

# Mercury technical manual

December  
2020

v.1

# Mercury technical manual

[v.1](#)

## [Mercury technical manual](#)

### [1. Introduction](#)

### [2. Connection details](#)

#### [2.1 Pin assignments](#)

#### [2.2 Connecting multiple units](#)

#### [2.3 USB2Mercury](#)

#### [2.4 Mercury power hub](#)

#### [2.5 Power-up protocol](#)

### [3. Communication protocol](#)

#### [3.1 Overview](#)

#### [3.2 Protocol](#)

#### [3.3 Request packet](#)

##### [3.3.1 Overview](#)

##### [3.3.2 Checksum description](#)

##### [3.3.3 Unique servo ID](#)

#### [3.4 Status packet](#)

##### [3.4.1 Overview](#)

##### [3.4.2 Error description](#)

### [4. Instructions](#)

#### [4.1 Instructions table](#)

#### [4.2 Instruction details](#)

##### [4.2.1 PING](#)

[Instruction packet](#)

[Status packet](#)

##### [4.2.2 READ](#)

[Instruction packet](#)

[Status packet](#)

##### [4.2.3 WRITE](#)

[Instruction packet](#)

[Status packet](#)

##### [4.2.4 REG\\_WRITE](#)

[Instruction packet](#)

[Status packet](#)

##### [4.2.5 ACTION](#)

[Status packet](#)

##### [4.2.7 RESET](#)

[Status packet](#)

##### [4.2.7 REBOOT](#)

[Status packet](#)

## [5. Control registers](#)

### [5.1 Register table](#)

### [5.2 Register limits](#)

### [5.3 Register details](#)

#### [5.3.1 Control enable](#)

#### [5.3.2 Baud rates](#)

#### [5.3.3 Acknowledgement packet response time](#)

### [5.4 Operating modes](#)

#### [5.4.1 Operating modes table](#)

#### [5.4.2 Mercury position control](#)

##### [5.4.2.1 Position controller architecture](#)

[Proportional \(position\) gain](#)

[Proportional \(velocity\) gain](#)

[Integral \(velocity\) gain](#)

[Feedforward \(velocity & current\) gains](#)

[Angular velocity profile](#)

[Acceleration profile](#)

[Profile modes](#)

##### [5.4.2.2 Joint \(single-turn\) and multi-turn position modes](#)

##### [5.4.2.3 Clockwise \(CW\) and counterclockwise \(CCW\) angle limits](#)

##### [5.4.2.4 Horn position offset](#)

##### [5.4.2.5 Dead zone](#)

##### [5.4.2.6 Further reading](#)

#### [5.4.3 Mercury velocity control](#)

##### [5.4.3.1 Wheel controller architecture](#)

[Target angular velocity](#)

[Proportional \(velocity\) gain](#)

[Integral \(velocity\) gain](#)

[Feedforward \(current\) gain](#)

[Acceleration profile](#)

#### [5.4.4 Torque mode](#)

#### [5.4.5 Stepper mode](#)

#### [5.3.10 Is moving](#)

#### [5.3.11 Hardware error status](#)

## 1. Introduction

This document provides detailed technical information on the Mercury range of digital servos from Robot Articulation.

## 2. Connection details

### 2.1 Pin assignments

Each Mercury servo has 6 input pins.

The pinouts are as follows:

PIN1: GND

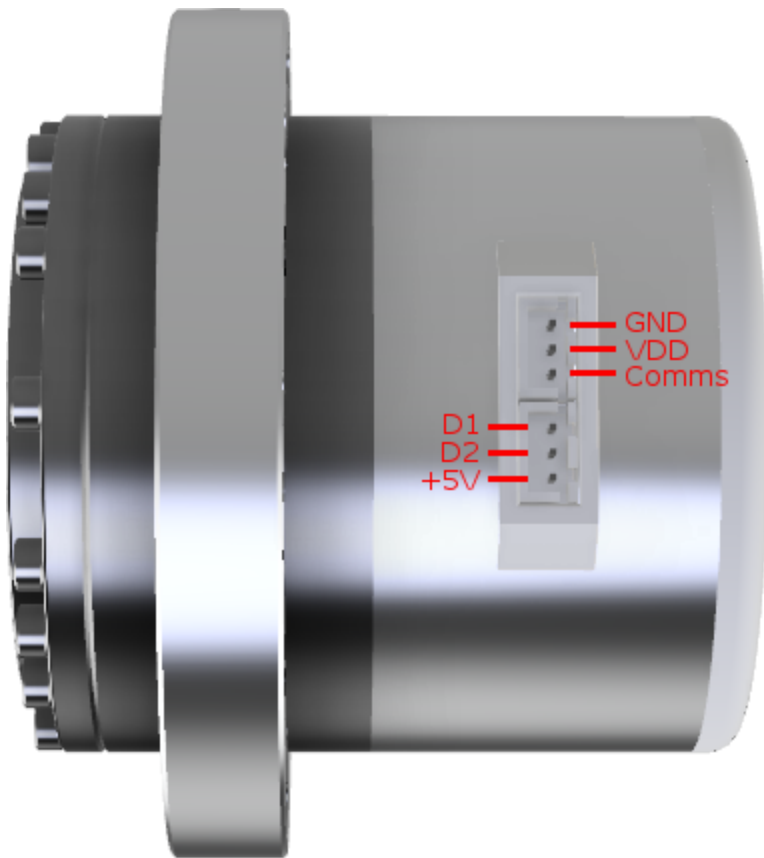
PIN2: VDD (15-24V DC)

PIN3: Comms (half-duplex, TTL levels)

PIN4: Digital input 1 (opto-isolated)

PIN5: Digital input 2 (opto-isolated)

PIN6: 5V common supply (to optocouplers)



## 2.2 Connecting multiple units

Multiple Mercury digital servos may be connected in parallel. Each servo must be configured to have its own unique address.

