CAN Open protocol specification and functions

Functions and settings description

Version 1.7

01/10/2024

Modification History:

Revision	Issue Date	Author	Changes
1.0	15/11/2021	SM	Concept
1.1	13/04/2022	SM	Update parameter names, guidelines for external line contactor management
1.2	20/10/2022	SM	Update External Line contactor schematic
1.3	06/02/2023	SM	Updated the error codes
1.4	08/03/2023	SM	Update references, update control type description (had wrong numbering). Fix parameter number for active H/L config at page 34. PDO preamp menu locked in case of free mapping enabled
1.5	09/11/2023	SM	Fix Control Via CAN activation condition in case of free mapping. Fix pump setpoint mode requiring the "Switch/no Switch" option to 0. It does not.
1.6	16/01/2024	SM	Update CAN safety menu and safety functions of controller available from HW V07.03 (Sigma2N) and V05.02 (SigmaLITE). Add SigmaLITE compatibility. Add explanation of SDO content with examples. Add section explaining how to reset faults Via CAN.
1.7	01/10/2024	SM	Add details about ramp times used in speed and torque setpoint mode

Table of Contents

1 INTRODUCTION 1

1.1 REFERENCES 1
1.2 ACRONYMS 1
1.3 ABSTRACT 1

2 CAN OPEN PROTOCOL DESCRIPTION 2

2.1 DMC CONTROLLERS NODE NUMBER 2
2.2 NODE BOOT SEQUENCE, NETWORK MANAGEMENT AND HEARTBEAT MESSAGE 2

2.2.1 Boot up 2
2.2.2 Network management 3
2.2.3 Heartbeat message 3
2.2.4 Reset application command 3

2.3 PDO COMMUNICATION 4
2.4 SAFETY IMPLEMENTED IN RX AND TX PDOS 4

2.4.1 Timeout function 4
2.4.2 Roll-over counter function 4

2.5 SDO COMMUNICATION 5

2.5.1 SDO read service 5
2.5.2 SDO write service 6
2.5.3 SDO abort transfer service 6

2.6 CLEARING AN ACTIVE FAULT VIA CAN 8

2.6.1 Fault reset trough SDO 8
2.6.2 Fault reset trough ControlWord 9

3 SETTING UP THE CONTROLLER 10

MENU 7 “CAN BUS SETUP” 10

Menu 7.1 “CAN general” for Sigma2N controller 10
Menu 7.1 “CAN general” for SigmaLITE controller 13
Menu 7.2 “PDO premapped setup” 14
Menu 7.3 “Shared line contactor” 18
Menu 7.4 “Display” 19
Menu 7.5 “Safe Stop 1” / “CAN Safety” 21

4 FREE MAPPING OF DMC CONTROLLER 25

4.1 HOW TO FREE-MAP A DMC CONTROLLER 28

4.1.1 RxPDO Communication objects (Index 0x1400 to 0x1407) 28
4.1.2 TxPDO Communication objects (Index 0x1800 to 0x1807) 29
4.1.3 RxPDO Mapping objects (Index 0x1600 to 0x1607) 30
4.1.4 TxPDO Mapping objects (Index 0x1A00 to 0x1A07) 32

5 CONTROL VIA CAN FUNCTION 33

5.1 HOW TO ENABLE THE CONTROL VIA CAN FUNCTION 33
5.2 SAFETY FUNCTIONS 34

5.2.1 Safe Torque Off (STO) 34
5.2.2 Safe Stop 1 (SS1) 35

5.3 GENERAL FUNCTION DESCRIPTION 38

5.3.1 "Setpoint" and "Remote Input" CAN control types 38
5.3.2 "Mixed inputs" control type 39
5.3.3 Boot up and Network management 39
5.3.4 Pre-mapped Receive (Rx) PDOs 41

5.4 THIRD PARTY VCU DEVELOPMENT 42

5.4.1 General suggestions 42
5.4.2 General remarks 42
5.4.3 Loop cycle implementation (for Pre-mapped operation mode) 43
5.4.4 Power up and communication sequence example (for Pre-mapped operation mode) 44

5.5 SLAVE CONTROLLER(S) INFORMATION 45

5.5.1 Slave controller(s) features (for Pre-mapped operation mode) 45
5.5.2 Slave controller(s) behaviour (for Pre-mapped operation mode) 45

5.6 DIFFERENCES BETWEEN TRACTION AND PUMP APPLICATION SOFTWARE 46

5.6.1 Notes on "HMI digital input word" or "Preset byte #1" for Traction 46
5.6.2 Notes on "HMI digital input word" and "Preset byte #1" for Pump 47
5.6.3 Notes on RPM setpoint and speed limit signals for Traction 48
5.6.4 Notes on RPM setpoint and speed limit signals for Pump 48

5.7 DEVICE MANAGEMENT WITH CONTROL WORD (OBJECT 0X6040) 49

5.7.1 Transitions description 51
5.7.2 Abort connection option code 51
5.7.3 Quick stop option code 51
5.7.4 Shutdown option code 51
5.7.5 Disable operation option code 51
5.7.6 Halt option code 51

5.8 DEVICE MONITORING WITH STATUSWORD (OBJECT 0X6041) 52
5.9 MISCELLANEOUS FAILURES - TROUBLESHOOTING 53

5.9.1 VCU fails to send Drive Command (RxPDO1) to slave controller 53
5.9.2 Slave controller fails to send Drive Status (TxPDO2) to VCU 53
5.9.3 VCU sends corrupted data to slave controller 53
5.9.4 Slave controller sends corrupted data to VCU 53
5.9.5 Hard fault occurs in slave controllers 53
5.9.6 Hard fault occurs in VCU 53

6 SHARED LINE CONTACTOR FUNCTION 54

6.1 EXAMPLES 54

6.1.1 How to set up for two controllers sharing line contactor 54
6.1.2 How to set up for 3 controllers sharing line contactor 55

6.2 INTERACTION BETWEEN SHARED LINE CONTACTOR AND CONTROLWORD 56
6.3 ERROR CODES RELATED TO SHARED LINE CONTACTOR 57

7 EXTERNAL LINE CONTACTOR MANAGEMENT 58

7.1 EXTERNAL LINE CONTACTOR POWER-ON SEQUENCE 59
7.2 SEPARATED CONTROL LOGIC SUPPLY MANAGEMENT 60
7.3 WIRING SCHEMATIC EXAMPLES FOR SIGMA2N 61

7.3.1 External line contactor without separated logic supply 61

7.3.2 External line contactor with separated logic supply 61

7.4 WIRING SCHEMATIC EXAMPLES FOR SIGMALITE 63

7.4.1 External line contactor without separated logic supply 63
7.4.2 External line contactor with separated logic supply 63

8 USE OF CAN OPEN CALIBRATOR AND DMC CONFIGURATOR 65
9 WIRING EXAMPLES SIGMA2N 66

9.1 SHARED LINE CONTACTOR WITH NON-ISOLATED CAN BUS 66
9.2 SHARED LINE CONTACTOR WITH ISOLATED CAN BUS 67
9.3 CONTROL VIA CAN WITH NON-ISOLATED CAN BUS 68
9.4 CONTROL VIA CAN WITH ISOLATED CAN BUS 69
9.5 CONTROL VIA CAN AND SHARED LINE CONTACTOR WITH NON-ISOLATED CAN BUS 70
9.6 CONTROL VIA CAN AND SHARED LINE CONTACTOR WITH ISOLATED CAN BUS 71

1 Introduction

1.1 References

[1]	Premapped Can Messages in CAN Open Protocol V1.3
[2]	Sigma2N Traction/Pump Advanced Manual V1.6
[3]	SigmaLITE Traction/Pump Advanced Manual V1.0
[4]	DMC Configurator Manual V1.1.4
[5]	CiA® 301 CANopen application layer and communication profile (Control in Automation standard)
[6]	CiA® 402 CANopen control device profile for Drives and motion control (Control in Automation standard)
[7]	DMC Object Dictionary Variables description V1.5
[8]	DMC Object dictionary V1.99
[9]	DMC Display manual V1.1
[10]	DMC BMS support V1.5
[11]	EN60204-1/2006 Safety of machinery

1.2 Acronyms

DMCC	DMC Controller
PDO	Process data object
SDO	Service Data object
Tx	Transmission from node
Rx	Received by the node
RPM	Revolutions Per Minutes
BCL	Battery Current Limit
SLC	Shared Line Contactor
BDI	Battery Discharge Indication
SOC	State Of Charge
COM	Can Open Master (i.e. VCU vehicle control unit)

1.3 Abstract

This document describes the CAN Open communication protocol implemented in the Sigma2N and SigmaLITE controllers, which is fully compliant with the DS301 and DS402 CAN Open standards.
The main feature of this product is its flexibility: it can in fact be used as a pre-mapped unit, with some feedback and monitoring capabilities already implemented and enabled as default, or it can be fully configured by the user to better match its application.
The Controller operates within 2 macro-areas: the “Communication” level and the “Motor-Application” one.
Both are hierarchically equivalent, but interact with each other for safety purposes.
The “Communication” area takes care of the so called “HMI”. It is based on the Standard DS301 Communication state machine (see “”) and is therefore responsible of the managing of input/output signal managing Via CAN.
The “Motor-Application” area takes care of running the function that the user want to operate. It is based on the Standard DS401 Application state machine (see “”) and is therefore responsible of dealing with the motor control.
As default, the Sigma2N and SigmaLITE controllers comes pre-configured with the CAN system deactivated. This means that the Controller could be used wired without taking care of managing the previously named state machines. A further step of configuration consists of enabling the CAN system in a pre-mapped situation, in which the PDO mapping is defined by DMC and many CAN functions are disabled for the sake of simplicity. Advanced used could then gain access to the full flexibility capability of the Controller: they can in fact map whatever object of the DMC Object Dictionary [7] in any PDO, changing the PDO configuration and many other advanced options that will be described in this document.

2 CAN Open protocol description

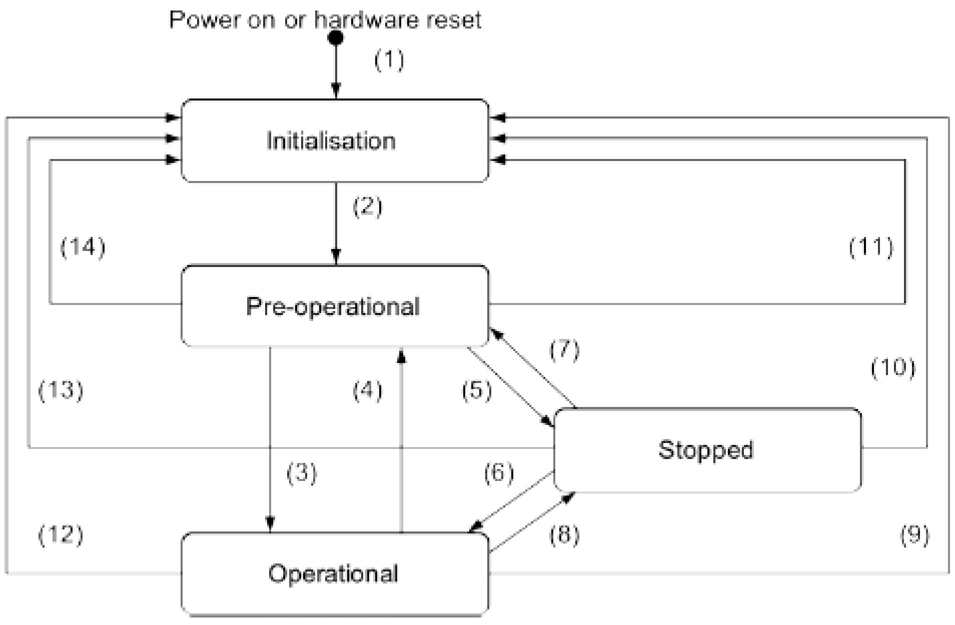
DMC Sigma2N and SigmaLITE controllers are fully compliant with CiA CAN Open communication protocol. It is therefore highly suggested to refer to the documentation issue by CiA [5] for details about this protocol.
In this manual only basic concepts about CAN Open for introducing the users to this protocol.

2.1 DMC Controllers node number

As the CAN Open standard specifies, in a CAN Open network there could be up to 128 nodes, numbered from 0 to 127. Node 0 is the so called “CAN Open Network Master (COM)” and is in charge of configuring and managing all the “slave” nodes. This device is usually a PLC/VCU, able to gather data from the field and issue commands to the actuators.
Nodes 1 to 127 are the so called “Slave nodes”. They are in charge of receiving commands from the master, actuating instructions and send back feedback about their status.
A DMC Sigma2N or SigmaLITE controller can only be assigned a node number from 1 to 127: it can in fact be used only as a slave node. Slight exception to this can be made when using the “Advanced Node Mapping” function, since that will enable the Sigma2N and SigmaLITE controllers to send or receive messages that should otherwise be reserved to COM only.

2.2 Node Boot Sequence, Network Management and Heartbeat message

The CiA DS301 CAN Open standard [5] specifies 4 “Communication” states that the CAN Open device could be fall into: “Initialization”, “Pre-operational”, “Operational” and “Stopped”.



The Sigma2N and SigmaLITE controllers implement transitions from 1 to 11 to allow the user to cycle between those states.

2.2.1 Boot up

If DMC Sigma2N or SigmaLITE controller is used as standalone (“Control Via CAN Open” function not used, “CAN pre-map” enabled), its status at boot-up will be “operational” and it will not be possible to change it. This means that transitions 1, 2 and 3 will be automatically performed after powering on the Controller and all other transitions are disabled.
If DMC Sigma2N or SigmaLITE controller is used with “Control Via CAN Open Compatible” function or the “CAN pre-map” is disabled, its status at Boot Up will be “pre-operational” and it will be up to the Can Open Master (COM), trough NMT messages (ID 0x000), to set the Sigma2N or SigmaLITE controller to “operational” or “stopped”.

2.2.2 Network management

If a DMC Sigma2N or SigmaLITE controller is used with “Control Via CAN Open Compatible” function or the “CAN pre-map” is disabled, the CAN Open Master (either a PLC/VCU) is in charge of managing the DMC Controller’s status trough NMT messages (ID 0x000).
This message has DLC = 2 and includes the following information:

- Byte 0: The command to actuate a transition.
- Byte 1: The target node.

The possible commands are:

- 0x01: Go to operational
- 0x80: Go to pre-operational
- 0x02: Go to stopped
- 0x81: Reset application

Notice that the target node has to be inserted in hexadecimal.
Notice that CAN Open also allows a “broadcast” NMT command. This allows to force all the nodes in the network to actuate the same command. To this means, the “target node” field should be set to 0x0.

The following table resumes the messages that the DMC Sigma2N or SigmaLITE controller can send/receive in each status.

	Stopped	**Pre-operational**	**Operational**
PDO (Tx and Rx)	×	×	✓
SDO (Tx and Rx)	×	✓	✓
SYNC	×	×	✓
EMCY	×	✓	✓
NMT	✓	✓	✓
Heartbeat	✓ (Tx only)	✓	✓

2.2.3 Heartbeat message

Unless parameter “M7.2-3 PDOs configuration “PDO sel”” is set to 4 and “CAN pre-map” is enabled (“M7.2-1PDO free mapping type “FreeMapT”” set to 0), see “3 Setting up the Controller” for details, DMC Sigma2N or SigmaLITE controller will always send an HEARTBEAT message (ID 0x7nn, with n the DMC Controller’s node number) for displaying the node status (operational, preoperational or stopped).

2.2.4 Reset application command

If an NMT message is sent with command to reset the application, the receiving node will boot up again and re-load the configuration. This operation is equivalent to cycle the key of the controller, but without removing power.
For safety purposes, the “reset application” command will be considered by the target node only when the controller is not energizing the motor.
When executing this command, the Controller will firstly open the line contactor.

2.3 PDO communication

When the DMC Controller is in “Operational” status, PDOs communication is enabled.
If “CAN pre-map” is enabled DMC Controller has the capability of receiving two pre-mapped RxPDOs:

- RxPDO1 is used when “Control Via CAN Open” function is enabled (see “5 Control via CAN function”) to give the DMC Controller the Drive Commands. Several pre-mapping of this RxPDO are available. The pre-mapping can be selected by means of parameter “M7.2-2 RxPDO premaped configuration “RxPDOmap””. Refer to [1] for details about the message content of different pre-maps.
- RxPDO2 is used when the “BCL Via CAN” function is enabled and contains the battery current limits and the BDI value.

If “CAN pre-map” is enabled DMC Controller has the capability of sending three pre-mapped TxPDOs. In the TxPDOs all status information are available. For details see [1].
If the DMC Controller is, from “Operational” status, set back to “Pre-operational”, PDOs communication is inhibited.

WARNING!

If “Control Via CAN Open” function is enabled and NMT request to go to “Pre-operational” status is sent to DMC Controller while motor is running, the motor is stopped (with neutral braking).
To start driving the motor when PDOs are first enabled in “Operational” status (i.e. when the Sigma2N or SigmaLITE controller is set from “Pre-operational” to “Operational” status), it is necessary to start from speed (or torque) setpoint equal to zero or in general from zero demand

2.4 Safety implemented in Rx and Tx PDOs

2.4.1 Timeout function

A time out for every RxPDOs in DMC Sigma2N or SigmaLITE controller is implemented.
In case of pre-mapped mode (see “M7.2-1 PDO free mapping type “FreeMapT””), the time out values for RxPDO1 and RxPDO2 are adjustable by means of “M7.2-9 RxPDO1 event timer “RxPDO1TO”” and “M7.2-10 RxPDO2 event timer “RxPDO2TO””.
In case of free-mapping mode, the timeout for each active RxPDO can be setup by means of DMC Configurator of SDO communication (see “4.1.1 RxPDO Communication objects (Index 0x1400 to 0x1407)” for details).
Timers are active only in “operational” status, starting from when the first RxPDO, respectively, is received. This means that the time out is not considered when:

- DMC Sigma2N or SigmaLITE controller is in “Operational” status but no RxPDO has been received yet;
- DMC Sigma2N or SigmaLITE controller is not in “Operational” status.

2.4.2 Roll-over counter function

A roll-over counter for every Tx and Rx PDOs in DMC Sigma2N or SigmaLITE controller is implemented:

- If message counter on RxPDO does not change or does not increase for 2 consecutive RxPDOs, the node goes in failure stare (F29) and motor is stopped.

A message counter is implemented in each TxPDOs:

- Each transmit TxPDO has its own independent message counter that is available for third party to check if TxPDO is updated.
  The TxPDO message counter is incremented every time a new TxPDO is sent by the DMC Sigma2N or SigmaLITE controller.

2.5 SDO communication

When the DMC Controller is in “Pre-Operational” or “Operational” status, SDOs communication is enabled
SDOs communication allows to read or write any parameter and to read any status variable included in the DMC Object dictionary [7].
Please notice that the CiA DS301 [5] defines standard format for SDO communication, that is in any case reported below:

- Byte 0: Command specifier
- Byte 1: Target object index LSB
- Byte 2: Target object index MSB
- Byte 3: Target object sub-index
- Byte 4 to 7: SDO content

The “command specifier” field embeds information about:

- the type of SDO service required (i.e. upload, download, block upload, block download, segmented upload, segmented download);
- the fact that the service is expedite or segmented;
- the fact that the data size is indicated or not.

The first byte of an SDO (be it Tx or Rx) is called “Command specifier” is co composed:

SDO (Tx or Rx) Byte 0							
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit
CCS (Rx) / SCS (Tx)			Reserved (keep 0)	Data size		E	S

Where:

- CCS/SCS (Client/Server Command Specifier) defines the type of the SDO service that is beige executed (.e. upload, download, block upload, block download, segmented upload, segmented download requests/responses).
- Data Size (valid only if both “E” and “S” bits have value 1) indicates the number of bytes in the SDO that do not contain useful data.
- E indicates that the requested service will only require 1 communication cycle (i.e. Rx -> Tx)
- S indicated whether the Data Size indication will be valid.

SDO communication is peer to peer. The DMC Controller can only reply with TxSDO when it receives a RxPDO from a VCU/PLC. In the following chapters, the two most common SDO services (SDO read request/response and SDO write request/response) will be discussed. For further details about SDO content and services see [1].

2.5.1 SDO read service

This SDO service allows a CAN Open Master (COM) device to interrogate a DMC Controller to know one of its object’s value.
An SDO read request must be sent from a COM device with the following characteristics:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7
0x40	Requested object index LSB	Requested object index MSB	Requested object sub-index	Reserved (keep 0)	Reserved (keep 0)	Reserved (keep 0)	Reserved (keep 0)

Once a DMC Controller will receive this message, it will reply with the following SDO read response:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7

0x40	Requested object index LSB	Requested object index MSB	Requested object sub-index	Requested object value LLSB	Requested object value LSB	Requested object value MSB	Requested Object value MMSB

2.5.2 SDO write service

This SDO service allows a CAN Open Master (COM) device to write a DMC Controller's object.
An SDO write request must be sent from a COM device with the following characteristics:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7
0x23	Target object index LSB	Target object index MSB	Target object sub-index	Download value LLSB	Download value LSB	Download value MSB	Download value MMSB

Once a DMC Controller will receive this message, it will reply with the following SDO write response acknowledge:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7
0x60	Target object index LSB	Target object index MSB	Target object sub-index	0	0	0	0

2.5.3 SDO abort transfer service

This SDO service can replace any SDO response in case the requested action cannot be carried out. The DMC controller will reply to a SDO request message with the following Tx SDO:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7
0x80	Target/Requested object index LSB	Target/Requested object index MSB	Target/Requested object sub-index	Abort code LLSB	Abort code LSB	Abort code MSB	Abort code MMSB

The abort error code can be used to investigate the reason why the SDO requested service could not be carried out. The possibles are listed below.

