CANopen User Manual

Firmware versions supported: 0x0000B020 or later

Revision History:

Revision	Date	Changes
V1.0.0	06/09/2021	- First Release
V1.0.1	30/09/2021	- SOLO MINI added
V1.0.2	01/04/2022	- Current limit description updated - Analogue Speed Resolution Division Coefficient added - Incremental Encoder Index Counts added - Position Feedback (Incremental Encoder and Hall sensors) updated - Position Reference description updated
V1.0.3	15/12/2022	- Adding Stall detection timeout - Adding Motion Profile settings and variables (st-curve time-based and time-optimal) - Adding Field Weakening definitions for Id current reference
V1.0.4	31/03/2022	- PDO messages are added - PDO examples are added
V1.0.5	10/05/2024	- Update in "Commanding Mode" write and read functions - Update in "Current Limit" range - Replacing "Drive Disable/Enable" instead of "Emergency Stop" and adding propper reading command - Update in "Ra" and 'La" range limit - Update in Feedback Control Mode" - Adding ZSFT parameters instead of previous sensorless parameters - Adding "Transition Speed" command write and read for sensorless HSO - Adding "Digital Outputs Register" write and read commands - Adding "Regeneration Current Limit" write and read command - Adding "Position Sensor Digital Filter Level" write and read command - Adding "PT1000 Sensor Voltage" read command - Adding "Digital Input Register" read command - Drive stop after expiry of life-time - Update in "Torque Reference (Iq/IM)" range - Update in "Output PWM Frequency (switching frequency) " default values - Update in "Magnetizing / Field Weakening Current Reference (Id)" range

Contents:

Introduction

Purpose of this user manual

This user manual intends to address the technical and practical aspects of CANopen communication for communicating with SOLO motor controller devices.

The CANopen interface for SOLO Motor Controllers follows the CiA DS301 communication profile, CiA (CAN in Automation) is the non-profit organization that governs the CANopen standard. They can be contacted at http://www.can-cia.org

CANopen is an open standard embedded machine control protocol. CAN is a serial communication interface and The CANopen protocol is developed for the CAN physical layer. In this document, CAN is reserved for physical layer descriptions, while CANopen refers to the communication protocol.

In this Manual the numbers in Hexa-decimal formats are shown either with an “h” at the end of the number like “0h” or with “0x” in the beginning of the number like “0x02”.

If after reading this manual or during your experimentations with our products you had any questions, you can use the SOLO Motor Controllers Forum to share with us the questions and get back your answers promptly.

CAN Bus Access points on SOLO UNO:

The CAN Bus on SOLO UNO can be accessed from two different sections as shown in Figure 1 below, The first section is the “CAN BUS/UART PINOUT” and the other one is located in “COMMUNICATION PORT” which both have the same functionality and identical circuitry.in Figure 1, CANH stands for CAN High and CANL stands for CAN Low connections for the CAN bus.

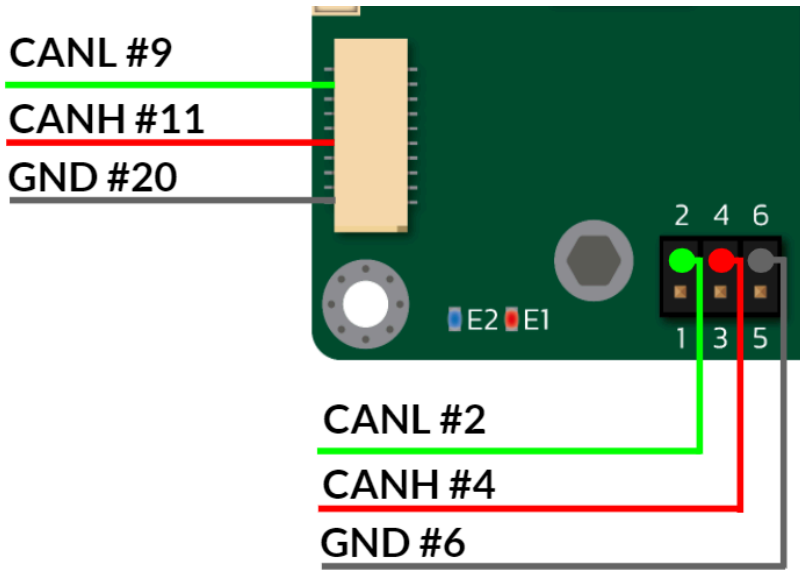


Figure 1- CAN bus access points on SOLO UNO

CAN Bus Access points on SOLO MINI:

The CAN Bus on SOLO MINI can be accessed from two different sections as shown in Figure 2 below, The first section is the “I/O port” and the other one is located in “COMMUNICATION PORT” which both have the same functionality and identical circuitry, in Figure 2, CANH stands for CAN High and CANL stands for CAN Low connections for the CAN bus.

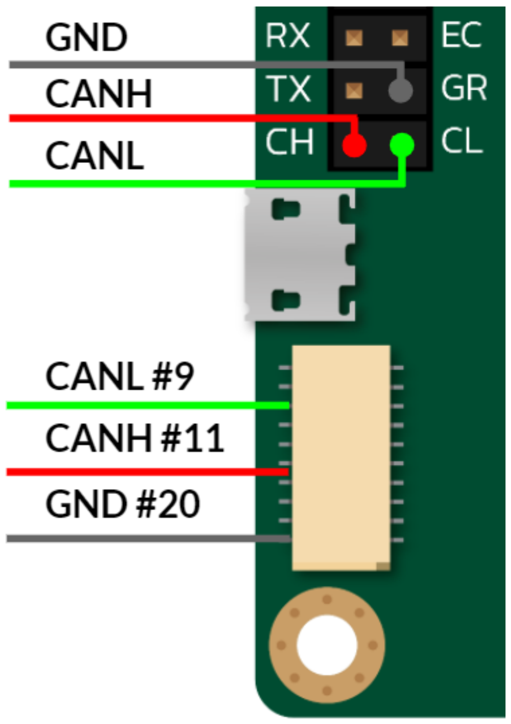


Figure 2- CAN bus access points on SOLO MINI

CAN Bus utilization on Legacy SOLO BETA models:

The CAN bus on legacy SOLO BETA models is not having the CAN transceiver IC mounted, so to use the CANopen functionality the user has to use an external CAN Transceiver module compatible with 3.3V CAN_TX and CAN_RX coming out of SOLO BETA as the CAN bus signals similar to Figure 3 below:

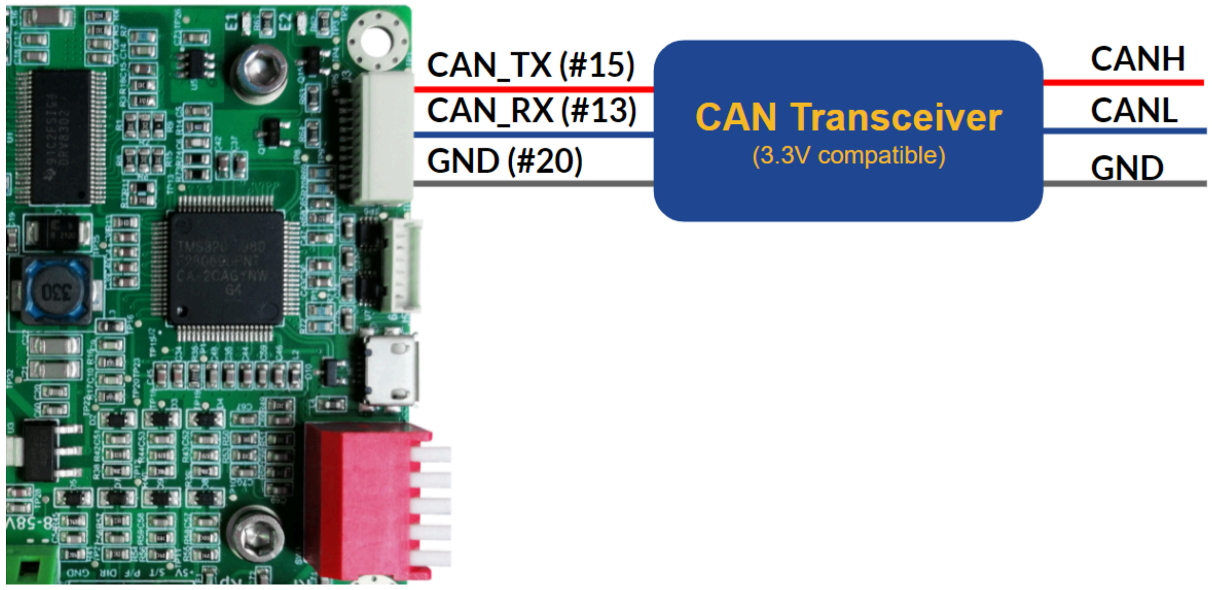


Figure 3 - CAN bus utilization on Legacy SOLO BETA models

CAN Bus Setup:

On the CAN bus, each units' CAN line is attached to the differential bus lines at CANH and CANL. Typically, the bus is a twisted pair of wires with a characteristic impedance of 120Ω, in the standard half-duplex multipoint topology. Each end of the bus should be terminated with 120Ω resistors in compliance with the standard to minimize signal reflections on the bus, the SOLO units, don't have the 120Ω termination on the board and the user must apply it externally at each end of the bus line as shown in Figure 4 below:

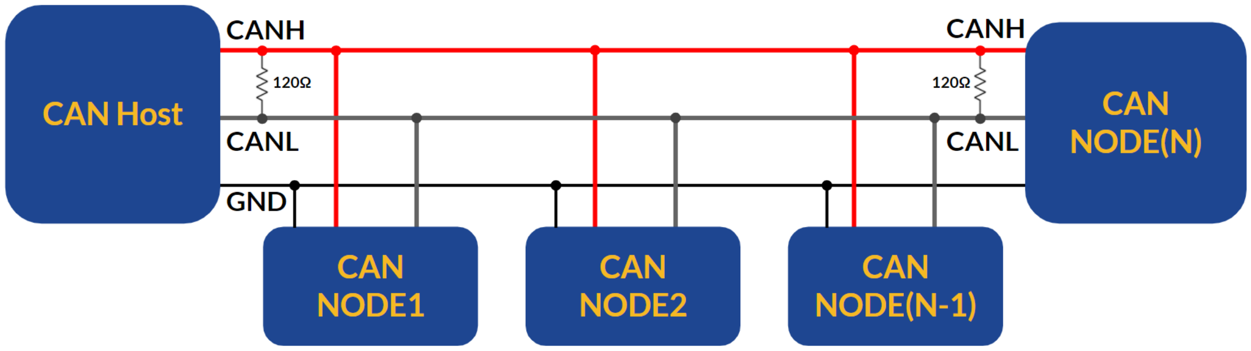


Figure 4 - CAN bus setup

Each Node shown above can be any module with CANopen enabled topology and based on CANopen standards, up to 127 nodes can be connected together on a CAN bus.

CANopen Objects:

All the functionalities that are supported by CANopen in SOLO motor controllers devices are brought into different objects, so in essence, each object can relate to a specific task or drive functionality. (Speed reference, current limit, position reference, etc)

The drive has a unique object for every parameter that needs to be stored or used. Access to the objects vary depending on what the object is used for. Objects may be writable, readable, or both, all the objects that are supported by SOLO are listed below.

Each object is accessible with a 16-bit address called the object index. Some objects contain sub components with 8-bit addresses called sub-indices. Reading and writing to objects is accomplished via CANopen Messages. Specific types of messages are designed to access specific objects.

Types of CANopen Objects:

SOLO at this point supports two types of CANopen objects:

NMT Control Objects 1000h – 1FFFh: These objects relate to CANopen Network Management and functionalities like Node Guarding, Lifeguarding, Heartbeat etc.

Manufacturer Specific Objects 3000h – 5FFFh: These objects are manufacturer Specific which are used to command the controller for setups or controlling, the details of all these objects can be found later in this manual.

