

Mercury technical manual

October
2023

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](#)

[4.2.8 CLEAR](#)

[Status packet](#)

[4.3 Using the broadcast ID](#)

[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 Input modes table](#)

[5.4.3 Mercury position control](#)

[5.4.3.1 Position controller architecture](#)

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

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

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

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

[Angular velocity profile](#)

[Acceleration profile](#)

[Profile modes](#)

[5.4.3.2 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](#)

[5.4.4 Mercury velocity control](#)

[5.4.4.1 Wheel controller architecture](#)

[Target angular velocity](#)

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

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

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

[Acceleration profile](#)

[5.4.5 Torque mode](#)

[5.4.6 Stepper mode](#)

[5.5 Is moving](#)

[5.6 Hardware status](#)

[6. Diagnostics](#)

[6.1 LED states](#)

[7. ROS2 integration](#)

[7.1 Github repositories](#)

[7.1 Building a ROS2 Mercury example](#)

1. Introduction

This document provides detailed technical information on the Mercury M1 digital servo from Robot Articulation.

2. Connection details

2.1 Pin assignments

Mercury servos have a number of input and output connectors:

Opto-isolated input / output connector:

PIN1: collector output

PIN2: emitter output

PIN3: +5v external supply for inputs

PIN4: step pulse input

PIN5: step direction input

CAN bus input/output:

PIN1: CAN HI

PIN2: CAN LOW

USART input / output (half duplex):

PIN1: usart IO

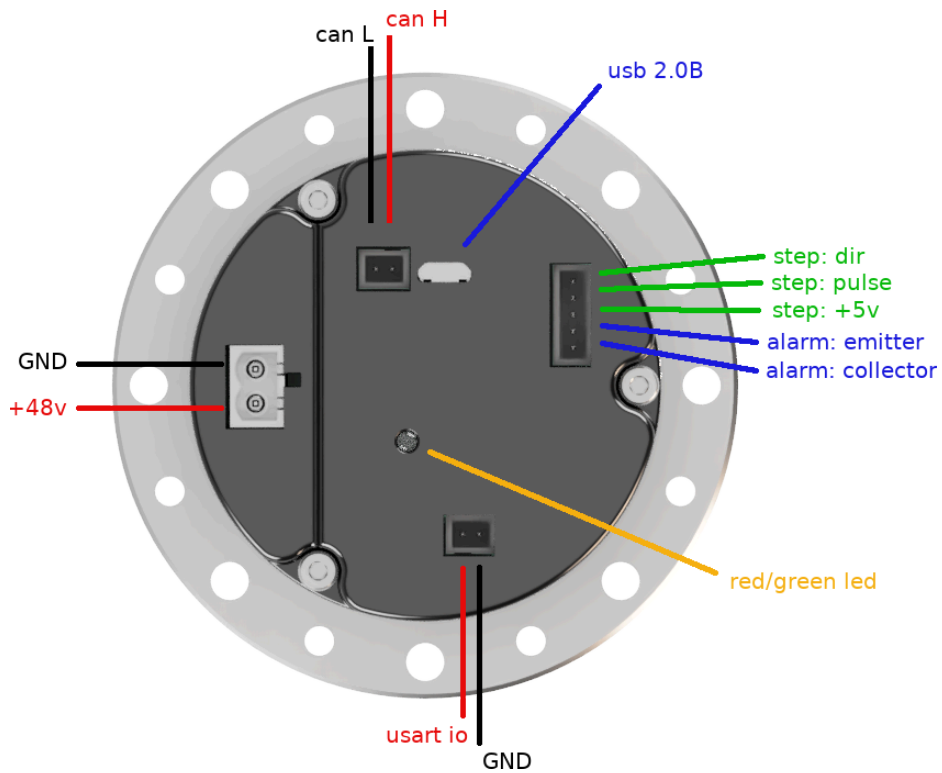
PIN2: GND

Servo power supply (DC):

PIN1: +48v (20A)

PIN2: GND

USB 2.0B



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.