Abort code	Description	Suggested action
0x 0503 0000	Toggle bit fault	In a segmented upload/download, the toggle bit is not managed correctly
0x 0504 0000	SDO timeout	
0x 0504 0005	Unsupported access	User might not have access rights to modify this parameter
0x 0601 0000	Trying to read a write-only object	Check the requested/target object index correctness.
0x 0601 0001	Trying to write a read-only object	Check the requested/target object index correctness.
0x 0602 0000	The requested/target object index does not exist in the object dictionary of the device	Check the requested/target object index correctness.
0x 0604 0041	Trying to map a non-mappable object in a PDO	Check the requested/target object index correctness.
0x 0609 0011	The requested/target object sub-index does not exist in the object dictionary of the device	Check the requested/target object index correctness.
0x 0609 0030	Trying to download an invalid value	Check the download value acceptable values from DMC documentation

0x 0609 0031	Trying to download a too high value	Check the download value acceptable values from DMC documentation
0x 0609 0032	Trying to download a too low value	Check the download value acceptable values from DMC documentation
0x 0800 0000	Generic fault	If trying to reset the error, notice that this is not possible when the object 0x6040 "ControlWord" is mapped in any RxPDOs. Please use the "reset fault" bit of the ControlWord to reset the active fault.
0x 0800 0020	Data storage error (hardware)	FRAM could not be accessed. Try again. If error persists, contact DMC
0x 0800 0021	Data storage error (local control)	The target object could not be written under the actual circumstances. If trying to clear an error, make sure the error is actually clearable (see "2.6 Clearing an active fault Via CAN"). If trying to write a command variable with a SDO, this operation is forbidden.

2.6 Clearing an active fault Via CAN

In DMC Controllers it is possible to recover from a faulty situation under certain circumstances. This chapter will discuss the circumstanced and the methods which can be used to recover from a fault.
Before describing the fault-reset conditions, it is worth describing the fault structure of DMC error codes. Fault codes are divided into 4 different categories, with increasing level of severity:

1. Controller warnings
2. Drive errors
3. Soft error
4. Hard errors

Controller warnings are self-recovering once the "warning" condition is removed. For example, the fault F2 (voltage getting low) will disappear automatically as soon as battery voltage will be restored above a configurable warning level. Trying to clear controller warnings is useless, as the controller will re-display the fault immediately after clearing, in case the warning condition persists.
Drive errors faults are not-sever faults that cause the controller to stop de-energizing the motor with a controlled ramp, but do not disconnect the power from the battery (i.e. not opening the Line Contactor). Resetting these faults is possible only with a "neutral cycle", meaning that no drive command must be present for the controller to self-recover from the faulty condition.
Soft errors indicate conditions which are not dangerous for the controller, thus the power is not disconnected from the battery, but drive is still immediately inhibited for precaution. These faults are generally self-recovering with a neutral cycle, meaning that no drive command must be present for the controller to self-recover from the faulty condition.
Hard errors are faults that arise when a dangerous condition is detected and those errors cause the immediate stop to energizing the motor and the disconnection from battery power. Those faults can be generally recovered by turning the controller off. Those faults are the only one worthy of a reset via CAN.

In order to reset a fault, the DMC Controller must be in neutral and the motor must be at standstill. Not all hard errors are clearable anyway, as some are directly related to safety (i.e. F27 Hide side mosfet shortcircuit). The list of clearable/not clearable faults is reported below:

Fault	Clearable	Fault	Clearable
19	Yes	30	Yes
20	Yes	31	Yes
21	Yes	32	No
22	Yes	33	No
23	No	34	No
24	No	35	No
25	Yes	36	No
26	Yes	37	No
27	No	38	No
28	No	39	No
29	No	40	No

To reset an active hard error in the DMC controller, two methods can be used: SDO reset of Reset trough ControlWord.

2.6.1 Fault reset trough SDO

The SDO fault reset method simply consists of "clearing" the fault code object (index 0x3840, sub-index 0x00) through an SDO download service. The SDO to be sent has the following characteristics:

ID = 600 + target node nr							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byre 5	Byte 6	Byte7
0x23	0x40	0x38	0x00	0x00	0x00	0x00	0x00

As a response, the DMC Controller will reply with either a standard acknowledgement of with an abort SDO, in case the clearing is not possible.
Please notice that SDO fault reset is only possible in cas the Object 0x6040 ControlWord is not mapped in any active RxPDO. In case the latter holds true, then the fault reset must be described as in the following section.

2.6.2 Fault reset trough ControlWord

The DMC Controllers comply to CAN Open DS402, as described in “5.7 Device management with Control Word (Object 0x6040)”. Trough a standard bit-significant object called “ControlWord” it is therefore possible to fully control the power status of the DMC controller’s application. This includes resetting an active fault to be able to re-apply power. Please refer to the proper section of this manual to gather further information about device-control trough the ControlWord object.

3 Setting up the Controller

Menu 7 "CAN BUS setup"

In this menu the user will find all the configurable parameters that concern the CAN bus. It is intended that if any of those parameters requires modification, it means that the DMC Sigma2N or SigmaLITE controller is properly wired in a CAN network, with adequate isolation of supply and termination resistances installed.
The menu is organized in sub-menus, each of them corresponding to a specific function.

Menu 7.1 "CAN general" for Sigma2N controller

This sub-menu encloses general parameters that the DMC Sigma2N or SigmaLITE controller will use for communicating with other devices.

Cal. Ref.	Parameter name "Calibrator text"	Range	Step size	Default	Index	Sub-index
1	CAN node number "CAN node"	1 - 127	1	1	0x2A03	0x0
2	CAN bit rate "CANbitRt"	0 - 3	1	2	0x2A04	0x0
3	CAN node read via digital input "CAN_DIG"	0 - 1	1	0	0x2A05	0x0
4	Producer heartbeat time "CANOHrBt"	100ms – 1000ms	10ms	1000ms	0x1017	0x0
5	DualMotor and SharedLC messages on 29 or 11 bit IDs "DM&LC ID"	0 - 1	1	0	0x2A0A	0x0
6	CAN communication powerup delay "PwrUpDly"	0.0s – 5.0s	0.1s	0.0s	0x2A0B	0x0
7	CAN Open master SYNC produced enable	0 - 1	1	0	0x2A0F	0x0
8	SYNC generation time	10ms – 1000ms	10ms	100ms	0x2A10	0x0

M7.1-1 CAN node number "CAN node"

This sets the CAN node number for the controller. It must be uniquely assigned to each node of the network. This is the ID on which SDO communication will be based.
In case "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0 or 1, this is the node-ID that will be used for constructing the identifier of each message that the DMC Sigma2N or SigmaLITE controller will be able to send/receive.
In case "M7.2-1 PDO free mapping type "FreeMapT"" is set to 2 or 3, the user has the possibility to assign any node number to the message identifiers, but **<u>SDO communication. DualMotor function and Shared LineContactor function will still be based upon this parameter</u>**.
Example: the user has set this parameter to #5. SDO communication will be performed trough messages 0x585 and 0x605, even though the user has assigned to TxPDO1 the message ID 0x182 (that should actually belong to node #2) and to RxPDO1 the message ID 0x87A (random possible assignable ID).

M7.1-2 CAN bit rate "CANbitRt"

It changes the Can communication rate of the controller:

- If set to **0**, the bitrate is 100 kbit/s
- If set to **1**, the bitrate is 125 kbit/s
- If set to **2**, the bitrate is 250 kbit/s
- If set to **3**, the bitrate is 500 kbit/s

The change is active only after a key power off-on cycle.

WARNING!

! VERY IMPORTANT: if the bit rate of the controller is changed also the calibrator bit rate has to be changed to communicated again with the controller.

To change the calibrator bit rate follow this procedure:

1) Press and hold the SEL button for more than 2 seconds in order to return to the calibrator’s main page;
2) Scroll with ⬇ button to “About & Setup” menu; this is the calibrator menu for info and bit rate setup;
3) Press ▷ and ◁ buttons at the same time and information about calibrator will appear;
4) Scroll with ⬇ button to “bitrate” entry;
5) Press ▷ or ◁ buttons to increase or decrease the Calibrator’s CAN bitrate.
6) When the desired bit rate is displayed cycle the key of controller for resetting the calibrator.
7) The calibrator at new power up will operate on CAN bus at the selected bit rate.

WARNING!

! VERY IMPORTANT: If the calibrator bit rate and controller bit rate are not set consistently, the calibrator is not communicating with the controller.

M7.1-3 CAN node read via digital input “CAN_DIG”

If this parameter is set to **1** it is possible to read/assign CAN Node Number from physical inputs. To make this active properly, also control Via CAN Open must be activated and the Can Node assigned by physical digital input must be between 1 and 15.
If any of the mentioned conditions is not fulfilled a fault code (F7 S014, F29 S013) will appear.

The Can node ID number is assigned by means of the pins reported in the following list

- DI 1 (pin 1) (LSB)
- DI 2 (pin 2)
- DI 3 (pin 3)
- DI 5 (pin 5) (MSB)

Inputs have to be considered normally open (if the switch connected to the input is open, then DIx = 0; if the switch connected to the input is closed, then DIx = 1).
The rule to assign Can node ID number is the following:

Can Node Number = 8*DI5 + 4*DI3 + 2*DI2 + DI1

The can node number ID is assigned once during controller power up.
The CAN node assignment is not compatible with the SafeStop function.

!IMPORTANT Notice. With this option is only possible to set node numbers from 1 to 15 !

M7.1-4 Producer heartbeat time “CANOHrBt”

This setting adjusts the transmit refresh rate or the Can Open Heartbeat messages.

M7.1-5 DualMotor and SharedLC messages on 29 or 11 bit IDs “DM&LC ID”

This parameter sets whether the CAN messages used to manage the DualMotor and Shared Line Contactor functions will be based on 11bit or 29bit CAN IDs.

- If set to **0 (29bit IDs)**, the Dual Motor and Shared Line Contactor functions will use the following CAN IDs:
  - 0x00000900 |Dual Motor Master Node Number
  - 0x00000900 | Dual Motor Slave Node Number
  - 0x00000A00 | Dual Motor Master Node Number
  - 0x00000A00 | Dual Motor Slave Node Number
  - 0x00000880 | Line Contactor Master Node Number
  - 0x00000880 | Line Contactor Slave #1 Node Number
  - 0x00000880 | Line Contactor Slave #n Node Number
- If set to **1 (11bit IDs)**, the Dual Motor and Shared Line Contactor functions will use the following CAN IDs:
  - 0x400 |Dual Motor Master Node Number
  - 0x400 | Dual Motor Slave Node Number
  - 0x500 | Dual Motor Master Node Number
  - 0x500 | Dual Motor Slave Node Number
  - 0x480 | Line Contactor Master Node Number
  - 0x480 | Line Contactor Slave #1 Node Number
  - 0x480 | Line Contactor Slave #n Node Number

Please notice that if using 11bit IDs for the said function, the user might create conflicts with the standard CAN Open IDs. Huge attention must be adopted when mapping the Controller’s PDOs.

M7.1-6 CAN communication powerup delay “PwrUpDly”

This parameter sets a delay in starting any activity on the CAN bus. Until the delay is elapsed, the DMC Controller will not try to send and/or receive any message on the bus.

This delay is useful in case the CAN network is isolated and powered by a DC-DC converter. Under this condition, in case the DC-DC and Controller are powered up together with the same Key switch, it might happen that the DC-DC takes time to charge and actually provide output power, while the Controller is already "ready to run" and tries to start CAN bus activities. This would result in application errors, since the CAN bus could not be accessed (being it not powered).

M7.1-7 CAN Open master SYNC produced enable “SYNCprod”

This parameter instructs the Sigma2N or SigmaLITE controller to be a SYNC producer (role usually covered by a CAN Open Master). As default, the SYNC COB-ID follows the standard CAN Open specification (i.e. ID 0x080). This SYNC will be a message with DLC 1, containing an increasing counter from 0 to 255 and then rolling to 0.
Mind that in a CAN Open network, only one device should be set to be a SYNC producer.

M7.1-8 SYNC generation time “SYNCtime”

The parameter is used to set the timer for SYNC generation. The DMC Sigma2N or SigmaLITE controller will generate the SYNC message at this rate when setup to be a SYNC producer. Mind that in a CAN Open network, only one device should be set to be a SYNC producer.

Menu 7.1 “CAN general” for SigmaLITE controller

This sub-menu encloses general parameters that the DMC Sigma2N or SigmaLITE controller will use for communicating with other devices.

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	CAN node number “CAN node”	1 - 127	1	1	0x2A03	0x0
2	CAN bit rate “CANbitRt”	0 - 3	1	2	0x2A04	0x0
3	Producer heartbeat time “CANOHrBt”	100ms – 1000ms	10ms	1000ms	0x1017	0x0
4	DualMotor and SharedLC messages on 29 or 11 bit IDs “DM&LC ID”	0 - 1	1	0	0x2A0A	0x0
5	CAN communication powerup delay “PwrUpDly”	0.0s – 5.0s	0.1s	0.0s	0x2A0B	0x0
6	CAN Open master SYNC produced enable	0 - 1	1	0	0x2A0F	0x0
7	SYNC generation time	10ms – 1000ms	10ms	100ms	0x2A10	0x0

M7.1-1 CAN node number “CAN node”

Refer to “M7.1-1 CAN node number “CAN node””.

M7.1-2 CAN bit rate “CANbitRt”

Refer to “M7.1-2 CAN bit rate “CANbitRt””.

M7.1-3 Producer heartbeat time “CANOHrBt”

Refer to “M7.1-4 Producer heartbeat time “CANOHrBt””.

M7.1-4 DualMotor and SharedLC messages on 29 or 11 bit IDs “DM&LC ID”

Refer to “M7.1-5 DualMotor and SharedLC messages on 29 or 11 bit IDs “DM&LC ID””.

M7.1-5 CAN communication powerup delay “PwrUpDly”

Refer to “M7.1-6 CAN communication powerup delay “PwrUpDly””.

M7.1-6 CAN Open master SYNC produced enable “SYNCprod”

Refer to “M7.1-7 CAN Open master SYNC produced enable”.

M7.1-7 SYNC generation time “SYNCtime”

Refer to “M7.1-8 SYNC generation time”.

Menu 7.2 “PDO premapped setup”

This submenu defines the options for having the DMC Sigma2N or SigmaLITE controller running with pre-mapped Tx and Rx PDOs.
In case parameter “M7.2-1 PDO free mapping type “FreeMapT”” is set to 0, the Controller at powerup will load the pre-mapped PDOs according to the configuration made in this submenu. In any other case the Controller will load the PDO configuration performed by means of DMC Configurator or SDO communication.

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	PDO free mapping type “FreeMapT”	0 - 3	1	0	0x2A06	0x0
2	RxPDO premaped configuration “RxPDOmap”	0 -6	1	0	0x2A07	0x0
3	PDOs configuration “PDO sel”	0 - 19	1	4	0x2A08	0x0
4	Battery current limit via CAN enable “BclCanMs”	0 - 1	1	0	0x2A09	0x0
5	TxPDO1 event timer “TxPDO1Rt”	10ms – 9000ms	5ms	1000ms	0x1800	0x5
6	TxPDO2 event timer “TxPDO2Rt”	10ms – 9000ms	5ms	100ms	0x1801	0x5
7	TxPDO3 event timer “TxPDO3Rt”	10ms – 9000ms	5ms	100ms	0x1802	0x5
8	TxPDO4 event timer “TxPDO4Rt”	10ms – 9000ms	5ms	1000ms	0x1803	0x5
9	RxPDO1 event timer “RxPDO1TO”	10ms – 1000ms	5ms	150ms	0x1400	0x5
10	RxPDO2 event timer “RxPDO2TO”	10ms – 1000ms	5ms	500ms	0x1401	0x5

M7.2-1 PDO free mapping type “FreeMapT”

This parameter effectively defines whether the DMC Sigma2N or SigmaLITE controller will use pre-mapped or free mapping modes PDOs. The available options are:

0. **CAN Open pre-mapped configuration mode;**
   If this option is selected, the controller at powerup will load the PDO mapping according to parameters choice in this menu.
   By means of parameter “M7.2-2 RxPDO premaped configuration “RxPDOmap”” the user can choose among several RxPDO1 maps. These maps are all representing different control message configurations and are intended to be used to give drive commands to the Controller.
   By means of parameter “M7.2-3 PDOs configuration “PDO sel”” the user can choose to active the reception of RxPDO1 and the output of TxPDO1 to TxPDO4.
   By means of parameter “M7.2-4 Battery current limit via CAN enable “BclCanMs”” the user can choose to activate the reception of RxPDO2.
1. **CAN Open basic free mapping mode;**
   If this option is selected, the controller at powerup will load the user-defined configuration performed either through the DMC Configurator mapper or trough SDO. This mode allows the user to enable/disable an arbitrary number of Tx and Rx PDO, each mapped with the desired content. For PDOs 1 to 4 the user can also select whether they will be on an 11-bit or 29-bit identifier (PDOs 5 to 8 are only available on 29-bit identifiers).
2. **CAN Open advanced free mapping mode;**
   This mode extends the basic free mapping, also introducing the possibility to setup custom message IDs. This means, for example, that to message TxPDO1 can actually be assigned an identifier different from “0x180 + Node nr” (which is the standard defined by CiA DS301). The user can choose to assign to this message either a different ID (keeping the node number of the controller), either a standard ID with a node number different from the Controller’s node number, either a complete nonstandard ID+node number combination. This mode gives huge flexibility of configuration, but the user should take care of possible conflicts between messages in the network. This solution is highly recommended to be adopted only by advanced users or with the help of DMC engineers.

3. **J-1939 Ready free mapping;**
   This mode further extends the CAN Open advanced free mapping in the sense that the CAN Open "communication" state machine is excluded and is forced to be in "operational" status. The user could adopt this mode to implement a J1939 compatible management of the controller, without the need of the Communication state machine management and its NMT messages. Please notice that this mode is highly exposed at risk of committing errors and falling out Controller's safety management and is therefore only suggested to be used by advanced users.

NOTICE!

In case this parameter is set different than 0 (not premapped configuration), the rest of the parameters in this menu will not be active and will not take any effect. To configure the controller's received and transmitted messages in their content, ID, frame type and timing, it will be mandatory to use the DMC Configurator "CAN configurator" tool or SDO communication.

M7.2-2 RxPDO premaped configuration "RxPDOmap"

This setting is active if parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0 and "M7.2-3 PDOs configuration "PDO sel"" is set lower than 4 ("Control Via CAN Open" mode).
The DMC Sigma2N or SigmaLITE controller is able to receive two kinds of pre-mapped RxPDO1:

- By setting this parameter from 0 to 2, the controller is expecting to receive the Drive Command as **"Remote Input mode"**. In other words the RxPDO1 has to embed the Drive Command in form of digital and analogue inputs (direction switches, accelerator and footbrake). All the standard functions of the Sigma2N or SigmaLITE controller are available, with the only difference that the inputs are coming Via CAN instead of from the wired connector. For more information refer to [1].
- By setting this parameter from 3 to 6 the controller is expecting to receive the Drive Command as **"Setpoint Mode"**, which is just a speed (or torque) setpoint. For more information refer to [1].

This parameter also defines the type of control Via CAN Open options.
The available options are: "*Remote Input*" or "*Setpoint*" mode with/without "*Speed Limit Via Can"* or "*Torque Limit Via CAN*"

These options can be enabled/disabled according to following table for traction software:

Value	Speed Limit VIA CAN	Torque Limit Via CAN	Control Mode
0	Disabled	Disabled	Remote Input Mode
1	Enabled	Disabled	Remote Input Mode
2	Disabled	Enabled	Remote Input Mode
3	Disabled	Disabled	Setpoint Mode, Speed control
4	Disabled	Enabled (speed control only)	Setpoint Mode, Speed control
5	Disabled	Disabled	Setpoint Mode, Torque control
6	Enabled (torque control only)	Disabled	Setpoint Mode, Torque control

These options can be enabled/disabled according to following table for pump software:

Value	Speed Limit VIA CAN	Torque Limit Via CAN	Control Mode
0	Disabled	Disabled	Remote Input Mode
3	Disabled	Disabled	Setpoint Mode

Speed Limit VIA CAN

If **Speed Limit Via CAN** is activated, it is possible to set a dynamic speed limit in the RxPDO1.
For the details and speed limit Via CAN format refer to [1].

Torque Limit VIA CAN

If **Torque Limit Via CAN** is activated, it is possible to set a dynamic torque limit in the RxPDO1.
For the details and torque limit Via CAN format refer to [1].

M7.2-3 PDOs configuration “PDO sel”

This setting allows to select the activation of the CAN HMI protocol if “M7.2-1 PDO free mapping type “FreeMapT”” is set to 0.
These are the possible options:

- 0: Control Via CAN, Input RxPDO1 output TxPDO1
- 1: Control Via CAN, Input RxPDO1 output TxPDO1,3
- 2: Control Via CAN, Input RxPDO1 output TxPDO1,2
- 3: Control Via CAN, Input RxPDO1 output TxPDO1,2,3
- 4: No message input/output (default)
- 5: Monitor Via CAN, output TxPDO1
- 6: Monitor Via CAN, output TxPDO2
- 7: Monitor Via CAN, output TxPDO1,2
- 8: Monitor Via CAN, output TxPDO3
- 9: Monitor Via CAN, output TxPDO2,3
- 10: Monitor Via CAN, output TxPDO1,3
- 11: Monitor Via CAN, output TxPDO1,2,3
- 12: Monitor Via CAN, output TxPDO4
- 13: Monitor Via CAN, output TxPDO1,4
- 14: Monitor Via CAN, output TxPDO2,4
- 15: Monitor Via CAN, output TxPDO3,4
- 16: Monitor Via CAN, output TxPDO1,2,4
- 17: Monitor Via CAN, output TxPDO1,3,4
- 18: Monitor Via CAN, output TxPDO2,3,4
- 19: Monitor Via CAN, output TxPDO1,2,3,4

Using CAN HMI messages allows to insert the controller in an existing CAN network with third party devices. Refer to “5 Control via CAN function” for all details.
For the RxPDO and TxPDO premapped content please refer to[1].

Here a short description of the contend of TxPDO and RxPDO when premapped.

- TxPDO1 Motor Speed, Battery Current, Fault Code Subcode
- TxPDO2 Drive status, Speed and Torque limit indicators, Motor Current, Torque %
- TxPDO3 Motor and Controller Temperatures, Battery Voltage, Motol Limit Indicator, Digital Ouput Status
- TxPDO4 Analog and Digital Input Status
- RxPDO1 Drive Commands (HMI analogue and digital input if Remote Input Mode) Speed or Torque setpoint (if setpoint mode selected).
- Rx PDO2 Drive and Regen Battery current limits, BDI (SOC) percentage.

