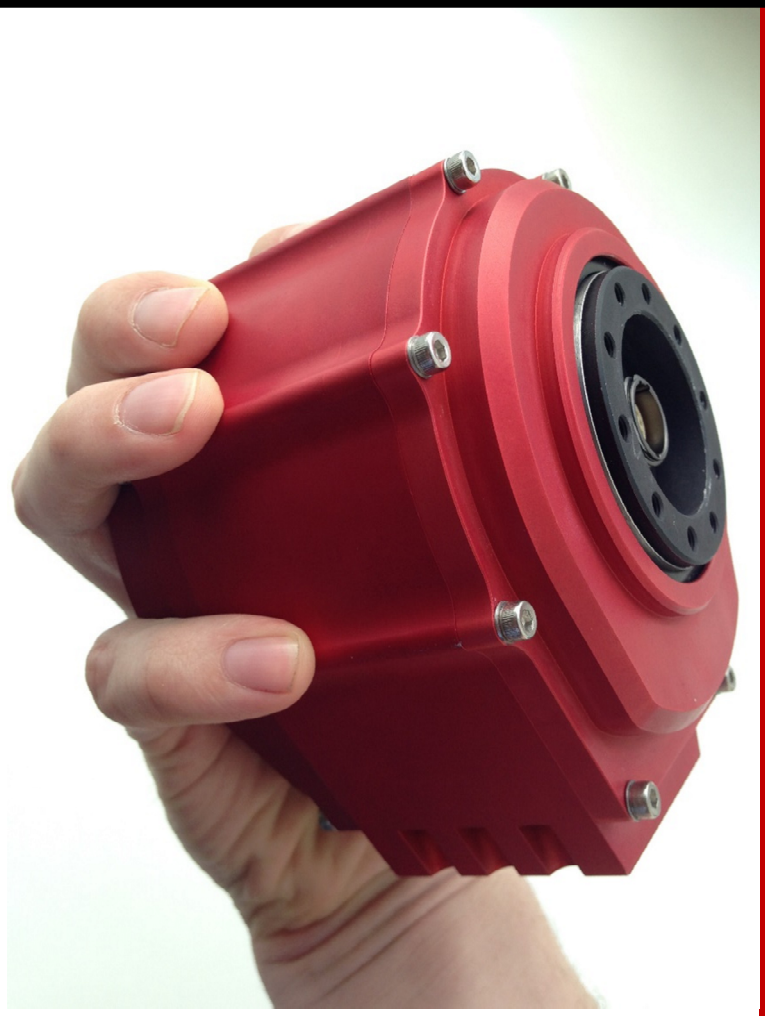
Servosila CANbus Protocol



Revision 5

www.servosila.com

TABLE OF CONTENTS

1. Introduction

Servosila CANbus protocol is a communication protocol designed for controlling various servo mechanisms produced by Servosila such as:

- Servosila Servo Drives,
- Servosila Grippers,
- Servosila Chassis including a pair of drive motors and a servo mechanism of the flipper threads,
- Servosila Robotic Arms.

The two-way message exchange protocol allows to:

- *Send* commands to the servo mechanisms,
- *Receive* state and status information back from the servo mechanism at regular time intervals.

The Servosila protocol uses CANbus as a physical and data layer protocol. Please refer to available online resources if CANbus network is a new subject to you. For example, please follow this link as a starting point:

https://en.wikipedia.org/wiki/CAN_bus

The rest of the document is written under assumption that the reader has a basic understanding of CANbus network.

2. CANbus Data Rate

- The default data rate for CANbus interfaces of all Servosila devices is **500 kbps.**
- The data rate can be changed via your device's configuration interface. In general, the data rate of the CANbus ranges from 250 kbps to 1 Mbit/s.

3. Node ID

- Every device (node) in a CANbus network shall have its own unique address, called Node ID.
- If you are interconnecting multiple devices via a CANbus network (e.g. a chain of servo drives and a robot's onboard computer), please make sure that each of those devices has a unique Node ID. The Node ID can be changed via your device's configuration interface.
- Please note that most Servosila devices come with the same factory-preconfigured Node ID. This means that the Node ID will almost certainly have to be changed when connecting it to a CANbus network.
- The protocol supports up to 126 slave devices with Node IDs ranging from 2 to 127.
- Please note that the Node ID of 0x01 has a special meaning; it is allocated for a single Master node/device (see below).