CANopen Message Structure:

The message format for a CANopen frame is based on the CAN frame format. In the CAN protocol, the Data is transferred in frames consisting of an 11-bit or 29-bit CAN-ID, control bits such as the remote transfer bit (RTR), start bit and 4-bit Data length field, and 0 to 8 bytes of Data. The COB-ID, commonly referred to in CANopen, consists of the CAN-ID and the control bits. In CANopen, the 11-bit CAN ID is split into two parts: a 4-bit function code and a 7-bit CANopen node ID. The 7-bit size limitation restricts the amount of devices on a CANopen network to 127 nodes. All COB-IDs must be unique at any level to prevent conflicts on the bus.

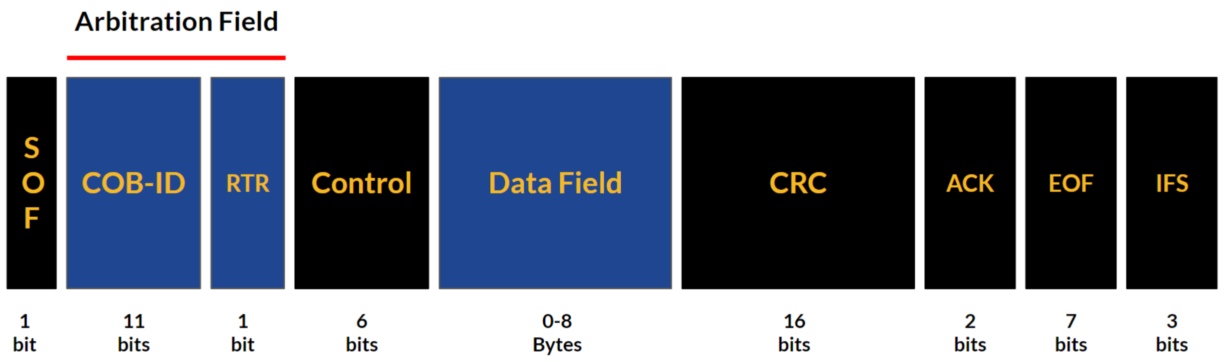


Figure 5- CANopen Frame bit sequence

Within CANopen protocol, which is actually a software layer over CAN physical layer, the only two sections that are important for us are shown in BLUE which are the Arbitration Field and the Data field(COB-ID, RTR and the Data Field), the rest of the sections are automatically arranged by CAN protocol or should be taken care independently from the CANopen structure of messaging.

The Arbitration Field (COB-ID + RTR):

The values in the arbitration field set the priority of the message. The closer the value is to zero, the higher the priority of the message. Higher priority messages will dominate, or take other messages on the CAN bus. Arbitration of the CAN bus is done at the CAN hardware level, thus ensuring that the highest priority message is transmitted first. CANopen message priority is determined by the message COB-ID bits and the RTR (Remote Transmit Request) bit.

COB-ID:

Every CANopen message has a unique COB-ID that identifies the message type and in case of node specific messages, the node number. In the case of a range of COB-IDs, the actual COB-ID for a message will depend on which node receives or transmits the message. These COB-IDs begin with a base number (assigned in CiA's DS301 specification) and the addition of the NODE-ID completes the COB-ID. In another word, The COB-ID is either a fixed number or it's the result of the summation of the command code and the real Node ID which is also known as the address of the device in the network. You can see on Table 1 below some COB-ID ranges for SOLO Motor Controllers:

Table 1: CANopen message types

Message Type	Description	COB-ID
NMT	Network Management (broadcast)	0h
NMT ERROR CONTROL	Network management error control	701h – 77Fh
BOOT-UP	Boot-Up Message	701h – 77Fh
SYNC	Synchronization message (broadcast)	80h
EMERGENCY	Emergency messages	81h - FFh
PDO	Process Data Objects	181h - 57Fh
SDO	Service Data Objects	581h – 67Fh

RTR Bit:

The remote transmission request (RTR) bit is used in some specific cases when the host would like to request information from a node. In particular, the RTR bit is used for node guard and TPDO requests. With the exception of these two cases, the RTR bit is always set to zero.

Node-ID

Every node on the CANopen network must have a unique node-ID, between 1 and 127. Node 0 is always considered as the host. You can use the USB connection to change your SOLO's Node ID (device address) using the Motion Terminal as the easiest way.

The Data Field:

The content of the Data field depends on the CANopen message type, for each object the Data type and the range are given below in this manual.

Little Endian Format

Numerical Data larger than 1 byte must be organized into "Little Endian" format. This means that the Data is broken into its individual bytes and sent Least-Significant Byte first (LSB first). For instance a generic 32 bits Data of 0x87963510 is sent like 0x10 0x35 0x96 0x87 as shown below:

Arbitration Field		Data Field							
COB-ID	**RTR**	**Byte1**	**Byte2**	**Byte3**	**Byte4**	**Byte5**	**Byte6**	**Byte7**	**Byte8**
XXXh	X	**0x10**	**0x35**	**0x96**	**0x87**	0x00	0x00	0x00	0x00

CANopen NMT state machine:

The CANopen Network Management on SOLO follows the following state machine shown in Figure 6 below, the values shown on the arrows are called the events and they indicate what transition between the states will happen if an NMT Message with respective event comes through the CANopen network for the Node with the same Node ID based on the message structure shown in Table 2 below.

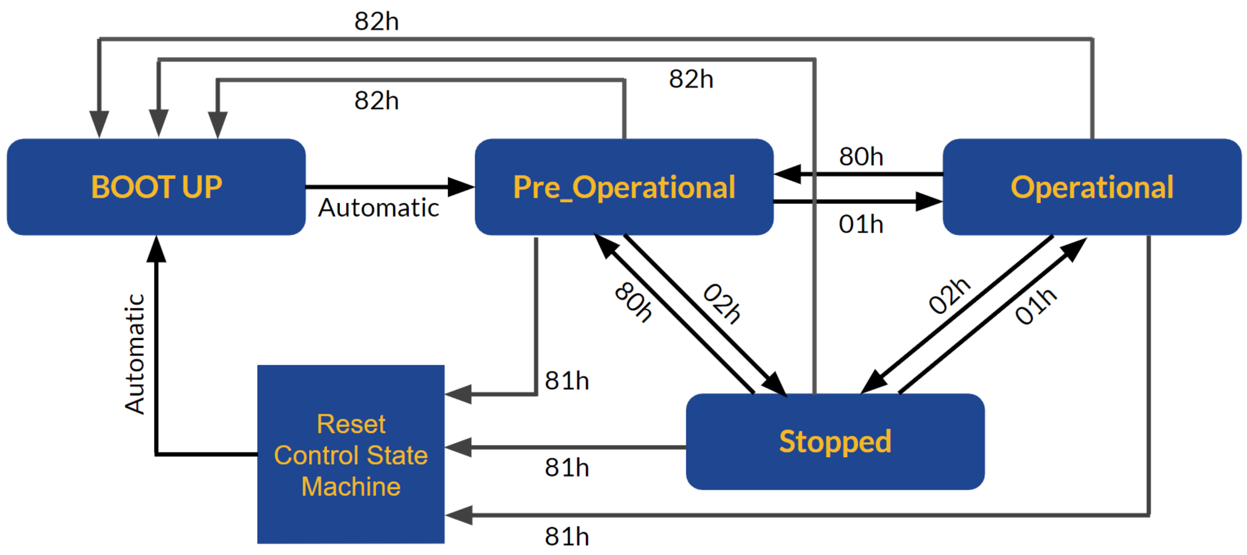


Figure 6 - Communication State Machine

Every CANopen device contains an internal Network Management server that communicates with an external NMT master. One device in a network, generally the host, may act as the NMT master. Through NMT messages, each CANopen device's network management server controls state changes within its built-in Communication State Machine. The Communication State Machine in all CANopen devices, is identical as specified by the DS301.
NMT messages have the highest priority. The 5 NMT messages that control the Communication State Machine each contain 2 Data bytes that identify the node number and a command to that node's state machine, table 2 below shows the general NMT state machine construction for sending these messages to SOLO Motor controllers devices:

Table 2 - NMT Message structure

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
000h	0	See Table 3	See Table 3	These Bytes are not sent					

Table 3 - NMT messages supported by SOLO Motor Controllers CANopen enabled devices

NMT Message	COB-ID	Data Byte 1	Data Byte 2	Description
Go to Operational State	0	01h	Node_ID	Sets the CANopen communication state machine on the designated node to Operational.
Go to Stopped State	0	02h	Node_ID	Sets the CANopen communication state machine on the designated node to Stopped.
Go to Pre-Operational State	0	80h	Node_ID	Sets the CANopen communication state machine on the designated node to Pre-Operational. In the pre-operational state, only NMT and SDO messages are allowed.
Reset the Node	0	81h	Node_ID	Resets the designated node (same as power recycle). Results in a Boot Up message sent by the node.
Reset the Communication	0	82h	Node_ID	Resets CANopen communication state machine on the designated node. Results in a Boot Up message sent by the node.

BOOT-UP State:

Upon Power up, the device will start the initialization process by checking critical functionalities and loading all the parameters saved on the memory. After Boot-up a message will be sent out and the drive will go to pre-operational state automatically.

Pre-Operational State:

Communication is limited to all message types except PDO messages. In this state, the Master can put the Device into any state possible shown in Table 3 based on the communication state diagram in Figure 6.

Operational State

Enables all message types including PDO messages. In this state, the NMT master can command the communication state to go into any state possible shown in Table 3 based on the communication state diagram in Figure 6.

Stopped State

Disables all message types except NMT messages; Node Guarding / Lifeguarding (explained below) remains active

NMT Error Control:

SOLO Motor Controller CANopen enable devices, support Node Guarding, Life Guarding, and Heartbeat protocol as NMT error controls.

Node Guarding

The NMT Master can monitor the communication status of each node using the Node Guarding protocol. During node guarding, a drive is polled periodically and is expected to respond with its communication state within a predefined time frame. Acceptable states are shown in Table 6. Note that responses indicating an acceptable state will alternate between two different values due to a toggle bit in the returned value. If there is no response, or an unacceptable state occurs, the NMT master reports an error to its host application. The Node Guard message is sent at time intervals, determined by the Guard Time (object 100Ch). The NMT slave (node) must reply to this message before the end of this time interval. Table 4 and Table 5 show the message format for an NMT master request and the correct NMT slave response. Note that the slave always responds with a toggle bit in byte 1, therefore the response will toggle between the two values shown in Table 6.

Life Guarding

The NMT slave monitors the status of the NMT master (Life Guarding). This event utilizes the Guard Time (object 100Ch) and Life Time Factor (object 100Dh) to determine a “Lifetime” for each NMT slave as shown in Figure 7 below (Lifetime = Guard Time * LifeTime Factor). If a node does not receive a Node Guard message within its Lifetime, the node assumes communication with the host is lost and triggers a communication error event and stops operating till the error register gets cleared. Each node may have a different Lifetime.

Table 4: NMT Master Node Guard Request Message Format (Host to Node)

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
700h + Node-ID	1	These Bytes are not sent.							

Table 5: NMT Slave Node Guard Reply Message Format (Node to Host)

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
700h + Node-ID	0	See Table 6	These Bytes are not sent.						

