1.2. Addressed and own node ID .....	14
7.1.3. Command ID & classification .....	14
7.1.4. Error frames.....	15
7.1.5. Byte Count.....	15
7.1.6. Data .....	15
7.1.7. Checksum.....	15
7.2. Transmission byte order.....	16
7.3. Control flow & errors .....	16
7.3.1. Communication flow examples.....	17
7.3.2. Recommended master operation .....	18
7.4. Error reset .....	18
8. I2C Interface .....	19
8.1. Electrical layer.....	19
8.1.1. Connections.....	19
8.1.2. Bus speeds.....	19
8.1.3. Pull up resistor sizing .....	19
8.1.4. Considerations.....	20
8.2. Data link layer .....	20
8.2.1. SuperModified slave addresses .....	20
8.2.2. General call address .....	21

8.3. 01Mech Protocol over I2C .....	21
8.3.1. Frame structure .....	21
8.3.2. Communication flow control .....	21
8.3.3. Communication example over I2C .....	21
9. UART, RS-485 interfaces .....	23
9.1. Electrical Layer.....	23
9.1.1. RS-485 specification .....	23
9.1.2. RS-485 Connections .....	23
9.1.3. UART specification .....	24
9.1.4. UART connections.....	24
9.1.5. Considerations.....	25
9.2. Data link layer .....	25
9.3. 01Mech Protocol over UART .....	25
10. RC Servo Interface .....	26
10.1. Connections .....	26
10.2. RC-Servo Interface limitations and capabilities .....	26
11. Command set .....	27
11.1. Position nomenclature.....	27
11.2. Set commands .....	27
11.2.1. Set PID gain P.....	27
11.2.2. Set PID gain I .....	27
11.2.3. Set PID gain D.....	28
11.2.4. Set profile acceleration.....	28
11.2.5. Set profile constant velocity.....	28
11.2.6. Set current limit.....	28
11.2.7. Set duration for current limit .....	29
11.2.8. Move with velocity .....	29
11.2.9. Move to absolute position.....	29
11.2.10. Move to relative position.....	30
11.2.11. Profiled move with velocity .....	30
11.2.12. Profiled move to absolute position .....	30
11.2.13. Profiled move to relative position .....	31
11.2.14. Set velocity setpoint .....	31
11.2.15. Set absolute position setpoint .....	31
11.2.16. Set relative position setpoint .....	32
11.2.17. Set profiled velocity setpoint.....	32
11.2.18. Set profiled absolute position setpoint .....	32
11.2.19. Set profiled relative position setpoint .....	33
11.2.20. Configure digital IOs.....	33
11.2.21. Set digital outputs.....	33
11.2.22. Set node ID.....	34
11.2.23. Set local acceptance mask.....	34



11.2.24. Set baud rate UART .....	35
11.2.25. Reset incremental position .....	35
11.2.26. Start .....	35
11.2.27. Halt .....	36
11.2.28. Stop .....	36
11.2.29. Set error reaction.....	36
11.2.30. Set anti-windup.....	38
11.2.31. Reset errors.....	38
11.3. Get Commands .....	38
11.3.1. Get PID gain P.....	38
11.3.2. Get PID gain I.....	38
11.3.3. Get PID gain D .....	39
11.3.4. Get profile acceleration .....	39
11.3.5. Get profile constant velocity .....	39
11.3.6. Get current limit .....	39
11.3.7. Get current limit duration .....	40
11.3.8. Get digital IO configuration.....	40
11.3.9. Get local acceptance mask .....	40
11.3.10. Get digital inputs.....	40
11.3.11. Get analog inputs .....	41
11.3.12. Get position .....	41
11.3.13. Get absolute position.....	41
11.3.14. Get velocity.....	42
11.3.15. Get current.....	42
11.3.16. Get Error Reaction .....	42
11.4. Broadcast commands .....	43
11.4.1. Do move .....	43
11.4.2. Global start .....	43
11.4.3. Global halt .....	43
11.4.4. Global stop .....	43
12. Error code reference.....	44

**Disclaimer:** The information in this document is provided in connection with 01 Mechatronics products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of 01Mechatronics products.

EXCEPT AS SET FORTH IN 01 MECHATRONICS TERMS AND CONDITIONS OF SALE LOCATED ON 01 MECHATRONICS WEB SITE, 01 MECHATRONICS ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL 01 MECHATRONICS BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF 01 MECHATRONICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

01 Mechatronics makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. 01 Mechatronics does not make any commitment to update the information contained herein. Unless specifically provided otherwise, 01 Mechatronics products are not suitable for, and shall not be used in, automotive applications. 01 Mechatronics's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2014 01 Mechatronics Corporation. All rights reserved. 01™ MECHATRONICS ®

01™ is a registered trademark of 01 Mechatronics. Other terms and product names may be trademarks of others.