4. Single Master, Multiple Slave Nodes

- CANbus supports multiple Slave devices and just a single Master device. At any time only one Master node may be connected to the bus.
- Your robot's control computer or a microcontroller shall normally be configured as Master node. Servosila Robotic Head is one example of such a computer that connects to CANbus as a Master node.
- All other Servosila devices (servo drives, grippers, etc.) are preconfigured as Slave nodes by default.
- The Master node has to have a Node ID of 0x01.

5. CANbus Data Frames (Packets)

Devices connected to a CANbus network can exchange information by sending and receiving data *frames* which carry information in a payload section of the frame.

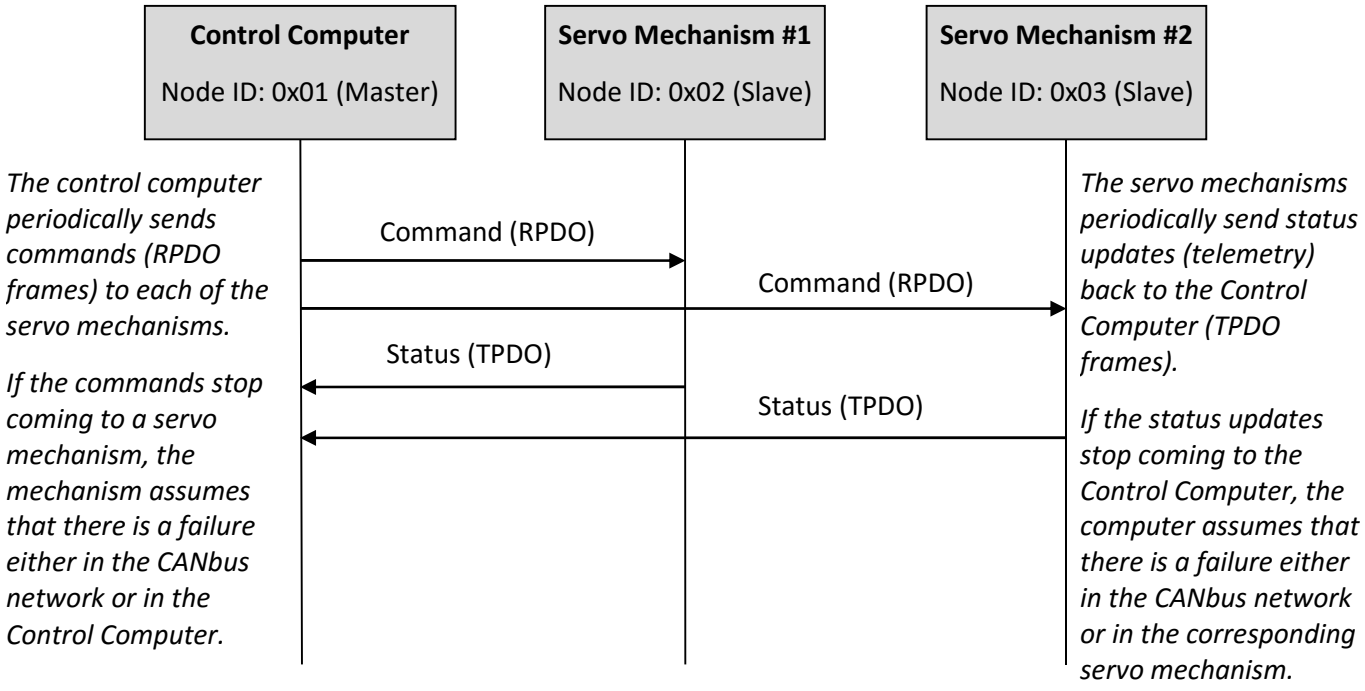
A standard CANbus data frame is shown in Table 1. Although there are *standard* and *extended* data frames in CANbus specification, the Servosila protocol makes use of only the *standard* frames; the *extended* frames are not used for communication to Servosila devices.

Table 1

	COB-ID	**RTR**	**Data Length**	**Data**
Length	11 bits	1 bit	4 bits	0-8 bytes

- **COB-ID** is a combination (a sum, see examples below) of a Node-ID and a Frame Type Code that tells what kind of information the packet carries. Supported Frame Type Codes differ from device to device and generally depend on what the device is capable of doing. This document provides reference tables (see sections below) for codes supported by various Servosila devices.
- **RTR** is set to 0.
- **Data** and **Data Length** fields are the *payload* fields of the packet. The format of the Data field depends on a device type. This document provides reference tables (see sections below) for payload fields of various packets sent by various Servosila devices.

6. Typical Packet Flow



The control computer/microcontroller is expected to continuously and asynchronously send commands to each of the servo mechanisms. For example, the computer might periodically send shaft position commands to a servo drive. The servo mechanisms are continuously and asynchronously send status, health and telemetry information to the single Master node in the CANbus network.