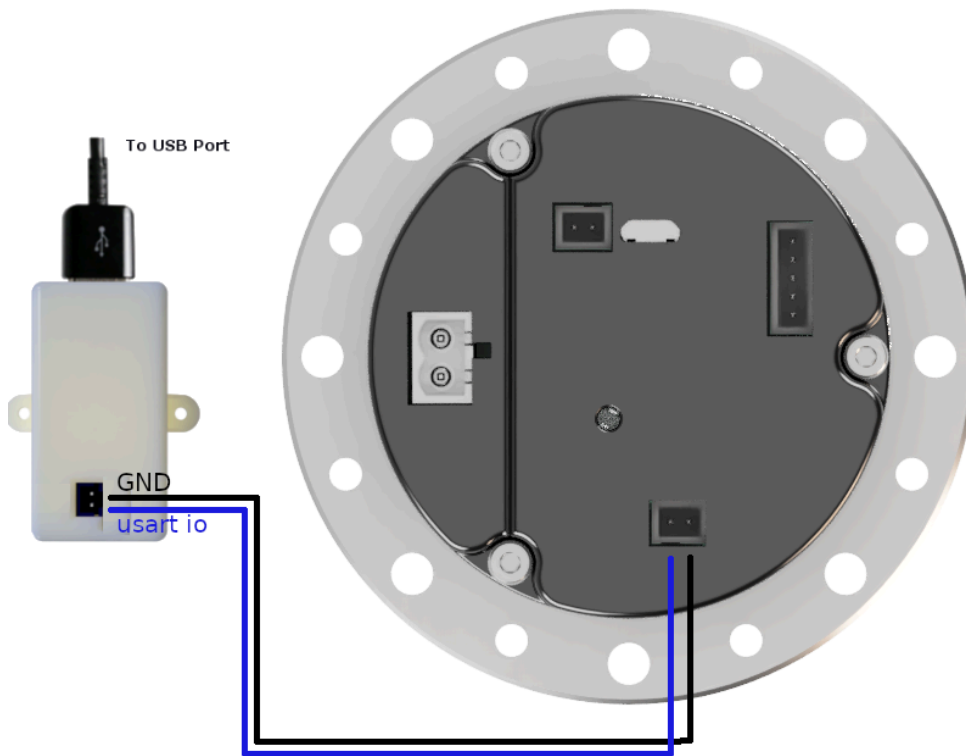
2.3 USB2Mercury

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 to connect to a Mercury servo
- 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-built 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 48V DC power source via a single 2-way PCB-mounted barrier terminal to supply a maximum of 30A. Fused and surge-protected
- provides 5 (in-parallel) 2-way WR-TBL 311 series power output connectors to power Mercury servos
- provides 5 (in-parallel) 2-way XH headers to connect directly to Mercury servo comms io connectors

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 exact state of the LED will depend on the state of the servo. See the [LED states](#) for detailed information.

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 description 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.

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 description)
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 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
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"> 1. An attempt is made to write to an address that is read-only 2. An attempt is made to read from a register that is write-only 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

4. Instructions

4.1 Instructions table

The Mercury servo range supports the following instructions:

Instruction	Description	Value	Number of parameters
PING	Returns a status servo from the targeted servo. No servo update is performed.	0x01	0
READ	Direct read of values from the register table.	0x02	4
WRITE	Direct write to the active register table.	0x03	4+
REG_WRITE	Rx packet is stored in a shadow rx packet. This instruction does not affect the current operation of the servo.	0x04	4+
ACTION	Commit the previously-written shadow rx packet to the active register table.	0x05	0
RESET	Reset the servo to the default (factory) settings	0x06	0
REBOOT	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. The broadcast ID (0xFE) can also be specified. For further details on the broadcast ping, see here .
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)

4.2.8 CLEAR

Instruction packet

The CLEAR instruction is used to reset the multiturn revolutions count to zero.

The Clear instruction can only be applied when **control_enable** = 0x0. If **control_enable** = 0x1 and a CLEAR instruction is sent, a Result Fail (0x01) will be set in the ERROR field of the Status Packet.

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	0x08	Length (low byte)
7	0x00	Length (high byte)
8	0x10	CLEAR instruction
9	0x01	P1
10	0x44	P2
11	0x58	P3

12	0x4C	P4
13	0x22	P5
14	~	Calculated checksum (low byte)
15	~	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	ERROR: 0 Result Fail (0x01) hardware error
10	~	Calculated checksum (low byte)
11	~	Calculated checksum (high byte)

4.3 Using the broadcast ID

The broadcast ID (0xFE) can be specified for a number of instructions. For write operations, this is straightforward as no status packet is generated for write operations.

For read operations, the broadcast ID is only supported for **PING** and **READ_SYNC** operations.

In order to avoid bus contention for broadcast read operations, each servo responds after a calculated delay.


- Calculate **time_per_byte**. e.g. 0.01s @ 1Mbps
- Calculate **status_packet_length** e.g. length == 14 for ping response
- Calculate **wait_length = servo_id * status_packet_length**
- Calculate **delay = (wait_length * time_per_byte) + (3.0 * servo_id)**

Using the calculated delay,, we avoid bus contention.

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 M1	
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 (0x05) (1,000,000 BPS)	See table below
5 (0x05)	1	Acknowledgement packet response time	RW	0→254	250 (0xFA)	See details below
6 (0x06)	1	Operating mode	RW	0-3	2 (single-turn mode)	See table below
7 (0x07)	2	Clockwise (CW) angle limit	RW	-8192→8191 (- π to π)	-8192 (- π)	See servo output position details below
9 (0x09)	2	Counter-clockwise (CCW) angle limit	RW	-8192→8191 (- π to π)	8191 (π)	
11 (0x0B)	1	Upper temperature limit	RW	40→75	70 (0x46)	Value in °C
12 (0x0C)	1	Lower input voltage limit	RW	0→120	100 (0x64)	Voltage (v) = register value / 5. e.g. 100 ⇔ 20V If the supply voltage falls below this figure, the Control enable register will be set to 0;
13 (0x0D)	1	Upper input voltage limit	RW	120→243	243 (0xF3)	As per above. 240 ⇔ 48V