M7.2-4 Battery current limit via CAN enable "BclCanMs"

This setting is active if parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0.
It basically sets a pre-mapped RxPDO2 embedding the BDI (SOC) value and two Battery Current Limits (Drive and Regenerative). For message content please refer to [1].

- If set to **0 (default)** a battery is managed with static battery current limit according to the settings "M6-10 Drive Battery Current Limit "IBattMax"" and "M6-11 Regen Battery Current Limit "IBattReg"" (see [2]). BDI (State of Charge Value) value is calculated inside controller (typical for lead acid battery).
- If set to **1 (BCL and BDI Via CAN Open enabled),** (BCLs and BDI Via CAN Open enabled), BDI (SOC) value and Drive and Regen Battery Current Limits are received Via CAN Open in the same BCL message (RxPDO2). This option is compatible with multi-controller multi-battery applications.

M7.2-5 TxPDO1 event timer "TxPDO1Rt"

This parameter defines the transmission refresh rate of TxPDO1.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

M7.2-6 TxPDO2 event timer "TxPDO2Rt"

This parameter defines the transmission refresh rate of TxPDO2.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

M7.2-7 TxPDO3 event timer "TxPDO3Rt"

This parameter defines the transmission refresh rate of TxPDO3.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

M7.2-8 TxPDO4 event timer "TxPDO4Rt"

This parameter defines the transmission refresh rate of TxPDO4.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

M7.2-9 RxPDO1 event timer "RxPDO1TO"

This parameter defines the timeout time of RxPDO1 (Drive command message).
In case RxPDO1 is enabled and not received within the set timeout a hard failure (F29S005) will occur.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

M7.2-10 RxPDO2 event timer "RxPDO2TO"

This parameter defines the timeout time of RxPDO2 (BCL and BDI message).
In case RxPDO2 is enabled and not received within the set timeout a hard failure (F29S005) will occur.
This parameter takes effect only in case parameter "M7.2-1 PDO free mapping type "FreeMapT"" is set to 0. In case any free mapping mode is enabled, the refresh rate of the said PDO must be configured either Via the DMC Configurator or through SDO communication.

Menu 7.3 “Shared line contactor”

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	Shared line contactor option “LCoption “	0 - 4	1	0	0x2A00	0x1
2	Shared line contactor reference node “Ref Node”	1 - 127	1	1	0x2A00	0x2
3	Shared line contactor refresh rate “SHLCRate”	15ms – 50ms	1ms	15ms	0x2A00	0x3

M7.3-1 Shared line contactor option “LCoption “

This parameter enables the Shared Line contactor function.

- If set **0,** the “Shared Line Contactor” and “External Line Contactor” functions are not active.
- If set **1,** the “Shared Line Contactor” function with Line contactor physically connected to the actual controller is active. This means the actual controller can be consider the one which is managing the Line Contactor and it is thus considered “Shared Line Contactor Master Controller”. Refer to “6 Shared line contactor function” for details.
- If set **2**, the “Shared Line Contactor” function with Line contactor physically connected to ANOTHER controller is active. This means the actual controller can be consider managed, regarding Line Contactor, by the “Shared Line Contactor Master Controller”. Thus the actual controller is considered as “Shared Line Contactor Slave Controller”. Refer to “6 Shared line contactor function” for details.
- If set to **3**, the “External Line Contactor” function is activated. Refer to “7 External Line Contactor management” for details.
- If set to **4**, the “External Line Contactor with independent logic supply” function is activated. This extends option #3 by leaving the user free to supply the Controller’s logic with an independent supply. Refer to “7 External Line Contactor management” and “7.2 Separated control logic supply management” for details.

M7.3-2 Shared line contactor reference node “Ref Node”

The meaning of this parameter depends upon the setting of “M7.3-1 Shared line contactor option “LCoption “”.

- If “M7.3-1Shared line contactor option “LCoption “is set to **1,** “Reference Node SLC” has to be set with the CAN node number of the last node in the CAN network that is sharing the line contactor.
- If n”M7.3-1Shared line contactor option “LCoption “” is set to **2,** “Reference Node SLC” has to be set with the CAN node number of the controller in the CAN network that is physically connected to the Line contactor and is managing the function. Thus “Reference Node SLC” in this case has to correspond to “Shared Line Contactor Master Controller” CAN node number.

M7.3-3 Shared line contactor refresh rate “SHLCRate”

This setting adjusts the refresh rate for the message used for shared line contactor management. It has to be set equal in all controllers. Suggested value is $< \frac{150}{\#\ of\ Sharing\ Slaves}$ ms, with a maximum of 75ms.

Menu 7.4 “Display”

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	Display node number “DispNode	1 - 127	1	73	0x2A01	0x1
2	Display Status field ”DispInfo”	1 - 9	1	7	0x2A01	0x2
3	Display refresh rate “RefRate”	100ms – 1000ms	10ms	200ms	0x2A01	0x3

M7.4-1 Display node number “DispNode”

If the DMC Controller is intended to communicate with a DMC Display, this parameter is used to set the node number of the DMC Display. It should be set coherently with the setting made inside the Display.
If the DMC Controller is intended to communicate with an iGauge, this parameter assumes different meanings depending on “M7.4-2 Display Status field ”DispInfo””:

- If the Controller is setup as an iGauge Master (“M7.4-2 Display Status field ”DispInfo”” set to 8), this parameter should indicate the node number of the last iGauge slave connected. In case no iGauge slave is connected, this parameter should match “M7.1-1 CAN node number “CAN node””.
- If the Controller is setup as an iGauge Slave (“M7.4-2 Display Status field ”DispInfo”” set to 9), this parameter should indicate the iGauge Master’s node number (which is in turn the iGauge node number).

NOTICE!

DMC Controller which are meant to communicate with an iGauge, should have consecutive node numbers, with the iGauge Master having the lowest node number between them.

M7.4-2 Display Status field ”DispInfo”

This sets the type of information that will appear in the General Indication Field of the DMC Display or how the Controller should behave with respect to the iGauge.

- If set to **0 (Send Device data)**, then the Controller will send only its device data to Display.
- If set to **1 (Accelerator/Demand)**, then the Accelerator/Demand as a percentage will be displayed, from 0% to 100%.
- If set to **2 (Motor Velocity)**, then the motor velocity-speed in units of RPM will be displayed, from 0 to the value set in the Autotuning menu (see [2] or [3]).
- If set to **3 (Vehicle Speed)**, then the vehicle speed in units of KPH or MPH (depending on the choice set in the DMC Display setting menu, see [9] for details) will be displayed, from 0 to the value set in Autotuning menu (see [2] or [3]), converted by means of parameter “M1.15-1 Motor/vehicle speed ratio “SpdRatio”” (see [2]).
  Mind that if MPH is chosen in the DMC Display menu, one must take care of including in the calculation of the parameter “” the scaling from KPH to MPH.
- If set to **4 (Steering)**, then the vehicle’s steering angle will be displayed using crosshairs.
- If set to **5 (Motor Current)**, then the motor current in units of A will be displayed, from 0 to the maximum rated current of the controller.
- If set to **6 (Battery Current)**, then the battery current in units of A will be displayed, from 0 to the maximum rated current of the controller.
- If set to **7 (Disabled)**, then the DMC or SigmaLITE controller will not send any message to the DMC Display.
- If set to 8 (**Disabled, use iGauge as Master**), then the DMC SuperSigma2N controller will not send any message to the DMC Display, but iGauge communication will be enabled. If more than one controller is in the network, this option must be set only in the controller physically connected to the iGauge.
- If set to 9 (**Disabled, use iGauge as Slave**), the Sigma2N controller will act as an iGauge slave, meaning that it will send only one message to the iGauge Master to also show its information on the iGauge.

NOTICE!

If more than one DMC Sigma2N or SigmaLITE controller **is in the CAN network, make sure only one of them has this parameter set between 1 and 6. This will be the DMC Sigma2N** or SigmaLITE controller **sending the drive and BDI information to the Display (and therefore the Display Master).**
In case of Dual motor configuration, the Dual Motor master must be also the Display master (i.e. the one with this parameter set different than 0).

M7.4-3 Display refresh rate “RefRate”

This parameter sets the refresh rate of the messages used to communicate with the DMC Display or with the iGauge master/slaves. It is not suggested to change this value unless there is too much traffic on the CAN bus.

Menu 7.5 “Safe Stop 1” / “CAN Safety”

This menu’s name and content vary depending on the product and on the hardware version of the specific product.

Sigma2N HW version later or equal than V07.03: “CAN Safety”

In Sigma2N with HW version later or equal than 7.03, the name of this menu is “CAN Safety” and it is used to configure the parameters related to both the Safe Torque Off and SafeStop1 safety functions.

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	Digital input 3-4 configuration “DI34cfg”	0 - 1	1	0	0x2A0D	0x0
2	Safe Torque Off safety configuration “STOsafe”	0 – 3	1	0	0x2A0C	0x0
3	SafeStop1 enable “SS1 Enab”	0 – 1	1	0	0x2A02	0x1
4	Safe Stop 1 safety configuration “SS1safe”	0 - 4	1	0	0x2A0E	0x0
5	Safe Stop 1 cut timer “SS1 Time”	0.1s – 10.0s	0.1s	5.0s	0x2A02	0x4
6	SafeStop1 torque “SS1 Torque”	0% - 100%	1%	20%	0x2A02	0x2
7	SafeStop1 ramp time “SS1 RmpTm”	0.1s – 10.0s	0.1s	1.0s	0x2A02	0x3

M7.5-1 Digital input 3-4 configuration “DI34cfg”

This parameter can be used to configure which safety function is assigned to digital inputs 3 and 4 when Control Via CAN is active (controller set in “Pre-mapping mode” (“M7.2-1 PDO free mapping type “FreeMapT”” set to 0) and Control Via CAN HMI is enabled (“M7.2-3 PDOs configuration “PDO sel”” set lower than 4) or if the controller is in any free-mapping mode (“M7.2-1 PDO free mapping type “FreeMapT”” set >= 1) and an RxPDO is mapped with some content):

- If set to **0 (SS1 on DI3, STO on DI4)**, then SafeStop1 function is assigned to DI3 and Safe Torque Off function is assigned to DI4.
- If set to **1 (STO on DI3, SS1 on DI4)**, then Safe Torque Off function is assigned to DI3 and SafeStop1 function is assigned to DI4.

The below table reports the compatibility between different parameter selections:

		M7.5-2 **Safe Torque Off safety configuration “STOsafe”**				M7.5-4 **Safe Stop 1 safety configuration “SS1safe”**				
		0	1	2	3	0	1	2	3	4
M7.5-1 Digital input 3-4 configuration “DI34cfg”	0	✓	✓	✓	×	✓	×	×	✓	✓
	1	✓	×	×	✓	✓	✓	✓	×	×

M7.5-2 Safe Torque Off safety configuration “STOsafe”

This parameter can be used to configure the Safety level for the Safe Torque Off function:

- If set to **0 (PLc level)**, then the Safe Torque Off function is performed with PLc level.
- If set to **1 (PLd level, no toggle needed)**, then the Safe Torque Off function is performed with PLd level. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 0. When this option is selected, the Safe Torque Off function is considered “armed” straight at powerup and the DI4 must be active (please check the Digital Input active H/L configuration in [2]) at any time the Sigma2N Controller is on, or the STO function will be triggered.
- If set to **2 (PLd level, toggle DI4 required)**, then the Safe Torque Off function is performed with PLd level. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 0. With this option, the drive will be inhibited until the Sigma2N recognises a DI4 rising edge/activation (please check the Digital Input active H/L configuration in [2]). In case DI4 is active at powerup, it must be deactivated and re-activated. Once this happens, the Safe Torque Off function is considered “armed” and the DI4 must be active at any time the Sigma2N Controller is on, or the STO function will be triggered.

- If set to **3 (PLc advanced level)**, then the Safe Torque Off function is performed with PLc enhanced level. This means that HW redundant safety is not applied, but the redundancy is provided by SW. Compared to option

#1, when STO is triggered, power to the motor and digital outputs will be cut by means of a SW triggered signal. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 1.

M7.5-3 SafeStop1 enable “SS1 Enab”

This parameter is used to enable the Safe Stop 1 function. The input which will trigger this function is defined by parameter “M7.5-1 Digital input 3-4 configuration “DI34cfg””.

M7.5-4 Safe Stop 1 safety configuration “SS1safe”

- If set to **0 (PLc level)**, then the SafeStop1 function is performed with PLc level.
- If set to **1 (PLd level, no toggle needed)**, then the SafeStop1 function is performed with PLd level. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 1. When this option is selected, the SafeStop1 function is considered “armed” straight at powerup and the DI4 must be active (please check the Digital Input active H/L configuration in [2]) at any time the Sigma2N Controller is on, or the SS1 function will be triggered.
- If set to **2 (PLd level, toggle DI4 required)**, then the SafeStop1 function is performed with PLd level. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 1. With this option, the drive will be inhibited until the Sigma2N recognises a DI4 rising edge/activation (please check the Digital Input active H/L configuration in [2]). In case DI4 is active at powerup, it must be deactivated and re-activated. Once this happens, the SafeStop1 function is considered “armed” and the DI4 must be active at any time the Sigma2N Controller is on, or the SS1 function will be triggered.
- If set to **3 (PLc advanced level)**, the SafeStop1 function is performed with PLc enhanced level. This means that HW redundant safety is not applied, but the redundancy is provided by SW. Compared to option #1, when SS1 is triggered and “M7.5-5 Safe Stop 1 cut timer “SS1 Time”” is elapsed, power to the motor and digital outputs will be cut by means of a SW triggered signal. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 0.
- If set to **4 (PLc advanced level, not cut on 0 speed)**, then the functionality is the same as option #3, with the only difference that, in case the motor reached zero speed and power has been removed from it before the timer defined by “M7.5-5 Safe Stop 1 cut timer “SS1 Time”” is elapsed, the digital outputs (with their associated functions) will not be cut. Mind that in case the motor is not recognised at 0 speed when the said timer is elapsed, motor power and all digital outputs will be cut. It can only be selected in case “M7.5-1 Digital input 3-4 configuration “DI34cfg”” is set to 0.

M7.5-5 Safe Stop 1 cut timer “SS1 Time”

This parameter is used to set the time after with a STO function is applied, once the SS1 function is triggered. For more detailed explanation, please refer to paragraph “5.2 Safety functions”.

M7.5-6 SafeStop1 torque “SS1 Torque”

This parameter is active only in case the Sigma2N controller is setup to work in torque control mode (see [2]).
It is used to define the torque level used to perform the SafeStop1 function.

M7.5-7 SafeStop1 ramp time “SS1 RmpTm”

This parameter assumes different meanings according to the Sigma2N control mode (see [2]).

- In **<u>Speed control mode</u>**, it defines the ramp time that the motor will try to follow, applying the required torque (up to 100%).
- In **<u>Torque control mode</u>**, it defines how quickly the braking torque defined with parameter “M7.5-6 SafeStop1 torque “SS1 Torque”” will be applied.

Sigma2N HW version earlier than V07.03: "Safe Stop 1"

In Sigma2N with HW version earlier then 7.03, the name of this menu is "Safe Stop 1" and it only embeds SafeStop1 configurable parameters.

Cal. Ref.	Parameter name "Calibrator text"	Range	Step size	Default	Index	Sub-index
1	SafeStop1 enable "SS1 Enab"	0 - 1	1	0	0x2A02	0x1
2	SafeStop1 torque "SS1 Torque"	0% - 100%	1%	20%	0x2A02	0x2
3	SafeStop1 ramp time "SS1 RmpTm"	0.1s – 10.0s	0.1s	1.0s	0x2A02	0x3

M7.5-1 SafeStop1 enable "SS1 Enab"

This parameter is used to enable the Safe Stop 1 function. The input which will trigger this function is fixed to Digital Input 3 (Pin A3 of 35-way AMPSEAL connector), which is a NC (Normally Close) input. This input must be always active (please check the Digital Input active H/L configuration in [2]), otherwise the SafeStop 1 function will be triggered.

M7.5-2 SafeStop1 torque "SS1 Torque"

This parameter is active only in case the Sigma2N controller is setup to work in torque control mode (see [2]).
It is used to define the torque level used to perform the SafeStop1 function

M7.5-3 SafeStop1 ramp time "SS1 RmpTm"

This parameter assumes different meanings according to the Sigma2N control mode (see [2]).

- In **Speed control mode**, it defines the ramp time that the motor will try to follow, applying the required torque (up to 100%).
- In **Torque control mode**, it defines how quickly the braking torque defined with parameter "M7.5-2 SafeStop1 torque "SS1 Torque"" will be applied.

SigmaLITE HW version later than V03.02: “Safe Stop 1”

In SigmaLITE with HW version earlier then 7.03, the name of this menu is “Safe Stop 1” and it only embeds SafeStop1 configurable parameters.

Cal. Ref.	Parameter name “Calibrator text”	Range	Step size	Default	Index	Sub-index
1	SafeStop1 torque “SS1 Torque”	0% - 100%	1%	20%	0x2A02	0x2
2	SafeStop1 ramp time “SS1 RmpTm”	0.1s – 10.0s	0.1s	1.0s	0x2A02	0x3
3	Safe Stop 1 cut timer “SS1 Time”	0.1s – 10.0s	0.1s	5.0s	0x2A02	0x4

M7.5-1 SafeStop1 torque “SS1 Torque”

This parameter is active only in case the SigmaLITE controller is setup to work in torque control mode (see [2]).
It is used to define the torque level used to perform the SafeStop1 function.

M7.5-2 SafeStop1 ramp time “SS1 RmpTm”

This parameter assumes different meanings according to the SigmaLITE control mode (see [2]).

- In **Speed control mode**, it defines the ramp time that the motor will try to follow, applying the required torque (up to 100%).
- In **Torque control mode**, it defines how quickly the braking torque defined with parameter “M7.5-2 SafeStop1 torque “SS1 Torque”” will be applied.

M7.5-3 Safe Stop 1 cut timer “SS1 Time”

This parameter is used to set the time after with a STO function is applied, once the SS1 function is triggered. For more detailed explanation, please refer to paragraph “5.2 Safety functions”.

4 Free mapping of DMC Controller

Being the Sigma2N and SigmaLITE controllers based on a CAN Open protocol, it introduces the possibility to freely configure the CAN bus messages of the device. This means that to any Tx and Rx PDOs an arbitrary content could be assigned and even the message ID of those messages can be configured. The Controller configuration is stored in the standard CAN Open objects 0x1400 to 0x1407 (RxPDO communication parameters), 0x1600 to 0x1607 (RxPDO mapping parameters), 0x1800 to 0x1807 (TxPDO communication parameters) and 0x1A00 to 0x1A07 (TxPDO mapping parameters). Detailed description of those objects will follow.
It is necessary to remark that CAN Open specifications define, for each device, 4 TxPDOs and 4 RxPDOs on the following IDs:

- 0x180 + node nr (TxPDO1)
- 0x280 + node nr (TxPDO2)
- 0x380 + node nr (TxPDO3)
- 0x480 + node nr (TxPDO4)
- 0x200 + node nr (RxPDO1)
- 0x300 + node nr (RxPDO2)
- 0x400 + node nr (RxPDO3)
- 0x500 + node nr (RxPDO4)

Those message are intended to have 11bit ID on the CAN bus.
DMC further extends this message capability by disposing of other 4 Tx and 4 Rx PDOs on 29bit ID:

- 0x00000580 + node nr (TxPDO5)
- 0x00000680 + node nr (TxPDO6)
- 0x00000780 + node nr (TxPDO7)
- 0x00000880 + node nr (TxPDO8)
- 0x00000600 + node nr (RxPDO5)
- 0x00000700 + node nr (RxPDO6)
- 0x00000800 + node nr (RxPDO7)
- 0x00000900 + node nr (RxPDO8)

As default the Sigma2N and SigmaLITE controllers comes with the “pre mapped” mode enabled. In this mode only TxPDO1 to 4 and RxPDO1 and 2 are usable with pre-mapped content.
To enable “Free mapping” mode the user should set parameter “M7.2-1 PDO free mapping type “FreeMapT”” greater than 0. This will result in increasing level of freedom when performing the mapping:

1. Setting that parameter to 1 will allow the user to use the whole set of PDOs (8x Tx and 8x Rx), keeping the standardized message IDs reported above. Notice that Tx and Rx PDOs from 1 to 4 can also be migrated on 29bit IDs if desired. The user can enable an arbitrary amount of messages and map any variable of the DMC object dictionary (see [7]) in them. Some restrictions apply to this, like the fact that the same variable cannot be mapped in more than one RxPDOs (since this would create uncertainty in message reception) or that the safety-specific variables of a PDO can only be mapped to the PDO they belong to (i.e. RollOverCounterTxPDO1 variable can only be mapped to TxPDO1). The user can still map the RxPDOs safety-specific variables into TxPDOs for monitoring purposes.
2. Setting that parameter to 2 will extend the user freedom in the sense that the message IDs of Tx and Rx PDOs can be freely changed. This means that, for example, the TxPDO1 message ID can be changed from the standard “0x180 + node nr” to an arbitrary ID. This can be
    - “0x180 + Any_node_nr”
    - “0xAny_ID + node nr”
    - “0xAny_ID + Any_node_nr”

   where “node nr” is the actual controller node number’s, and “Any_node_nr” is an arbitrary node number decided by the user.
   This gives the possibility, for example, to the Controller with node number 5 to actually output the RxPDO1 belonging to node number 6, therefore the Controller would act as a VCU sending drive commands to another node in the network.
   There are some restrictions to the message IDs that can be assigned to Tx and RxPDOs:

- An ID can only be assigned to a PDO (i.e. two PDOs cannot have the same ID);
- It is not possible to set the same ID on 11 and 29 bit (example: not possible to have ID 0x00000181 on TxPDO1 and 0x181 on TxPDO2);
- It is not possible to assign “standard” CAN Open IDs of Tx and Rx PDOs 5 to 8 on 11bit (i.e. IDs from 0x580 to 0x7FF are forbidden);
- It is not possible to assign “standard CAN Open IDs for SYNC and EMCY (0x80 to 0xFF), neither on 11 or 29bit IDs.
- It is not possible to assign “standard CAN Open IDs for TIMESTAMP (0x100) on 11bit ID;
- It is not possible to assign “standard CAN Open IDs for NMT (0x000)), neither on 11 or 29bit IDs;
- It is not possible to assign “standard CAN Open IDs for LSS (0x7E4 and 0x7E5) on 11bit ID;

3. Setting that parameter to 3 does not provide extension to the mapping capability itself, but actually changes the way in which the communication is handled by the Sigma2N or SigmaLITE controller. This settings in fact disables the CAN Open Communication state machine and forces the controller to be always in “operational” status. This means that the user does not have to worry about sending an NMT message to change the controller’s operational status, since it will always able to send and receive messages.