The special names, RPDO and TPDO, are used to describe the frames. Those names originally come from CANopen protocol upon which the Servosila protocol is loosely based. The names don't have any special meanings in the context of this document, and are used in the protocol description primarily for historical/legacy reasons. Just remember that "RPDO" refers to frames sent by a control computer to a servo mechanism; while "TPDO" refers to frames sent by a servo mechanism back to a control computer.

The control computer shall normally periodically send commands to servo mechanisms via the RPDOs frames. If a servo mechanism doesn't receive any RPDO frames within a preconfigured period of time (default, 5 sec), it assumes that either the CANbus network has failed or the control computer has failed, and halts operation as a safety measure until a fresh RPDO frame is received.

The servo mechanisms periodically send status updates (the TPDO frames) back to the Master device which is usually a control computer/microcontroller of a robot or a robotic vehicle. The sending frequency is configurable in the range of 0.5 to 100 Hz, and generally depends on the device type and its place and role in overall robotic control system.

7. Binary Frame Format

This section describes the binary formats of the servo command frames (RPDO) and the servo status frames (TPDO).

The protocol defines just 2 basic formats (Table 2) for servo *command* frames, called RPDO1 and RPDO2:

- RPDO1 type of the frame has a 2 bytes long data payload section and is generally used to deliver position or speed commands to the servo mechanism, be it a shaft position, or speed of a continuous rotation motor. The two-bytes-long payload section allows carrying a signed int16 value which in most cases refers to either shaft position or motor speed.
- RPDO2 type of the frame has just 1 byte in the payload section and is typically used to set specific bits that control behavior of a servo mechanism. For example, an emergency stop bit can be set by sending an RPDO2 frame down to a servo mechanism.

Table 2. Servo Command Frames (RPDO frames)

	COB-ID	**Num of bytes**	**1 Byte**	**2 Byte**
RPDO1 (set numerical value)	0x200+ Node ID	2	RData1 (0^{th} byte)	RData1 (1^{st} byte)
RPDO2 (set bit flags)	0x500+ Node ID	1	RData2	

Please note that the COB-ID value is used to differentiate between the two types of the RPDO frames. The COB-ID value is a sum of a Frame Type Code (either 0x200 or 0x500) and a Node ID. Please refer to Table 2 for an explanation about how COB-ID should be computed.

The payload section of both RPDO messages (the RData1 0^{th} byte, RData1 1^{st} byte and RData2 fields) are used differently by different servo mechanisms (please see reference tables in subsequent sections of the document).

The protocol also defines 4 different types of servo status feedback frames (Table 3). There are so many types of feedback frames, because a standard CANbus frame is not big enough to carry all the required status information in a single frame.

Table 3. Servo Status Feedback Frames (TPDO frames)

	COB-ID	**Num of bytes**	**1 – 4 Byte**				**5 - 8 Byte**	
TPDO0	0x180+ Node ID	8	TData 1				TData 2	
TPDO1	0x280+ Node ID	8	TData 3				TData 4	
TPDO2	0x380+ Node ID	8	TData5	-	TData6	-	-	-
TPDO3	0x480+ Node ID	4	TData7	-	TData8	-	-	-

The payload fields of the TPDO frames carry different information for different kinds of servo mechanisms (please see reference tables in subsequent sections of the document).

8. Reference 1: Servosila Servo Drives, Frame Format Specification

(Applicable to all servo drives produced by Servosila, including the flipper servo mechanism in the chassis, and the servo mechanism of the rotating grippers)

This section lists payload formats for Servo Drive *Commands* and Servo Drive *Status Feedback* frames.

Table 4. Servo Drive Command Fields

Payload Section	**Type**	**Range**	**Description**
RData 1 (0th byte)	uint16	0x0001 … 0x0FFF	**The commanded position of servo drive shaft in discrete units.** The 0…360deg range of the shaft rotation is divided into 4096 discrete points. One discrete unit is thus 0.087 deg. Any value out of range is ignored
RData 1 (1st byte)			
RData 2	byte	-	**The bit command byte 1.** This byte includes flags that affect behavior of the device (see applicable bits below).

Bits utilization in the Command Byte for RData2:

7 0

Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	ESTOP

ESTOP – Emergency stop bit. If it is set to 1, the servo drive will stop immediately.