						If the supply voltage goes above this figure, the Control enable register will be set to 0;
14 (0x0E)	2	Phase current limit	RW	M1: 0→6250	M1: 6250 (0x186A)	Maximum phase current in mA.  To avoid damage to your Mercury servo, do not routinely exceed the rated torque figure.
16 (0x10)	2	Angular velocity limit	RW	0 → 3200	3200 (0xC80)	2000 equates to 2000 milli-rad/s.
18 (0x12)	2	Acceleration limit	RW	0→48000	0	0 indicates the maximum possible acceleration.
20 (0x14)	4	Horn position offset	RW	Single-turn: -4096→4095 (- $\pi/2$ to $\pi/2$) Multi-turn: -4,177,920 → 4,177,919	0	See details below.
24 (0x18)	2	Moving threshold	RW	0 → 2,000	20 (0x0014)	Moving velocity threshold in milli-rad/s. See details below.
26 (0x1a)	1	Input mode	RW	0 → 1		See input modes for more details.
Non-volatile default registers are copied to their volatile counterparts on reset or power-up.						
27 (0x1b)	2	Default position proportional gain (Kp)	RW	0 → 2000	16 (0xa0)	See position details below.
29 (0x1d)	2	Default Velocity feedforward gain	RW	0 → 16383	0	See feedforward details below.

31 (0x1f)	2	Default current feedforward gain	RW	0 → 1000	0	
33 (0x21)	2	Default velocity integral gain	RW	0 → 4000	2000	See velocity integral details below.
35 (0x23)	2	Default velocity proportional gain	RW	0 → 4000	500	See velocity proportional details below.
37 (0x25)	2	Default angular velocity profile	RW	0→Angular velocity limit	300	See the angular velocity profile details below.
39 (0x27)	2	Default acceleration profile	RW	0→Acceleration limit	500	See acceleration profile details below.
41 (0x29)	2	Reserved				
...	5	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 details 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→ 2000	From default position proportional gain (Kp)	See position details below.
56 (0x38)	2	Velocity feedforward gain	RW	0→ 16,383	From default Velocity feedforward gain	See feedforward details below.
58 (0x3A)	2	Current feedforward gain	RW	0→ 1000	From default current feedforward gain	
60 (0x3C)	2	Velocity integral gain	RW	0→ 4000	Default velocity integral gain	See velocity integral details below.

62 (0x3E)	2	Velocity proportional gain	RW	0→ 4000	Default velocity proportional gain	See velocity proportional details below.
64 (0x40)	2	Reserved				
66 (0x42)	2	Reserved				
68 (0x44)	2	Angular velocity profile	RW	0→Angular velocity limit	From default angular velocity profile	See the angular velocity profile details below.
70 (0x46)	2	Acceleration profile	RW	0→Acceleration limit	From default acceleration profile	See acceleration profile details below.
72 (0x48)	2	Reserved				
74 (0x4A)	2	Reserved				
76 (0x4C)	2	Reserved				
78 (0x4E)	4	Target position	RW	Single-turn: 0→16383 (- π to π) Multi-turn: -4,177,920 → 4,177,919	Value from actual position register.	See servo output position section below
82 (0x52)	2	Target angular velocity	RW	0→3200	0	Target angular velocity in milli-rad/s. See details below. Only used in continuous rotation mode.
84 (0x54)	2	Target torque	RW	0→value from Torque limit register.	0	Target torque in units of 10 mNm. See torque mode details below.
86 (0x56)	2	Actual phase current	R			1 ⇔ 1mA
88 (0x58)	2	Actual angular velocity	R			Angular velocity in milli-rad/s. e.g. 3200 milli-rad/s ⇔ 3.2 rad/s ⇔ ~30 rpm

90 (0x5a)	4	Actual position	R	Single-turn: -8192→8191 (-π to π) Multi-turn: -4,177,920 → 4,177,919	-	See servo output position section below
94 (0x5E)	2	Actual torque	R			Torque in units of 10 mNm 100 ⇔ 1Nm
96 (0x60)	1	Actual voltage	R			Voltage = register value / 10. e.g. 200 ⇔ 20V
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 details below.
100 (0x64)	1	Trajectory status	R			Indicates the current trajectory status. See details below.
101 (0x65)	2	Calculated angular velocity profile trajectory	R			
103 (0x67)	4	Calculated position profile trajectory	R			
107 (0x6b)	1	Hardware status	R			See details below

Please note that **two's complement** is used to represent all 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
---------------	-------------	-----------	-----------