NOTICE: mapping any of the following objects will result in automatic activation of the “Control Via CAN” function, therefore causing incompatibility with other functions (i.e. Dual Motor slave controller must not have Control Via CAN function active, as only the master must be driven).

The following table refers to Sigma2N:

Object	Name	Object	Name
0x3800	HMI Analogue input 1	0x3833	RPM demand
0x3801	HMI Analogue input 2	0x3835	Percentage demand %4096
0x3802	HMI Analogue input 3	0x3846	RPM speed limit
0x3806	HMI Digital input word	0x3847	Percentage speed limit &4096
0x3808	HMI Digital input 1	0x3848	Percentage torque limit
0x3809	HMI Digital input 2	0x6040	DS402 ControlWord
0x380A	HMI Digital input 3	0x6071	DS402 torque demand
0x380B	HMI Digital input 4	0x6072	DS402 torque limit
0x380C	HMI Digital input 5	0x6080	DS402 speed limit
0x380D	HMI Digital input 6	0x60FF	DS402 speed demand
0x380E	HMI Digital input 7		

The following table refers to SigmaLITE:

Object	Name	Object	Name
0x3875	Accelerator	0x3889	Generator enable functional input
0x3876	Brake	0x388A	Pump accelerator #2
0x3877	Steerpot	0x388B	Pump speed #1 enable functional input
0x3878	Generator speed setpoint	0x388C	Pump speed #2 enable functional input
0x3879	Generator battery current limit	0x388D	Pump speed #3 enable functional input
0x387A	HMI preset byte #1	0x388E	Pump speed #4 enable functional input
0x387B	HMI preset byte #2	0x388F	Pump speed #5 enable functional input
0x387C	Forward functional input	0x3890	Pump powersteer enable functional input
0x387D	Reverse functional input	0x3891	Pump inhibit functional input
0x387E	Footswitch functional input	0x3833	RPM demand
0x387F	Interlock functional input	0x3835	Percentage demand %4096
0x3880	Speed limit #1 functional input	0x3846	RPM speed limit
0x3881	Speed limit #2 functional input	0x3847	Percentage speed limit &4096
0x3882	Speed limit #3 functional input	0x3848	Percentage torque limit
0x3883	Advanced mode #1 functional input	0x6040	DS402 ControlWord
0x3884	Advanced mode #2 functional input	0x6071	DS402 torque demand
0x3885	Handbrake functional input	0x6072	DS402 torque limit
0x3886	Inching forward functional input	0x6080	DS402 speed limit
0x3887	Inching reverse functional input	0x60FF	DS402 speed demand
0x3888	Digital footbrake functional input		

Return to Table Of Contents

4.1 How to free-map a DMC Controller

In order to gain advantage of the full flexibility of the Sigma2N or SigmaLITE controller, the user should be aware of how the free mapping is actually performed. DMC supports standard objects defined by CiA DS301 for configuring the PDOs in the Controllers. Those are divided in 2 categories: Communication and Mapping objects. The formers define the COB-ID of the messages, their timing and other properties, while the latter actually embed the indication of which variable should be contained in which position of any PDO.

4.1.1 RxPDO Communication objects (Index 0x1400 to 0x1407)

Those objects contain the communication parameters for RxPDOs. Object 0x1400 is referred to RxPDO1, object 0x1401 is referred to RxPDO2 and so on.
Each object includes a number of sub-indexes, each corresponding to a different configuration setup for the specific RxPDO. In the following every Sub-index will be described in details.

Sbuindex 0 – NOF ENTRIES

Defines the number of available configurable parameters (i.e. the maximum SubIndex value within this object).

SubIndex 1 – COB-ID

It is a 32bit variable that defines the message ID, the message type (11 or 29 bit) and its enabling. It is defined as follows:

31	30	29	28	11	10	0
Enabled	Reserved	ID type	0x00000		11bit CAN ID	
			29bit CAN ID			

- Bits 0 to 28 are used to define the message ID;
- Bit 29 is used to specify whether the ID is standard (i.e. 11 bit) or extended (i.e. 29 bit). In the first case, the bit should be set to 0, otherwise it should be set to 1;
- Bit 30 is reserved and does not have any influence;
- Bit 31 defines whether the PDO is valid (i.e. it is enabled) or not (i.e. it is disabled). In case the PDO has to be enabled this bit should be set to 0, otherwise it should be set to 1;

SubIndex 2 – TRANSMISSION TYPE

This object defines how the RxPDO should be processed. Admissible values are:

- 0x00: Actuate on SYNCH. RxPDO will be processed with the reception of the next SYNCH message.
- 0xFE or 0xFF: Actuate on reception. RxPDO will be processed asynchronously, as soon as the message is received.

SubIndex 3 – INHIBIT TIME

Not supported.

SubIndex 4 – RESERVED

This object is reserved and its value should be left to 0.

SubIndex 5 – EVENT TIMER

This object is used for deadline monitoring. It is basically the timeout timer for the RxPDO. In case the next RxPDO is not received within this time, the application will report an error.

4.1.2 TxPDO Communication objects (Index 0x1800 to 0x1807)

Those objects contain the communication parameters for TxPDOs. Object 0x1800 is referred to TxPDO1, object 0x1801 is referred to TxPDO2 and so on.
Each object includes a number of sub-indexes, each corresponding to a different configuration setup for the specific TxPDO. In the following every Sub-index will be described in details.

Sbuindex 0 – NOF ENTRIES

Defines the number of available configurable parameters (i.e. the maximum SubIndex value within this object).

SubIndex 1 – COB-ID

It is a 32bit variable that defines the message ID, the message type (11 or 29 bit) and its enabling. It is defined as follows:

31	**30**	**29**	**28 11**	**10 0**
Enabled	Reserved	ID type	0x00000	11bit CAN ID
			29bit CAN ID	

- Bits 0 to 28 are used to define the message ID;
- Bit 29 is used to specify whether the ID is standard (i.e. 11 bit) or extended (i.e. 29 bit). In the first case, the bit should be set to 0, otherwise it should be set to 1;
- Bit 30 is reserved and does not have any influence;
- Bit 31 defines whether the PDO is valid (i.e. it is enabled) or not (i.e. it is disabled). In case the PDO has to be enabled this bas should be set to 0, otherwise it should be set to 1;

SubIndex 2 – TRANSMISSION TYPE

This object defines how the TxPDO should be send out. Admissible values are:

- 0x00: Send on SYNCH. TxPDO will be sent with the reception of the next SYNCH message.
- 0xFE or 0xFF: Send on timer. TxPDO will be send asynchronously, with its timer setting.

SubIndex 3 – INHIBIT TIME

Not supported.

SubIndex 4 – RESERVED

This object is reserved and its value should be left to 0.

SubIndex 5 – EVENT TIMER

This object is used for timing the message output. The TxPDO will be sent out with this value as refresh rate.

4.1.3 RxPDO Mapping objects (Index 0x1600 to 0x1607)

Those objects contain the mapping for RxPDOs. Object 0x1600 is referred to RxPDO1, object 0x1601 is referred to RxPDO2 and so on.
Each object includes a number of sub-indexes, each corresponding to a different configuration setup for the specific RxPDO. In the following every Sub-index will be described in details.

Sbuindex 0 – NOF MAPPED VARIABLES

Defines the number of expected mapped variables. This value should match the highest sub-index with a value different than 0.

SubIndex 1 to 8 – Mapped variables

It is a 32bit constant that defines the mapped variable. It is defined as follows:

31	16	15	8	7	0
Object index		Object sub-index		Object length in bits	

- Bits 0 to 7 are used to define the object data size in bit (i.e. a 8bit variable will have this value to 0x8, a 16bit variable will have this value to 0x10, a 32pit variable will have this value to 0x20);
- Bit 8 to 15 are used to specify the sub-index of the mapped object;
- Bit 16 to 31 are used to specify the index of the mapped object.

Example:
The user wants to map the "Target velocity", which is object 0x60FF Sub 0x00 and it is a s32 variable (0x20 bits in hexadecimal), in first position of RxPDO1.
Then the Value of the Index 0x1600 Subindex 0x1 must be 0x60FF0020.

Example:
The user wants to map the "Control word", which is object 0x6040 Sub 0x00 and it is a u16 variable (0x10 bits in hexadecimal), in third position of RxPDO3.
Then the Value of the Index 0x1602 Subindex 0x3 must be 0x60400010.

NOTICE!

The user must take care of not exceeding the maximum mappable length of 8 bytes for each Tx or Rx PDO.
This means that the user can map 8x 8bit variables, 4x 16bit variables, 2x 32bit variables or combinations of them.
Mapping, for example, 3x 8bit variables, 1x16bit variable and 1x 32 bit variable, would result in mapping a total 9 bytes, which is not practically feasible for a CAN message and would therefore result in an error when powering up the Controller.

NOTICE!

The variables must be mapped consecutively in a CAN message, without leaving blank spaces (i.e. Subindexes with value 0x0) in between.
This means that the user cannot assign a value to object 0x1600 Subindex 0x01, then leave Subindex 0x02 empty and finally assign a value to Subindex 0x03.
If the user wants to leave a "blank" space, he must use the DMC "Dummy" variable (Index 0x3845 Subindex 0x00, which is an 8bit) created on purpose to space content. Take care that this variable increases the "Number of mapped objects" and should be therefore taken into account when assigning a value to the Subindex 0x00 of the RxPDO mapping objects.
Blank spaced are of course allowed at the end of a mapping object, since the maximum size of the CAN message can be reached by mapping less than the 8 available mapping slots.

The following picture reports various mapping examples. Things highlighted in RED correspond to mapping mistakes, while things highlighted in GREEN represent possible fixes and correct solutions.

Object 0x1600		
SubIndex	**Value**	**Description**
0x00	0x3	Mappe objects
0x01	0x60400010	Control Word
0x02	0x00000000	Empty
0x03	0x38000010	AI1 - Accel
0x04	0x38060008	DI word
0x05	0x00000000	Empty
0x06	0x00000000	Empty
0x07	0x00000000	Empty
0x08	0x00000000	Empty

Mapped lenght 0x10+0x10+0x8 = 40bits = 5bytes OK

Object 0x1600		
SubIndex	**Value**	**Description**
0x00	0x3	Mappe objects
0x01	0x60400010	Control Word
0x02	0x38450008	Spacing variable
0x03	0x38000010	AI1 - Accel
0x04	0x38060008	DI word
0x05	0x00000000	Empty
0x06	0x00000000	Empty
0x07	0x00000000	Empty
0x08	0x00000000	Empty

Mapped lenght 0x10+0x80x10+0x8 = 48bits = 6bytes OK

Object 0x1600		
SubIndex	**Value**	**Description**
0x00	0x6	Mappe objects
0x01	0x60400010	Control Word
0x02	0x38450008	Spacing variable
0x03	0x38000010	AI1 - Accel
0x04	0x38060008	DI word
0x05	0x38010010	AI2 - brake
0x06	0x38460010	Speed limit RPM
0x07	0x00000000	Empty
0x08	0x00000000	Empty

Mapped lenght 0x10+0x10+0x8 = 40bits = 10bytes NOT OK

Object 0x1600		
SubIndex	**Value**	**Description**
0x00	0x4	Mappe objects
0x01	0x60400010	Control Word
0x02	0x38450008	Spacing variable
0x03	0x38000010	AI1 - Accel
0x04	0x38060008	DI word
0x05	0x00000000	Empty
0x06	0x00000000	Empty
0x07	0x00000000	Empty
0x08	0x00000000	Empty

Mapped lenght 0x10+0x80x10+0x8 = 48bits = 6bytes OK

4.1.4 TxPDO Mapping objects (Index 0x1A00 to 0x1A07)

Those objects contain the mapping for TxPDOs. Object 0x1A00 is referred to TxPDO1, object 0x1A01 is referred to TxPDO2 and so on.
Each object includes a number of sub-indexes, each corresponding to a different configuration setup for the specific TxPDO. In the following every Sub-index will be described in details.

Sbuindex 0 – NOF MAPPED VARIABLES

Defines the number of expected mapped variables. This value should match the highest sub-index with a value different than 0.

SubIndex 1 to 8 – Mapped variables

It is a 32bit constant that defines the mapped variable. It is defined as follows:

31	16	15	8	7	0
Object index		Object sub-index		Object length in bits	

- Bits 0 to 7 are used to define the object data size in bit (i.e. a 8bit variable will have this value to 0x8, a 16bit variable will have this value to 0x10, a 32pit variable will have this value to 0x20);
- Bit 8 to 15 are used to specify the sub-index of the mapped object;
- Bit 16 to 31 are used to specify the index of the mapped object.

Example:
The user wants to map the “Velocity actual value”, which is object 0x606C Sub 0x00 and it is a s32 variable (0x20 bits in hexadecimal), in first position of TxPDO1.
Then the Value of the Index 0x1A00 Subindex 0x1 must be 0x606C0020.

Example:
The user wants to map the “Status word”, which is object 0x6041 Sub 0x00 and it is a u16 variable (0x10 bits in hexadecimal), in third position of TxPDO3.
Then the Value of the Index 0x1A02 Subindex 0x3 must be 0x60410010.

NOTICE!

The user must take care of not exceeding the maximum mappable length of 8 bytes for each Tx or Rx PDO.
This means that the user can map 8x 8bit variables, 4x 16bit variables, 2x 32bit variables or combinations of them.
Mapping, for example, 3x 8bit variables, 1x16bit variable and 1x 32 bit variable, would result in mapping a total 9 bytes, which is not practically feasible for a CAN message and would therefore result in an error when powering up the Controller.

NOTICE!

The variables must be mapped consecutively in a CAN message, without leaving blank spaces (i.e. Subindexes with value 0x0) in between.
This means that the user cannot assign a value to object 0x1600 Subindex 0x01, then leave Subindex 0x02 empty and finally assign a value to Subindex 0x03.
If the user wants to leave a “blank” space, he must use the DMC “Dummy” variable (Index 0x3845 Subindex 0x00, which is an 8bit) created on purpose to space content. Take care that this variable increases the “Number of mapped objects” and should be therefore taken into account when assigning a value to the Subindex 0x00 of the TxPDO mapping objects.
Blank spaced are of course allowed at the end of a mapping object, since the maximum size of the CAN message can be reached by mapping less than the 8 available mapping slots.

5 Control via CAN function

The "Control via CAN" function allows to send the Drive Commands through the CAN bus. The Drive Commands are generated by a VCU/PLC and sent to all DMC Sigma2N and SigmaLITE controllers, that are set up to be Slaves on the CAN bus.

5.1 How to enable the Control via CAN function

To enable this function, either:

- Set controller in "Pre-mapping mode" ("M7.2-1 PDO free mapping type "FreeMapT"" set to 0) and set parameter "M7.2-3 PDOs configuration "PDO sel"" lower than 4;
- Set controller in any free-mapping mode ("M7.2-1 PDO free mapping type "FreeMapT"" set >= 1) and activate the reception of any RxPDO (with or without content) containing one of the following objects:

The following table refers to Sigma2N:

Object	Name	Object	Name
0x3800	HMI Analogue input 1	0x3833	RPM demand
0x3801	HMI Analogue input 2	0x3835	Percentage demand %4096
0x3802	HMI Analogue input 3	0x3846	RPM speed limit
0x3806	HMI Digital input word	0x3847	Percentage speed limit &4096
0x3808	HMI Digital input 1	0x3848	Percentage torque limit
0x3809	HMI Digital input 2	0x6040	DS402 ControlWord
0x380A	HMI Digital input 3	0x6071	DS402 torque demand
0x380B	HMI Digital input 4	0x6072	DS402 torque limit
0x380C	HMI Digital input 5	0x6080	DS402 speed limit
0x380D	HMI Digital input 6	0x60FF	DS402 speed demand
0x380E	HMI Digital input 7		

The following table refers to SigmaLITE:

Object	Name	Object	Name
0x3875	Accelerator	0x3889	Generator enable functional input
0x3876	Brake	0x388A	Pump accelerator #2
0x3877	Steerpot	0x388B	Pump speed #1 enable functional input
0x3878	Generator speed setpoint	0x388C	Pump speed #2 enable functional input
0x3879	Generator battery current limit	0x388D	Pump speed #3 enable functional input
0x387A	HMI preset byte #1	0x388E	Pump speed #4 enable functional input
0x387B	HMI preset byte #2	0x388F	Pump speed #5 enable functional input
0x387C	Forward functional input	0x3890	Pump powersteer enable functional input
0x387D	Reverse functional input	0x3891	Pump inhibit functional input
0x387E	Footswitch functional input	0x3833	RPM demand
0x387F	Interlock functional input	0x3835	Percentage demand %4096
0x3880	Speed limit #1 functional input	0x3846	RPM speed limit
0x3881	Speed limit #2 functional input	0x3847	Percentage speed limit &4096
0x3882	Speed limit #3 functional input	0x3848	Percentage torque limit
0x3883	Advanced mode #1 functional input	0x6040	DS402 ControlWord
0x3884	Advanced mode #2 functional input	0x6071	DS402 torque demand
0x3885	Handbrake functional input	0x6072	DS402 torque limit
0x3886	Inching forward functional input	0x6080	DS402 speed limit
0x3887	Inching reverse functional input	0x60FF	DS402 speed demand
0x3888	Digital footbrake functional input		

5.2 Safety functions

The Sigma2N and SigmaLITE controllers embed two CAN Safety function, namely Safe Torque Off (STO) and SafeStop1 (SS1). Those functions are triggered by means of digital inputs.
On the Sigma2N controller family, the said functions are assigned, as default, respectively to DI4 and DI3 (namely pins A4 and A3 on 35-way Ampseal connector A [2], respectively). Starting from HW version V07.03, the DI4 (namely pin A4 on 35-way Ampseal connector A [2]) can reach safety level PLd and the activating input for the two functions can be freely selected so that one of those function can be activated with PLd safety level, while the other can be at PLc level.
On the SigmaLITE controller family, the functional input triggering STO and SS1 can be freely assigned to any digital input. The safety level reached by those function, independently to the pins they are assigned to, is PLd.
This paragraph will now describe the STO and SS1 functions, which activation and parametrization can be done in "Menu 7.5 "Safe Stop 1" / "CAN Safety"".

5.2.1 Safe Torque Off (STO)

The Safe Torque Off function allows the user to immediately remove power from the motor and from the whole controller on an hardware wired signal. This function correspond to a stop function of category 0 according to EN60204-1/2006 Safety of ([11]).

When the Safe Torque Off function is triggered, the following happens:

1- Power from the motor is immediately removed by SW.
2- The mosfet drivers are deactivated by HW (only with PLd safety level configuration).
3- All controller's digital outputs are immediately deactivated by SW (i.e. Line Contactor opens, EMbrake is applied, etc..).
4- All controller's digital outputs are immediately deactivated by HW (i.e. Line Contactor opens, EMbrake is applied, etc..) (only with PLd safety level configuration).

WARNING!

If an STO function is triggered, the IPM overvoltage protection by means of motor short circuit can not be applied. Therefore, the user must take care of not running IPM motors at too high speed or performing appropriate risk calculations.

STO on Sigma2N controllers

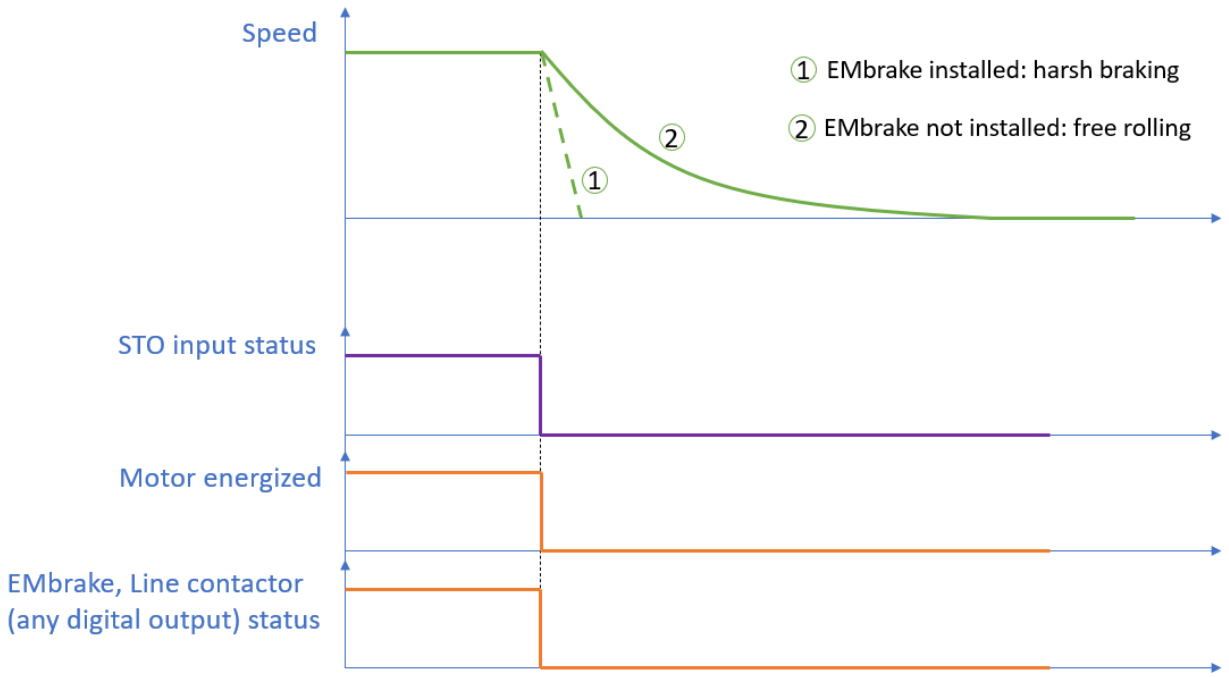
This function is active by default when Control Via CAN is activated and cannot be deactivated by any means. By default, this function is triggered by DI4 and its safety level is PLc.
Starting from HW version V07.03, the user can select whether this function is activated trough DI3 or DI4 (respectively Pin A3 and A4 of the 35-way connector A). In case this function is assigned to DI4, then its safety level can be raised to PLd level by means of parameter "M7.5-2 Safe Torque Off safety configuration "STOsafe"".
The selected input must be "active" (check the Digital Input active H/L configuration in [2]) at all time once the function is "armed" (depending on "M7.5-2 Safe Torque Off safety configuration "STOsafe""). Deactivating the input leads to triggering the Safe Torque Off function.