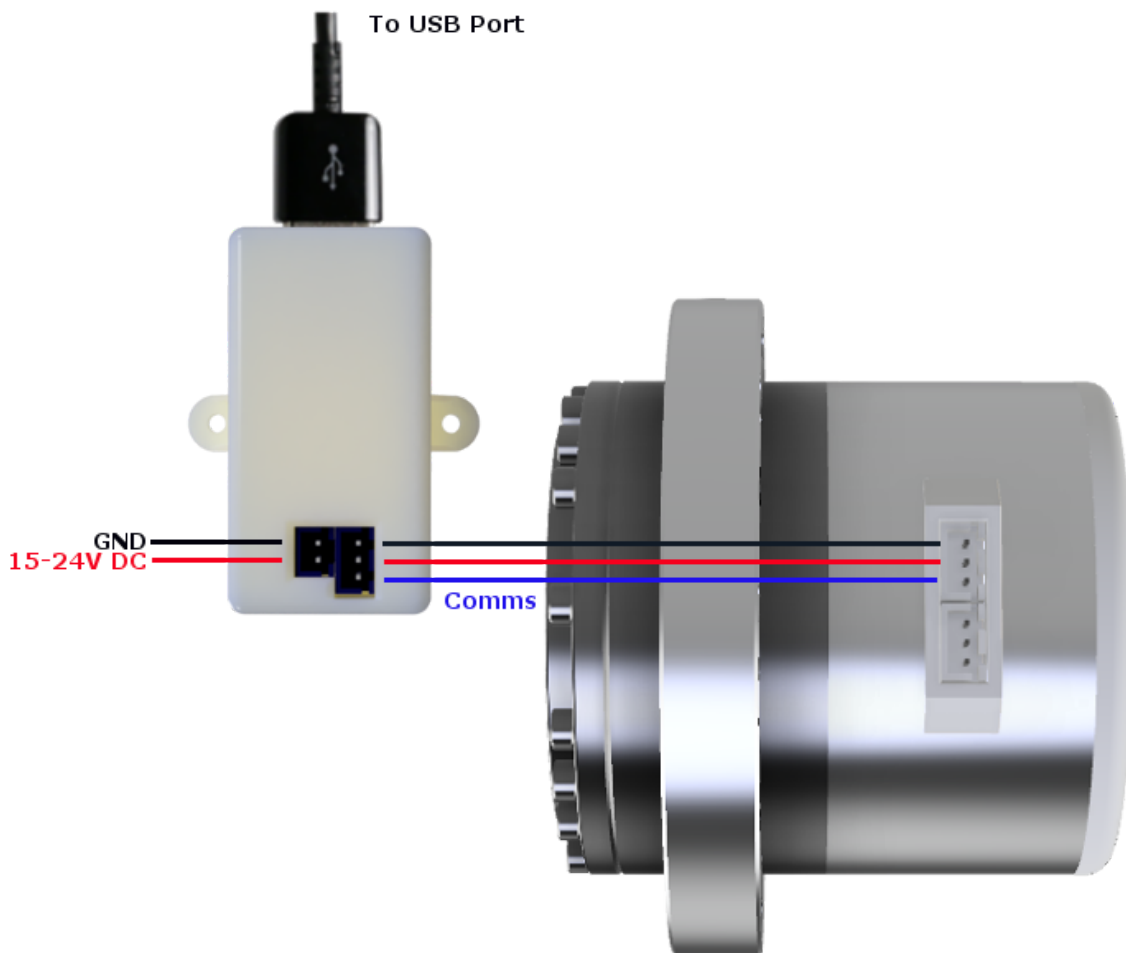
## 2.3 USB2Mercury

In order to control (or configure) Mercury digital servos from a PC, a USB2Mercury unit should be used.

The USB2Mercury unit contains:

- a USB 2.0 type A connector which connects to a USB port on a PC
- a single 2-way XH-series header for connection to an external 15-24V DC power source
- a single 3-way XH-series header to connect to a Mercury servo
- reverse voltage and surge protection

The USB2Mercury unit is connected in the following way:



Please see the USB2Mercury document for more details.

## 2.4 Mercury power hub

Mercury digital servos may be connected in parallel to form a network of Mercury servos. To facilitate such parallel operations, a Mercury Power Hub should be used.

The Mercury Power Hub has the following capabilities:

- contains an in-build USB2Mercury to interface between a PC's USB port and a network of Mercury servos
- accepts a single USB 2.0 type A connection from a PC
- accepts an external DC power source of between 15 and 24V
- provides 5 (in-parallel) XH-series headers to connect directly to Mercury servos.
- can supply a maximum of 15A to the connected Mercury servos - assuming that the external DC power source can supply this amount of current.
- reverse voltage and surge protection

Please see the Mercury Power Hub document for more details.

## 2.5 Power-up protocol

When a Mercury digital servo is powered-up, the red LED will be illuminated for 1 second. The red LED will then be switched off and the Green LED illuminated. The green LED will remain illuminated.

If the red LED is illuminated after the power-up protocol, this will indicate a fault condition.

# 3. Communication protocol

## 3.1 Overview

The USB2Mercury communicates with the connected Mercury digital servos, by sending and receiving a series of data packets. There are two types of packets - Request and Status packets. Request packets are sent from the USB2Mercury unit to the Mercury servos. Acknowledgement packets are sent from Mercury servos to the USB2Mercury unit.

## 3.2 Protocol

Communication between Mercury servos and the USB2Mercury unit is performed using an asynchronous serial protocol of 8 bits, with 1 stop bit and no parity.

See the [table](#) (below) for details on supported baud rates.

## 3.3 Request packet

### 3.3.1 Overview

The structure of the Request Packet is as follows:

Start bytes	Reserved	Unique servo ID / Broadcast ID (0xFE)	Length (2 bytes) (ParamN + 3)	Instruction	Param1	...	Param N	Checksum (2 bytes) (see <a href="#">description</a> )
-------------	----------	---------------------------------------	-------------------------------	-------------	--------	-----	---------	---

										below)
0xFF	0xFF	0xFD	0x00	ID	Length	Instruction	Param1	...	Param N	Checksum

### 3.3.2 Checksum description

The 16-bit checksum is calculated using the CRC-16 IBM/ANSI scheme with a  $x^{16} + x^{15} + x^2 + 1$  polynomial with a 0x8005 polynomial representation.

### 3.3.3 Unique servo ID

A servo ID in the range of 0→252 may be specified in the request packet.

Alternatively, a broadcast ID (0xFE) may be specified where applicable. This has the effect of sending the instruction packet to all Mercury servos on the network. Please note that no acknowledgement packet is returned when the broadcast ID is specified.

## 3.4 Status packet

### 3.4.1 Overview

The Status Packet is sent by a Mercury servo in response to a Request Packet. The structure of the Status packet is as follows:

Start bytes			Reserved	Unique servo ID / Broadcast ID (0xFE)	Length (2 bytes) See below	Instruction	Error See below	Param1	...	ParamN	Checksum (2 bytes) (see <a href="#">description</a> )
0xFF	0xFF	0xFD	0x00	ID	Length	Instruction	Error	Param1	...	ParamN	Checksum

The 16-bit length field is the field count of:

Instruction + error + CRC\_Low + CRC\_High + number of parameters.

Please note that for RESET and REBOOT instructions, a zero is always returned in the Error byte.

### 3.4.2 Error description

Bit 7	Bit 6 ~ bit 0
Alert	Error number

The Alert bit is set when an error condition has been encountered. The [Hardware error status](#) registered should be read in order to determine the nature of the error.

The error number indicates the problem encountered with the processing of the instruction packet:

Value	Error number	Description
0x01	Process failure	Failed to process the sent instruction packet
0x02	Instruction error	Undefined instruction specified, or a ACTION instruction is issued with no pending reg_write instruction
0x03	CRC error	The CRC of the instruction packet does not match the calculated CRC.
0x04	Data range error	Set if an attempt has been made to write data that is outside the min/max range of the relevant register
0x05	Data length error	Set when an attempt is made to write data that has a length that is less than the target register's length  Note, this is <u>not</u> checked when performing a bulk write to the register table.
0x06	Data limit error	The data to be written to the target register is outside of the register's limit value.
0x07	Access error	<ol style="list-style-type: none"><li>1. An attempt is made to write to an address that is read-only</li><li>2. An attempt is made to read from a register that is write-only</li><li>3. An attempt is made to write to the non-volatile region of the register table when the control enable register is equal to 1</li></ol> Note, points 1 & 2 are <u>not</u> checked when performing a bulk write to the register table.