Non-volatile			
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	8191 (0x1FFF)
9 (0x09)	Counter-clockwise (CCW) angle limit	0	-8192 (0xE000)
11 (0x0B)	Upper temperature limit	0	75 (0x4b)
12 (0x0C)	Lower input voltage limit	100 (0x64)	120 (0x78)
13 (0x0D)	Upper input voltage limit	120 (0x78)	243 (0xF3)
14 (0x0E)	Phase current limit	0	6250 (0x186A)
16 (0x10)	Angular velocity limit	0	3200 (0xC80)
18 (0x12)	Acceleration limit	0	48000 (0xBB80)
20 (0x14)	Home position offset	Single-turn: -4096 Multi-turn: -4,177,920	Single-turn: 4095 Multi-turn: 4,177,919
24 (0x18)	Moving threshold	0	2,000 (0x7D0)
26 (0x1a)	Input mode	0	1
Non-volatile default registers are copied to their volatile counterparts on reset or power-up.			
27 (0x1b)	Position proportional gain	0	2000 (0x7D0)
29 (0x1d)	Velocity feedforward gain	0	16,383 (0x3FFF)
31 (0x1f)	Current feedforward gain	0	1000 (0x3E8)
33 (0x21)	Velocity integral gain	0	4000 (0xFA0)
35 (0x23)	Velocity proportional gain	0	4000 (0xFA0)
37 (0x25)	Angular velocity profile	0	Angular velocity limit
39 (0x27)	Acceleration profile	0	Acceleration limit

Volatile			
48 (0x30)	Control enable	0	1
49 (0x31)	Pending shadow instruction	0	1
54 (0x36)	Position proportional gain	0	2000 (0x7D0)
56 (0x38)	Velocity feedforward gain	0	16,383 (0x3FFF)
58 (0x3A)	Current feedforward gain	0	1000 (0x3E8)
60 (0x3C)	Velocity integral gain	0	4000 (0xFA0)
62 (0x3E)	Velocity proportional gain	0	4000 (0xFA0)
68 (0x44)	Angular velocity profile	0	Angular velocity limit
70 (0x46)	Acceleration profile	0	Acceleration limit
78 (0x4E)	Target position	Single-turn: CW angle limit Multi-turn: -4,177,920	Single-turn: CCW angle limit Multi-turn: 4,177,919
80 (0x50)	Target angular velocity	0	Angular velocity limit
82 (0x52)	Target torque	0	Torque limit
96 (0x60)	Pending shadow instruction	0	1

5.3 Register details

5.3.1 Control enable

This register controls the following:

Value	Description
Bit 0 (R/W)	0: <ul style="list-style-type: none"> • unlocks all non-volatile registers • cuts all power to the motor 1: <ul style="list-style-type: none"> • can only be set if the motor has already been calibrated. • locks all non-volatile registers • enables the motor
Bit 1 (R/W)	1: <ul style="list-style-type: none"> • Calibrates the motor. This causes the motor

	<p>to rotate by a small amount in order to align the rotor with the zero position of the shaft encoder. Maximum output movement = 0.0785 radians. Cleared automatically when calibrated.</p>
--	--

5.3.2 Baud rates

The supported baud rates of the Mercury servo are as follows:

Value	Baud Rate
0 (0x00)	9600
1 (0x01)	19200
2 (0x02)	38400
3 (0x03)	57600
4 (0x04)	115200
5 (0x05)	1000000

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 2µ seconds * the register value.

5.4 Operating modes

5.4.1 Operating modes table

Mode	Value	Description
Torque mode	0	<p>Controls the output torque of the Mercury servo.</p> <p>Makes use of:</p> <ul style="list-style-type: none"> • Acceleration limit • Acceleration profile • Torque limit • Target torque • Angular velocity limit
Continuous rotation (Wheel) mode	1	<p>Controls the angular velocity of the Mercury servo.</p> <ul style="list-style-type: none"> • Rotates the servo at the target angular velocity. • The direction bit (15) controls the direction of rotation. <p>Makes use of:</p> <ul style="list-style-type: none"> • Acceleration limit

		<ul style="list-style-type: none"> • Acceleration profile • Angular velocity limit • Target angular velocity • Velocity proportional gain (vKp) • Velocity integral gain (vKi) • Current feedforward gain cKff
Single-turn (Joint) position mode	2	<p>Moves to the target position based on the specified velocity and acceleration profiles.</p> <p>Makes use of:</p> <ul style="list-style-type: none"> • Home position offset • CW & CCW angle limits • Input mode • Acceleration limit • Acceleration profile (trajectory input mode only) • Angular velocity limit • Angular velocity profile • Position proportional gain (pKp) • Velocity feedforward gain vKff • Velocity integral gain (vKi) • Velocity proportional gain (vKp) • Current feedforward gain cKff
Multi-turn position mode	3	<p>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°. Maximum turns are -256→256.</p> <p>Makes use of:</p> <ul style="list-style-type: none"> • Home position offset • Input mode • Acceleration limit • Acceleration profile (trajectory input mode only) • Angular velocity limit • Angular velocity profile • Position proportional gain (pKp) • Velocity feedforward gain vKff • Velocity integral gain (vKi) • Velocity proportional gain (vKp) • Current feedforward gain cKff
Stepper mode	4	<p>In this mode, digital pins D1 and D2 are used as 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"> • Home position offset • Acceleration limit • Angular velocity limit

5.4.2 Input modes table

Input modes allow for different control strategies to be applied to input commands.