Table 6: Acceptable NMT slave return values for the states

Communication Status	Possible return values
Pre-Operational	0x7F or 0xFF
Operational	0x05 or 0x85
Stopped	0x04 or 0x84

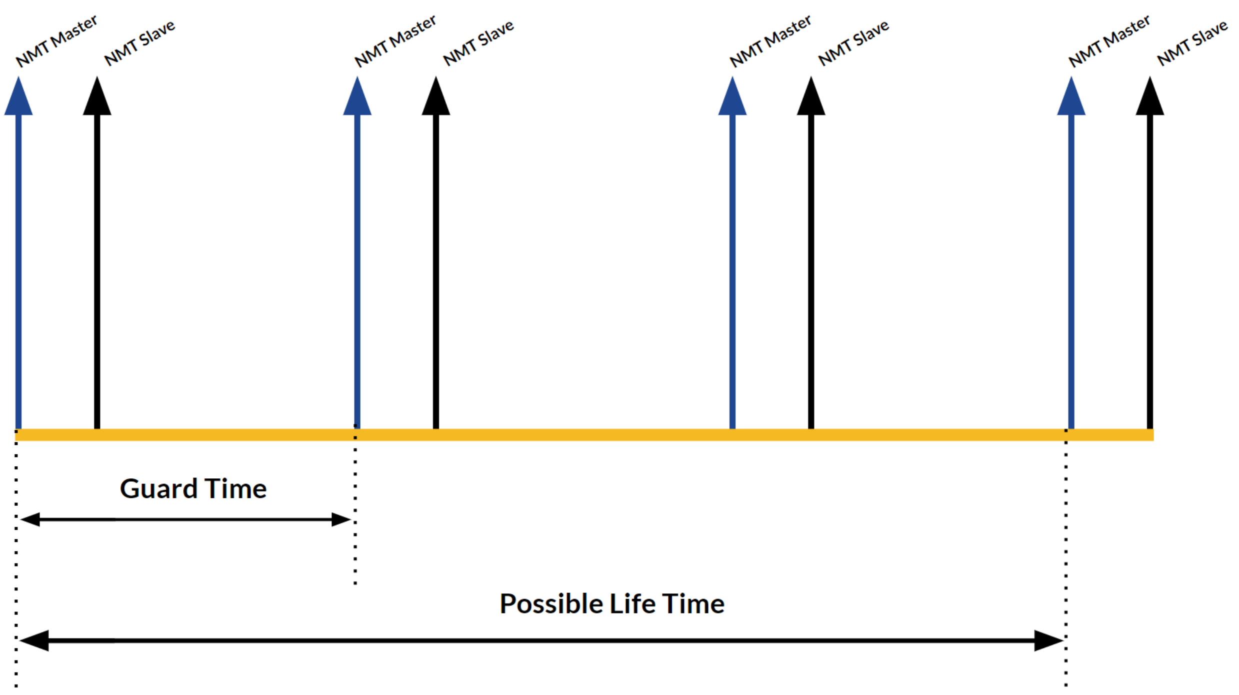


Figure 7- Life Time and Guard Time relation

As can be seen in Figure 7, the lifetime can be an integer factor of the Guard time.

Heartbeat:

The heartbeat error control method uses a producer to generate a periodic message. One or more consumer devices on the network listen for this message. If the producer fails to generate a message within a specified time frame, the consumer acts accordingly. Any drive on the network can be configured to be a producer of heartbeat based on the most recent firmware on SOLO controller, The producer heartbeat time (object 1017h) represents the time in milliseconds between successive heartbeat messages. It can be any integer value between 1 and 65535. When set to zero, the producer heartbeat is disabled.

BOOT-UP Message:

SOLO transmits a boot-up message after power up, communication reset, or power recycling. The CANopen master can monitor this and report an error if no boot-up message was received. The boot-up message of SOLO uses the same COB-ID as a Node Guard reply.

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
700h + Node-ID	0	0x00	These Bytes are not sent.						

EMERGENCY Message:

EMERGENCY messages are sent by the CANopen nodes to provide crucial status information to the CANopen host controller. An emergency message is transmitted only once per error event by the controller, it uses the message ID of “0x80” plus the Node ID as the COB-ID, as can be seen below, the index of Error register object is 1001h, and the errors can be overwritten using the object 3020h explained in object dictionary:

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
80h + Node-ID	0	0x00	0x00	0x00	Error Register (see table 7)	0x00	0x00	0x00	0x00

Table 7: Emergency Message Error Register bit description

Bit#	Error Status
0	Generic error
1	Current error
2	Voltage error
3	Temperature error
4	Communication error (overrun, error state)
5	Device profile specific error
6	Reserved (always 0)
7	Manufacturer-specific error

SYNC Message:

SYNC message acts like a "CLOCK" or in other words a "Triggering" mechanism over the CANopen network, in a sense that all the nodes can be synchronized to act on this message or after receiving a certain number of SYNC messages, SOLO Motor Controller drives can be set to act or send specific feedbacks to the network upon receiving SYNC messages using PDO messaging structure, the SYNC messages are having the following structure:

Arbitration Field		Data Field							
COB-ID	RTR	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
80h	0	These Bytes are not sent.							

SDO vs. PDO Message:

There are two methods for reading and writing Data to an object, Service Data Object (SDO) and Process Data Object (PDO) messages. An SDO consists of an outgoing message from host to node, possibly some intermediate messages between host and node, and a reply message from node to host; this is referred to as confirmed messaging. A PDO consists of a single unconfirmed message that requires less bus traffic relative to its SDO counterpart. Although PDOs make more efficient use of the CAN bus than do SDOs, PDO messages must be configured prior to use.

SDO Messages:

Service Data objects (SDOs) enable access to all entries of a CANopen object dictionary. One SDO consists of two CAN Data frames with different CAN-Identifiers. This is a confirmed communication service. With an SDO, a peer-to-peer client-server communication between two CANopen devices can be established on the broadcast medium CAN. The owner of the accessed object dictionary acts as a server of the SDO. The device that accesses the object dictionary of the other device is the SDO client.

SDO Read Request:

The SDO read request is sent from Host to the respective Node to read a parameter that is "Readable" and the Node will reply back to the message as can be seen in the following format, it worth mentioning that each object in the object dictionary can have a sub-index which will define all the related sub-objects to the main object, and each can have a different functionality. If an object has a special sub-index it will be mentioned in the object dictionary.

Host Initiating the SDO Read Command:

Arbitration Field	Data Field							
COB-ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
600h + Node-ID	0x40	Object index (LSB)	Object index (MSB)	Sub-index	Use 0x00 for each byte			

Node replying to the SDO Read Request:

Arbitration Field	Data Field							
COB-ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
580h + Node-ID	0x42	Object index (LSB)	Object index (MSB)	Sub-index	Data, LSB first			

SDO Write Request:

The SDO Write Request is sent from Host to the respective Node to write a parameter or set a desired reference point for an object which is “Writable”, similar to Read request, the Write request is also composed out of a request followed by a message from the Node as below:

Host attempts the SDO Write Command:

Arbitration Field	Data Field							
COB-ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
600h + Node-ID	0x22	Object index (LSB)	Object index (MSB)	Sub-index	Data, LSB first			

Node replying to the SDO Write Request:

Arbitration Field	Data Field							
COB-ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
580h + Node-ID	0x60	Object index (LSB)	Object index (MSB)	Sub-index	ignore			

SDO Abort Transfer Messages

When an error occurs during reading or writing an object, the node sends an abort transfer message to the host.

Arbitration Field	Data Field							
COB-ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
580h + Node-ID	0x80	Object index (LSB)	Object index (MSB)	Sub-index	See Table 8 below for abort codes			

Table 8 : Node indicating error in SDO communication

Abort Codes	Description
0x06020000	Object does not exist in the object dictionary
0x06090030	Value range of parameter exceeded (only for write access)

PDO Messages:

PDO messages are messages without any acknowledgment, serving two major roles in a CANopen network:

1. Reducing the CAN bus traffic by minimizing the number of bytes sent, specially since there is no confirmation (ACK) from the Node to the Host
2. Proving the possibility of having synchronous or asynchronous messages from the Nodes, preprogrammed to act based on an internal or external event.

There are two types of PDO messages known as TPDO and RPDO that are explained below. To learn more about PDOs and how to use them in practice you can refer to “PDO Examples” at the end of this document.

Receive Process Data Object (RPDO):

RPDOs are configured to receive a specific data from the Host and apply it in the Drive immediately or upon receiving SYNC message, each RPDO has to be configured before and has to be enabled, each RPDO should have a unique address with respect to the whole CANopen network ranging from 0x200 to 0x27F, this will be user’s responsibility to make sure the RPDOs are having unique and non-conflicting addresses with respect to other nodes in the network.

Transmit Process Data Object (TPDO):

TPDOs are configured to send a specific data from the Node to the Host upon receiving a number of SYNC messages or RTR request, each TPDO has to be configured before and has to be enabled, each TPDO should have a unique address with respect to the whole CANopen network ranging from 0x280 to 0x2FF, this will be user’s responsibility to make sure the TPDOs are having unique and non-conflicting addresses with respect to other nodes in the network.

PDO Configuration:

The configuration and setup of each PDO is achieved by setting the communication parameters visible in "PDO Communication Parameter table" in below , each PDO has a unique communication parameter ID ranging from 1400 h-15FFh and 1800h-19FFh for RPDOs and TPDOs respectively, the user has to select the right PDO for their purpose to set or receive the desired values.

PDO Communication Parameter table

PDO	PDO Communication Parameter Object	1st object mapping	COBID range
RPDO1	0x1414	Target Position	[0x200-0x27F]
RPDO2	0x1415	Target Velocity	[0x200-0x27F]
RPDO3	0x1416	Target Torque[Iq]	[0x200-0x27F]
RPDO4	0x1417	Target Direct Current[Id]	[0x200-0x27F]
RPDO5	0x1418	Control Mode	[0x200-0x27F]
RPDO6	0x1419	Motor Direction	[0x200-0x27F]
TPDO1	0x1814	Feedback Position	[0x280-0x2FF]
TPDO2	0x1815	Feedback Velocity	[0x280-0x2FF]
TPDO3	0x1816	Feedback Iq	[0x280-0x2FF]
TPDO4	0x1817	Feedback Id	[0x280-0x2FF]
TPDO5	0x1818	Error Register	[0x280-0x2FF]
TPDO6	0x1819	Board Temperature	[0x280-0x2FF]

PDO Communication Parameter Object

This object for each PDO provides the possibility of setting the COBID plus activation or deactivation of the respective PDO through its first sub-index(0x01) , it also provide the possibility of defining the Transmission type of each PDO through its second sub-index(0x02), for instance the COBID of the RPDO1 can be found in object 1414.01h and the transmission type resides in object 14014.02.

PDO Triggering Mechanism and COB-ID Configuration:

Each PDO is supposed to have a unique COB-ID with respect to the whole CANopen network, based on the “PDO Communication Parameter table” mentioned abve, the COB-ID for each PDO object can be assigned using the PDO Communication parameter object using its first sub-index(0x01), where the RPDO objects can have a range from 0x200 to 0x27F and the TPDO objects can have a value in range of 0x280 to 2FF for their COB-ID.
First sub-index (0x01) of each PDO’s Communication Parameter object contains the COB-ID and is a 32-bit data field segmented into five components as shown in below within its DATA part:

Bit 31	Bit 30	Bit 29	Bits 28-11	Bits 10 to 0
0/1	0/1	0	000000000000000000	COB-ID

The whole definition of this 32 bit is as below:

COB-ID Configuration definition table:

Bit Number	Value	Description
31 (MSB)	0	PDO message is disabled and will not respond to triggering mechanism, this is the default state of each PDO
	1	PDO message is Enabled and will respond to triggering mechanisms.
30	0	RTR is not allowed on this PDO
	1	RTR is allowed on this PDO
29	0	Not used
28-11	0	Not used
10-0 (LSB)	11-bits ID	Contains the COB-ID of each PDO within the allowed range.

PDO Transmission type Configuration

Each sub-index 02h of PDO's communication parameter is an 8-bit data field that defines the Transmission type of the respective PDO based on the Tables below and the type of the PDO:

TPDO Transmission Type configuration Table:

Value	TPDOs
0x00	Send the data at the next SYNC or immediately after RTR request
0x01-0xF0	Send the data at the next "n" number of SYNCs where "n" is a number ranging from 0x01 to 0xF0, or immediately after receiving RTR request

RPDO Transmission Type configuration Table:

Value	RPDOs
0x00-F0	The received message is held until the next SYNC message to be applied.
0xFE-0xFF	The received message will be applied immediately

RTR bit and TPDOs:

For TPDOs that are enabled,the host can send the COB-ID of the TPDO with RTR bit set to 1 to request an immediate receive of data from a TPDO, this will result in instant response of the TPDO and can help to increase the CAN bus efficiency further.

Object Dictionary:

NMT Control Objects:

1001h : Read Error Register

Code: 0x1001	Read Error Register				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-254]	N/A	R	V	0

Description:
This object reads the CANopen Error register, if the value of Data in the response is interpreted bit-wise, meaning that if each bit in the answer is 1 corresponding to error as shown in table below, to reset or overwrite the errors you can use the object 3020h.