! IMPORTANT: Schematics reported in section "9 Wiring examples" refer to digital input option "Active L/H" selected for Active Low input.

STO on SigmaLITE controllers

This function is activated by assigning either both the "STO Normally Open" and "STO Normally Closed" or the "STO toggle" functional input to any digital input. The safety level reached by the function, independently from the assignment choice, will be PLd cat III.
In case the function is performed trough "STO Normally Open" and "STO Normally Closed" functional inputs, the status of the said inputs has to be consistent, meaning that they inputs must be either OPEN-CLOSE (STO ready) or CLOSE-OPEN (STO triggered). In case the inputs are not consistent, a wireoff fault will be triggered.

The image below represents an example of STO behaviour (in case the STO function is triggered by a single input):



5.2.2 Safe Stop 1 (SS1)

The SafeStop1 function allows, on an hardware wired signal, to perform a controlled deceleration of the motor before removing power from the motor and from the whole controller. It is a basically controlled deceleration of the motor followed by an STO. This function correspond to a stop function of category 1 according to EN60204-1/2006 Safety of ([11]).
For extra safety, a configurable timer is available to make sure that, in case the motor deceleration could not be followed, power from the motor itself and the whole controller is cut anyway.
When the SafeStop1 is triggered, the following happens:

1- The independent timeout counter defined by “M7.5-5 Safe Stop 1 cut timer “SS1 Time”” starts counting down.
2- A controlled deceleration is performed by the motor, using the parameters “M7.5-6 SafeStop1 torque “SS1 Torque”” and “M7.5-3 SafeStop1 ramp time “SS1 RmpTm””.
3- When either the motor is at 0 speed OR the timer has elapsed:
    a. Power from the motor is immediately removed by SW.
    b. The mosfet drivers are deactivated by HW (only with PLd safety level configuration).
    c. All controller’s digital outputs are immediately deactivated by SW (i.e. Line Contactor opens, EMbrake is applied, etc..).
    d. All controller’s digital outputs are immediately deactivated by HW (i.e. Line Contactor opens, EMbrake is applied, etc..) (only with PLd safety level configuration).

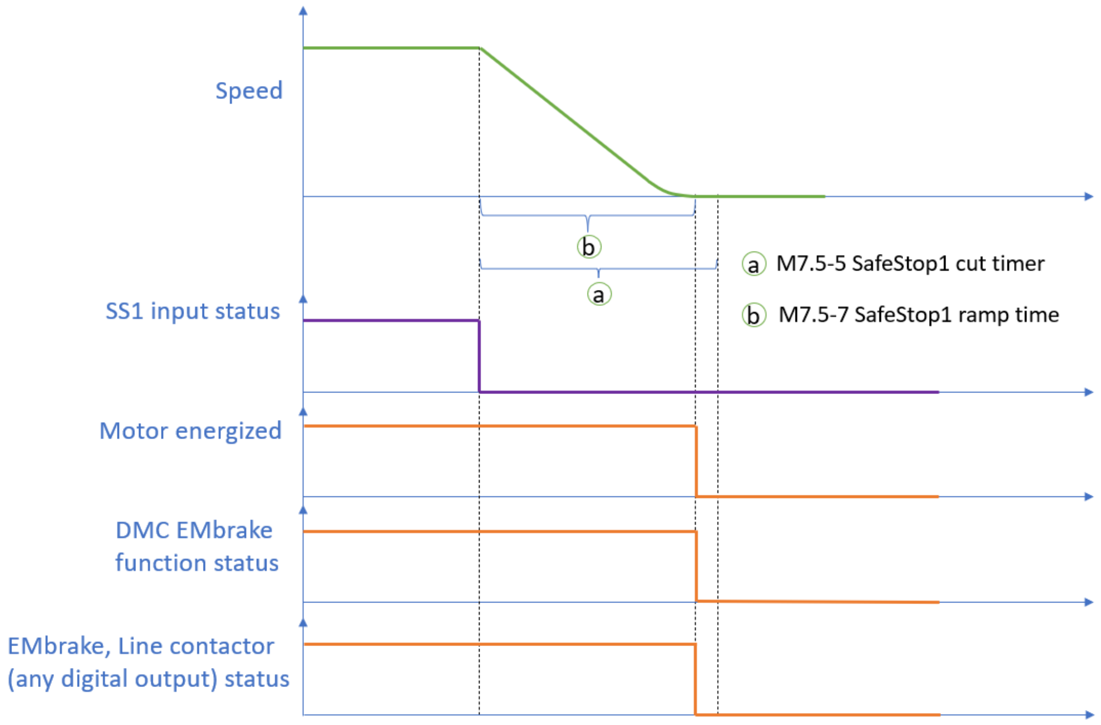
SS1 on Sigma2N controllers

This function can be activated by means of parameter “M7.5-3 SafeStop1 enable “SS1 Enab””, when also Control Via CAN is active. By default, this function is triggered by DI3 and its safety level is PLc.
Starting from HW version V07.03, the user can select whether this function is activated trough DI3 or DI4 (respectively Pin A3 and A4 of the 35-way connector A). In case this function is assigned to DI4, then its safety level can be raised to PLd level by means of parameter “M7.5-4 Safe Stop 1 safety configuration “SS1safe””.
The selected input must be “active” (check the Digital Input active H/L configuration in [2]) at all time once the function is “armed” (depending on “M7.5-4 Safe Stop 1 safety configuration “SS1safe””). Deactivating the input leads to triggering the SafStop1 function.

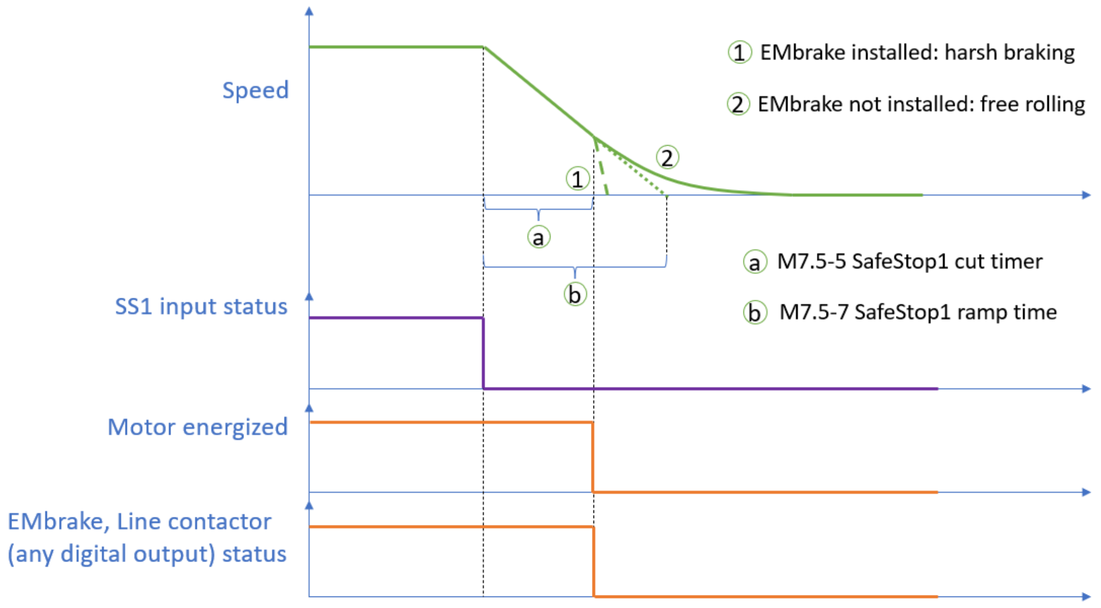
SS1 on SigmaLITE controllers

This function is activated by assigning either both the “SS1 Normally Open” and “SS1 Normally Closed” or the “SS1 toggle” functional input to any digital input. The safety level reached by the function, independently from the assignment choice, will be PLd cat III.
In case the function is performed trough “SS1 Normally Open” and “SS1 Normally Closed” functional inputs, the status of the said inputs has to be consistent, meaning that they inputs must be either OPEN-CLOSE (STO ready) or CLOSE-OPEN (STO triggered). In case the inputs are not consistent, a wireoff fault will be triggered.

The image below represents a possible SS1 behaviour (in case the SS1 function is triggered by a single input), where the motor stops with its ramp and the safety is provided by the recognition at 0 speed, before the SS1 timer even elapsed:


Speed
SS1 input status
Motor energized
DMC EMbrake function status
EMbrake, Line contactor (any digital output) status
b
a
a M7.5-5 SafeStop1 cut timer
b M7.5-7 SafeStop1 ramp time

Another possible situation is represented in the below image (still in case the SS1 function is triggered by a single input), where the motor does not manage to reach 0 speed, and it is therefore the safety timer that expires and “commands” power removal:


Speed
1 EMbrake installed: harsh braking
2 EMbrake not installed: free rolling
1
2
a
b
a M7.5-5 SafeStop1 cut timer
b M7.5-7 SafeStop1 ramp time
SS1 input status
Motor energized
EMbrake, Line contactor (any digital output) status

Please notice that in the above images the motor is supposed to be controller in speed mode for ease or representation. To this means, also notice that in the second image, the fine-dashed line represent the theoretical ramp that the motor would have followed in case the timer expiration didn't happen.
For completeness of speed mode, it is worth specifying that:

- The behaviour in speed mode is dependent upon the selection of parameter "M1.12-1 Allow drive torque during braking", which specifies whether the controller could also provide drive torque during the motor deceleration to strictly follow the set ramp, or whether the drive torque is inhibited during braking, thus the motor could naturally stop in a shorter time. Please refer to the said parameter description in [2].
- The End Of Braking function naturally smooths the motor deceleration at its end. So the configured ramp time in speed mode will always results elongated depending on the End Of Braking configuration.

In case the motor was controlled in torque mode, it would not be decelerating following a ramp, but instead with a variable time, depending on the applied torque (configurable by means of "M7.5-6 SafeStop1 torque "SS1 Torque"").

5.3 General function description

5.3.1 “Setpoint” and “Remote Input” CAN control types

This paragraph will briefly explain the distinction between the two types of “CAN control types” that can be adopted to drive a DMC Sigma2N or SigmaLITE controller via CAN Open. Some general knowledge about them has already been given while describing parameter “M7.2-2 RxPDO premaped configuration “RxPDOmap”” (for pre-mapping situation).
From a general point of view, the main distinction between “Remote Input” and “Setpoint” mode can be described as:

- **Remote Input** mode aims at copying a situation in which the Sigma2N or SigmaLITE controller is used wired. To this means, the Controller expects to have as inputs several digital inputs (FootSwitch, FW direction bit, RV direction bit, Seatswitch signal etc..) and several analogue inputs (Accelerator, brake pot, etc..). In Remote input mode the inputs are not given physically to controller, but they are sent Via CAN.
- **Setpoint** mode aims at hugely simplifying the drive function, as it only requires a speed or torque setpoint to drive the motor. The Sigma2N or SigmaLITE controller will only aim at reaching that setpoint, without any “drive safety” provided by the Footswitch and Seatswitch (just to give an example). Setpoint mode is commonly used to command actuators or pumps.
  A note must be made for speed setpoint mode, as the behaviour of the controller changes according to the mapping of the “HMI power enable” variable:
  - If the “HMI power enable” variable is <u>not</u> mapped to any RxPDO, then the motor will be energized only when any speed demand different than 0 is requested. When the motor is spinning (i.e. with a 100RPM demand) and then 0-speed is requested, the controller will perform a “neutral brake” operation, removing power from the motor once 0 speed is reached. In this way is it therefore not possible to achieve 0-speed control.
  - If the “HMI power enable” variable is mapped to any RxPDO, then the motor will be energized only when this bit is set to 1, no matter on the requested demand. Therefore the said bit must be set to 1 for driving the motor (i.e. any speed demand is discarded when “HMI power enable” bit is set to 0). In this scenario, the motor will be energized at any time in which the “HMI power enable” bit is set, making it possible to request 0-speed setpoints. If fact when the motor is spinning (i.e. with a 100RPM demand) and then 0-speed is requested, if “HMI power enable” bit is set to 1, a simple “deceleration” operation is performed and then the motor is kept energized and in control at 0 speed. Instead, if the motor is spinning and the “HMI power enable” bit is set to 0, a “neutral brake” operation will be performed and when 0 speed will be reached power will be removed from the motor.

  When the Sigma2N or SigmaLITE controller is controlled trough “Setpoint mode”, different ramp times are used according to the requested speed/torque set point:
  - If the speed/torque setpoint is changed from positive to negative and vice versa, direction brake ramp is used.
  - If the torque setpoint is zeroed, neutral brake ramp is used.
  - If the speed setpoint is zeroed and “HMI power enable” bit is not mapped, neutral brake ramp is used.
  - If the speed setpoint is zeroed and “HMI power enable” bit is mapped and set to 1, deceleration ramp is used.
  - If the motor is spinning and “HMI power enable” bit is mapped and set to 0, neutral brake ramp is used.
  - If the speed/torque setpoint is increased/decreased, acceleration/deceleration is used.

When the Sigma2N or SigmaLITE controller is controlled trough “Setpoint mode”, ramp times have to be set according to the desired responsiveness to the speed/torque set point.
If the speed/torque setpoint is changed from positive to negative and viceversa, direction brake ramp is used.
If the speed/torque setpoint is zeroed, neutral brake ramp is used.
If the speed/torque setpoint is increased/decreased, acceleration/deceleration is used.

When the DMC Sigma2N or SigmaLITE controller is used with “pre-mapped” PDOs (i.e. parameter “M7.2-1 PDO free mapping type “FreeMapT”” set to 0), the Control type selection is performed simply by means of parameter “M7.2-2 RxPDO premaped configuration “RxPDOmap””.
On the other hand, when the DMC Sigma2N or SigmaLITE controller is used in “free-mapping” mode, the Control Type selection is performed automatically by the Controller, based on the variables mapped by the user in the RxPDOs.
If the user maps, for example, any “setpoint” variable (i.e. objects 0x3833, 0x3835, 0x6071 or 0x60FF, see [7] for their descriptions), the Controller will automatically use “Setpoint” Can control type.

If the user maps, for example, any “remote input” variable (i.e. objects 0x3800, 0x3801, 0x3802, 0x3806 or 0x3808 to 0x380F, see [7] for their descriptions), the Controller will automatically use “Remote Input” CAN control type.
Please notice that in [7] a compatibility table of mappable variables is reported. In fact, the Controller at each powerup will check if any incompatible variable coexist in the mapping, in order to avoid uncertainties on the CAN control type to adopt. In case any incompatibility is found in the mapping, an error is thrown and the drive functions are disabled.

5.3.2 “Mixed inputs” control type

This paragraph enhances the description of the previously introduced “Remote Input” CAN control type, by introducing the “Mixed inputs” control type.
This feature, only available when using “free mapped” RxPDO configuration, allows to mix the source of the analogue and digital inputs of the Controller. This means that several inputs can be sent Via CAN, while others can be wired physically to the DMC Sigma2N or SigmaLITE controller.
To enable this control type, just map a restricted set of “Remote input” variables (i.e. objects 0x3800, 0x3801, 0x3802, 0x3806 or 0x3808 to 0x380F, see [7] for their descriptions) in RxPDOs and provide the remaining inputs by physically wiring them.
For Traction Controllers, a minimum set of FW (or RV), FS, ST and Accelerator must be provided in order to run the motor.
For pump Controllers, a minimum set of SPD1+Accelerator OR any SPDx must be provided in order to run the motor.
It is up to the user to decide which signal to wire and which signal to provide Via CAN.
Please notice that if a signal is both mapped in a RxPDO and wired physically, the priority is given to the value coming Via CAN (i.e. the physical value is not taken into account).

5.3.3 Boot up and Network management

When “Control Via CAN Open” function is active (i.e. “M7.2-1 PDO free mapping type “FreeMapT”” set to 0 and“M7.2-3 PDOs configuration “PDO sel”” is set lower than 3, OR “M7.2-1 PDO free mapping type “FreeMapT”” set to 1 or 2 and some variable is mapped in enabled RxPDOs), after boot-up the DMC Sigma2N or SigmaLITE controller will enter a state called “pre-operational”, as specified by the CAN Open standard (see [5]).
It will be up to the Can Open Master (COM), trough NMT messages (ID 0x000), to set the Sigma2N Controller to “operational” or “stopped”.
If the “Control Via CAN Open” function is active, regardless of the state in which the DMC Sigma2N or SigmaLITE controller is in, it will always send an Heartbeat message (ID 0x70n, with n the node number) for displaying the node status (operational, preoperational or stopped).
Possible status transitions are:

- “Pre-operational” -> “Stopped”
- “Pre-operational” -> “Operational”
- “Operational -> “Pre-operational”
- “Operational -> “Stopper”
- “Stopped” -> “Pre-operational”
- “Stopped” -> “Operational”

The following table resumes the messages that the DMC Sigma2N or SigmaLITE controller can send/receive in each status.

	Stopped	Pre-operational	Operational
PDO (Tx and Rx)	✗	✗	✓
SDO (Tx and Rx)	✗	✓	✓
SYNC	✗	✓	✓
EMCY	✗	✓	✓
NMT	✓	✓	✓
Heartbeat	✓ (Tx only)	✓	✓

The Node Status transition and definition are compliant to Can Open Standard DS301 (see [5] for details).
For information about the NMT and HEARTBEAT messages format and content refer to [1].

! VERY IMPORTANT According to the table reported above, the DMC Sigma2N and SigmaLITE controllers are able to receive (and process) RxPDOs only in “operational” state. It is therefore up to the Can Open Master (COM) to set the nodes to “operational” state for being able to drive the motors.

! VERY IMPORTANT Each time a DMC controller is set to “Operational” status, the VCU must send the first message containing 0 demand. Basically the first message must be sent empty, otherwise F12 S15 is signalled.

5.3.4 Pre-mapped Receive (Rx) PDOs

When the “Control Via CAN Open” function is active and “M7.2-1 PDO free mapping type “FreeMapT”” is set to 0, the DMC Sigma2N or SigmaLITE controller is capable of receiving several pre-mapped RxPDOs. According to parameter “M7.2-2 RxPDO premaped configuration “RxPDOmap””, in both Traction and Pump applications two types of Control type can be enabled: “Remote Input” mode and “Setpoint” mode.

With “**Remote input**” mode RxPDO1 should contain the standard digital and analogue input signals, further than Speed or Torque limits.
For example, in traction applications, the accelerator, footbrake and direction inputs must be sent to DMC Controller.
For traction software, the motor can be controlled either in speed or torque mode.
For pump software, the motor can be controlled in speed mode only.
All standard drive functions of DMC Sigma2N and SigmaLITE controllers are available (see [2]) when using “Remote Input” mode.

With “**Setpoint**” mode RxPDO1 should contain a speed setpoint in RPM or a torque setpoint in % (depending on the control mode selected), further than Speed or Torque limits.
For traction software, the motor can be controlled either in speed or torque mode:

- If “**speed mode**” is selected, the RPM speed demand is positive for forward driving and negative for reverse driving.
- If “**torque mode**” is selected, with the % torque demand it is possible to ask positive torque for driving forward (or braking in reverse driving) and negative torque for driving reverse (or braking in forward driving).
  Please notice that in this situation, parameter “M8-16 Regenerative BCL enabled during Direction Braking “BCLRegDB”” has a big impact. If that parameter is set to 1 the controller will in fact have the braking action always limited by the regenerative BCL.
  Example:
  The motor is running FW with positive torque request. Parameter “BCLRegDB” is set to 1 (BCL regen active when Direction Braking). If the user asks for negative torque (for braking the motor), the Controller performs a “Direction Brake” operation. If the regenerative BCL is set low enough to cut the braking torque, this will be cut and the requested braking action will not be performed.
  On the other hand, if “BCLRegDB” is et to 0 (BCL regen inactive when Direction Braking), the motor will always be able to brake with the maximum torque available, but the active regenerative BCL will not be considered and the battery will have to absorb the generated power.
  For pump software, the motor can be controlled in speed mode only:
  The RPM speed setpoint can only be positive, corresponding to forward drive. This means that braking torque is never applied to the motor.

5.4 Third party VCU development

5.4.1 General suggestions

Here below are some suggestion for the VCU software development:

- VCU should act as the CAN Open Network Master. It should therefore send the NMT messages to all DMC Controllers to set them in the proper state.
- VCU should start sending NMT messages after a power up time is expired.
  Controller and VCU may have different hardware power up time and user must make sure they are both properly started up before starting the communication.
  Important: Controllers don't have any "first message time out" implemented, thus VCU can wait several seconds before start sending NMT and Drive Commands (RxPDOs).
  Important: before to send the first Drive Command (RxPDO) to a controller, make sure that the digital inputs meant to trigger the STO and SS1 functions (On Sigma2N DI3 or DI4, according to HW version and configuration setup of "M7.5-1 Digital input 3-4 configuration "DI34cfg"*", see "*Safety functions*" for details) are closed.*
- VCU should start sending Drive Commands (RxPDOs) to slave controllers only after "operational" state is detected (through HEARTBEAT message).
- VCU should send the first Drive Commands (RxPDOs) empty (i.e. containing 0 demand, be it a speed, torque or percentage demand and, if "Remote Input mode" is used, all drive inputs signals to 0).
- VCU must report the error if CAN communication error occurs (see later chapter for error description) and cut down the "Safe Torque Off input" (On Sigma2N controller, Digital input 4 - HW Enable Signal, see "5 Hardware installation – Safe Torque Off"; On SigmaLITE controller, inputs depends on active configuration).
  VCU can detect the CAN communication failure by checking the roll-over counter and timeout function.