## 4. Instructions

### 4.1 Instructions table

The Mercury servo range supports the following instructions:

Instruction	Description	Value	Number of parameters
<a href="#">PING</a>	Returns a status servo from the targeted servo. No servo update is performed.	0x01	0
<a href="#">READ</a>	Direct read of values from the register table.	0x02	4
<a href="#">WRITE</a>	Direct write to the active register table.	0x03	4+
<a href="#">REG_WRITE</a>	Rx packet is stored in a shadow rx packet. This instruction does not affect the current operation of the servo.	0x04	4+
<a href="#">ACTION</a>	Commit the previously-written shadow rx packet to the active register table.	0x05	0
<a href="#">RESET</a>	Reset the servo to the default (factory) settings	0x06	0
<a href="#">REBOOT</a>	Reboots the servo	0x08	0

### 4.2 Instruction details

#### 4.2.1 PING

##### Instruction packet

The PING instruction is used to request a status packet from a particular Mercury servo specified by an id.  
Instruction packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x03	Length (low byte)
7	0x00	Length (high byte)

8	0x01	PING instruction
9	~	Calculated checksum (low byte)
10	~	Calculated checksum (high byte)

#### Status packet

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x07	Length (low byte)
7	0x00	Length (high byte)
8	0x55	Instruction
9	~	Error - 0x00 if no error
10	~	Model number LSB
11	~	Model number MSB
12	~	Firmware version
13	~	Calculated checksum (low byte)
14	~	Calculated checksum (high byte)

#### 4.2.2 READ

##### Instruction packet

The READ instruction is used to read data directly from the register table of a Mercury servo

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2

3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID
6	0x07	Length (low byte)
7	0x00	Length (high byte)
8	0x02	READ instruction
9	0+	Register table start address (low byte)
10	0+	Register table start address (high byte)
11	1+	Number of bytes to read (low byte), starting from the (above) start address
12	0+	Number of bytes to read (high byte)
13	~	Calculated checksum (low byte)
14	~	Calculated checksum (high byte)

#### Status packet

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	4+	Length (low byte)
7	0+	Length (high byte)
8	0x55	Instruction
9	~	Error - 0x00 if no error
10	~	Data byte 1
11	~	Data byte N
N+6	~	Calculated checksum (low byte)
N+7	~	Calculated checksum (high byte)

### 4.2.3 WRITE

#### Instruction packet

The WRITE instruction is used to write data directly to the register table of a Mercury servo.

No status packet is returned if the broadcast ID is used.

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID. The broadcast ID (0xFE) can also be specified.
6	N+5	Length (low byte) - the number of data bytes to be written + 5
7	N+5	Length (high byte)
8	0x03	WRITE_DIRECT instruction
9	0+	Start address (low byte)
10	0+	Start address (high byte)
11	0+	Data byte 1
...	0+	Data bytes 2→ N-1
11 + Number of data bytes	0+	Data byte N
11 + Number of data bytes + 1	~	Calculated checksum (low byte)
11 + Number of data bytes + 2	~	Calculated checksum (high byte)

#### Status packet

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved

5	0→0xFC (252)	ID of the targeted servo
6	0x01	Length (low byte)
7	0	Length (high byte)
8	0x55	Instruction
9	~	Error - 0x00 if no error
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

#### 4.2.4 REG\_WRITE

##### Instruction packet

The **REG\_WRITE** instruction results in the rx packet being stored in a shadow rx packet on the Mercury servo. No update of the register table takes place. A **REG\_WRITE** instruction has therefore no direct effect on the operation of the Mercury servo.

The Pending shadow instruction register is set to 1 following a **REG\_WRITE** instruction.

No status packet is returned if the broadcast ID is used.

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID. The broadcast ID (0xFE) can also be specified.
6	N+5	Length (low byte) - the number of data bytes to be written + 5
7	N+5	Length (high byte)
8	0x03	REG_WRITE instruction
9	0+	Start address (low byte)
10	0+	Start address (high byte)
11	0+	Data byte 1
...	0+	Data bytes 2→ N-1
11 + Number of data bytes	0+	Data byte N

11 + Number of data bytes + 1	~	Calculated checksum (low byte)
11 + Number of data bytes + 2	~	Calculated checksum (high byte)

### Status packet

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x01	Length (low byte)
7	0	Length (high byte)
8	0x55	Instruction
9	~	Error - 0x00 if no error
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

## 4.2.5 ACTION

### *Instruction packet*

The ACTION instruction is used to update the register table with the data held in the shadow rx packet on a Mercury servo. A Mercury servo will only execute an ACTION instruction if its **Pending Shadow Instruction** register has a value of 1.

The Pending shadow instruction register is reset to 0 following an ACTION instruction.

No status packet is returned if the broadcast ID is used.

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3

4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x03	Length (low byte)
7	0x00	Length (high byte)
8	0x05`	ACTION instruction
9	~	Calculated checksum (low byte)
10	~	Calculated checksum (high byte)

### Status packet

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x01	Length (low byte)
7	0	Length (high byte)
8	0x55	Instruction
9	~	Error - 0x00 if no error
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

#### 4.2.7 RESET

##### *Instruction packet*

The RESET instruction is used to reset the register table of a Mercury servo to the original factory defaults.

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x03	Length (low byte)
7	0x00	Length (high byte)
8	0x01	RESET instruction
9		0xFF: Reset all 0x01: Reset all except ID 0x02: Reset all except ID and Baud rate
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

##### **Status packet**

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x04	Length (low byte)
7	0	Length (high byte)



8	0x55	Instruction
9	0	P1
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

#### 4.2.7 REBOOT

##### *Instruction packet*

The REBOOT instruction is used to reboot the Mercury servo. All non-volatile register settings will be set to their default values.

Packet details:

Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved
5	0→0xFC (252)	ID of the targeted servo
6	0x03	Length (low byte)
7	0x00	Length (high byte)
8	0x01	REBOOT instruction
9	~	Calculated checksum (low byte)
10	~	Calculated checksum (high byte)

##### **Status packet**


Byte	Value	Description
1	0xFF	Start byte 1
2	0xFF	Start byte 2
3	0xFD	Start byte 3
4	0x00	Reserved

5	0→0xFC (252)	ID of the targeted servo
6	0x04	Length (low byte)
7	0	Length (high byte)
8	0x55	Instruction
9	0x00	P1
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

## 5. Control registers

### 5.1 Register table

Non-volatile registers						
Address	Size	Description	R/W	Range	Default Value	Notes
0 (0x00)	1	Model number minor	R		E.g. 0x01 for model minor version 1	
1 (0x01)	1	Model number major	R		E.g. 30 (0x1E) for Mercury M30	
2 (0x02)	1	Firmware version	R			
3 (0x03)	1	ID	RW	0→252	1 (0x01)	Note that the USB2Mercury reserves ID 253 (0xFD)
4 (0x04)	1	Baud rate	RW	1→254	1 (0x01) (1,000,000 BPS)	See <a href="#">table</a> below
5 (0x05)	1	Acknowledgement packet response time	RW	0→254	250 (0xFA)	See <a href="#">details</a> below
6 (0x06)	1	Operating mode	RW	0-3	2 (single-turn mode)	See <a href="#">table</a> below
7 (0x07)	2	Clockwise (CW) angle limit	RW	-2047→2047 (-π to π)	-2047 (-π)	See <a href="#">servo output position</a> details below
9 (0x09)	2	Counter-clockwise (CCW) angle limit	RW	-2047→2047 (-π to π)	2047 (π)	
11 (0x0B)	1	Upper temperature limit	RW	0→254	55 (0x37)	Value in °C