Bit#	Error Status
0	Generic error
1	Current error
2	Voltage error
3	Temperature error
4	Communication error (overrun, error state)
5	Device profile specific error
6	Reserved (always 0)
7	Manufacturer-specific error

100Ch : Guard Time

Code: 0x100C	Guard Time				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-65535]	ms	R/W	V	0
Description: This object reads or writes the value of the Guard Time in Milliseconds					

100Dh : Life Time Factor

Code: 0x100D	Lifetime Factor				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-255]	N/A	R/W	V	0
Description: This object reads or writes the value of the Life Time Factor. Lifetime = Guard Time * Lifetime Factor					

1017h : Producer Heartbeat Time

Code: 0x100D	Producer Heartbeat Time				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-65535]	ms	R/W	V	0
Description: This object reads or writes the value of the Heartbeat time in Milliseconds for a node to produce heartbeats at requested time.					

SOLO CANopen Objects:

1414h : 1st RPDO Communication Parameter:

This PDO can be used to hold the desired Position Reference once the device is in Position Mode, subsequently it won't be valid in other modes, the final effect, data types and the range is identical to Object 301B.

Code: 0x1414.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1414.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1415h : 2nd RPDO Communication Parameter:

This PDO can be used to hold the desired Speed Reference once the device is in Velocity or Speed controlling Mode, subsequently it won't be valid in other modes, the final effect, data types and the range is identical to Object 3005.

Code: 0x1415.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1415.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1416h : 3rd RPDO Communication Parameter:

This PDO can be used to hold the desired Iq Reference (Torque References) once the device is in Torque Mode, subsequently it won't be valid in other modes, the final effect, data types and the range is identical to Object 3004.

Code: 0x1416.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1416.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1417h : 4th RPDO Communication Parameter:

This PDO can be used to hold the desired Id Reference (Direct Current References) once the device is in Position, Speed or Torque Mode, to be used for Field-Weakening purposes or to create magnetizing current for ACIM motors, the final effect, data types and the range is identical to Object 301A.

Code: 0x1417.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1417.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1418h : 5th RPDO Communication Parameter:

This PDO can be used to change the Control Mode Type to Position, Speed or Torque modes, the final effect, data types and the range is identical to Object 3016.

Code: 0x1418.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1418.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1419h : 6th RPDO Communication Parameter:

This PDO can be used to change the Direction of the Motor, the final effect, data types and the range is identical to Object 300C.

Code: 0x1419.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1419.02	Transmission Type				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1814h : 1st TPDO Communication Parameter:

This PDO can be used to read the Position Feedback pulses, the final effect, data types and the range is identical to Object 3037.

Code: 0x1814.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1814.02	Transmission Type				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1815h : 2nd TPDO Communication Parameter:

This PDO can be used to read the Speed feedback read from the motor, the final effect, data types and the range is identical to Object 3036.

Code: 0x1815.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1815.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1816h : 3rd TPDO Communication Parameter:

This PDO can be used to Iq Feedback (Torque Feedback) read from the Motor, the final effect, data types and the range is identical to Object 3034.

Code: 0x1816.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1816.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1817h : 4th TPDO Communication Parameter:

This PDO can be used to read the Id Feedback (Direct Current Feedback) read from the Motor, the final effect, data types and the range is identical to Object 3035.

Code: 0x1817.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1817.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1818h : 5th TPDO Communication Parameter:

This PDO can be used to read the Error Register from the Controller, the final effect, data types and the range is identical to Object 3020.

Code: 0x1818.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1818.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

1819h : 6th TPDO Communication Parameter:

This PDO can be used to read the Board Temperature from the Controller, the final effect, data types and the range is identical to Object 3039.

Code: 0x1819.01	COB-ID Configuration with Enabling/Disabling the PDO				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	M	0
Description: This object holds the COBID for the respective PDO as well as other parameters, to know more please refer to "PDO configuration section".					

Code: 0x1819.02	Transmission Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0-255	N/A	R/W	M	0
Description: This object defines how the PDO is transmitted, to know more please refer to "PDO Transmission Type configuration".					

3001h : Set Device Address

Code: 0x3001	Set Device Address				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[1-254]	N/A	R/W	M	1

Description:
This object Sets or Reads the desired device address for a SOLO unit; the address can be used to network multiple SOLO's in a single network if the address assigned to each unit is unique. In the CANOpen network the address of "0" is not acceptable for a SOLO unit, and if the address is set at "0", it will be considered as "1" automatically.

3002h : Commanding Mode

Code: 0x3002	Commanding Mode				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0 -2]	N/A	R/W	M	0

Description:
This command sets the mode of the operation of SOLO in terms of operating in pure Analogue mode, Digital Mode or Analogue mode with some gains taken digitally, based on the value of DATA in the packet following the table below:

DATA	Actions
0 (0x00000000)	Puts SOLO in Analogue Mode
1 (0x00000001)	Puts SOLO in Digital Mode
2 (0x00000002)	Put SOLO in Analogue Mode with Speed controller gains taken from the Digital setup (replacement of the Kp and Ki potentiometers)

3003h : Current Limit

Code: 0x3003	Current Limit				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - Max Device Limit]	Amps	R/W	M	32

Description:
This command defines the maximum allowed current into the motor in terms of Amps limited to the max continuous current of the controller.
By setting the current limit value at zero (both at Analogue or Digital Modes) SOLO will stop the switching at its output and generate OCP error, this will allow for free-wheeling of the Motor without injection of any current however to recover the drive from Error state, the Error register has to be cleared.

3004h : Torque Reference (Iq/IM)

Code: 0x3004	Torque Reference (Iq/IM)				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Sfxt(32-17)	[0 - Max Device Limit]	Amps	R/W	V	0

Description:
This object Sets or Reads the amount of desired current that acts in torque generation. In 3-phase motors this is called Iq or quadrature current as SOLO operates in FOC mode, however for DC brushed motors this value sets the reference for IM in DC brushed motors.
This command will be effective only once SOLO is in closed-loop digital Torque mode as in analogue mode the Torque reference is set through the “S/T” input pin once SOLO is in Torque Mode.
For all the motors including DC, BLDC, PMSM and ACIM the value of the torque reference can relate to Torque on the shaft of the motor based on following relation:
Requested Torque [N.m] = Torque Reference [A] x Motor’s Torque constant [N.m/A]

3005h : Speed Reference

Code: 0x3005	Speed Reference				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	[0 - 30,000]	RPM*	R/W	V	0

Description:
This object Sets or Reads the speed reference for SOLO once it's in Digital Speed Mode, as in analogue mode the reference is set through the "S/T" input pin with analogue voltages or PWM pulses once SOLO is in Speed Mode.
*The Speed Unit depends on the type of the Motor as well as the type of operation, and if SOLO is in Open-loop or Closed-loop it can take different meanings as shown below:

Motor Type (**Analogue or Digital Mode)**	**Closed-loop Sensor-less**	**Closed-loop Sensor-based**	**Open-loop**
BLDC - PMSM	RPM	RPM	RPM
BLDC - PMSM Ultrafast	RPM	RPM	RPM
DC Brushed	Motor Dependant**	RPM	Duty Cycle***
AC Induction Motor	RPM	RPM	RPM

Basically the major consideration will be for DC brushed motors while they are in Sensor-less or Open-loop Mode with the following conditions:

***Speed Unit for DC brushed motor in Closed-loop Sensor-less mode:** This is a qualitative value based on the observer gain defined for the sensorless speed estimator for DC brushed motors, the value can be ranged from 0 to 30,000 and the final speed of the motor for each value depends on the characteristics of the Motor itself like BEMF constant, the increase in the Motor's Speed with respect to the reference will be linear up until the nominal speed of the motor.

*****Speed Unit for DC brushed motor in Open-loop mode:** in This case the speed reference will act as the duty cycle percentage at the output on the Motor, the value can be between 0 to 30,000 which will be mapped into 0% duty cycle to 100% duty cycle, going from no speed to max speed.

3006h : Power Reference

Code: 0x3006	Power Reference				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 100.0]	N/A	R/W	V	0

Description:
This object Sets or Reads the amount of power percentage during only Open-loop mode for 3-phase motors. The value is from 0.0 to 100.0 standing for 0% to 100% output power on the shaft of the Motor.

3007h : Motor's Parameters Identification

Code: 0x3007	Motor Parameters Identification				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0 or 1	N/A	W	V	0

Description:
This object starts the Motor ID by putting value of 1 in the Data section of a packet sent with this command, SOLO will start identifying the electrical parameters of the Motor connected, The identification will take 1 second to be done and after that the Motor Inductance, Resistance and some internal parameters are Identified (or re-identified).
Identification process depends on the type of the motor selected, so before running the Identification the user has to make sure they have properly selected their motor type both in Analogue or Digital Mode.

3008h : Drive Disable/Enable

Code: 0x3008	Drive Disable/EnableStop				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uin 32	0/1	N/A	R/W	V	1

Description:
This command Disables or Enables the Controller resulting in deactivation or activation of the switching at the output, by disabling the drive, the effect of the Controller on the Motor will be almost eliminated (except for body diodes of the Mosfets) allowing freewheeling, this command shall be used only in Emergency conditions or when the Motor is nearly stopped and it shouldn't replace normal start and stopping of the motor using Torque, Speed or Position reference settings.

DATA	Action
0	Disable the Drive (output switching shut-down)
1	Enable the Drive (output switching activated)

3009h : Output PWM Frequency (switching frequency)

Code: 0x3009	Output PWM Frequency (switching frequency)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[8 - 8000]	Hz/kHz	R/W	M	20 / 80

Description:
This object Sets the output switching frequency of the whole power unit on the Motor, SOLO supports switching frequencies from 8 to 80kHz and the user can set their desired frequency with steps of 1kHz from 8 to 80kHz once writing into this object, however once you Read the switching frequency using this object, the returned value will be in Hertz format.
As a rule of thumb, Higher switching frequencies are necessary for Motors with Low inductance (below 200uH), but the increase or decrease of this value should be done carefully, as increasing the switching frequency is not always good and it can cause saturation and excessive loss for both the magnetic cores of the Motor and the SOLO unit itself if the Motor under control can not support such high frequencies. The whole internal algorithms including samplings in SOLO are synchronized to the switching frequency.

300Ah : Speed Controller Kp Gain

Code: 0x300A	Speed Controller Kp gain				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 300.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the Speed controller Kp Gain, and it will be functional only in Digital Closed-loop mode, since in Analogue mode this gain is set using the Potentiometer named with “Kp” locally on the board.
This is the proportional gain of the PI controller that SOLO uses to control to stabilize the speed of a motor on a given reference point in close-loop mode, This gain is normally for most motors takes a value in between 0.001 to 0.5 but there might be some cases that you need to go even higher, the user has to increase/decrease these gains with care as they can cause instability for the device under test if not selected properly.

300Bh : Speed Controller Ki Gain

Code: 0x300B	Speed Controller Ki gain				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 300.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the Speed controller Ki gain, and it will be functional only in Digital Closed-loop mode, since in Analogue mode this gain is set using the Potentiometer named with “Ki” locally on the board.
This is the integral gain of the PI controller that SOLO uses to control to stabilize the speed of a motor on a given reference point, This gain is normally for most motors takes a value in between 0.0001 to 0.01 but there might be some cases that you need to go even higher, the user has to increase/decrease these gains with care as they can cause instability for the device under test if not selected properly. In theory, this gain helps the system to have zero steady-state error.

300Ch : Motor's Direction of Rotation

Code: 0x300C	Motor Direction of Rotation				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	0/1	N/A	R/W	V	0

Description:
This object Sets or Reads the direction of the rotation of the motor either in ClockWise rotation or Counter Clockwise Rotation based on the table below,(In sensorless Modes, the order of the wirings of the motor can invert the direction of rotation with respect to table below)

Data	Desired Rotational Direction
0 (0x00000000)	Counter ClockWise
1 (0x00000001)	ClockWise

300Dh : Motor's Phase or Armature Resistance

Code: 0x300D	Motor's Phase or Armature Resistance				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0001 - 25.0]	Ohm	R/W	M	0

Description:
This object Sets or Reads the amount of the Phase or Armature resistance for 3-phase or DC Brushed motors respectively. This value is automatically identified by SOLO after Motor Identification but in any case it's possible for the user to alter the values as they like. After changing these values manually, for them to take effect a power recycle is required.