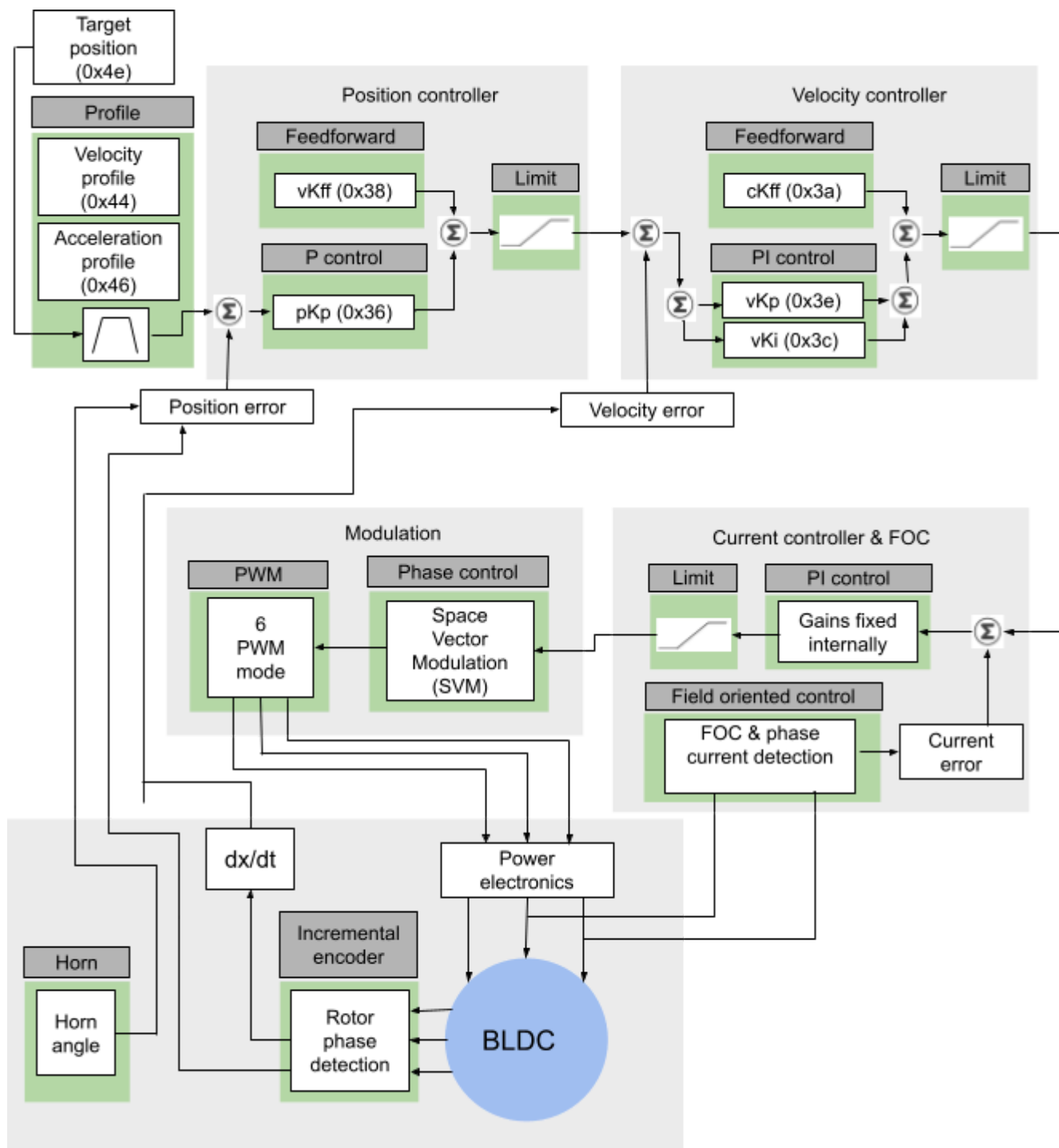
Mode	Value	Description
Trajectory mode	0	In this mode, the Mercury servo generates a trajectory in order to traverse to the target position. See profile modes for more details.
Position mode	1	In this mode, no trajectory profiles are applied. Instead, the resultant profiles are determined only by the controller settings and the target position. The angular velocity limit is set by the angular velocity profile register value.

5.4.3 Mercury position control

5.4.3.1 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 position mode, the velocity feedforward quantity ($vKff$) is added to the calculated desired velocity value.

In wheel mode, the current setpoint is set directly from the current feedforward ($cKff$) quantity;

The same is also true in position mode, but only in Position input mode. In trajectory mode, the current setpoint is generated from the trajectory.

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 trajectory velocity profile in Joint and Multi-turn operating mode. Profile angular velocity is represented in milli-rad/s. The maximum angular velocity is approximately 20 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 milli-rad/s^2 . Valid values are $0 \rightarrow 48000$, representing a maximum profile acceleration of 48 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.