Table 5. Servo Drive Status Feedback Fields

Payload Field	Type	Range	Description
TData 1	uint32	0x0001… 0x0FFF	Last received commanded shaft position. This value is a copy of RData1 received in the latest RPDO frame.
TData 2	uint32	0x0001… 0x0FFF	Current position of the shaft in discrete units.
TData 3	int32	0x0000… 0xFFFF	Current motor Speed in rev/min
TData 4	uint32	0x00000000 0x0000FFFF	DC Power source voltage in Volts*10 units. i.e. 245 means 24.5 V
TData 5	byte	-	Fault Byte (see a section 11 below)
TData 6	byte	-	Status Byte (see a section 12below)
TData 7	-	-	-
TData 8	-	-	-

9. Reference 2: Servosila Chassis Motors (continuous rotation motors)

Table 6. Chassis Motor Commands

Payload Field	Type	Range	Description
RData 1 LSB	int16	0xFC18 … 0x03E8	Desired speed in discrete units. One discrete unit = 0.1*Max drive shaft speed. I.e. 1000 involves motor to rotate with max speed, and -1000 involves to rotate with max speed in opposite direction. Any value out of range is ignored
RData 1 MSB			
RData 2	-	-	-

Table 7. Payload Fields for Status Feedback Frames – Chassis Motors

Payload Field	Type	Range	Description
TData 2	int32	0xFFFFFFFF… 0x00000000	Current speed of the motor's shaft in rpm.
TData 3	int32	0xFFFFFFFF… 0x00000000	Current value of the motor's phase in A*10 Units, i.e. 12 means 1.2 A
TData 4	uint32	0x00000000… 0x0000FFFF	DC Power source voltage in Volts*10 units. i.e. 245 means 24.5 V
TData 5	byte	-	Fault Byte (see a section below)
TData 6	byte	-	Status Byte (see a section below)
TData 7	-	-	-
TData 8	-	-	-

10. Reference 3: Servosila Grippers

There two type of grippers, a regular gripper and a rotating gripper:

- The **regular gripper** uses a continuous rotation motor to actuate the fingers of the gripper (the close/open motion). The control interface is the same as for the chassis main motors (see a spec in a section above).
- The **rotating gripper** is a combination of a servo mechanism for rotating the fingers, and as in a regular gripper, it includes a continuous rotation motor to close and open the fingers. The rotation of the gripper is controlled via the same interface as a servo drive (see a section above) while the finger open/close motion is controlled via the same interface as the chassis main motors (see a section above).

11. Reference 4: Format of the Faults Byte

7 0

F8	F7	F6	F5	F4	F3	F2	F1

f1 = overheat of the motor controller electronics
f2 = overvoltage, reduce power source voltage
f3 = under-voltage of the power source.
f4 = short circuit
f5 = emergency stop, sets if ESTOP command sent.
f7 = -
f8 = startup configuration fault, device firmware fault

12. Reference 6: Format of the Status Byte

7 0

S8	S7	S6	S5	S4	S3	S2	S1

s1 =
s2 =
s3 =
s4 = Power stage is off. It happens when overcurrent protection is latched.
s5 = Motor Stall, sets to 1 if motor shaft is blocked or the load torque is too high. It's a fault condition for servo drives and moving drives. It clears automatically after motor shaft is unblocked. But this is a normal situation for gripper drives. They hold the items with a shaft blocked.
s6 = Limits of the shaft position reached (only in servo drives).
s7 =
s8 = The drive controller program has been started successfully and is operating properly.