12 (0x0C)	1	Lower input voltage limit	RW	0→254	150 (0x96)	<p>Voltage = register value / 10. 200 ⇔ 20V</p> <p>If the supply voltage falls below this figure, the Control enable register will be set to 0;</p>
13 (0x0D)	1	Upper input voltage limit	RW	0→254	250 (0xFa)	<p>As per above. 250 ⇔ 25V</p> <p>If the supply voltage goes above this figure, the Control enable register will be set to 0;</p>
14 (0x0E)	2	Phase current limit	RW	M30: 0→3000 M50: 0→5000 M65: 0→6500	M30: 3000 (0x258) M50: 5000 (0x3E8) M65: 6500 (0x5DC)	<p>Maximum phase current in mA.</p>  <p>To avoid damage to your Mercury servo, do not routinely exceed the <b>rated</b> torque figure.</p>
16 (0x10)	2	Angular velocity limit	RW	0 → 5000	5000 (0x1388)	5000 equates to 5000 milli-rad/s.
18 (0x12)	2	Acceleration limit	RW	0→100	0	0 indicates the maximum possible acceleration.
20 (0x14)	4	Horn position offset	RW	<p>Single-turn: -1023→1023 (-π/2 to π/2)</p> <p>Multi-turn: -1,044,479 → 1,044,479</p>	0	See <a href="#">details</a> below.
24 (0x18)	2	Moving threshold	RW	0 → 2,000	200 (0x00C8)	Moving velocity threshold in

						milli-rad/s. See <a href="#">details</a> below.
26 (0x1a)	1	Reserved				
...	21	Reserved				
47 (0x2F)	1	Reserved				

Volatile registers						
Address	Size	Description	R/W	Range	Default Value	Notes
48 (0x30)	1	Control enable	RW	0→1	0	See <a href="#">details</a> below.
49 (0x31)	1	Pending shadow instruction	RW	0→1	0	1 = pending REG_WRITE instruction.
50 (0x32)	2	Reserved				
52 (0x34)	2	Reserved				
54 (0x36)	2	Position proportional gain (Kp)	RW	0→16,383	1000 (0x3E8)	See position <a href="#">details</a> below.
56 (0x38)	2	Velocity feedforward gain	RW	0→16,383	0	See feedforward <a href="#">details</a> below.
58 (0x3A)	2	Current feedforward gain	RW	0→16,383	0	
60 (0x3C)	2	Velocity integral gain	RW	0→16,383	1500 (0x5DC)	See velocity integral <a href="#">details</a> below.
62 (0x3E)	2	Velocity proportional gain	RW	0→16,383	200 (0xC8)	See velocity proportional <a href="#">details</a> below.
64 (0x40)	2	Reserved				
66 (0x42)	2	Reserved				
68 (0x44)	2	Angular velocity profile	RW	0→Angular velocity limit	0	See the angular velocity profile <a href="#">details</a> below.
70 (0x46)	2	Acceleration profile	RW	0→Acceleration limit	0	See acceleration profile <a href="#">details</a> below.

72 (0x48)	2	Dead zone	RW	0→1024	0	Position control only. See dead zone <a href="#">details</a> below.
74 (0x4A)	2	Reserved				
76 (0x4C)	2	Reserved				
78 (0x4E)	4	Target position	RW	Single-turn: 0→4095 (-π to π)  Multi-turn: -1,048,575 → 1,048,575	Value from actual position register.	See <a href="#">servo output position</a> section below
82 (0x52)	2	Target angular velocity	RW	0→10000	0	Target angular velocity in milli-rad/s. See <a href="#">details</a> below. Only used in continuous rotation mode.
84 (0x54)	2	Target torque	RW	0→value from Torque limit register.	0	Model dependant target torque in units of 10 mNm. See torque mode <a href="#">details</a> below.
86 (0x56)	4	Actual position	R	Single-turn: -2048→2048 (-π to π)  Multi-turn: -1,048,575 → 1,048,575	-	See <a href="#">servo output position</a> section below
90 (0x5A)	2	Actual angular velocity	R			Angular velocity in milli-rad/s. e.g. 4400 milli-rad/s ⇔ 4.4 rad/s ⇔ 42 rpm
92 (0x5C)	2	Actual torque	R			Torque in units of 10 mNm 100 ⇔ 1Nm
94 (0x5E)	1	Actual voltage	R			Voltage = register value /

						10. e.g. 200 ⇔ 20V
95 (0x5F)	2	Actual current	R			1 ⇔ 1mA
97 (0x61)	1	Actual temperature	R			Value in °C
98 (0x62)	1	Reserved				
99 (0x63)	1	Moving	R	0→1		Indicates if the servo is moving. See <a href="#">details</a> below.
100 (0x64)	1	Trajectory status	R			Indicates the current trajectory status. See <a href="#">details</a> below.
101 (0x65)	2	Calculated angular velocity profile trajectory	R			
103 (0x67)	4	Calculated position profile trajectory	R			
107 (0x6b)	1	Hardware error status	R			See <a href="#">details</a> below

Please note that **two's complement** is used to represent all negative negative numbers.

## 5.2 Register limits

Each writable register has an associated minimum and maximum value. Write instructions made outside of valid ranges will return an out-of-range status error, and no update will take place.

The following table details the data range for each register. 16 bit registers must be written atomically within the same instruction packet.

Write address	Description	Min value	Max value
3 (0x03)	ID	0	252 (0xFC)
4 (0x04)	Baud rate	1	254 (0xFE)
5 (0x05)	Acknowledgement packet response time	0	254 (0xFE)
6 (0x06)	Operating mode	0	4
7 (0x07)	Clockwise (CW) angle limit	0	4095 (0xFFFF)

9 (0x09)	Counter-clockwise (CCW) angle limit	0	4095 (0xFFFF)
11 (0x0B)	Upper temperature limit	0	55 (0x37)
12 (0x0C)	Lower input voltage limit	150 (0x64)	254 (0xFE)
13 (0x0D)	Upper input voltage limit	150 (0x64)	254 (0xFE)
14 (0x0E)	Torque limit	0	M30: 1,800 (0x708) M50: 3,000 (0xBB8) M65: 4,000 (0xFA0)
16 (0x10)	Angular velocity limit	0	5000 (0x1388)
18 (0x12)	Acceleration limit	0	100 (0x64)
20 (0x14)	Home position offset	Single-turn: -1535 Multi-turn: -32,767	Single-turn: 1535 Multi-turn: 32,767
23 (0x18)	Moving threshold	0	2,000 (0x7D0)
48 (0x30)	Control enable	0	1
50 (0x32)	Position derivative gain	0	16,383 (0x3FFF)
52 (0x34)	Position integral gain	0	16,383 (0x3FFF)
54 (0x36)	Position proportional gain	0	16,383 (0x3FFF)
56 (0x38)	Position feedforward gain 1	0	16,383 (0x3FFF)
58 (0x3A)	Position feedforward gain 2	0	16,383 (0x3FFF)
60 (0x3C)	Velocity integral gain	0	16,383 (0x3FFF)
62 (0x3E)	Velocity proportional gain	0	16,383 (0x3FFF)
68 (0x44)	Angular velocity profile	0	Angular velocity limit
70 (0x46)	Acceleration profile	0	Acceleration limit
72 (0x48)	Dead zone	0	1024
78 (0x4E)	Target position	Single-turn: CW angle limit Multi-turn: -32,767	Single-turn: CCW angle limit Multi-turn: 32,767 (0x7FFF)
80 (0x50)	Target angular velocity	0	Angular velocity limit