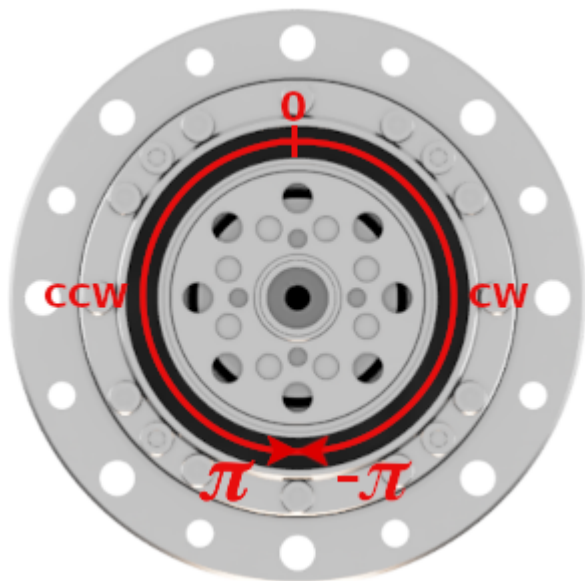
The angular velocity of the horn at the point when the horn starts to decelerate towards the position setpoint, is less (or equal) to the specified angular velocity profile value. The resulting profile is thus triangular.

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.

5.4.3.2 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 **actual position** register values at π and $-\pi$ radians 8191, and -8192 respectively. This is with a **Home position offset** register value of 0.

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

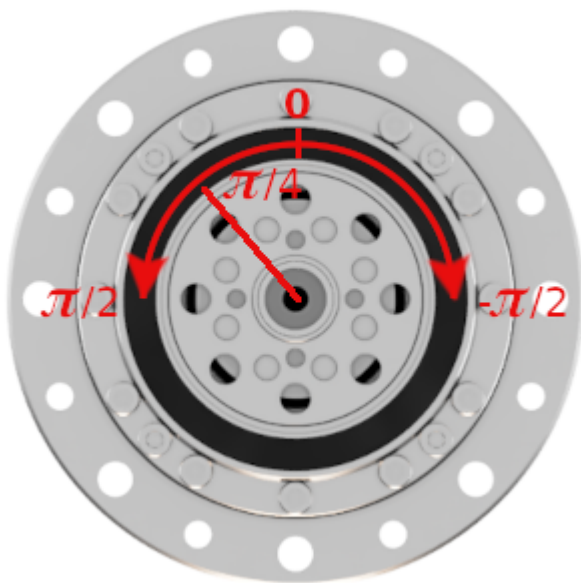
CW and CCW angle limits register are only relevant in joint mode. In multi-turn mode, the CW and CCW angle limits are ignored..

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 2π radians [360°]), the valid range is $-4096 \rightarrow 4095$. 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 $-4,177,920 \rightarrow 4,177,919$. This is the equivalent of ± 255 turns.



In the above image:

- The real position is $\pi/4$ radians [45°]
- The home position offset is -2048
- The actual position register value is 0.

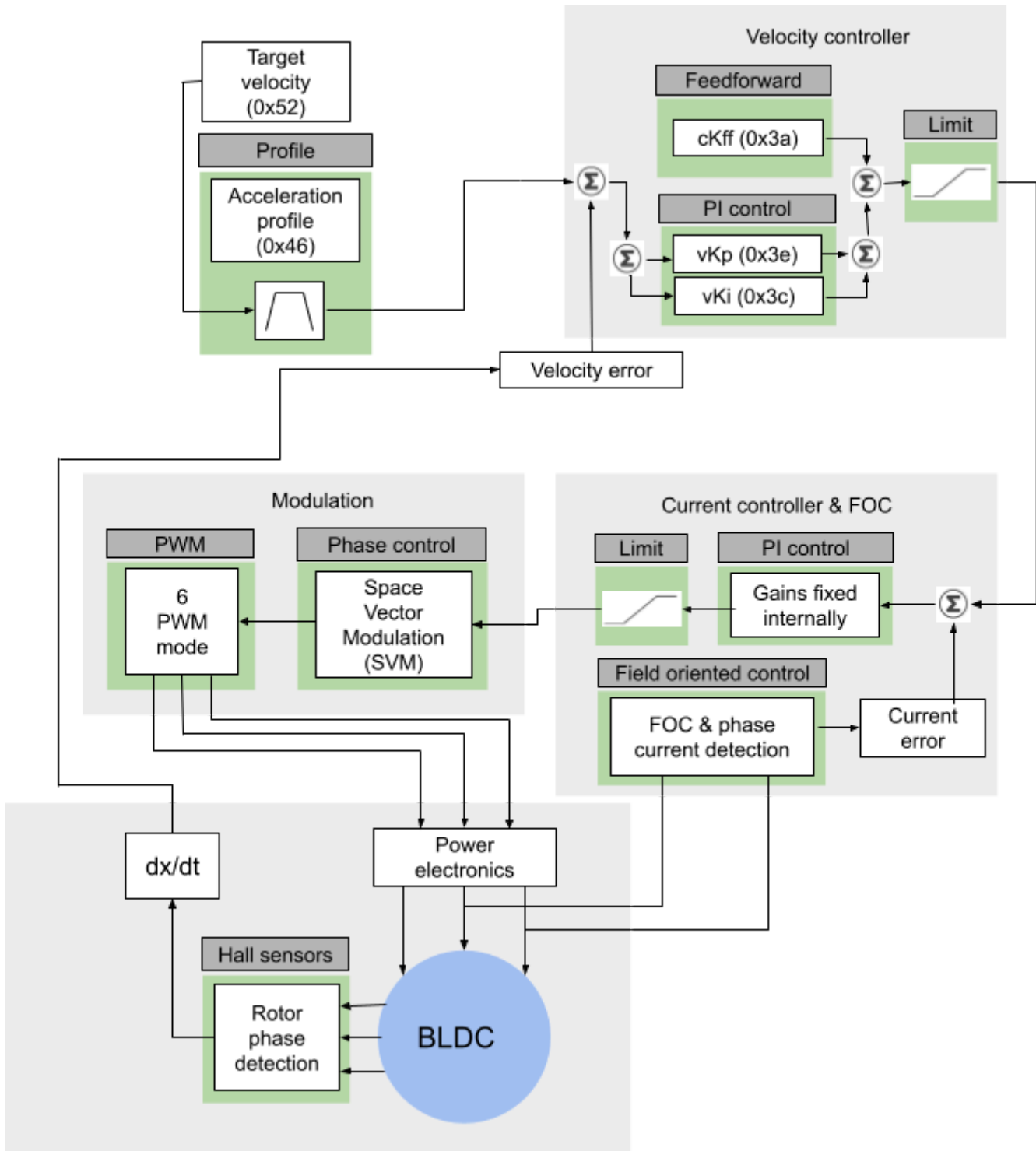
That is, the actual position register value = real position + home position offset.