13. Appendix I. Factory Settings for the Node IDs of Mobile Robot "Engineer"

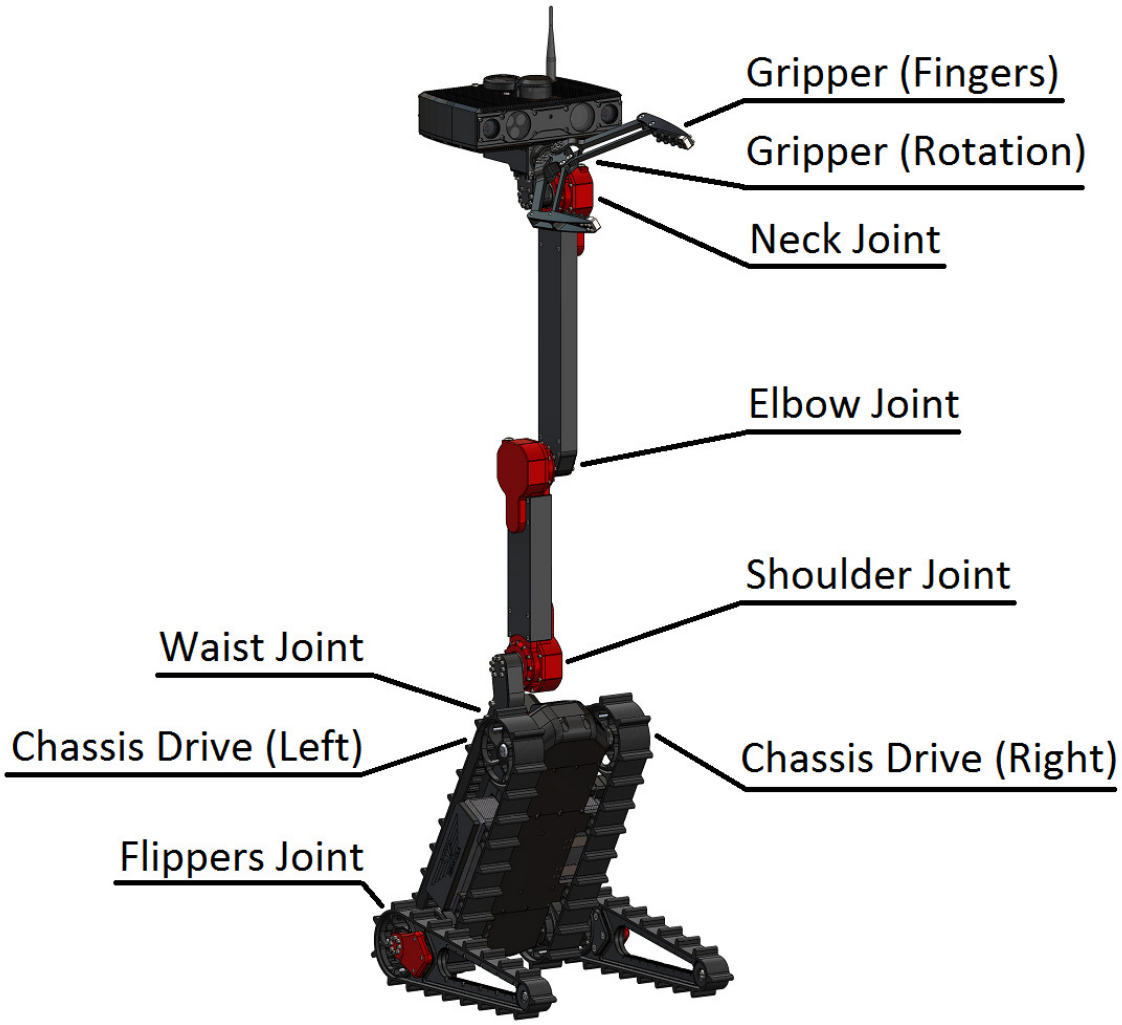


Table 8

Drive	Type of Mechanism	Node ID
Gripper (Fingers)	Continuous Rotation Motor	0x02
Gripper (Rotation)	Servo Drive	0x03
Elbow Joint	Servo Drive	0x04
Neck Joint	Servo Drives	0x05
Shoulder Joint	Servo Drive	0x06
Waist Joint	Servo Drive	0x07
Chassis Drive (Right)	Continuous Rotation Motor	0x08
Chassis Drive (Left)	Continuous Rotation Motor	0x09
Flippers Joint	Servo Drive	0x0A

14. Appendix II. Binary Examples of Frames

Servo Status Frames (TPDO frames)

185 [8] 0B 0C 00 00 34 0C 00 00

285 [8] 16 FF 00 00 EF 00 00 00

385 [8] 00 00 81 00 00 00 00 00

Description

A servo mechanism with a Node ID = 0x05 is a servo drive, moving to 0x0C0B position. The current shaft position is 0x0C34.

A motor inside the servo drive is rotating with -234 rpm (0xFF16) speed.

The voltage is 239/10 = 23.9 V (0xEF)

There are no faults, and a controller program inside the servo mechanism is started and is functioning properly.

Servo Command Frames (RPDO frames)

1) 205 [2] 0B 0C
2) 505 [1] 01
3) 205 [2] 0B 0C
4) 505 [1] 00
5) 205 [2] 0B 0C

Description

The first frame commands a servo drive with Node ID = 0x05 to move to the 0x0C0B position.

The second frame immediately stops the motor and removes voltage from the motor inside the servo drive.

The third command given immediately after the second it will be ignored since the drive is in a ESTOP mode

The forth command puts the servo drive back into the operation mode.

15. About Servosila

Servosila is a technology company that designs, produces and markets a range of mobile robots, servo drives, and robotic control systems as well as software that makes the mobile robots intelligent.

www.servosila.com

www.youtube.com/user/servosila/videos