WARNING!

IT IS UP TO VCU PROGRAMMER, ACCORDING TO THE HARDWARE and SOFTWARE AVAILABLE IN ITS OWN VCU, TO CHOOSE IN WICH WAY TO IMPLEMENT SAFETY AND LOOP COMMUNICATION TIMING.
THE ABOVE SUGGESTEST ACTION ARE GIVEN CONSIDERING A SINGLE TASK LOOP SOFTWARE.

5.4.2 General remarks

Hereafter some important remarks are reported:

- DMC Sigma2N or SigmaLITE controller will start processing RxPDOs content only after the reception of at least one per kind of each active RxPDO with mapped content. This means, for example, that if 3 RxPDOs (RxPDO1, RxPDO2 and RxPDO3) are mapped with content, they will not be processed until both RxPDO1, RxPDO2 and RxPDO3 are received at least once.
- Everytime the DMC Sigma2N or SigmaLITE controller exits "Opertional" state (see "2.2.2 Network management") or "OperationEnabled" state (see "5.7 Device management with Control Word (Object 0x6040)"), the above condition is reset.
- Once the first message of each RxPDO active with mapped content is received, in order to drive the motor, the "Drive Commands" must be received to 0 at least once. This applies to the objects which are used to perform the Control via CAN function (see "5 Control via CAN function" for the full list). [7] for object descriptions.

5.4.3 Loop cycle implementation (for Pre-mapped operation mode)

The VCU designer has to respect few basic advices in order to send proper Drive Commands (RxPDO1) to the controllers:

1. Wait that power up timer elapses: we advise a time of at least 2s;
2. Depending on used controller:
    - On Sigma2N, "close" the STO input and, if the function active, the SS1 input (the input corresponding to those functions is dependent on "M7.5-1 Digital input 3-4 configuration "DI34cfg"") of all controllers ("close" to ground or battery plus according to "M2-8 (for traction) or M2-2 (for pump) Active High Low "Actv L/H"" setting in "Menu 2 – Controller setup", see [2]).
    - On SigmaLITE, depending on the input configuration performed in "Menu 8 – I/O configuration", see [3], make sure the STO and SS1 functions are not triggered, by having the "Normally Open" and "Normally Closed" inputs in the "active" position (i.e. NO switch -> Open; NC switch Closed).
3. If an HEARTBEAT message with "pre-operational" information from a slave node is detected, VCU could set that node into "operational" status through NMT message.
4. Check for slave nodes' HEARTBEAT messages;
5. Repeat step 4) for all slave nodes in the network. See remark below.
6. Update the Drive Commands (RxPDO1) for all slave nodes (also increasing the roll-over counters);
7. **Send an empty Drive Command** (RxPDO1) to the first slave node and start timeout timer;
8. Wait to receive Drive Status (TxPDO2) from the first slave node;
9. If the Drive Status (TxPDO2) is received from the Slave node within timeout timer, check the roll-over counter;
10. If the Drive Status is failed to receive or roll over counter does not match (see "2.4.2 Roll-over counter function"), report error and, depending on the used controller,
    - On Sigma2N, "open" the STO input or, if the function active, the SS1 input (the input corresponding to those functions is dependent on "M7.5-1 Digital input 3-4 configuration "DI34cfg"") of all controllers.
    - On SigmaLITE, depending on the input configuration performed in "Menu 8 – I/O configuration", see [3], make sure one between STO and SS1 functions is triggered, by having the "Normally Open" and "Normally Closed" inputs in the "inactive" position (i.e. NO switch -> Closed; NC switch Open).

    otherwise see next point;
11. Move to next Slave node and send Drive Command (RxPDO1) to it. Repeat cyclically from 7 to 10 until one loop is finished.
12. Send the actual desired Drive Command (RxPDO1) to the first slave node and start timeout timer;
13. Wait to receive Drive Status (TxPDO2) from the first slave node;
14. If the Drive Status (TxPDO2) is received from the Slave node within timeout timer, check the roll-over counter;
15. If the Drive Status is failed to receive or roll over counter does not match (see "2.4.2 Roll-over counter function"), report error and cut down the "Safe Torque Off input" (Digital input 4 - HW Enable Signal, see "5 Hardware installation – Safe Torque Off"), otherwise see next point;
16. Move to next Slave node and send Drive Command (RxPDO1) to it. Repeat cyclically from 12 to 15 until one loop is finished.
17. Wait loop time is expired.
18. Start a new "sending message loop" (Go to point 6 of this list).

Remark: It is not compulsory to set ALL slave nodes to "operational" status at power up. Only the slave that have to be commanded should be put into "operational" status. If, at any time, a slave does not need to be commanded, it could be put back into "pre-operational" status. To command that slave node again, it should be put back to "operational" status (and again an empty first message must be sent to it).
If a slave node has been set to "operational" status and a first Drive Command (RxPDO) has been sent to it, that slave will start an internal timeout function. If a new message in not received within the timeout time, that node will report an error. If a slave node is set back to pre-operational, it will stop checking for Drive Commands (RxPDO) and the timeout function will be disabled. The function will be re-enabled when the slave node is set to "operational" status and a Drive Command (RxPDO) will be sent to him again. See "2.4.1 Timeout function" for more information.

5.4.4 Power up and communication sequence example (for Pre-mapped operation mode)

Here is an example of power up sequence and Drive Command messages timing for 3 controllers (node 1, 2, 3) managed Via CAN.

Power up sequence:

1) Power up the controllers by "key input";
2) Depending on used controller:
   a. On Sigma2N, "close" the STO input and, if the function active, the SS1 input (the input corresponding to those functions is dependent on "M7.5-1 Digital input 3-4 configuration "DI34cfg"") of all controllers ("close" to ground or battery plus according to "M2-8 (for traction) or M2-2 (for pump) Active High Low "Actv L/H"" setting in "Menu 2 – Controller setup", see [2]).
   b. On SigmaLITE, depending on the input configuration performed in "Menu 8 – I/O configuration", see [3], make sure the STO and SS1 functions are not triggered, by having the "Normally Open" and "Normally Closed" inputs in the "active" position (i.e. NO switch -> Open; NC switch Closed).
3) Wait 1s;
4) Send NMT message with "go to operational" command to node 1.
5) Check that node 1 is in "operational" status trough HEARTBEAT message.
6) Send NMT message with "go to operational" command to node 2.
7) Check that node 2 is in "operational" status trough HEARTBEAT message.
8) Send NMT message with "go to operational" command to node 3.
9) Check that node 3 is in "operational" status trough HEARTBEAT message.
10) VCU can start sending Drive Command messages to drive/operate (recall that first message to each controller must be sent empty).

NOTICE!

- Step 2 must be performed before step 10, otherwise **F29 S7** occurs.
- There is no time out for performing step 10.
- Step 2 can be also be performed before of step 1.

Communication sequence normal operation:

The communication sequence and timing in VCU strongly depends on HW and SW language type of VCU itself. Here is a simple example for a basic implementation on VCU using a polling single task to communicate to the DMC controller:

1) Send RxPDO1 message to node1;
2) Check TxPDO2 answer from node 1;
3) Send RxPDO1 message to node;
4) Check TxPDO2 answer from node 2;
5) Send RxPDO1 message to node 3;
6) Check TxPDO2 answer from node 3;
7) Repeat from 1) after a "loop time" ("loop time" shorter than "CAN_TO") is elapsed, increasing the roll-over counter in each RxPDO messages before sending it.

NOTICE!

- The controller must receive the Drive Command (RxPDO1) message within "M7.2-9 RxPDO1 event timer "RxPDO1TO"" setting, thus the refresh rate of VCU for transmitting the Drive Command (RxPDO1) message has to be lower than that time.
  For example if "CAN_TO" = 150ms, keep refreshing Drive Command (RxPDO1) message at least every 100ms.
- The controller sends out the Drive Status (TxPDO2) message as an "answer" to each Drive Command (RxPDO1) message received, within 2ms. This means that the refresh rate of the Drive Status (TxPDO2) message depends on the Drive Command (RxPDO1) message's refresh rate.

- If a multi task VCU is available, the Drive Command (RxPDO1) messages can be sent in different task with different refresh time. It is important that each controller receives a Drive Command (RxPDO1) message with a refresh rate consistent with its own time out “CAN_TO”.

5.5 Slave controller(s) information

5.5.1 Slave controller(s) features (for Pre-mapped operation mode)

- Slave controllers can be chosen to receive Drive Command (RxPDO1) via CAN or not. Slave controller can choose which status information (TxPDO1 and/or TxPDO3) to be sent back to VCU.
- If a controller is not set to use Control Via CAN Open, it works as a normal controller and receives Drive Commands Via analogue and digital wired inputs. If it is set to be using Control Via CAN Open, it sends Drive Status (TxPDO2) information back to VCU as soon as it receives the Drive Command (RxPDO1) as acknowledge message.
- Controllers can be chosen to send other status information back to VCU periodically according to their settings.
- Slave controllers report error if CAN communication error occurs.
  Slave can detect the CAN communication failure by checking the roll-over counter and set timeout function.
  Slave can be stopped if the VCU cut down the physical switch in order to stop reading Drive Command via CAN.

5.5.2 Slave controller(s) behaviour (for Pre-mapped operation mode)

The controllers configured to be driven Via CAN by the VCU will follow these rules:

- Start timeout timer when they receive the very first Drive Command (RxPDO1);
- If they receive a Drive Command (RxPDO1) within timeout timer, check the roll-over counter;
- Copy the Drive Command’s (RxPDO1) roll-over counter into Drive Status (TxPDO2) and send it to VCU, together with drive information;
- Extract the Drive Command (RxPDO1) and assign it to functional inputs for traction or pump;
- Send Controller/Motor&Battery Status back to VCU if it is required.

5.6 Differences between Traction and Pump application software

The Control Via CAN Open is available for both traction and pump application software. The Drive Command signals in the RxPDO1 message assume different meaning depending on the application software installed in the DMC Sigma2N or SigmaLITE controller and on the CAN Control Type selected (see "M7.2-2 RxPDO premaped configuration "RxPDOmap"").
In the message data format sheet ([1]) the first meaning is for traction application software and the meaning indicated between brackets in italics is for pump.

VERY IMPORTANT!

In case objects corresponding to "fictious" copies of digital inputs are mapped (i.e. objects 0x3806, 0x3808, 0x3809, 0x380A, 0x380B, 0x380C, 0x380D, 0x380E), they must be treated as "physical" digital inputs.
For example, in Sigma2N controller, if a physical digital input is Normally Open, to make it active, 0 must be written in the corresponding bit of the "HMI digital input word" (object 0x3806). An example of this could be the Speed Limits 1 and 2 in Traction Control: if speed limits have to be excluded bits 5 and 6 in "HMI digital input word" (object 0x3806) have to be both set to 1.
Also in pump control, if some digital input are programmed to be Normally Open, the meaning in the "HMI digital input word" (object 0x3806) is:

1 = input not active;
0 = input active.

In the SigmaLITE controller, the same has to be applied to objects 0x3880, 0x3881, 0x3882, 0x3883, 0x3884 and to the bits of object 0x387A and 0x387B corresponding to "Speed limit #1, #2 and #3".

5.6.1 Notes on "HMI digital input word" or "Preset byte #1" for Traction

Digital inputs in "HMI digital input word" correspond to controller physical inputs. They work as the physical input signals. For details please refer to [2].
At any rate, a table with meanings is reported below.

	"HMI digital input word" for Traction						
Bit	0	1	2	3	4	5	6
Meaning	DI1 (forward direction)	DI2 (reverse direction)	DI3 (footswitch)	DI4 (Seatswitch / Enable)	DI5 (Speed Limit 1)	DI6 (Speed Limit 2)	DI7 (Handbrake / Speed Limit 3)

Drive Forward or Reverse

Commands sequence:

1) Set Bit 3 (Input4 - Seat/Enable (seat switch)) to 1, keeping Bits 0, 1 and 2 (Inputs 1, 2 and 3 – Forward, Reverse and Footswitch) to zero.
2) Select direction (Forward or Reverse) be means of Bits 0 and 1.
3) Set Bit 2 (Input 3 – FootSwitch) to 1 and increase throttle signal.

Step 2 and 3 can be swapped. It is important the first activation of Bit 3 (Seat switch/Enable) must be done with Bits 0, 1 and 2 (Inputs 1, 2 and 3) set to zero. If this sequence is not respected, F12 failure occurs; to reset it there is no need to perform a key power off on cycle: just reset the bits and start properly from step 1.

Input 5 and 6 (bit 4 and 5) - Speed limiting signals

Input 5 and 6 are used for limiting speed. They are “Normally Closed” logic. So if Input 5 or 6 bit (bit 4 and 5) are set to 0, the speed limit is active.
Here is the table for proper use of these limit signals:

Input 5 (bit 4)	Input 6 (bit 5)	Active Speed Limit
1	1	Max. speed FW or Max speed RV
0	1	Speed Limit 1
1	0	Speed Limit 2
0	0	Lowest between Speed Limit 1 and Speed Limit 2

In general, if more than one speed limit is active at the same time, the lowest value one takes precedence.

Input 7 (bit 6) - Speed Limit 3 / Handbrake

The configuration of this parameter depends on “M3-6T – I/O Pin 7 “Spd3/Hbk””.

If input 7 is configured as Handbrake is it a Normally Open input:
Bit 6 = 0 handbrake released (no limit);
Bit 6 = 1 handbrake activated (Handbrake torque and speed limit);
If Input 7 is configured as Speed limit 3 is a Normally Closed input:
Bit 6 = 0 Speed limit 3 active
Bit 6 = 1 Speed limit 3 disabled
In general, if more than one speed limit is active at the same time the lowest value one takes precedence.

5.6.2 Notes on “HMI digital input word” and “Preset byte #1” for Pump

Digital inputs in “HMI digital input word” correspond to controller physical inputs. They work as the physical input signals. For details please refer to [2].
At any rate, a table with meanings is reported below.

	“HMI digital input word” for Pump						
Bit	0	1	2	3	4	5	6
Meaning	DI1 (Pump pot / switch 1)	DI2 (Pump switch 2)	DI3 (Pump switch 3)	DI4 (Pump switch 4)	DI5 (Pump switch 5)	DI6 (Pump switch 6)	DI7 (Pump inhibit)

Pump pot/switch 1

Input 1 is active or not according to setting “M3-5P Enable pot with switch “NoSw/Sw”” in “Menu 3 - Controller setup” (refer to [2] or [3]).

Pump switch 6 / powersteer

Input 6 (adopted for power steer) can be configured as Normally Open or Normally closed according to “M3-2P Speed 6 input normally closed or normally open “Sp6NO/NC”” (refer to [2] or [3]).

- If set to **NC**, if input (bit 5) is set to 0 the pump runs at Speed 6, if input (bit 5) is set to 1 the pump is not running.
- If set to **NO**, if input (bit 5) is set to 1 the pump runs at Speed 6, if input (bit 5) is set to 0 the pump is not running.

Pump inhibit

Input 7 (adopted for inhibit) can be configured as Normally Open or Normally closed according to “M3-3P Inhibit input normally closed or normally open “HibNO/NC”” (refer to [2] or [3]).

- If set to **NC**, if input (bit 6) is set to 0 the pump operation is inhibited, if input (bit 6) is set to 1 the pump can run.
- If set to **NO**, if input (bit 6) is set to 1 the pump operation is inhibited, if input (bit 6) is set to 0 the pump can run

5.6.3 Notes on RPM setpoint and speed limit signals for Traction

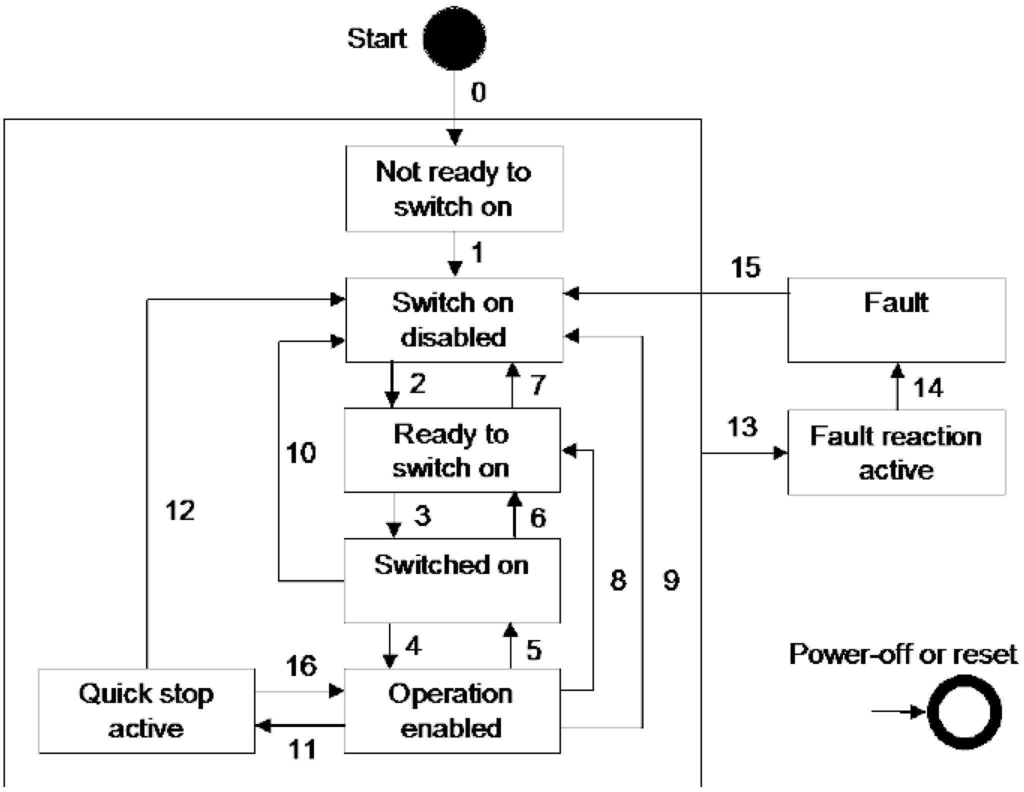
Regardless of the RPM setpoint sent by means of objects 0x3833 or 0x60FF and regardless of RPM speed limits sent by means of objects 0x3846 or 0x6080, the Adjustment “M7.1-2 Maximum speed forward “SpdMaxF”” and Adjustment “M7.1-3 Maximum speed reverse “SpdMaxR”” will act as static speed limitation for forward and reverse direction. At any case the motor will never spin faster than “M3-13 Maximum desired motor frequency “Fmotmax*”” selected in “ Menu 4 – Autotuning).
Other Speed Limits (Speed 1, 2 and 3) are not active.
Ramp adjustments are active and have to be set according to desired responsiveness to the setpoint variation.
For details of cited adjustments and parameters refer to [3] and [4].

5.6.4 Notes on RPM setpoint and speed limit signals for Pump

Regardless of the RPM setpoint sent by means of objects 0x3833 or 0x60FF and regardless of RPM speed limits sent by means of objects 0x3846 or 0x6080, set Adjustment “M1-5 Pump Speed 1 “PotMax1”” to the maximum speed you want run. At any case the motor will never spin faster than “M4-13 Maximum desired motor frequency “Fmotmax*”” selected in “ Menu 5 – Autotuning).
Other Speed settings (Speed 2, 3, 4, 5 and 6) are not active.
Ramp adjustments are active and have to be set according to desired responsiveness to the setpoint variation.
For details of cited adjustments and parameters refer to [3] and [4].

5.7 Device management with Control Word (Object 0x6040)

All DMC Sigma2N and SigmaLITE controllers implement the standard CiA DS402 application profile. This consist of a standardized way to manage the Controller functions, that enables having a high safety level and allows the user to have a complete control on the Drive operations at any time.
This management is performed by means of the finite state machine reported below:



The functions are managed in each state according to the table below:

	Not ready to switch on	Switch on disabled	Ready to switch on	Switched on	Operation enabled	Quick stop active	Fault reaction active	Fault
EM brake applied	Yes	Yes	Yes	Yes	Yes/No	Yes/No	Yes/No	Yes
Low-level power applied	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
High level power applied	No	No	No	Yes	Yes	Yes	Yes	No
Drive enabled	No	No	No	No	Yes/No	Yes/No	Yes/No	No

If the ControlWord (Object 0x6040) is not mapped in any RxPDOs, then the Sigma2N or SigmaLITE controller will manage this state machine itself and the user should not take any action.
If the user wants to have control over the states transitions, then he should map the ControlWord into any RxPDO and manage the transitions autonomously.

The ControlWord is a 16 bit unsigned integer composed as follows:

15	11	10	9	8	7	6	4	3	2	1	0
ms		r	oms	h	fr	oms		eo	qs	ev	so

ms: manufacturer specific

r: reserved

oms: operation mode specific

h: halt

fr: fault reset

eo: eneble operation

qs: quick stop

ev: enable voltage

so: switch on

The commands to device can be given by setting the proper bits of the ControlWord according to the following table (an X indicates that any value is accepted):

Command	Bits of ControlWord					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Switch on + enable operation	0	1	1	1	1	3+4
Disable voltage	0	X	X	0	X	7, 9, 10, 12
Quick stop	0	X	0	1	X	7, 10, 11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4, 16
Fault reset	1	X	X	X	X	15

Notice that the “Halt” bit of the ControlWord does not determine a state transition, but a specific action is still taken. In fact, that bit is considered only when the drive is in “Operation Enabled” state and its setting/reset will not trigger a transition. The action taken when the “Halt” bit is set to 1 is described in “5.7.6 Halt option code”.