300Eh :Motor's Phase or Armature Inductance

Code: 0x300E	Motor's Phase or Armature Inductance				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0000001 - 0.5]	Henry	R/W	M	0

Description:
This object Sets or Reads the amount of the Phase or Armature Inductance for 3-phase or DC Brushed motors respectively. This value is automatically identified by SOLO after Motor Identification but in any case it's possible for the user to alter the values as they like. After changing these values manually, for them to take effect a power recycle is required.

300Fh : Motor’s Number of Poles

Code: 0x300F	Motor’s Number of Poles				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[1-254]	N/A	R/W	M	8

Description:
This object Sets or Reads the number of the Poles of a 3-phase motor commissioned with SOLO, in case of BLDC or PMSM motors, the Number of poles is equal to the number of magnets on the rotor of the Motor, for ACIM motors, the user has to refer to the technical Datasheet of their motor to find this parameter.

3010h : Incremental Encoder’s Lines

Code: 0x3010	Incremental Encoder’s Lines				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[1-200,000]	pre-quad	R/W	M	1000

Description:
This object Sets or Reads the pre-quad number of physical lines of an incremental encoder engraved on its disk, in another form it can be seen as the number of pulses generated on 1 line of the Encoder output once it is rotated for 1 exact round.

3011h : Speed Limit

Code: 0x3011	Speed Limit				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-30,000]	RPM	R/W	M	30,000

Description:
This object Sets or Reads the allowed speed during trajectory following in closed-loop position controlling mode, it can be used to change the speed of a position follower during motion, or it can be fixed on a desired value.

3013h : Feedback Control Mode

Code: 0x3013	Feedback Control Mode				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-3]	N/A	R/W	M	0

Description:
This command sets the type of the feedback control SOLO has to operate with based on table below both in Analogue or Digital Mode:

DATA	Operation
0 (0x00000000)	Operates in Sensor-less feedback Mode (HSO)
1 (0x00000001)	Operates in Incremental Encoder feedback Mode (calibration required)
2 (0x00000002)	Operates in HALL Sensors feedback Mode (calibration required)
3 (0x00000003)	Operates in Sensor-less Zero Speed Full Torque feedback Mode (ZSFT)

3014h : Reset Factory

Code: 0x3014	Reset Factory				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	1	N/A	R/W	V	0

Description:
This object resets SOLO to its factory setting to all the default parameters, after this command a power recycle is required so that the default values take effect, the Data sent within this object to reset the device should be “0x0000001”.

3015h : Motor Type

Code: 0x3015	Motor Type				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-3]	N/A	R/W	M	0

Description:
This object Sets or Reads the Motor type that is connected to SOLO in Digital Mode based on the following table if the requested Data is placed in a packet sent with the object 0x3015, the user has to note that in Analogue Mode the Motor Type is only selected by Piano Switch in SOLO UNO using PIN 1 and 2 and on SOLO MINI it's selected by M1 and M2 pins on the "I/O Port" (refer to respective SOLO user manual to know more)

Data	Motor Type
0 (0x00000000)	Selects DC brushed Motor in Digital Mode
1 (0x00000001)	Selects Normal BLDC-PMSM Motor in Digital Mode
2 (0x00000002)	Selects ACIM Motor in Digital Mode
3 (0x00000003)	Selects Ultra fast BLDC-PMSM Motor in Digital Mode

3016h : Control Mode Type

Code: 0x3016	Control Mode Type				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	[0-2]	N/A	R/W	M	1

Description:
This object Sets or Reads the Control Mode in terms of Torque, Speed or Position only in Digital Mode based on the following table, in Analogue Mode this functionality is selected by using Piano switch PIN 4 in SOLO UNO and using FN pin on SOLO MINI for only Torque and Speed controlling functionalities. (refer to respective SOLO user manual to know more)

Data	Control Mode Type
0 (0x00000000)	Operates in Speed Mode
1 (0x00000001)	Operates in Torque Mode
2 (0x00000002)	Operates in Position Mode

3017h : Current Controller Kp Gain (Torque controller)

Code: 0x3017	Current Controller Kp (Torque controller)				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Sfxt(32-17)	[0 - 16000.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the value for Current Controller Kp or proportional gain, which will be used in the PI controller that controls the current (torque) inside of the motor both in Analogue or Digital Mode, this value is automatically identified by SOLO after Motor Identification but in any case it's possible for the user to alter this value as they like. After changing this value manually, for them to take effect a power recycle is required.

3018h : Current Controller Ki Gain (Torque controller)

Code: 0x3018	Current Controller Ki Gain (Torque controller)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 16000.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the value for Current Controller Ki or integral gain, which will be used in the PI controller that controls the current (torque) inside of the motor both in Analogue or Digital Mode, this value is automatically identified by SOLO after Motor Identification but in any case it's possible for the user to alter this value as they like. After changing this value manually, for them to take effect a power recycle is required.

301Ah : Magnetizing/Field Weakening Current Reference (Id)

Code: 0x301A	Magnetizing Current Reference (Id)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 -Device Current Limit]	Amps	R/W	V	0

Description:
This object Sets or Reads the Id or Direct current reference depending on the Motor type:

1. **In case of BLDC or PMSM motors:**
   Sets the Field Weakening current reference to help the motor reach speeds higher than nominal values, by increasing this current the maximum deliverable torque by the motor will reduce, thus the user has to find the best maximum value for the given motor at a given condition.
2. **In case of AC Induction Motors:**
   Sets the desired magnetizing current (Id) required for controlling ACIM motors in FOC in Amps, it can be used only in Closed-loop digital mode, since in Analogue mode the Magnetizing current reference is set through Pin "P/F" (refer to SOLO user manual to know more)

301Bh : Position Reference

Code: 0x301B	Position Reference				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Int32	[-2,147,483,647 to 2,147,483,647]	Quad-Pulse	R/W	V	0

Description:
This command sets the desired Position reference in terms of quadrature pulses while SOLO operates with the Incremental Encoders or in terms of pulses while while SOLO operates with Hall sensors, this functionality will be available only once SOLO is in Close-loop Digital Position mode both for Incremental encoders as well as Hall sensors.
Once using St-curve profiles (time-based or time-optimal), the maximum allowed relative position (the absolute distance between current position and the target position) can not be more than 60 mechanical turns in each iteration, for instance for an encoder with 8000 pulses per turn, the maximum relative distance shall be less than or equal to +/-480,000 pulses.
The relative distance will not limit the user to reach to higher values, however they need to allow the controller to traverse between two points first and then set the new target to go further than the 60 mechanical turns limit.
In case of using Step response, there will be no limit between the relative positions between the current position and the target position.

301Ch : Position Controller Kp Gain

Code: 0x301C	Position Controller Kp Gain				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 16000.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the value for Position Controller Kp or proportional gain, which will be used in the PI controller that controls the position.

301Dh : Position Controller Ki Gain

Code: 0x301D	Position Controller Ki Gain				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 16000.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the value for Position Controller Ki or integrator gain, which will be used in the PI controller that controls the position.

301Fh : Reset Position to Zero (Home)

Code: 0x301F	Reset Position to Zero (Home)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	1	N/A	W	V	0

Description:
This object Sets the position counter back to zero if in the Data part of the packet the value of “0x00000001” is placed.

3020h : Device Error Register

Code: 0x3020	Device Error Register				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R/W	V	0

Description:
This object Reads or Overwrites the reported errors in Device Error Register. The purpose of error overwriting is to allow the user to see the problems transparently and decide what to do, but SOLO will always keep track of catastrophic errors and it will not ignore them. The following table is the bit arrangement of the error register, if any of the mentioned errors occurs, you will receive "1" in the position of the bit corresponding to the error in a 32 bits register shown below, so if the value of the error register is anything other than Zero, it means there is a problem somewhere,which can be found exactly by checking the following bits. By putting zero in the place of each bit and sending a command with an object "0x3020" the error bits can be overwritten, similarly by sending a Data part with only zeros inside, the whole error register will be overwritten. Once the error is over-written if its cause of existence is removed, SOLO will return back to normal operation, otherwise it will stay in fault mode without any switching at the output.

Bit Number	Error Description
0	Over-current error, while the current in the motor rises above 62A for more than 0.1ms
1	Over-voltage on BUS error, when the BUS voltage rises above 57V for more than 35us
2	Over Temperature Error, when the board temperature rises above 85 degrees
3	Encoder Calibration timeout - Index pulse is missing
4	Hall Sensors Calibration timeout
5	CAN Communication Lost (lifeTime expired)
6	Stall Time out
7	N/A

3021h :Sensorless Zero Speed Full Torque Injection Amplitude

Code: 0x3021	Sensorless Zero Speed Full Torque Injection Amplitude				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0 - 0.55]	N/A	R/W	M	0.15
Description: Once in Zero Speed Full Torque algorithm (ZSFT) for controlling the speed of a BLDC or PMSM in sensorless fashion, this parameter defines the strength of signal injection into the motor, the user has to make sure this value is not selected too high or too low, to learn more about this please check our website.					

3022h : Sensorless Zero Speed Full Torque Polarity Amplitude

Code: 0x3022	Sensorless Zero Speed Full Torque Polarity Amplitude				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0 - 0.55]	N/A	R/W	M	0.25
Description: Once in Zero Speed Full Torque algorithm (ZSFT) for controlling the speed of a BLDC or PMSM in sensorless fashion, this parameter defines the strength of signal injection into the motor to identify the polarity of the Motor at the startup, the user has to make sure this value is not selected too high or too low, to learn more about this please check our website.					

3023h : Sensorless Observer Gain for DC Brushed Motors

Code: 0x3023	Sensorless Observer Gain for DC Brushed Motors				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.01 - 1000]	N/A	R/W	M	50

Description:
This object Sets or Reads the observer gain for the Non-linear observer that estimates the speed of a DC brushed once the motor type is selected as DC brushed. This Observer Gain basically deals with the Motor Back EMF Estimation, and normally it has a value from 10 to 100 for regular operations. This observer gain will be used both in Analogue and Digital Mode for Sensor-less operations and the gain can be modified dynamically in run-time.

3024h : Sensorless Zero Speed Full Torque Injection Frequency

Code: 0x3024	Sensorless Zero Speed Full Torque Injection Frequency				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-10]	N/A	R/W	M	0

Description:
Once in Zero Speed Full Torque algorithm (ZSFT) for controlling the speed of a BLDC or PMSM in sensorless fashion, this parameter defines the frequency of signal injection into the Motor in runtime, by selecting zero the full injection frequency will be applied which allows to reach to higher speeds, however for some motors, it’s better to increase this value, the user has to make sure this value is not selected too high or too low, to learn more about this please check our website.

3025h : Sensorless Transition Speed

Code: 0x3025	Sensorless Transition Speed				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	[1 - 5000]	RPM	R/W	M	500

Description:
Once in Sensorless speed or torque controlling of a BLDC or PMSM motors, this parameter defines the speed in which the Low speed algorithm has to switch to high speed algorithm, the condition will be:

1. For **"HSO sensorless algorithm"** : This speed defines the point of transition between startup part which is partially closed-loop to full closed-loop control, for most motors in this case 20% of the nominal speed will be a great starting point, the user can reduce this speed further to find the minimum robust and repeatable transition speed.

2. For **"ZSFT sensorless algorithm"** : This speed defines the point of transition between zero speed algorithm (ZSFT) to high speed algorithm (HSO), for Motors with existing saliency, this transition speed is better to be defined as low as possible, based on our experimentations, a starting point of 5% of the Nominal speed is good, in general once using ZSFT, it's better to keep this value below 150-200 RPM.