82 (0x52)	Target torque	0	Torque limit
96 (0x60)	Pending shadow instruction	0	1

### .3 Register details

#### 5.3.1 Control enable

This register controls the following:

Value	Description
0 (default)	<ul style="list-style-type: none"> <li>• unlocks all non-volatile registers</li> <li>• cuts all power to the motor</li> </ul>
1 (0x01)	<ul style="list-style-type: none"> <li>• locks all non-volatile registers</li> <li>• enables the motor</li> </ul>

#### 5.3.2 Baud rates

The baud rate of the Mercury servo is calculated as follows:

Baud rate (BPS) =  $2000000 / (\text{value} + 1)$

Standard baud rates:

Value	Baud Rate
1 (0x01)	1000000
3 (0x03)	500000
4 (0x04)	400000
7 (0x05)	250000
9 (0x09)	200000
16 (0x10)	115200
34 (0x22)	57600
103 (0x67)	19200
207 (0xcf)	9600

The baud rate formula is: Speed (BPS) =  $2000000 / (\text{code} + 1)$

The baud rate margin of error is set at < 3%.



### 5.3.3 Acknowledgement packet response time

This register controls the (approximate) elapsed time between the reception of the request packet and the transmission of the acknowledgement packet. The elapsed time is given by  $2\mu\text{seconds} \times \text{the register value}$ .

## 5.4 Operating modes

### 5.4.1 Operating modes table

Mode	Value	Description
Torque mode	0	Controls the output torque of the Mercury servo.  Makes use of: <ul style="list-style-type: none"><li>• Acceleration limit</li><li>• Acceleration profile</li><li>• Torque limit</li><li>• Target torque</li><li>• Angular velocity limit</li></ul>
Continuous rotation (Wheel) mode	1	Controls the angular velocity of the Mercury servo. <ul style="list-style-type: none"><li>• Rotates the servo at the target angular velocity.</li><li>• The direction bit (15) controls the direction of rotation.</li></ul> Makes use of: <ul style="list-style-type: none"><li>• Acceleration limit</li><li>• Acceleration profile</li><li>• Angular velocity limit</li><li>• Target angular velocity</li><li>• Velocity proportional gain (Kp)</li><li>• Velocity integral gain (Ki)</li></ul>
Single-turn (Joint) position mode	2	Moves to the target position based on the specified velocity and acceleration profiles.  Makes use of: <ul style="list-style-type: none"><li>• Home position offset</li><li>• CW &amp; CCW angle limits</li><li>• Acceleration limit</li><li>• Acceleration profile</li><li>• Angular velocity limit</li><li>• Angular velocity profile</li><li>• Position proportional gain (pKp)</li><li>• Velocity integral gain (vKi)</li><li>• Velocity derivative gain (vKd)</li><li>• Velocity feedforward gain vKff</li><li>• Current feedforward gain cKff</li></ul>
Multi-turn position mode	3	Moves to the target position based on the specified velocity and acceleration profiles. Allows a (real) target angle to be specified that is greater than $360^\circ$ . Maximum turns are $-16 \rightarrow 16$ .  Makes use of: <ul style="list-style-type: none"><li>• Home position offset</li><li>• Acceleration limit</li></ul>

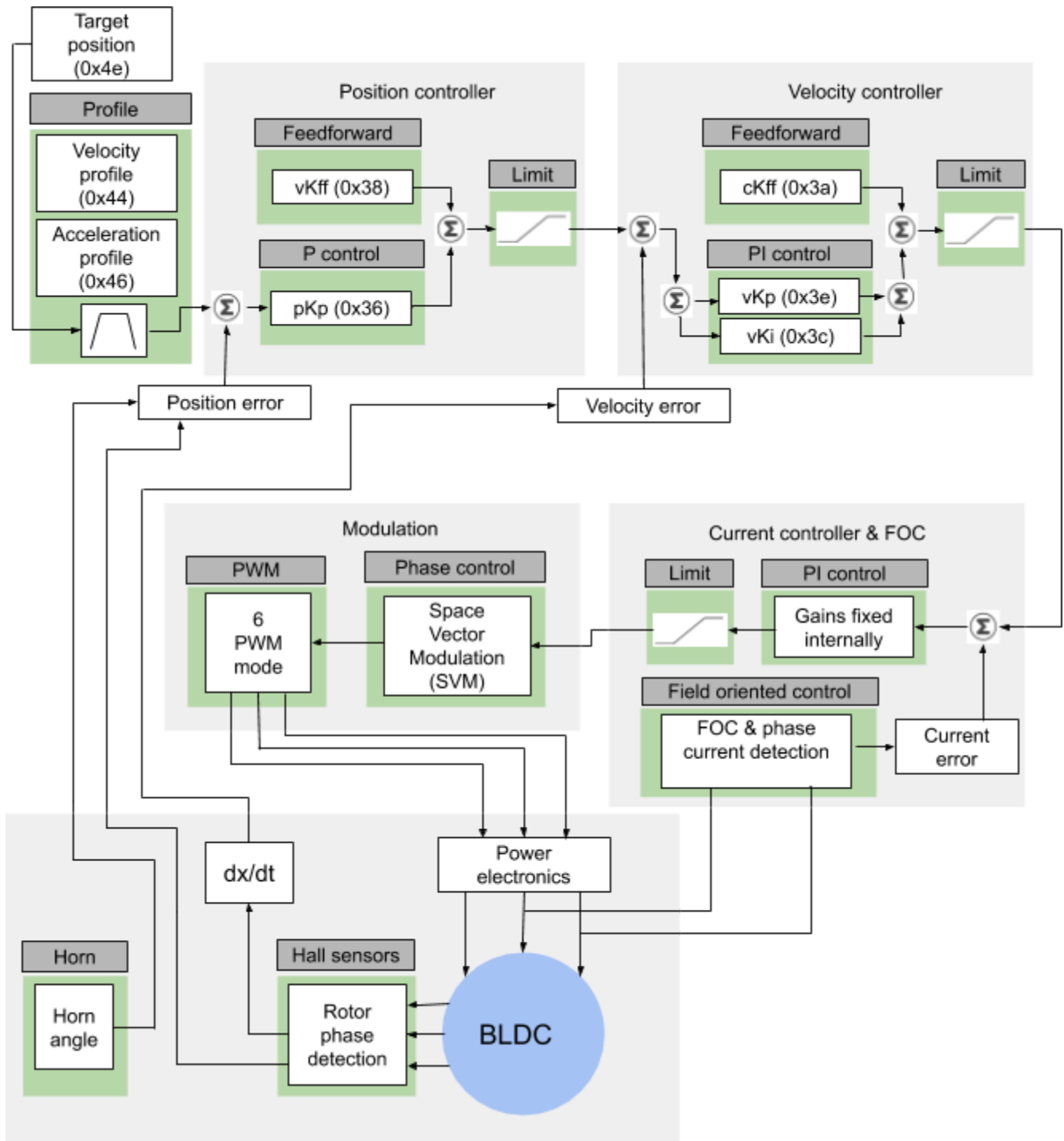
		<ul style="list-style-type: none"> <li>• Acceleration profile</li> <li>• Angular velocity limit</li> <li>• Angular velocity profile</li> <li>• Position proportional gain (pKp)</li> <li>• Velocity integral gain (vKi)</li> <li>• Velocity derivative gain (vKd)</li> <li>• Velocity feedforward gain vKff</li> <li>• Current feedforward gain cKff</li> </ul>
Stepper mode	4	<p>In this mode, digital pins D1 and D2 are used as a STEP and DIRECTION inputs respectively. Each step will advance the output horn (CW or CCW depending on the polarity of the DIRECTION input pin D2) by 1/4096 revolutions.</p> <p>Makes use of:</p> <ul style="list-style-type: none"> <li>• Home position offset</li> <li>• Acceleration limit</li> <li>• Angular velocity limit</li> </ul>