5.7.1 Transitions description

For completeness, a table describing the action taken during each transition is reported hereafter

Transition	Triggering event	Action performed
0	Automatic after power-on	Device initialization
1	Automatic after initialization ended	Communication enabled
2	SHUTDOWN command from VCU or local signal	Finalize precharge*
3	SWITCHON command from VCU or local signal	The Line Contactor is closed
4	ENABLEOPERATION command from VCU or local signal	The drive function is enabled
5	DISABLEOPERATION command from VCU or local signal	The drive function is disabled
6	SHUTDOWN command from VCU or local signal	The Line Contactor is opened
7	DISABLEVOLTAGE command from VCU or local signal	None
8	SHUTDOWN command from VCU or local signal	The drive function is disabled and the Line Contactor is opened
9	DISABLEVOLTAGE command from VCU or local signal	The drive function is disabled and the Line Contactor is opened
10	DISABLEVOLTAGE command from VCU or local signal	The Line Contactor is opened
11	QUICKSTOP command from VCU or local signal	The Quick stop function is performed
12	Quick-stop option code dependent	The drive function is disabled and the Line Contactor is opened
13	A fault happens	Configured fault reaction dependent
14	Automatic after fault signalling	The drive function is disabled and the Line Contactor is opened
15	FAULTRESET command from VCU or local signal	The fault is reset (if no other fault is present on the device)
16	Quick-stop option code dependent	The drive function is enabled

Notice*: the transition #2 is actuated only when the pre-charge of the device is finalized.

5.7.2 Abort connection option code

This object specifies the action taken by the device in case the communication with the VCU is interrupted (i.e. in case the node is set into stopped or pre-operational status).
In case this happens, DMC implements a manufacturer-specific action, consisting in forcing a “DISABLEOPERATION” command, so that the drives enters in a safe state, where power is applied but the drive function is inhibited. Refer to “5.7.5 Disable operation option code” for the description of the actions performed.

5.7.3 Quick stop option code

This object specifies the action taken by the device when the “QUICKSTOP” command is received.
In case this happens, DMC implements a standardized action consisting of slowing down on a deceleration ramp or with a defined braking torque (respectively in speed or torque mode), configurable in the “Drive Response” sub-menu or “Adjustments” menu, and then transiting into “SWITCH ON DISABLED” state.

5.7.4 Shutdown option code

This object specifies the action taken by the device when the “SHUTDOWN” command is received.
In case this happens, DMC implements a standardized action consisting of immediately removing power from motor (so it is free to rotate uncontrolled), opening the Line Contactor and then transiting into “SWITCH ON DISABLED” state.

5.7.5 Disable operation option code

This object specifies the action taken by the device when the “DISABLEOPERATION” command is received.
In case this happens, DMC implements a standardized action consisting of immediately removing power from motor (so it is free to rotate uncontrolled) and then transiting into “SWITCHED ON” state (when speed is zero).

5.7.6 Halt option code

This object specifies the action taken by the device when the “HALT” command is received.
In case this happens, DMC implements a standardized action consisting of slowing down on a deceleration ramp or with a defined braking torque (respectively in speed or torque mode), configurable in the “Drive Response” sub-menu or “Adjustments” menu, and then staying into “OPERATION ENABLED” state. When this bit is cleared, the drive will re-start accelerating the motor immediately with normal drive ramp/torque.

5.8 Device monitoring with StatusWord (Object 0x6041)

When using a DMC Sigma2N or SigmaLITE controller, the user might find useful to monitor its Drive state, especially if he has mapped the ControlWord and wants to properly manage the state transitions.
To this means, an object called "StatusWord" can be mapped in any TxPDO and it will provide information about the actual state of the Drive.
The StatusWord is a 16 bit unsigned integer composed as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ms		oms		ila	tr	rm	ms	w	sod	qs	ve	f	oe	so	rtso

ms: manufacturer specific

oms: operation mode specific

ila: internal limit active

tr: target reached

rm: remote

w: warning

sod: switch on disabled

qs: quick stop

ve: voltage enabled

f: fault

oe: operation enabled

so: switched on

rtso: ready to switch on

The status of the Drive can be retrieved from the StatusWord according to the following table (an X indicates that that value is not influent):

Status	Bits of StatusWord					
	Bit 6	Bit 5	Bit 3	Bit 2	Bit 1	Bit 0
Not ready to switch on	0	X	0	0	0	0
Switch on disabled	1	X	0	0	0	0
Ready to switch on	0	1	0	0	0	1
Switched on	0	1	0	0	1	1
Operation enabled	0	1	0	1	1	1
Quick stop active	0	0	0	1	1	1
Fault reaction active	0	X	1	1	1	1
Fault	0	X	1	0	0	0

Independently on the status of the Drive, the following bits provide further indication:

- Bit 4 indicates whether the Line Contactor is closed (1 if closed, 0 therwise);
- Bit 7 indicates a warning state (1 if the Drive has a fault between 15 and 19, 0 otherwise);
- Bit 9 indicated the reception of a ControlWord (1 if received, 0 otherwise);
- Bit 10 indicates that the Drive has reached the desired target (1 if reached, 0 otherwise);
- Bit 11 indicated that an internal limitation is active on the Drive (1 if active, 0 otherwise).

5.9 Miscellaneous failures - troubleshooting

5.9.1 VCU fails to send Drive Command (RxPDO1) to slave controller

Slave controller starts a timer after it receives the Drive Command (RxPDO1). The maximum time out time is set by parameter “M7.2-9 RxPDO1 event timer “RxPDO1TO””. If the slave controller fails to receive a Drive Command (RxPDO1) from VCU within that time, it reports error and stops working.
Note: controller doesn’t start timeout timer until it receives the very first Drive Command (RxPDO1) to prevent asynchronized power up between VCU and slave controller.
This will not cause safety problem, because vehicle stays still until it receive the first Drive Command (RxPDO1).

5.9.2 Slave controller fails to send Drive Status (TxPDO2) to VCU

VCU starts a timeout timer after it sends out Drive Command (RxPDO1) to a slave controller. If the VCU fails to receive the slave’s Dive Status (TxPDO2) within this timeout time, it has to cut down the “Safe Torque Off input” (Digital input 4 - HW Enable Signal, see “5 Hardware installation – Safe Torque Off”) and all the slave controllers will stop reading the Drive Commands, therefore resetting all the digital/analogue inputs and stop working. It is up to VCU programmer to implement this safety feature.

5.9.3 VCU sends corrupted data to slave controller

There is a roll-over counter in Drive Command (RxPDO1) message and a roll-over counter in Drive Status (TxPDO2) message.
VCU has to increase the roll-over counter in Drive Command (RxPDO1) at every new loop/message sending.
For example:
In first loop, VCU sends roll-over counter 0 to all the slave controllers.
In second loop, VCU sends roll-over counter 1 to all the slave controllers.
Controller slave N must check if the roll-over counter has increased **by one unit**, respect to the roll-over counter received in the previous Drive Command (RxPDO1) message. If not, slave controller reports error and stop working.
Note: Data corrupted can’t be detected in the first loop; if the loop time is shorter than 150ms, then problem will be detected within 150ms.

5.9.4 Slave controller sends corrupted data to VCU

Slave controller copy the roll-over counter received in Drive Command (RxPDO1) message and send it back in Drive Status message (TxPDO2).
For example:
VCU sends roll-over counter to 59 to slave controller #1.
Slave controller #1 copies the roll-over counter value and send Drive Status (TxPDO2) with roll-over counter to 59 to VCU.
VCU must check the consistency between the sent and received roll-over counter.

5.9.5 Hard fault occurs in slave controllers

If an hard fault occurs in a slave controller, it stops working even if it is still reading the “Safe Torque Off input” (Digital input 4 - HW Enable Signal, see “5 Hardware installation – Safe Torque Off”) and proper Drive Command (RxPDO1) is still sent by the VCU.
To start again, user has to perform a key power cycle, repeat the boot-up procedure and reset the Drive Command (RxPDO1 message.
The VCU can detect the hard failure in the controllers by reading the Drive Status (TxPDO2) message, which contain the failure code.

5.9.6 Hard fault occurs in VCU

These two situations can occur:

1. Stop sending Drive Command (RxPDO1) to slave controllers: slave controllers will stop after timeout timer is expired.
2. Cut down the “Safe Torque Off input” (see “Safety functions”): slave controllers will reset all the inputs, stop pulsing (cutting to zero immediately current and torque) and stop driving, signalling **F29 S007**.

6 Shared line contactor function

Shared line contactor management is operated by one of the DMC controllers connected in CAN network which is set up as “Shared Line contactor master controller”. To this controller the physical line contactor must be connected. Other DMC controllers are set up as and considered “Shared Line contactor slaves controller”. “Shared Line contactor master controller” is not a CAN Open network master, it is “master” related to share line contactor operation.
The DMC controllers share information for shared line contactor by means of TxPDO4 message (ID 0x480 + node number)) in case the 11 bit message ID is chosen or on TPDO9 in case the 29bit option (ID 0x00000980 + node number).
The 11/29 bit option can be chosen by means of the parameters “M7.1-5 DualMotor and SharedLC messages on 29 or 11 bit IDs “DM&LC ID” “
The prerequisites indicated in the following list are necessary.

- Maximum controllers sharing line contactor 5.
- Can Nodes set up from 1-127
- Shared line contactor must be connected to the DMC controller with the lowest CAN node number. The “Shared Line contactor HMI master controller” has therefore to be set up with the lowest CAN node number.
- There Controllers sharing the Line Contactor must have consecutive node numbers.

For wiring examples please refer to section “9 Wiring examples”.

6.1 Examples

6.1.1 How to set up for two controllers sharing line contactor

Here is an example on how to set up two controllers, for example traction + pump, connected in the CAN network. The controllers are set up to be sharing a line contactor.
Important settings are in bold, differences are highlighted

Parameter	TRACTION CONTROLLER	PUMP CONTROLLER
CAN node number “CAN node”	**8**	**9**
CAN bit rate “CANbitRt”	Same value	Same value
Shared line contactor option “LCoption “	**1:** Shared Line contactor CAN master	**2:** Shared Line contactor CAN slave
Shared line contactor reference node “Ref Node”	**9**	**8**
Shared line contactor refresh rate “SHLCRate”	**Same value (suggested 60 ms)**	**Same value (suggested 60 ms)**

In this example the Line contactor must be connected physically to traction controller node 8

6.1.2 How to set up for 3 controllers sharing line contactor

Here is an example on how to set up three controllers, for example traction + traction + pump, connected in the CAN network. The controllers are set up to be sharing a line contactor.
Important settings are in bold, differences are highlighted.

Parameter	**TRACTION CONTROLLER (1)**	**TRACTION CONTROLLER (2)**	**PUMP CONTROLLER**
CAN node number “CAN node”	**57**	**58**	**59**
CAN bit rate “CANbitRt”	Same value	Same value	Same value
Shared line contactor option “LCoption “	**1:** Shared Line contactor CAN master	**2:** Shared Line contactor CAN slave	**2:** Shared Line contactor CAN slave
Shared line contactor reference node “Ref Node”	**59**	**57**	**57**
Shared line contactor refresh rate “SHLCRate”	**Same value (suggested 50 ms)**	**Same value (suggested 50 ms)**	**Same value (suggested 50 ms)**

In this example the Line contactor must be connected physically to traction controller node 57.

6.2 Interaction between Shared Line contactor and ControlWord

There can be the case in which the user wishes to use both the “Shared Line Contactor” function and the “Control Via CAN” with the ControlWord object mapped for some DMC Drives. Here different situations can arise:

- The “Shared Line Contactor Master” has the ControlWord mapped and no slave does.
- The “Shared Line Contactor Master” has the ControlWord mapped and only some slave does.
- The “Shared Line Contactor Master” has the ControlWord mapped and also every slave do.
- The “Shared Line Contactor Master” does not have the ControlWord mapped and some slave does.
- The “Shared Line Contactor Master” does not have the ControlWord mapped and every slave do.

Independently on the above situation, the Line Contactor will be managed by the “Shared Line Contactor Master” according to its own and the Slave’s Application State machine.

This means that the Line Contactor will be closed only if all the devices sharing the line contactor are in “Ready to switch on” state (which means they have finished precharge and they are ready to have power applied).

To this mean the user must lead every device with ControlWord mapped to reach the “Ready to switch on state” and also give them the command to “Switch on”. Devices without ControlWord mapped will automatically aim at reaching the “Operation enabled” state and do not have to be managed. The status of each device can be monitored by means of the StatusWord.

In case a “Shutdown” or a “DisableVoltage” command is given to any device with ControlWord mapped, then all the devices (also those without “ControlWord mapped”) will elaborate that command and the power will be removed safely (i.e. the Line Contactor will be opened only after devices has stopped energizing the motor).

6.3 Error codes related to Shared Line Contactor

If shared line contactor option is selected, the Sigma2N or SigmaLITE controller perform some checks at power up to make sure that the settings of all controllers are consistent.
If some inconsistency is found, the controllers will display a failure code and safely inhibit driving/working.

Base fault Code	Sub fault code	Description	Solution
Hard error faults - Immediately stops pulsing and open line contactor			
7	2	“Shared Line Contactor slaves Controllers” are not set up as shared line contactor. This happens in the “Shared Line contactor master controller” (“ShareLC” set to 1) if the connected slaves sharing line contactor are not configured as “Shared Line contactor slave controller” (“ShareLC” set to 2).	Set in every “Shared Line contactor slave controller” parameter “ShareLC” to 2.
	16	“Shared Line contactor master controller” CAN node number (“CAN Node”) is higher/equal than last node (“LstNode”). This happens in the “Shared Line contactor master controller” (“ShareLC” set to 1), if “CAN Node” >= “LstNode”, which is a non-sense.	Check node assignment and make sure that in “Shared Line contactor master controller” (“ShareLC” set to 1) “CAN Node” < “LstNode”
	17	“Shared Line contactor slave controller” CAN node number (“CAN Node”) is lower/equal than “Shared Line contactor master controller” CAN node number. This happens in the controller with the setting “ShareLC” set to 2, if “CAN Node” >= “LstNode”, which is a non-sense	Check node assignment and make sure that in every “Shared Line contactor slave controller” (“ShareLC” set to 2) “CAN Node” > “LstNode”
29	1	“Shared Line Contactor slave” node/s time out. It happens in the “Shared Line Contactor master” if one or more “Shared Line Contactor slaves” stop communicating	Check can wiring;
	4	“Shared Line Contactor master” node time out. It happens in the “Shared Line Contactor slaves” if “Shared Line Contactor master” stop communicating	Check can wiring;
	20	“Shared Line Contactor master” node/s failure. It happens in the “Shared Line Contactor slaves” if the “Shared Line Contactor master” undergo a failure	Check Master’s failure code
	21	“Shared Line Contactor slave” node failure. It happens in the “Shared Line Contactor master” if any “Shared Line Contactor slave” undergo a failure	Check Slaves’ failure codes
	22	“Shared Line Contactor slave” node failure. It happens in the “Shared Line Contactor slave” if any other “Shared Line Contactor slave” undergo a failure	Check Slaves’ failure codes
39	3	Time out fault. Happens if, within 10s from power up, the controllers set up as shared line contactor are not able to communicate with each other via CAN because not properly set up or because of wiring issues. This prevents to drive in the cases of inconsistencies not covered by the failure indicated above.	Refer to example tables (“Examples How to set up for two controllers sharing line contactor” and “How to set up for 3 controllers sharing line contactor”) to check setting consistency and check wiring.

7 External Line Contactor management

The "External Line Contactor" function allows the user to independently drive the Line Contactor. This means that the Line Contactor's coil is not driven by the DMC controller itself; instead, a third party device (i.e. a BMS, a PLC or a VCU) is in charge of opening and closing the contactor and managing the Line contactor related safety function.
To this means, the user should notice that the DMC Controller has an internal pre-charge circuit and it is performing several checks at powerup aimed at providing Line contactor's safety. Moreover the DMC Controller, when operating the Line contactor itself, may be choosing to remove power from the controller at any time in case of severe faults. Therefore, if a user wishes to use the "External Line Contactor" function, he must take care of the following:

- That the DS402 standard object 0x6040 (i.e. ControlWord) and the DS402 standard object 0x6041 (i.e. StatusWord) must be mapped to the DMC Controller;
- That a power-on sequence must be followed to avoid controller's faults;
- That continuous monitoring of the controller status should be performed in order to remove power on request.

7.1 External Line contactor power-on sequence

The user should implement in its VCU/PLC the following sequence for applying the Line Contactor at powerup. This will assure that the DMC Controller will pass all the power-up safety checks and will correctly perform its pre-charge, before allowing to close the Line Contactor.
Please notice that this function requires the user to perform a specific PDO mapping (refer to "4 Free mapping of DMC Controller" for details) in order to work. In fact, DS402 standard object 0x6040 (i.e. ControlWord) and the DS402 standard object 0x6041 (i.e. StatusWord) must be mapped to the DMC Controller.
IMPORTANT: The described sequence must be applied to every DMC Controller in the network!

1. Power on the DMC Controller (i.e. provide the Key input, Pin 13 of Connector B [2]);
2. Set the DMC controller in "Operational" status trough the NMT message (see "2.2 Node Boot Sequence, Network Management and Heartbeat message");
3. Start monitoring the TxPDO mapped with the DS402 standard object 0x6041 (i.e. StatusWord);
4. Start sending the RxPDO mapped with the DS402 standard object 0x6040 (i.e. ControlWord).
5. In case of independent logic supply, start sending the RxPDO mapped with the DMC Battery Volage object 0x3918 (if different from RxPDO containing the DS402 standard object 0x6040 mentioned at previous point).
   Alternatively, in case the DMC Sigma2N or SigmaLITE controller is setup to work with a compatible BMS (see [2] or [10] for details), make sure that the BMS is outputting its CAN messages.
6. Monitor the Controller's Status Via the StatusWord until it is in "Switch On Disabled" state (i.e. value = 0bxxxx xxxx x1xx 0000, x: don't care).
7. In case of independent logic supply, apply high level power through a pre-charge resistor to power battery plus B+ terminal of the controller;
8. Trough the "ControlWord", send the "SHUTDOWN" command (i.e. value = 0bxxxx xxxx 0xxx x110, x: don't care). This will instruct the DMC Controller to start monitoring the pre-charge process;
9. Monitor the Controller's Status Via the StatusWord until it is in "Ready To Switch On" state (i.e. value = 0bxxxx xxxx x01x 0001, x: don't care). At this stage, the DMC Controller's pre-charge phase is ended;
10. Trough the "ControlWord", send the "SWITCH ON" command (i.e. value = 0bxxxx xxxx 0xxx x111, x: don't care);
11. Monitor the Controller's status Via the StatusWord until the "Voltage applied" bit is high (i.e. value = 0bxxxx xxxx xxx1 xxxx, x: don't care). This will be true as soon as the Controller enters "Switched On" state (i.e. StatusWord's value = 0bxxxx xxxx x01x 0011, x: don't care);
12. Close the Line Contactor and bypass the precharge resistor;
13. Trough the "ControlWord", send the "ENABLE OPERATION" command (i.e. value = 0bxxxx xxxx 0xxx 1111, x: don't care);
14. Monitor the Controller's Status Via the StatusWord until it is in "Operation enabled" state (i.e. value = 0bxxxx xxxx x01x 0111, x: don't care);
15. Start sending the commands via CAN or driving with analogue inputs.

The device management described above, falls under the case of "5.7 Device management with Control Word (Object 0x6040)" and "5.8 Device monitoring with StatusWord (Object 0x6041)". Please refer to those chapters for further explanations and details about DS402 device management.
Please notice that also backward transitions are supported, but the user should take care of keeping the Line Contactor (and precharge resistor, in case of independent logic supply) physical statuses coherent with the DS402 application state machine's status. This can be achieved by looking at the StatusWord's "Voltage applied" bit (i.e. bit #4, counting from 0). If the said bit is high, the physical Line Contactor is expected to be closed; it the said bit is low, the physical Line Contactor is expected to be open.
Furthermore, the Controller should be totally disconnected from the power battery (i.e. Line Contactor open and, in case of independent logic supply, precharge resistor excluded) in the following cases:

- in "FAULT" status
- in "SWITCH ON DISABLED" status before having sent the "SHUTDOWN" command to start precharging.

The "voltage applied" bit, together with the "fault" bit (i.e. bit #3 counting from 0) of the StatusWord, must be monitored continuously in order to open the Line contactor (and exclude the precharge resistor, in case of independent logic supply) in case the Controller is undergoing a fault that requires the high level power to be removed immediately and completely.

7.2 Separated control logic supply management

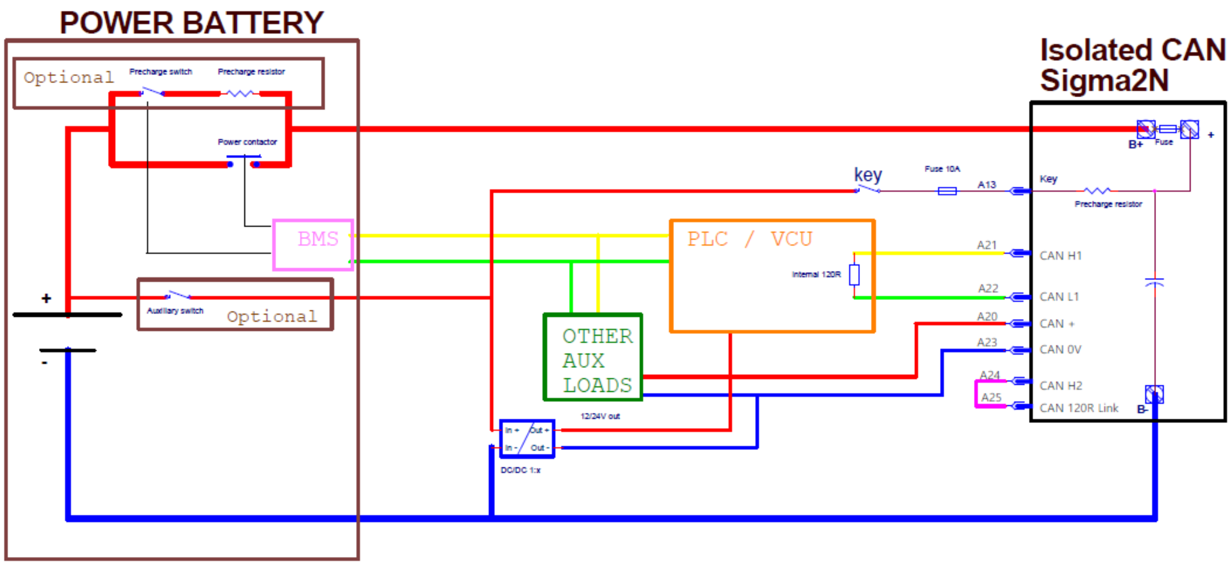
In case “M7.3-1 Shared line contactor option “LCoption “” is set to 4, the DMC Controller expects its logic to be powered from an independent supply other than Battery. This allows, for example, to avoid that the DMC Controller shuts down in case the logic power supply (i.e. the Key input, Pin 13 of Connector B [2]) comes from a non-directly controlled source (which might be integrated inside a BMS), that completely removes power to the system.