3026h : UART Baud-Rate

Code: 0x3026	Set UART Baud-Rate				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	0/1	Bits/s	R/W	M	937500

Description:
This object Sets or Reads the baud-rate of the UART line, currently there are two baudrates supported by SOLO, and they can be selected by sending either 0 or 1 in Data section of the packet , if the value of the Baud-rate is changed, to make it effective a power recycle is necessary.

Data	Baud-Rate [bits/s]
0 (0x00000000)	937500 (compatible with 921600)
1 (0x00000001)	115200

3027h : Encoder or Hall Sensors Calibration Start/Stop

Code: 0x3027	Encoder or Hall Sensors Calibration Start/Stop				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-2]	N/A	W	V	0

Description:
This object starts or stops the process of sensor calibration based on the table below if the Data is sent as table below. The sensor calibration is done in full torque mode limited to Current Limit value. The Encoder or Hall sensor calibration is only necessary for 3-phase motors and in case of incremental encoders the presence of index pulse is mandatory to find the correct mechanical offset of the shaft of the motor with respect to the control unit.

Data	Action
0 (0x00000000)	Stop the calibration process
1 (0x00000001)	Start Incremental Encoder Calibration
2 (0x00000002)	Start Hall Sensors Calibration

3028h : Per-Unit Encoder or Hall sensor Counter Clockwise offset

Code: 0x3028	Per-Unit Encoder or Hall sensor Counter Clockwise offset				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-1.0]	N/A	R/W	M	0

Description:
This object starts or stops the per-unit offset identified after sensor calibration for Encoder or Hall sensors in C.C.W direction, the value must be between 0 to 1 and it gets automatically updated each time after sensor calibration, however the users can insert their desired value dynamically.

3029h : Per-Unit Encoder or Hall sensor Clockwise offset

Code: 0x3029	Per-Unit Encoder or Hall sensor Clockwise offset				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-1.0]	N/A	R/W	M	0

Description:
This object Sets or Reads the per-unit offset identified after sensor calibration for Encoder or Hall sensors in C.W direction, the value must be between 0 to 1 and it gets automatically updated each time after sensor calibration, however the users can insert their desired value dynamically.

302Ah : Speed Acceleration Value

Code: 0x302A	Speed Acceleration Value				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-1600.0]	Rev/S^2	R/W	M	0

Description:
This object Sets or Reads the acceleration value of the Speed for speed controller both in Analogue and Digital modes in Revolution per square seconds, this value only has effect once SOLO is in Speed control mode and it will be ignored once in Torque or Position Mode.

302Bh : Speed Deceleration Value

Code: 0x302B	Speed Deceleration Value				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Sfxt(32-17)	[0.0-1600.0]	Rev/S^2	R/W	M	0

Description:
This object Sets or Reads the deceleration value of the Speed for speed controller both in Analogue and Digital modes in Revolution per square seconds, this value only has effect once SOLO is in Speed control mode and it will be ignored once in Torque or Position Mode. Defining deceleration can reduce the regenerative power going back to supply as the speed reduces slowly. The deceleration will be by passed (fast stop) if the user sends the following speed references based on following table:

Mode	Speed Reference for fast stop with regeneration
Digital Control	0
Analogue Control PWM input	Duty cycle of 0.5% or less
Analogue Control Voltage input	Voltage of 5mV or less

302Ch : CAN Bus Baud-rate

Code: 0x302C	CAN Bus Baud-rate				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-4]	kbits/s	R/W	M	1000

Description:
This object Sets or Reads the Baud-rate of CAN bus in CANopen network based on the following table, after changing the Baud-rate a power cycle is required for SOLO so the new Baud-rate can take effect.

Data	CAN bus Baud-rate [kbits/s]
0 (0x00000000)	1000
1 (0x00000001)	500
2 (0x00000002)	250
3 (0x00000003)	125
4 (0x00000004)	100

302Dh : Phase-A voltage

Code: 0x302D	Phase-A voltage				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-60 - 60]	Volts	R	V	0

Description:
This object reads the phase-A voltage of the motor connected to the “A” pin output of SOLO for 3-phase Motors.

302Eh : Phase-B voltage

Code: 0x302E	Phase-B voltage				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-60 - 60]	Volts	R	V	0

Description:
This object reads the phase-B voltage of the motor connected to the “B” pin output of SOLO for 3-phase Motors.

302Fh : Phase-A Current

Code: 0x302F	Phase-A Current				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-32 - 32]	Amps	R	V	0

Description:
This object reads the phase-A current of the motor connected to the “A” pin output of SOLO for 3-phase Motors.

3030h : Phase-B Current

Code: 0x3030	Phase-B Current				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-32 - 32]	Amps	R	V	0

Description:
This object reads the phase-B current of the motor connected to the “B” pin output of SOLO for 3-phase Motors.

3031h : BUS Voltage (Input Supply / Battery)

Code: 0x3031	BUS Voltage (Input Supply / Battery)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0 - 60]	Volts	R	V	0
Description: This object reads the input BUS voltage.					

3032h : DC Motor Current (IM)

Code: 0x3032	DC Motor Current (IM)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-32 - 32]	Amps	R	V	0
Description: This object reads the current inside the DC brushed motor connected to “B” and “C” outputs of SOLO.					

3033h : DC Motor Voltage (VM)

Code: 0x3033	DC Motor Voltage (VM)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-60 - 60]	Volts	R	V	0
Description: This object reads the voltage of the DC brushed motor connected to “B” and “C” outputs of SOLO.					

3034h : Quadrature Current (Iq)

Code: 0x3034	Quadrature Current (Iq)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-64.0 - 64.0]	Amps	R	V	0

Description:
This object reads the actual monetary value of "Iq" that is the current acts in torque generation in FOC mode for 3-phase motors, the amount of Torque on the shaft of the motor can be calculated using the following formula:
Actual Torque [N.m] = Iq [A] * Motor's Torque constant [N.m/A]

3035h : Direct Current / Magnetizing Current (Id)

Code: 0x3035	Direct Current / Magnetizing Current (Id)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-64.0 - 64.0]	Amps	R	V	0

Description:
This object reads the actual monetary value of Id that is the direct current acting in FOC, this current for ACIM motors is known as Magnetizing current as well.

3036h : Speed Feedback

Code: 0x3036	Speed Feedback				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Int32	[-30,000 - 30,000]	RPM	R	V	0

Description:
This object reads the actual speed of the motor measured or estimated by SOLO in sensorless or sensor-based modes respectively.

3037h : Position Feedback (Incremental Encoder and Hall sensors)

Code: 0x3037	Position Feedback (Incremental Encoder and Hall sensors)				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Int32	[-2,147,483,647 to 2,147,483,647]	Quad-Pulses	R	V	0

Description:
This command reads the number of counted pulses from the Incremental Encoder or Hall sensors. The counted pulses for Incremental Encoders will be in Quadrature format, however the counted pulses for the Hall sensors will be based on the following :
Hall sensor counted pulses in 1 revolution = Motor's number of poles x 3

3038h : Motor's Angle

Code: 0x3038	Motor's Angle				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-2 - 2]	Per Unit	R	V	10

Description:
This object reads the measured or estimated per-unit angle of the 3-phase motors.

3039h : Board Temperature

Code: 0x3039	Board Temperature				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[-30.0 - 150.0]	°C	R	V	0

Description:
This object reads the momentary temperature of the board in centigrade.

303Ah : Firmware Version

Code: 0x303A	Firmware Version				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R	M	N/A

Description:
This object reads the Firmware version existing currently on the SOLO unit, the Data section of the received command will contain a Number like “0x0000B009” which will indicate the firmware version.

303Bh : Hardware Version

Code: 0x303B	Hardware Version				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	N/A	N/A	R	M	N/A

Description:
This object reads the Hardware version of the SOLO unit connected.

303Ch: Analogue Speed Resolution Division Coefficient (ASRDC)

Code: 303C	Analogue Speed Resolution Division Coefficient (ASRDC)				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	[0.0001-10000]	NA	W/R	M	1

Description:
This object Sets or Reads the resolution of the speed at S/T input while SOLO operates in Analogue mode based on the following formula:
The max Speed on S/T by applying 5V = Max default speed defined by the motor type / ASRDC
The default Max speed for each motor type can be found in the following table:

Motor Type	Closed-loop Sensor-less	Closed-loop Sensor-based
BLDC - PMSM	8000 [RPM]	8000 [RPM]
BLDC - PMSM Ultrafast	30,000 RPM	30,000 [RPM]
DC Brushed	30,000 [N/A]**	8000 [RPM]
AC Induction Motor	4000 [RPM]	4000 [RPM]

** This is a qualitative value based on the observer gain defined for the sensorless speed estimator for DC brushed motors, the value can be ranged from 0 to 30,000 and the final speed of the motor for each value depends on the characteristics of the Motor itself like BEMF constant.

303Dh: Incremental Encoder Index Counts

Code: 303D	Incremental Encoder Index Counts				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Uint32	[0 - 2,147,483,647]	Pulses	W/R	V	0

Description:
This object reads the number of counted index pulses seen on the Incremental Encoder's output, the Index pulse should occur always at a fixed and certain mechanical position with respect to the stator of the Motor. While the Motor is turning the assumption is the index pulse occurs only once per revolution.

303Eh: Stall Detection Time-out

Code: 303E	Stall Detection Time-out				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-100000]	Milli seconds	W/R	M	5000

Description:
This object reads or writes the maximum allowed time interval for the controller to push current into the motor before sensing any motion, in other words, if the controller doesn't sense a change in the received position feedback from the motor within this period it will go into Stall Error condition.
Stall detection will only work in Sensor-based Speed and Position controlling.
Stall Error will cause the drive to stop the current flow into the motor, by deactivating the switching.
By putting Zero into the "Stall Detection Time-out" register, the Stall detection functionality will be deactivated.

303Fh: Motion Profile Mode

Code: 303F	Motion Profile Mode				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-2]	Uint32	W/R	M	0

Description:
This object reads or writes the type of the Motion Profile that is being used in Speed or Position Modes, SOLO currently offers the following 3 types as below:

1. **Step or RAMP response:**
   In this Mode the set point will be treated as a step response, thus the controller will react immediately to the reference, however if the Acceleration or Deceleration parameters are set in Speed mode, the controller will treat the reference as a RAMP input with the slope defined by the user.
2. **Time-based St-Curve:**
   In this mode the controller depending on the control type (Speed or Position) will create a smooth st-curve profile strictly limited to the timings defined in user inputs known as Motion Profile Variables.
3. **Time-optimal St-Curve:**
   In this mode the controller depending on the control type (Speed or Position) will create a smooth st-curve profile strictly limited to the maximum parameters like max jerks, max accelerations and max velocities defined in user inputs known as Motion Profile Variables.