## 5.4.2 Mercury position control

### 5.4.2.1 Position controller architecture

For position control, Mercury servos use a 3-stage cascaded controller consisting of a:

- position controller
- velocity controller
- current controller.



### **Proportional (position) gain**

The position proportional component depends only on the difference between the goal position and the actual position. In general, increasing  $pKp$  will increase the speed of the servo's response. However, if  $pKp$  is too large, oscillations could occur.

### **Proportional (velocity) gain**

The velocity component depends only on the difference between the desired velocity and the actual velocity. As with position proportional control, increasing  $vKp$  will increase the speed of the servo's response. If  $vKp$  is too large, oscillations could occur.

### **Integral (velocity) gain**

The integral component sums the velocity error over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the steady-state error to zero. Steady-state error is the final difference between the servo's actual velocity and the desired velocity. If the integral gain ( $vKi$ ) is too small, the servo response will be sluggish. If  $vKi$  is set too high, oscillation may occur.

### **Feedforward (velocity & current) gains**

In a feedforward system, knowledge about the system is used to calculate an output component that is added to the output of the relevant controller stage.

With Mercury digital servos, velocity and current feedforward gains represent some prior knowledge of the velocity and current gains respectively. The PWM signal that is (ultimately) sent to the motor is therefore influenced by these feedforward gains.

### **Angular velocity profile**

The angular velocity profile register maintains the velocity profile in Joint and Multi-turn operating mode. Profile angular velocity is represented in milli-rad/s. The maximum angular velocity is approximately 23 rpm. That is, a register value of 1000 equates to  $1000 \text{ milli-rad/s} \Leftrightarrow 1 \text{ rad/s} \Leftrightarrow 9.549 \text{ rpm}$ .

Angular velocity profile is used only in **Position control** mode.

### **Acceleration profile**

The acceleration profile register maintains the acceleration profile for the relevant profile type. Profile acceleration is represented in units of  $\text{milli-rad/s}^2$ . Valid values are  $0 \rightarrow 48000$ , representing a maximum profile acceleration of  $48 \text{ rad/s}^2$ .

When set to 0, the applied acceleration corresponds to the maximum acceleration of the motor.

Acceleration profile is used in both **Angular velocity control** mode and **Position control** mode.

## Profile modes

In Joint and Multi-turn control modes, 4 different angular velocity profile control schemes are available:

Mode	Angular velocity profile value	Acceleration profile value	Notes
Step	0	X	Angular velocity = Angular velocity limit (0x10). If angular velocity limit = 0, then maximum achievable angular velocity.  Acceleration = maximum achievable
Rectangle	> 0	0	Angular velocity = Angular velocity profile value  Acceleration = Acceleration limit (0x12). If acceleration limit = 0, then maximum achievable acceleration.
Triangle	> 0	> 0	Angular velocity = constantly changing  Acceleration = As specified
Trapezoidal	> 0	> 0	Angular velocity = changing until angular velocity profile value is reached. This angular velocity is maintained until the deceleration point is reached.  Acceleration = As specified

### 1. Step

In step mode, the acceleration and angular velocity profiles are not used. Instead, the horn accelerates at a maximum rate to the maximum angular velocity before decelerating at a maximum rate to the goal position.

### 2. Rectangle

In Rectangular mode, the horn accelerates at a maximum rate to the specified angular velocity profile value. The horn then maintains the calculated angular velocity profile value until reaching the goal position minus the deceleration time. The horn decelerates at the maximum rate.

### 3. Triangle

In Triangular mode, the horn accelerates at the calculated acceleration profile trajectory value.

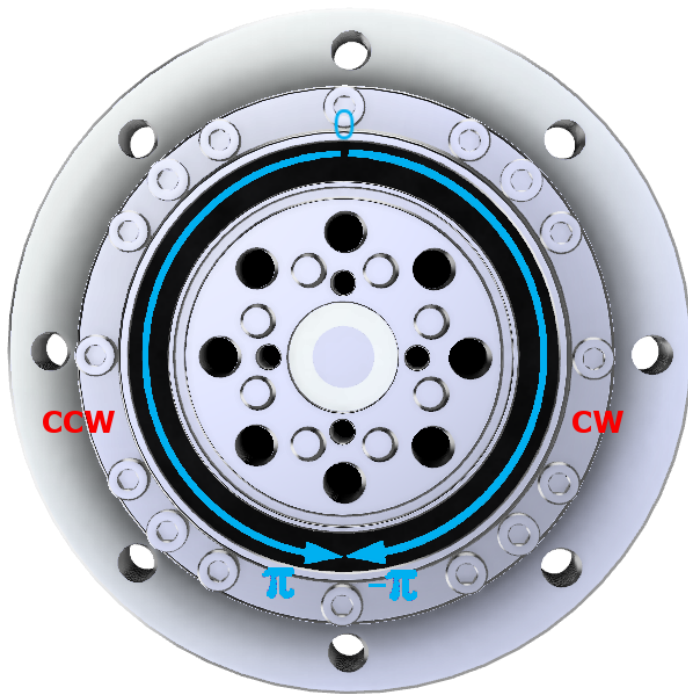
The angular velocity of the horn at the point when the horn starts to decelerate towards the position setpoint, is less (or equal) to the specified angular velocity profile value. The resulting profile is thus triangular.

### 4. Trapezoidal

In Trapezoidal mode, the horn accelerates at the calculated acceleration profile value until the calculated angular velocity profile value is reached. This velocity is then maintained before decelerating to the position setpoint.

#### 5.4.2.2 Joint (single-turn) and multi-turn position modes

A front-facing view of a Mercury servo is shown below.



In the (default) joint mode, the values in brackets represent the **actual position** register values at  $\pi$  radians ( $180^\circ$ ) and with a **Home position offset** register value of 0.