5.4.4 Mercury velocity control

5.4.4.1 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 20 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

5.4.5 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)

5.4.6 Stepper mode

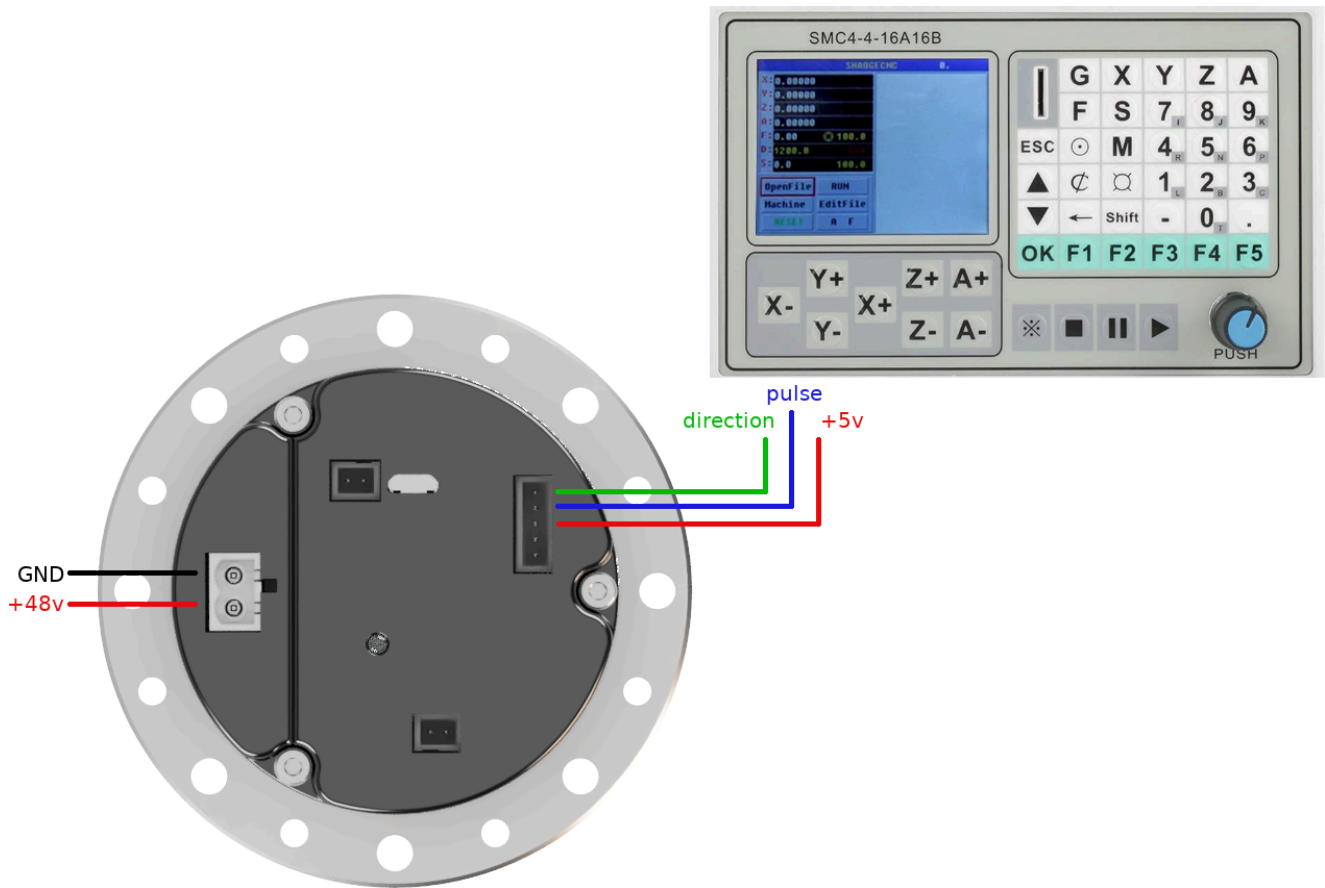
This control mode allows for the straightforward connection of Mercury digital servos to CNC-type controllers with STEP and DIRECTION outputs. Unlike conventional open-loop stepper motors, the Mercury servo's control electronics will ensure the correct output response based on input steps and chosen 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 <= 50ms

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

5.5 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.