DATA	Motion Profile Type
0x00000000	Step or RAMP response
0x00000001	Time-based St-Curve
0x00000002	Time-optimal St-Curve

3040h: Motion Profile Variable1

Code: 3040	Motion Profile Variable1				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Sfxt(32-17)	[0.0-16000.0]	N/A	W/R	M	0

Description:
This object reads or writes on the Motion Profile Variable 1, depending on the Motion Profile Mode and the Control Type this parameter will express one of the following meanings:

Motion Profile Mode	Control Type	Definition
St-curve Time-based	Position mode	Segments 1 and 3 Timings [seconds]
St-curve Time-optimal	Position mode	Maximum Velocity Constraint[RPM]
St-curve Time-based	Speed mode	Segment 1 and 3 Timings [seconds]
St-curve Time-optimal	Speed mode	Maximum Takeoff and landing Acceleration Constraint[RPM/s]

3041h: Motion Profile Variable2

Code: 3041	Motion Profile Variable2				
Data Type	**Data Range**	**Units**	**Accessibility**	**Memory Storage**	**Default Value**
Sfxt(32-17)	[0.0-16000.0]	N/A	W/R	M	0

Description:
This object reads or writes on the Motion Profile Variable 2, depending on the Motion Profile Mode and the Control Type this parameter will express one of the following meanings:

Motion Profile Mode	Control Type	Definition
St-curve Time-based	Position mode	Segment 2 Timing [seconds]
St-curve Time-optimal	Position mode	Maximum Takeoff Acceleration Constraint[RPM/s]
St-curve Time-based	Speed mode	Segment 2 Timing [seconds]
St-curve Time-optimal	Speed mode	Maximum Jerk Constraint[RPM/s^2]

3042h: Motion Profile Variable3

Code: 3042	Motion Profile Variable3				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-16000.0]	N/A	W/R	M	0

Description:
This object reads or writes on the Motion Profile Variable 3, depending on the Motion Profile Mode and the Control Type this parameter will express one of the following meanings:

Motion Profile Mode	Control Type	Definition
St-curve Time-based	Position mode	Segment 4 Timing [seconds]
St-curve Time-optimal	Position mode	Maximum landing Acceleration Constraint[RPM/s]
St-curve Time-based	Speed mode	N/A
St-curve Time-optimal	Speed mode	N/A

3043h: Motion Profile Variable4

Code: 3043	Motion Profile Variable4				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-16000.0]	N/A	W/R	M	0

Description:
This object reads or writes on the Motion Profile Variable 4, depending on the Motion Profile Mode and the Control Type this parameter will express one of the following meanings:

Motion Profile Mode	Control Type	Definition
St-curve Time-based	Position mode	Segments 5 and 7 Timings [seconds]
St-curve Time-optimal	Position mode	Maximum take-off Jerk Constraint[RPM/s^2]
St-curve Time-based	Speed mode	N/A
St-curve Time-optimal	Speed mode	N/A

3044h: Motion Profile Variable5

Code: 3044	Motion Profile Variable5				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-16000.0]	N/A	W/R	M	0

Description:
This object reads or writes on the Motion Profile Variable 5, depending on the Motion Profile Mode and the Control Type this parameter will express one of the following meanings:

Motion Profile Mode	Control Type	Definition
St-curve Time-based	Position mode	Segment 6 Timing [seconds]
St-curve Time-optimal	Position mode	Maximum landing Jerk Constraint[RPM/s^2]
St-curve Time-based	Speed mode	N/A
St-curve Time-optimal	Speed mode	N/A

3047h: PT1000 Sensor Voltage

Code: 3047	PT1000 Sensor Voltage				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0-3.3]	V	R	V	0.0

Description:
This command reads the value of the voltage sensed at the output of PT1000 temperature sensor amplifier, this command can be used only on devices that come with PT1000 input.

3048h: Digital Outputs Register

Code: 3048	Digital Outputs Register				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-(2^32-1)]	Uint32	R/W	V	0

Description:
This command sets or reads the state of the Digital Outputs Register as a 32 bits register, where each bit represent the state of each output as shown in the table below:

Bit Number	Digital Ouput
0	DO_0
1	DO_1
….	…
31	DO_32

3049h: Digital Inputs Register

Code: 3049	Digital Inputs Register				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-(2^32-1)]	Uint32	R	V	0

Description:
This command reads the value of the Digital Input Register as a 32 bits register, where each bit represent the state of each input as shown in the table below:

Bit Number	Digital Iuput
0	DI_0
1	DI_1
….	…
31	DI_32

304Ah: Analogue Input Read

Code: 304A	Analogue Input Read				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	[0-4095]	Uint32	R	V	0

Description:
This command reads the quantized value of an Analogue Input as a number between 0 to 4095 depending on the maximum voltage input possible at the analogue inputs for the controller, for instance to read the Channel 0 value which shows the sensed voltage at S/T input in the data section of the read command we have to place 0 (0x00000000) and to read Channel 1 value which shows the sensed voltage at P/F input in the data section we have to place 1 (0x00000001).

Input Number	DATA	Analogue Input
0	0x00000000	CH0 (S/T input)
1	0x00000001	CH1 (P/F input)

304Bh: Regeneration Current Limit

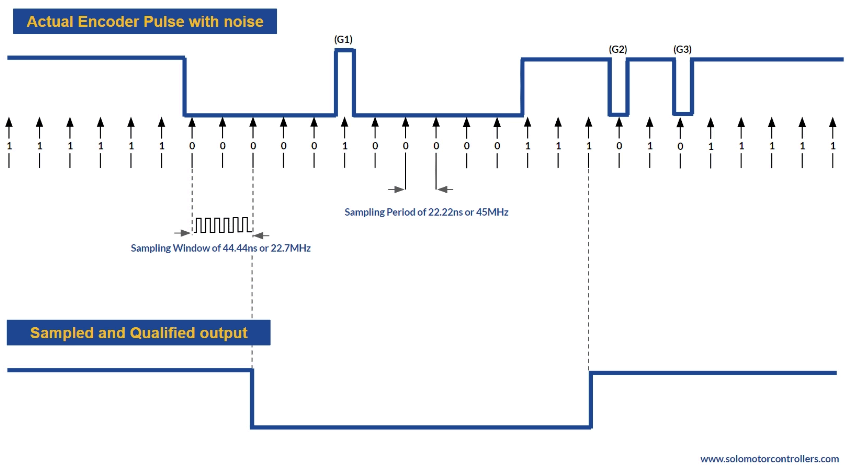
Code: 304B	Regeneration Current Limit				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Sfxt(32-17)	[0.0 - Device Current Limit]	Amps	R/W	M	0.0

Description:
This command reads or writes the maximum allowed regeneration current sent back from the Motor to the Power Supply during decelerations, in general once using uni-directional supplies like DC-DC converters only a small amount of current is allowed to go back to the supply to avoid Over-voltage issues (in some cases no reverse current is allowed), however in battery systems, this current can be used to charge the batteries.
The user has to make sure this value is defined in such a way that it can work for their system, and it worth noticing that reducing this value above a certain limit will reduce the dynamic performance of the Drive significantly, so usually it's good to allow at least up to 25% of the required current to drive the Motor (current limit) for this parameter.

304Ch: Position Sensor Digital Filter Level

Code: 304C	Position Sensor Digital Filter Level				
Data Type	Data Range	Units	Accessibility	Memory Storage	Default Value
Uint32	Sensor-dependant	Uint32	R/W	M	Sensor-dependant

Description:
This command reads or writes the the sampling window of qualification digital filter applied to the output of the position sensor before being processed by DSP, allowing to tune this parameter to achieve higher noise resilience or increasing the measurement frequency for high-speed applications, to learn how this filter works please check this article , in general here we set the sampling window shown in the image below, and if "n" number of samples are all having the same level, the qualified signal level will be passed to be processed at the DSP level, increasing this parameter will increase the filtering level and it can be used for heavily noisy encoders, however the user has to make sure they don't increase this value too much as it will negatively effect high-frequency measurements in case of using Encoders with High line counts .


Actual Encoder Pulse with noise
(G1)
(G2)
(G3)
1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1
Sampling Period of 22.22ns or 45MHz
Sampling Window of 44.44ns or 22.7MHz
Sampled and Qualified output
www.solomotorcontrollers.com

The range and default value of this digital filter depends on the type of the selected feedback based on the table below:

Feedback Type	Range	Default
Incremental Encoder	[1-255]	6
Hall sensor	[70-255]	100

Data TYPES:

The Data types in SOLO are categorized into three main types as shown in table below:

Name	Value	Size[bits]	Range	Resolution
UINT32	Unsigned Integer	32	[0 to 4,294,967,295]	+/- 1
INT32	Signed Integer	32	[-2,147,483,647 to 2,147,483,647]	+/- 1
Sfxt(32-17)	Fixed Point	32	[-16,384.000 to 16,384.000]	+/- 0.00000762

UINT32:

This Data type is used for Unsigned Integer values and it occupies 32 bits.

INT32:

This Data type is used for Signed Integer values and it occupies 32 bits.

Sfxt(32-17):

This Data type is used to represent the variables with floating point and it occupies 32 bits.

To send and receive commands or feedback properly during communication with SOLO, you need to convert your Data into one of these forms based on the command or feedback that you are using, as each command has a specific Data type.

In all of these formats, the Data section occupies 32 bits or 4 bytes, below you can see how one can convert these Data types to tangible numbers from Hexadecimal format that is the default way of sending or receiving Data with SOLO, in this manual the Hexadecimal numbers are either shown with “0x” in the beginning of the number or “h” at the end of the number.

DATA Types Conversions:

Converting Sfxt(32-17) data type to floating point data type:

If the DATA section of a packet received from SOLO, contains a number in Fixed-Point format, you can use the following two methods to convert it back to a floating point data type depending on the sign of the received data and whether if it's a positive value or a negative value:

Condition1) If the data read from SOLO is less than or equal to 0x7FFE0000 (Hex) or 2,147,352,576 (decimal), This means the data is positive, so follow the following steps

1) Convert the hex data read from SOLO into Decimal format
2) The float number = data read from SOLO (in Decimal format) / 131072

Condition2) If the data read from SOLO is greater than 0x7FFE0000 (Hex) or 2147352576 (decimal), This means the data is negative, so the conversion will be as :

1) Subtract the data from 0xFFFFFFFF and then add a 0x1 to it (add 1 to it)
2) Convert the result of step "1" into Decimal format
3) The float number = (data in Decimal format taken from step "2" /131072) * -1

Example1 _ positive Numbers:
Data read from SOLO is "0x00030000"
So it's a Positive Numbers Conversion since the data read from SOLO is less than or equal to 0x7FFE0000 so the conversion will be:
1) Decimal (0x00030000) = 196608
2) 196608 / (131072) = 1.5
Example2 _ negative Numbers:
Data read from SOLO is "0xFFFCCDD2"
So it's a Negative Numbers Conversion since the data read from SOLO is greater than 0x7FFE0000, so the conversion will be:

1) (0xFFFFFFFF – 0xFFFCCDD2) + 0x1 = 0x0003322E
2) Decimal (0x0003322E) = 209454
3) (209454 / (131072))* -1 = -1.598

Converting float data type to Sfxt(32-17) data type :

If the DATA section of a packet to be sent to SOLO, contains a number in Fixed-Point format, you can use the following two methods to convert your real world float number into a Sfxt(32-17) data type depending on the sign of the float data and whether if it’s a positive value or a negative value:

Condition1) If the float number is Positive:

1- Multiply the float number into 131072
2- Round down the result into the nearest integer value
3- convert this value to HEX

Condition2) If the float number is Negative:

1- Multiply the float number into 131072
2- Round down the result into the nearest integer value
3- Ignore the sign of the integer and convert this value to HEX (compute the Absolute value)
4- Subtract data from "0xFFFFFFFF"

Example 1_ positive float number:
Data to be sent : 4.2
Conversion:

1) 4.2 * 2^17 = 550,502.4
2) Round(550502.4) = 550502
3) Hex (550502) = 0x0086666

Example 2_ negative float number:
Data to be sent : -14.36
Conversion:

1) -14.36 * 2^17 = -1,882,193.92
2) Round(-1,882,193.92) = -1,882,193
3) Hex (abs (-1,882,193))= 0x 001CB851
4) 0xFFFFFFFF - 0x001CB851 = 0xFFE3 47AE

Converting 32 bits Hex data to signed INT32 format:

If you want to convert a DATA part of a packet read from SOLO into the real world Int32 format, based on the sign of the DATA you will fact the following two conditions for the conversion:

Condition1) If the data read from SOLO is less than or equal to 0x7FFFFFFF(Hex) or 2147483647 (decimal), This means the data is positive

When the data is positive, we can treat it like a normal unsigned value, so you can just directly convert the Hex value to decimal with known methods.