#### 5.4.2.3 Clockwise (CW) and counterclockwise (CCW) angle limits

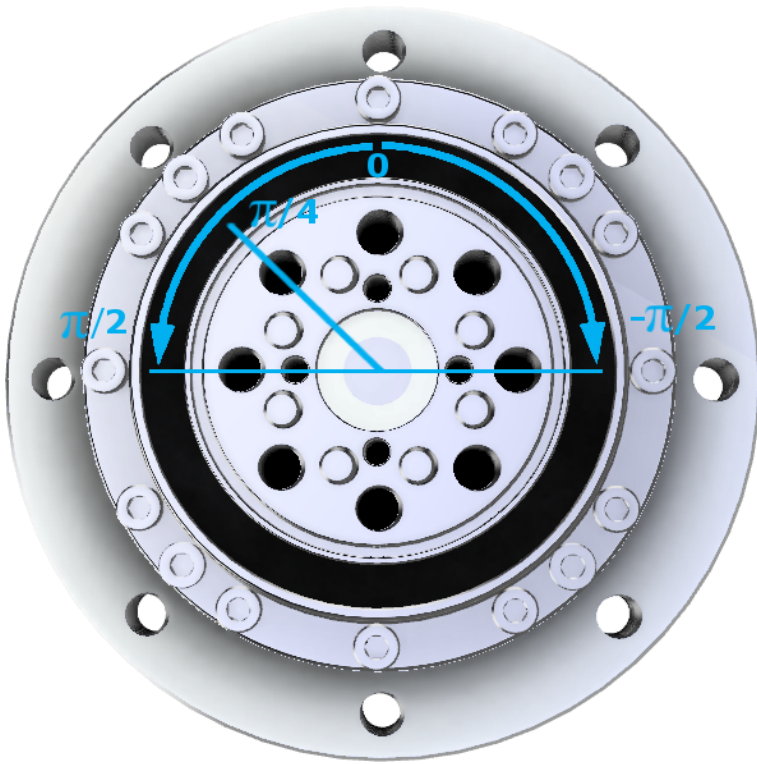
CW and CCW angle limits register are only relevant in joint mode. In multi-turn mode, the CW and CCW angle limits are ignored..

#### 5.4.2.4 Horn position offset

This 2's complement figure represents the horn position offset.

In joint (single-turn) mode (where the maximum rotation is  $2\pi$  radians [ $360^\circ$ ]), the valid range is  $-1024 \rightarrow 1024$ . A value outside of this range will be considered an error condition, and a value of zero will be assumed.

In multi-turn position mode, the range is  $-1,044,479 \rightarrow 1,044,479$ . This is the equivalent of  $\pm 255$  turns.



In the above diagram:

- The real position is  $\pi/4$  radians [ $45^\circ$ ] (512)
- The home position offset is -512
- The actual position register value is 0.

That is, the actual position register value = real position + home position offset.

#### 5.4.2.5 Dead zone

The dead zone register allows a setpoint error tolerance to be defined. With a (default) value of 0, power to the Mercury servo will only be cut when the exact target position is reached. With a non-zero dead zone value, the dead zone value is applied symmetrically to the target position.

For example:

- Dead zone tolerance: 5
- Target position: 2048

Motor power will be cut between actual positions: 2043→2053

#### 5.4.2.6 Further reading

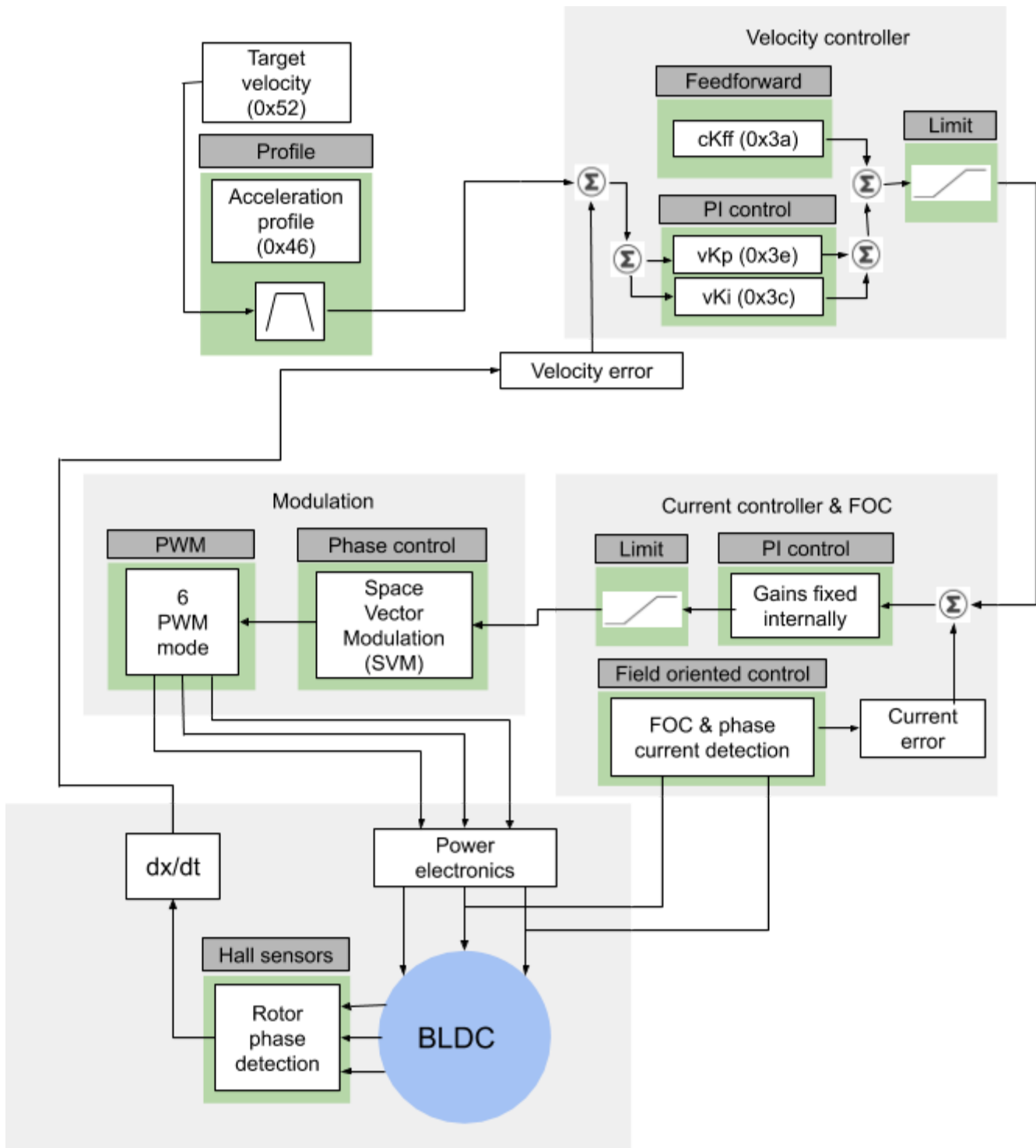
A detailed description of the Mercury control algorithms can be found in the accompanying document **Mercury control algorithms explained**.

### 5.4.3 Mercury velocity control

#### 5.4.3.1 Wheel controller architecture

For velocity control, Mercury servos use a 2-stage cascaded controller consisting of a:

- velocity controller
- current controller.





In *Wheel operating* mode, the servo will always attempt to maintain the target angular velocity. The velocity PI parameters are used to maintain the actual angular velocity at the target angular velocity. The profile acceleration register value is used to control the acceleration to the target velocity.