5.6 Hardware status

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

Bit	Status condition	Description
7	n/a	

6	Motor shutdown	This error can occur following an FOC (field oriented control) error that has forced the motor to shutdown.
5	Overload error	This error can be the result of one of the following conditions: <ul style="list-style-type: none"> • An overcurrent alarm has been generated by the BLDC driver. • An under-voltage has been detected by the BLDC driver The control_enable register is set to zero when this error occurs.
4	Angle limit error	The horn angle is either: <ul style="list-style-type: none"> • Outside of the specified CW or CCW limits in Joint mode • > +/- 255 revolutions from the zero point in Multi-turn mode
3	Encoder error	An encoder error has been detected. The control_enable register is set to zero when this error occurs.
2	Overheating error	The internal temperature of the Mercury servo has exceeded the specified temperature limit. This can be the result of one of the following conditions: <ul style="list-style-type: none"> • Measured temperature (97) has exceeded the temperature limit (11) • An over-temperature alarm generated from the tmc6200 driver. This temperature limit is fixed at 120°C. This error results in a device shutdown. To recover from a shutdown, the device must first be powered off. The device will not be re-enabled until both of the above conditions are no longer true. In the event of a tmc6200 driver error, the exact details of the error can be seen at register XXXX. See the "Register fields" tab for details.
1	Motor not synchronized	The motor is not synchronized with the encoder index The control_enable register bit 0 cannot be set while the motor is unsynchronised.
0	Input voltage error	The input (supply) voltage is either: <ul style="list-style-type: none"> • less than the specified minimum voltage limit • greater than the maximum specified voltage limit The control_enable register is set to zero when this error occurs.

6. Diagnostics

6.1 LED states

The following table details the various possible LED states of the Mercury servo motor:

LED State	Description
Solid RED	The servo is initialising.

Solid GREEN	The servo is both initialised and synchronised
GREEN LED toggles between ON and OFF every 1s.	The servo has been initialised, but the motor has not been synchronised.
RED LED toggles between ON and OFF every 1s.	Overload error, encoder error or input voltage error.
RED LED toggles between ON and OFF every 0.25s.	Overheating error.
Toggling between RED and GREEN LEDs every 1s.	Motor shutdown has occurred.

7. ROS2 integration

7.1 Github repositories

GitHub repository	Branch	Description
dynamixel_hardware	rolling	ROS2 Iron: ros2_control implementation for mercury & dynamixel servo motors.
dynamixel-workbench	mercury-ros2	ROS2 Iron: Implementation that supports mercury servo motors while maintaining support for dynamixel servos.
MercurySDK	ros2	A modified version of the MercurySDK that allows integration with dynamixel-workbench.
mercury_ros2_examples	master	Example robot description(s).

7.1 Building a ROS2 Mercury example

1. Create a workspace folder:

```
mkdir -p ~/mcy_ws/src  
cd ~/mcy_ws
```

2. Clone the required GitHub repos:

```
cd src
```

```
git clone git@github.com:RobotArticulation/dynamixel\_hardware.git  
git checkout rolling
```

```
git clone git@github.com:RobotArticulation/dynamixel-workbench.git  
git checkout mercury-ros2
```

```
git clone git@github.com:RobotArticulation/MercurySDK.git  
git checkout ros2
```

```
git clone git@github.com:RobotArticulation/mercury\_ros2\_examples.git
```

3. Build the workspace:

```
cd ~/mcy_ws  
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

4. Running an example robot description:

This example connects a single Mercury servo and allows control of the servo via the chosen ros2 action interface. In this example, we use the FollowJointTrajectory action.

Servo configuration:

Set the Mercury servo's configuration in:

~/mcy_wd/src/mercury_ros2_examples/mercury_description/urdf/mercury.ros2_control.xacro is correct for the connected servo motor(s).

Terminal #1:

```
cd ~/mcy_ws
source install/setup.bash

ros2 launch mercury_description mercury.launch.py
```

Terminal #2:

Send goal positions:

```
ros2 action send_goal /joint_trajectory_controller/follow_joint_trajectory
control_msgs/action/FollowJointTrajectory -f "{
  trajectory: {
    joint_names: [a_axis_servo_to_a_axis_horn],
    points: [
      { positions: [1.5], time_from_start: { sec: 2 } },
      { positions: [-1.5], time_from_start: { sec: 6 } },
      { positions: [0], time_from_start: { sec: 8 } },
      { positions: [0], time_from_start: { sec: 10 } }
    ]
  }
}"
```