Condition2) If the data read from SOLO is greater than 0x7FFFFFFF(Hex) or 2147483647(decimal), This means the data is negative, so the conversion will be as :

1) Subtract the data from 0xFFFFFFFF and then add a 0x1 to it (add 1 to it)
2) Convert the result of step "1" into Decimal format
3) Multiply the result of step 3 to "-1"

Example 1_ positive Numbers:

Data read from SOLO is "0x0003F393"

- The data is smaller than 0x7FFFFFFF so it becomes : Dec (0x0003F393) = +258963

Example 2_ negative Numbers:

Data read from SOLO is "0xFFFCA8AD"

- The data is bigger than 0x7FFFFFFF so we will have:

1. (0xFFFFFFFF - 0xFFFCA8AD) + 1 = 0x00035753
2. Dec(0x00035753) = 218963
3. 218963 * -1 = -218963

Converting signed INT32 to 32 bits Hex format:

If you want to convert an INT32 value to a Hex formatted number to place in the DATA part of a packet to send to SOLO, based on the sign of the data you want to send, you will face the following two conditions for the conversion:

Condition1) If the data is positive:

If the number you want to send is positive, the only thing you need to do is converting the integer into HEX like an unsigned value

Condition1) If the data is Negative:

For sending negative Integers with Hex format to SOLO, you need to follow the following steps:

1. Subtract the absolute value of your number from 4294967295(Decimal) or 0xFFFFFFFF(Hex)
2. Add 1 to the result of step 1
3. Convert the result of 2nd step to Hex

Example 1_ positive Numbers:

Data to be sent: 1536

- Hex (1536) = 0x00000600

Example 2_ negative Numbers:

Data to be sent: -56329

- Hex (4294967295 - Abs (-56329) + 1) = 0xFFFF23F7

CANopen Data communication examples

The following is a set of examples showing how one can set or read some of the objects on SOLO in CANopen network, in all of these examples:

1. The Node or Device address is considered as "1"
2. All of the numbers in the table are having Hexadecimal format
3. The Remote messages will have an "r" extension in front of their COB-ID
4. Inside the "Message and Data" section the order of the bytes from left to right is Byte 0 to Byte8.

Set the Guard Time:

In this example we want to set the Guard time of the CANopen on SOLO at 3000ms, so SOLO within this guard time should report the NMT state once asked by a remote packet from the server as below:

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 0C 10 00 B8 0B 00 00	Set the Guard time at 3000 ms by server
581	8	60 0C 10 00 00 00 00 00	Node 1 responds back
701r	0	RTR set	Server Sends a remote Guard Request
701	1	FF	Node 1 responds indicating pre-operational mode within 3 seconds
701r	0	RTR set	Server Sends a remote Guard Request
701	1	7F	Node 1 responds indicating pre-operational mode within 3 seconds
701r	0	RTR set	Server Sends a remote Guard Request
701	1	FF	Node 1 responds indicating pre-operational mode within 3 seconds

Set the Heartbeat production:

This example wants to set the SOLO unit to produce a heartbeat with intervals of 1000ms in between.

COB-ID	Number of Bytes	Message and Data	Description	Time stamp[ms]
601	8	22 17 10 00 E8 03 00 00	Server requests the 1s heartbeat	854.73
581	8	60 17 10 00 00 00 00 00	Node 1 responds back	854.73
701	1	7F	Node 1 indicates in pre-operational state	855.72
701	1	7F	Node 1 indicates in pre-operational state	856.71
701	1	7F	Node 1 indicates in pre-operational state	857.70

Control the Speed of a PMSM using Encoders:

In this example we want to do a close-loop speed control on a PMSM with 1000 lines of encoder lines, the goal is to set the speed at 1000 RPM, before sending the commands you just need to make sure SOLO is in Closed-loop mode (PIN NO# 5 of Piano Switch on SOLO is Down or the Control Mode Switch on SOLO MINI is on C.L), so in this case we just need to tune the Speed controller Kp and Ki gains.
Another point to consider is in this example we have identified the motor parameters beforehand and the wirings and the calibration values for the Encoder and the Motor are found using this Tutorial.

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 15 30 00 01 00 00 00	Set Motor type to Normal BLDC-PMSM
581	8	60 15 30 00 00 00 00 00	Node 1 responds
601	8	22 16 30 00 00 00 00 00	Set the Control mode type on Speed Mode
581	8	60 16 30 00 00 00 00 00	Node 1 responds
601	8	22 0A 30 00 33 33 00 00	Set the Speed Controller Kp Gain to 0.01
581	8	60 0A 30 00 00 00 00 00	Node 1 responds
601	8	22 0B 30 00 68 00 00 00	Set the Speed Controller Ki Gain to 0.0008
581	8	60 0B 30 00 00 00 00 00	Node 1 responds
601	8	22 0F 30 00 08 00 00 00	Set the Number of Poles at 8
581	8	60 0F 30 00 00 00 00 00	Node 1 responds
601	8	22 13 30 00 01 00 00 00	Set Control Mode on Sensor-based with Incremental Encoders
581	8	60 13 30 00 00 00 00 00	Node 1 responds
601	8	22 10 30 00 E8 03 00 00	Set the Encoder Lines at 1000 lines
581	8	60 10 30 00 00 00 00 00	Node 1 responds
601	8	22 02 30 00 01 00 00 00	Put SOLO in Digital Command Mode
581	8	60 02 30 00 00 00 00 00	Node 1 responds
601	8	22 05 30 00 E8 03 00 00	Set the Speed reference at 1000 RPM
581	8	60 05 30 00 00 00 00 00	Node 1 responds

Control the Speed of a BLDC using HALL sensors:

In this example we want to do a close-loop speed control on a BLDC to set the speed at 1000 RPM, before sending the commands you just need to make sure SOLO is in Closed-loop mode (PIN NO# 5 of Piano Switch on SOLO is Down or the Control Mode Switch on SOLO MINI is on C.L), so in this case we just need to tune the Speed controller Kp and Ki gains. Another point to consider is in this example we have identified the motor parameters beforehand and the wirings and the calibration values for the Encoder and the Motor are found using this Tutorial.

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 15 30 00 01 00 00 00	Set Motor type to Normal BLDC-PMSM
581	8	60 15 30 00 00 00 00 00	Node 1 responds
601	8	22 16 30 00 00 00 00 00	Set the Control mode type on Speed Mode
581	8	60 16 30 00 00 00 00 00	Node 1 responds
601	8	22 0A 30 00 00 00 01 00	Set the Speed Controller Kp Gain to 0.5
581	8	60 0A 30 00 00 00 00 00	Node 1 responds
601	8	22 0B 30 00 68 00 00 00	Set the Speed Controller Ki Gain to 0.0008
581	8	60 0B 30 00 00 00 00 00	Node 1 responds
601	8	22 0F 30 00 08 00 00 00	Set the Number of Poles at 8
581	8	60 0F 30 00 00 00 00 00	Node 1 responds
601	8	22 13 30 00 02 00 00 00	Set Control Mode on Sensor-based with Hall sensors
581	8	60 13 30 00 00 00 00 00	Node 1 responds
601	8	22 02 30 00 01 00 00 00	Put SOLO in Digital Command Mode
581	8	60 02 30 00 00 00 00 00	Node 1 responds
601	8	22 05 30 00 E8 03 00 00	Set the Speed reference at 1000 RPM
581	8	60 05 30 00 00 00 00 00	Node 1 responds

Control the Speed of a Brushless motor in Sensor-less mode:

In this example we want to do a sensorless close-loop speed control on a Brushless motor to set the speed at 6000 RPM, before sending the commands you just need to make sure SOLO is in Closed-loop mode (PIN NO# 5 of Piano Switch on SOLO is Down or the Control Mode Switch on SOLO MINI is on C.L), so in this case we just need to tune the Speed controller Kp and Ki gains. (We imagined the Motor Identification is done beforehand; to auto tune the current controller gains at least one time)

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 15 30 00 01 00 00 00	Set Motor type to Normal BLDC-PMSM
581	8	60 15 30 00 00 00 00 00	Node 1 responds
601	8	22 16 30 00 00 00 00 00	Set the Control mode type on Speed Mode
581	8	60 16 30 00 00 00 00 00	Node 1 responds
601	8	22 0A 30 00 33 33 00 00	Set the Speed Controller Kp Gain to 0.01
581	8	60 0A 30 00 00 00 00 00	Node 1 responds
601	8	22 0B 30 00 68 00 00 00	Set the Speed Controller Ki Gain to 0.0008
581	8	60 0B 30 00 00 00 00 00	Node 1 responds
601	8	22 0F 30 00 08 00 00 00	Set the Number of Poles at 8
581	8	60 0F 30 00 00 00 00 00	Node 1 responds
601	8	22 13 30 00 00 00 00 00	Set Control Mode on Sensor-less
581	8	60 13 30 00 00 00 00 00	Node 1 responds
601	8	22 02 30 00 01 00 00 00	Put SOLO in Digital Command Mode
581	8	60 02 30 00 00 00 00 00	Node 1 responds
601	8	22 05 30 00 70 17 00 00	Set the Speed reference at 6000 RPM
581	8	60 05 30 00 00 00 00 00	Node 1 responds

PDO Examples:

Below you can find some examples on how to Configure and use various TPDO or RPDOs.

RPDO Examples:

Setting the Position Reference to be applied Immediately on the Device

The RPDO object address assigned: 0x201
RTR: Enabled
Target Position value: 0x001A1A00

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 14 14 01 01 02 00 C0	Set the communications parameters, enabling RPDO with the RPDO address set at 0x201 on Node1.
581	8	60 14 14 01 00 00 00 00	ACK received from Node 1
601	8	22 14 14 02 FF 00 00 00	Set the Transmission type on Immediate update
581	8	60 14 14 02 00 00 00 00	ACK received from Node 1
201	4	00 1A 1A 00	Set the desired Position on RPDO with address of 0x201

Setting the Position Reference to be applied after the next SYNC on the Device

The RPDO object address assigned: 0x201
RTR: Enabled
Target Position value: 0x001A1A00

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 14 14 01 01 02 00 C0	Set the communications parameters, enabling RPDO with the RPDO address set at 0x201 on Node1.
581	8	60 14 14 01 00 00 00 00	ACK received from Node 1
601	8	22 14 14 02 00 00 00 00	Set the Transmission type on Next SYNC
581	8	60 14 14 02 00 00 00 00	ACK received from Node 1
201	4	00 1A 1A 00	Set the desired Position on RPDO with address of 0x201
80	0		Send 1 SYNC message and the RPDO will be applied immediately after.

TPDO Examples:

Getting the Position Feedback after 1 SYNC Message

The TPDO object address assigned: 0x280
RTR: Enabled

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 14 18 01 80 02 00 C0	Set the communications parameters, enabling TPDO with the TPDO address set at 0x280 on Node1.
581	8	60 14 18 01 00 00 00 00	ACK received from Node 1
601	8	22 14 18 02 00 00 00 00	Set the Transmission type on Next SYNC
581	8	60 14 18 02 00 00 00 00	ACK received from Node 1
80	0		Send 1 SYNC message across the CAN network.
280	4	60 D5 71 FE	TPDO with address of 0x280 answers back with Position Feedback.
80	0		Send 1 SYNC message across the CAN network.
280	4	F1 CA 71 FE	TPDO with address of 0x280 answers back with Position Feedback in terms of Pulses.

Getting the Position Feedback after 4 SYNCs Messages

The TPDO object address assigned: 0x280
RTR: Enabled

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 14 18 01 80 02 00 C0	Set the communications parameters, enabling TPDO with the TPDO address set at 0x280 on Node1.
581	8	60 14 18 01 00 00 00 00	ACK received from Node 1
601	8	22 14 18 02 04 00 00 00	Set the Transmission type to send the data after4 SYNC Messages
581	8	60 14 18 02 00 00 00 00	ACK received from Node 1
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
280	4	F1 CA 71 FE	TPDO with address of 0x280 answers back with Position Feedback in terms of Pulses.
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
80	0		Send 1 SYNC message across the CAN network.
280	4	60 D5 71 FE	TPDO with address of 0x280 answers back with Position Feedback in terms of Pulses.

Getting the Position Feedback after RTR Message

The TPDO object address assigned: 0x280
RTR: Enabled

COB-ID	Number of Bytes	Message and Data	Description
601	8	22 14 18 01 80 02 00 C0	Set the communications parameters, enabling TPDO with the TPDO address set at 0x280 on Node1.
581	8	60 14 18 01 00 00 00 00	ACK received from Node 1
601	8	22 14 18 02 00 00 00 00	Set the Transmission type on Next SYNC
581	8	60 14 18 02 00 00 00 00	ACK received from Node 1
280R	0		RTR request to TPDO with address of 0x280 is sent.
280	4	F1 CA 71 FE	TPDO with address of 0x280 answers back with Position Feedback in terms of Pulses.