### Target angular velocity

The target angular velocity register maintains the target velocity in milli-rad/s. That is, a register value of 1000 equates to 1000 milli-rad/s  $\Leftrightarrow$  1 rad/s  $\Leftrightarrow$  9.549 rpm.

The maximum angular velocity is approximately 23 rpm.

Bit 15 of the target angular velocity register maintains the direction of rotation:

- 0  $\Leftrightarrow$  counterclockwise (CCW)
- 1  $\Leftrightarrow$  clockwise (CW)

**Note** that the target angular velocity register is distinct from the profile angular velocity profile register.

### Proportional (velocity) gain

See [details](#) above.

### Integral (velocity) gain

See [details](#) above.

### Feedforward (current) gain

See [details](#) above.

### Acceleration profile

See [details](#) above

#### 5.4.4 Torque mode

In Torque mode, the target angular velocity is ignored. Instead, the servo will attempt to maintain a target torque. The result of this is that the angular velocity will depend only on the target torque setting and the load on the servo.

In Torque mode, the following parameters are ignored:

- Target angular velocity
- CW and CCW limits
- Goal position
- PID (position and velocity) parameters

Bit 15 of the target torque register maintains the direction of rotation:

- 0 ⇔ counterclockwise (CCW)
- 1 ⇔ clockwise (CW)

#### 5.4.5 Stepper mode

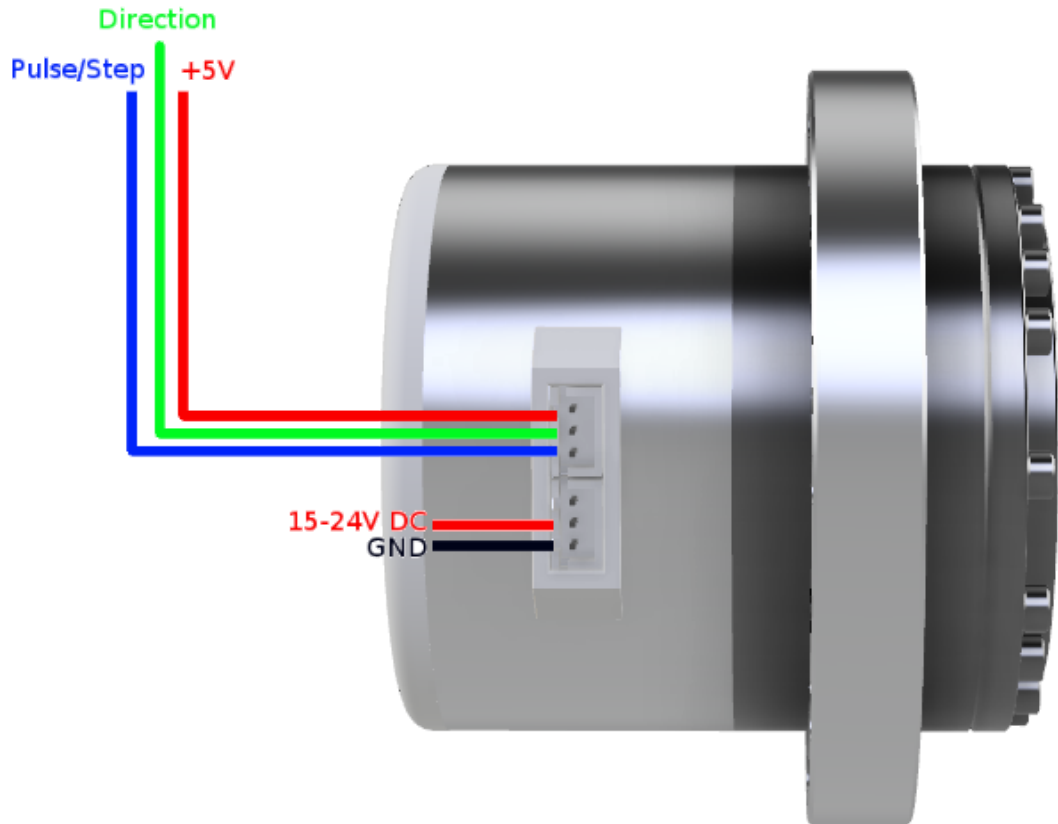
This control mode allows for the straightforward connection of Mercury digital servos to CNC-type controllers with STEP and DIRECTION outputs. Unlike conventional open-loop stepper motors, the Mercury servo's control electronics will ensure the correct output response based on input steps and chosen control parameters.

The two opto-isolated digital inputs (D1 & D2) are **active low**. See the [Connection details diagram](#) for more information. Please note that most CNC controllers allow the output polarity of their STEP and DIRECTION pins to be specified.

Input D1 (pin4) must be connected to the STEP output of the CNC controller.

Input D2 (pin 5) must be connected to the DIRECTION output of the CNC controller.

The 5v common (pin 6) must be connected to the 5 volt output of the CNC controller.



The CNC controller motor limits should ideally be set to the following settings:

- 3600 steps per 360° (0.1° degrees per step)
- 1000 steps per second
- Max acceleration/deceleration pulse period  $\leq 50\text{ms}$

For optimal tracking of the incoming step pulses, ensure that acceleration & deceleration are not too high.

### 5.3.10 Is moving

The Moving register is set to 1 if the Mercury digital servo is deemed to be moving. Otherwise the Moving register is set to zero indicating that the servo is stationary.

Determining whether the servo is moving or is stationary is done by comparing the actual velocity with the moving threshold value. If the (absolute) actual velocity exceeds the threshold value, the servo is deemed to be moving.

### 5.3.11 Hardware error status

This register maintains details of the current hardware error status. It's primary function is to protect the Mercury servo from out-of-range conditions:

Bit	Error condition	Description
7	n/a	
6	n/a	
5	Overload error	<p>This error can be the result of one of the following conditions:</p> <ul style="list-style-type: none"><li>• The output torque has exceeded the maximum specified torque</li><li>• An overcurrent alarm has been generated by the BLDC driver.</li></ul> <p>The control_enable register is set to zero when this error occurs.</p>
4	Angle limit error	<p>The horn angle is either:</p> <ul style="list-style-type: none"><li>• Outside of the specified CW or CCW limits in Joint mode</li><li>• &gt; +/- 255 revolutions from the zero point in Multi-turn mode</li></ul>
3	Encoder error	<p>An encoder error has been detected.</p> <p>The control_enable register is set to zero when this error occurs.</p>
2	Overheating error	<p>The internal temperature of the Mercury servo has exceeded the specified temperature limit. This can be the result of one of the following conditions:</p> <ul style="list-style-type: none"><li>• Measured temperature (97) exceed the temperature limit (11)</li><li>• An over-temperature alarm generated from the BLDC driver.</li></ul> <p>This error results in a device shutdown. To recover from a shutdown, the device must first be powered off.</p>
1	Hall effect sensor error	<p>An error has been detected with the motor's hall effect sensors.</p> <p>The control_enable register is set to zero when this error occurs.</p>
0	Input voltage error	<p>The input (supply) voltage is either:</p> <ul style="list-style-type: none"><li>• less than the specified minimum voltage limit</li><li>• greater than the maximum specified voltage limit</li></ul> <p>The control_enable register is set to zero when this error occurs.</p>