Supplying the logic from a different source allows the Controller to be kept powered also if the main power supply is missing. This can be useful for live fault recovery or system diagnostic.

7.3 Wiring schematic examples for Sigma2N

7.3.1 External line contactor without separated logic supply

This example refers to a situation in which the Line Contactor is not driven directly by the DMC Sigma2N controller.

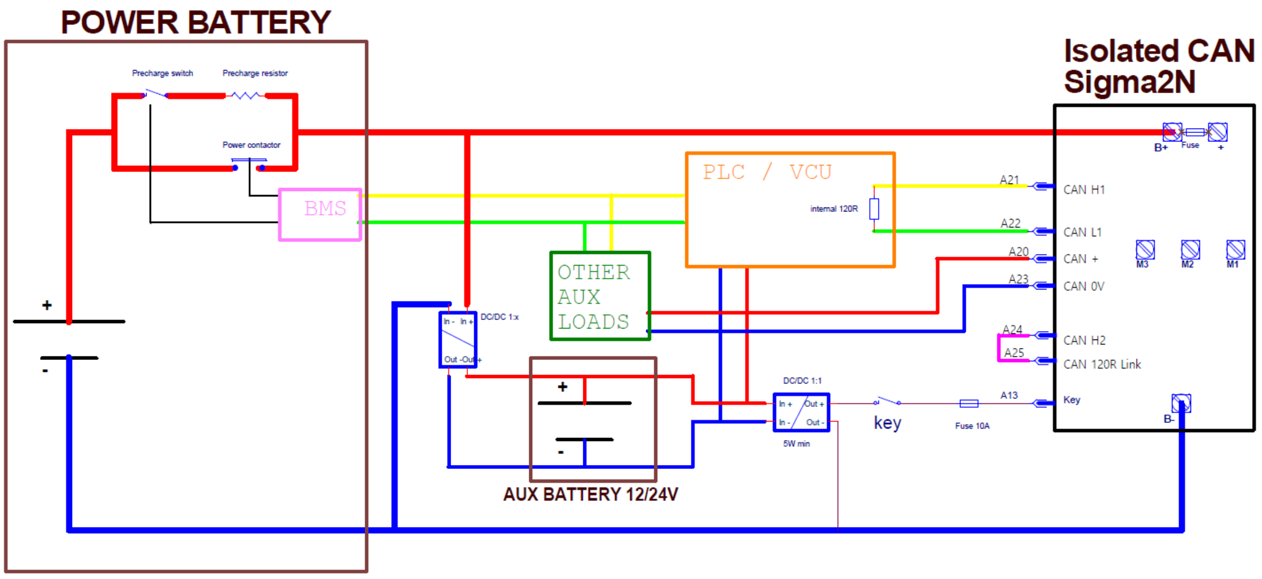


! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

7.3.2 External line contactor with separated logic supply

The following examples represents the suggested wiring for a situation in which the Key line for DMC Sigma2N Controller is separated from the main power source.

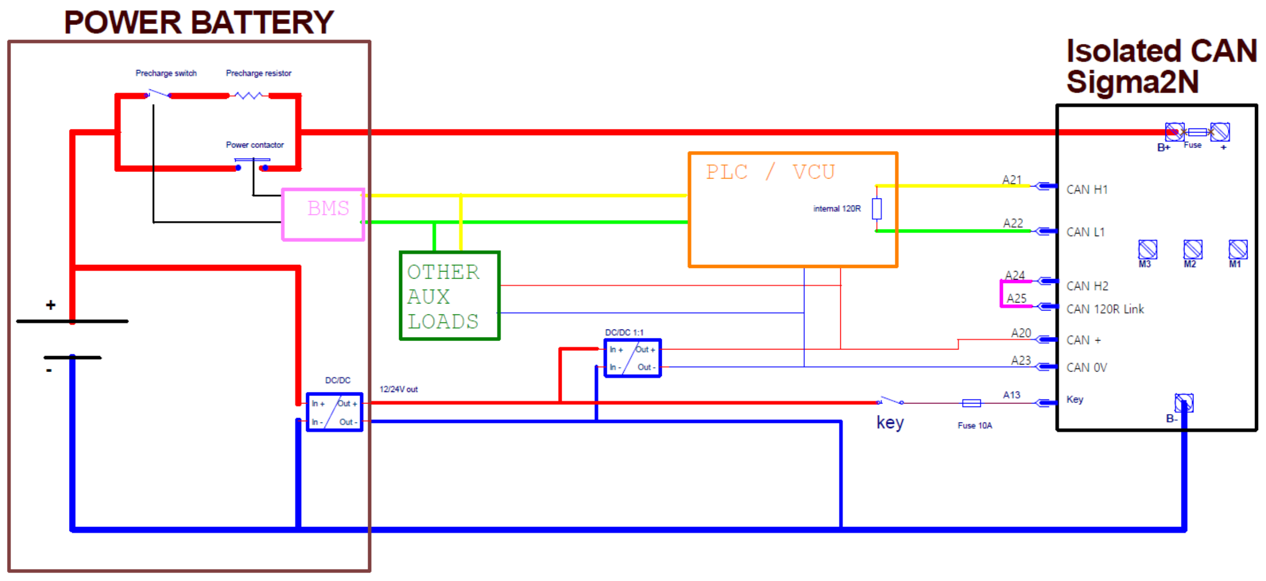
Please notice that the DC/DC converter supplying the Sigma2N key line must be at least capable of outputting 5W for supplying the control logic. In case the application needs the DMC Sifgma2N Controller to drive one or more Digital Outputs, the user should consider upsizing the overmentioned DC/DC converter taking into account 2A for each driven output. Mind that the outputs will be driven at Key line voltage.



! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

The following example represents the case in which the power battery embeds a dedicated output at 12 or 24V and therefore an auxiliary battery is not needed. Notice that in this case this output must be capable of outputting enough power for satisfying the consumption of DMC Sigma2N Controller and all the connected auxiliary devices.

The consideration at the begin of this section remain valid.

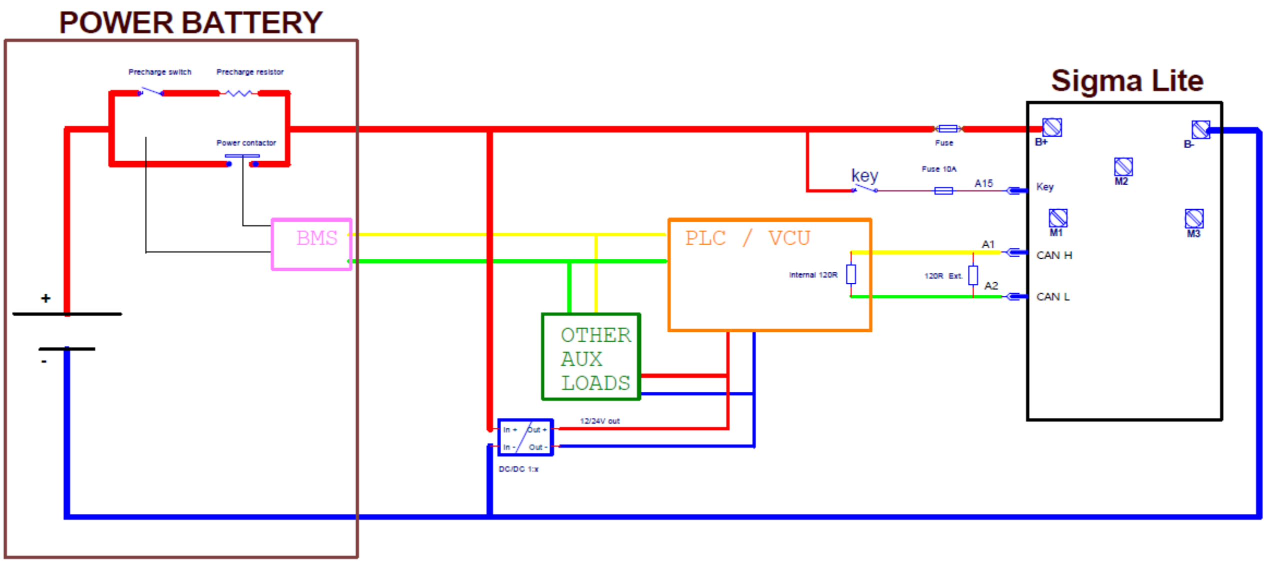


! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

7.4 Wiring schematic examples for SigmaLITE

7.4.1 External line contactor without separated logic supply

This example refers to a situation in which the Line Contactor is not driven directly by the DMC SigmaLITE controller.

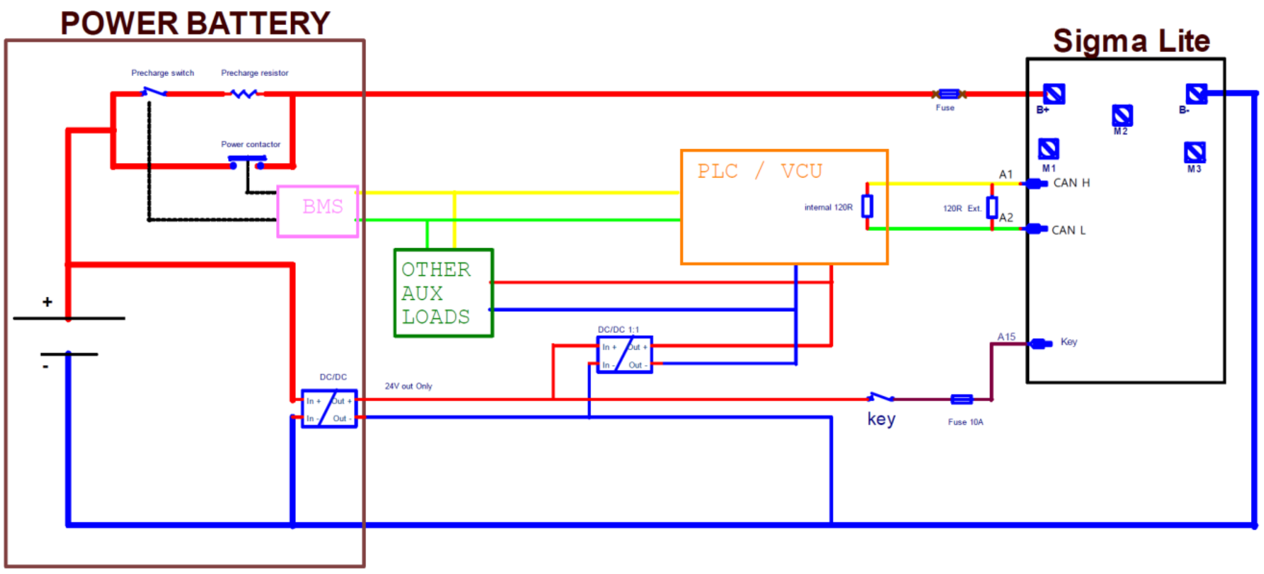


! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

7.4.2 External line contactor with separated logic supply

The following examples represents the suggested wiring for a situation in which the Key line for DMC SigmaLITE Controller is separated from the main power source.

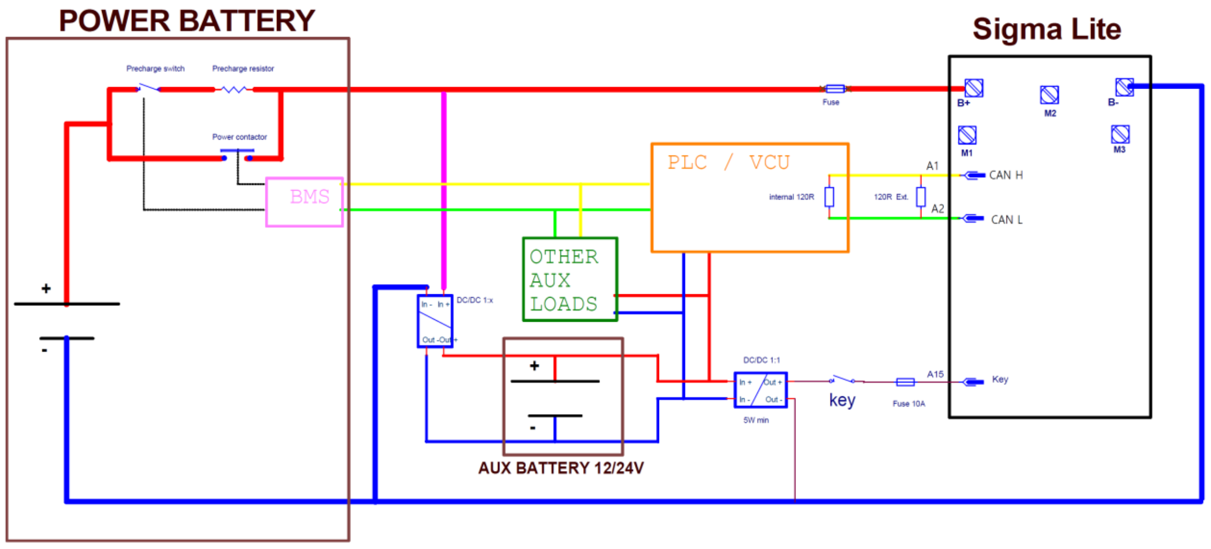
Please notice that the DC/DC converter supplying the SigmaLITE key line must be at least capable of outputting 5W for supplying the control logic. In case the application needs the DMC Sifgma2N Controller to drive one or more Digital Outputs, the user should consider upsizing the overmentioned DC/DC converter taking into account 2A for each driven output. Mind that the outputs will be driven at Key line voltage.



! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

The following example represents the case in which the power battery embeds a dedicated output at 12 or 24V and therefore an auxiliary battery is not needed. Notice that in this case this output must be capable of outputting enough power for satisfying the consumption of DMC SigmaLITE Controller and all the connected auxiliary devices.

The consideration at the begin of this section remain valid.



! Notice: the above scheme has to be followed strictly. Any variation must be agreed with DMC engineering first.

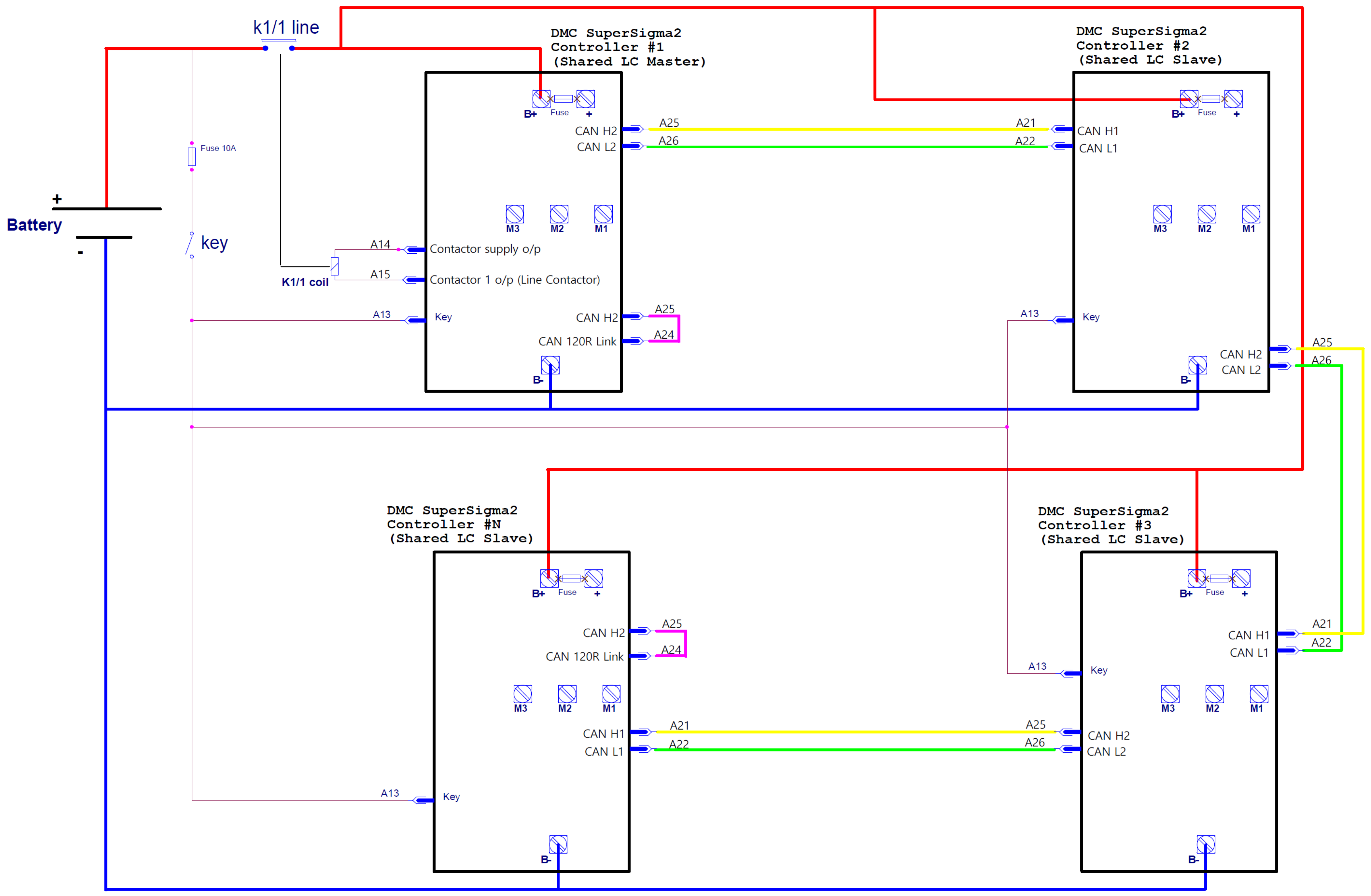
8 Use of Can Open Calibrator and DMC Configurator

DMC Calibrator can be connected in the Can Open Network since every DMC Sigma2N and SigmaLITE controller has a dedicated “node” number assigned.

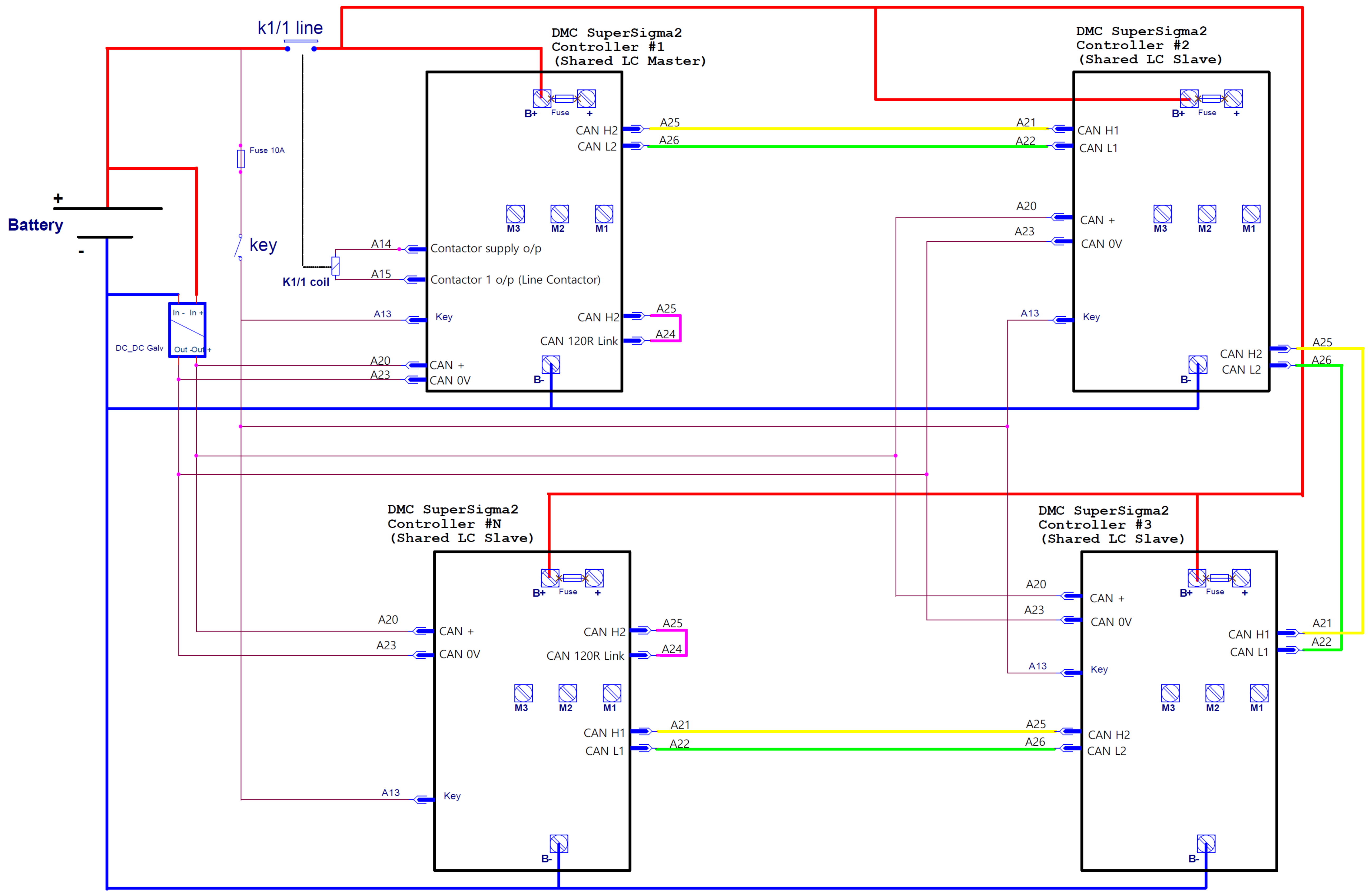
Connecting and using the DMC Configurator program (using the DMC Can Open Calibrator as dongle) has no limitation.

9 Wiring examples Sigma2N

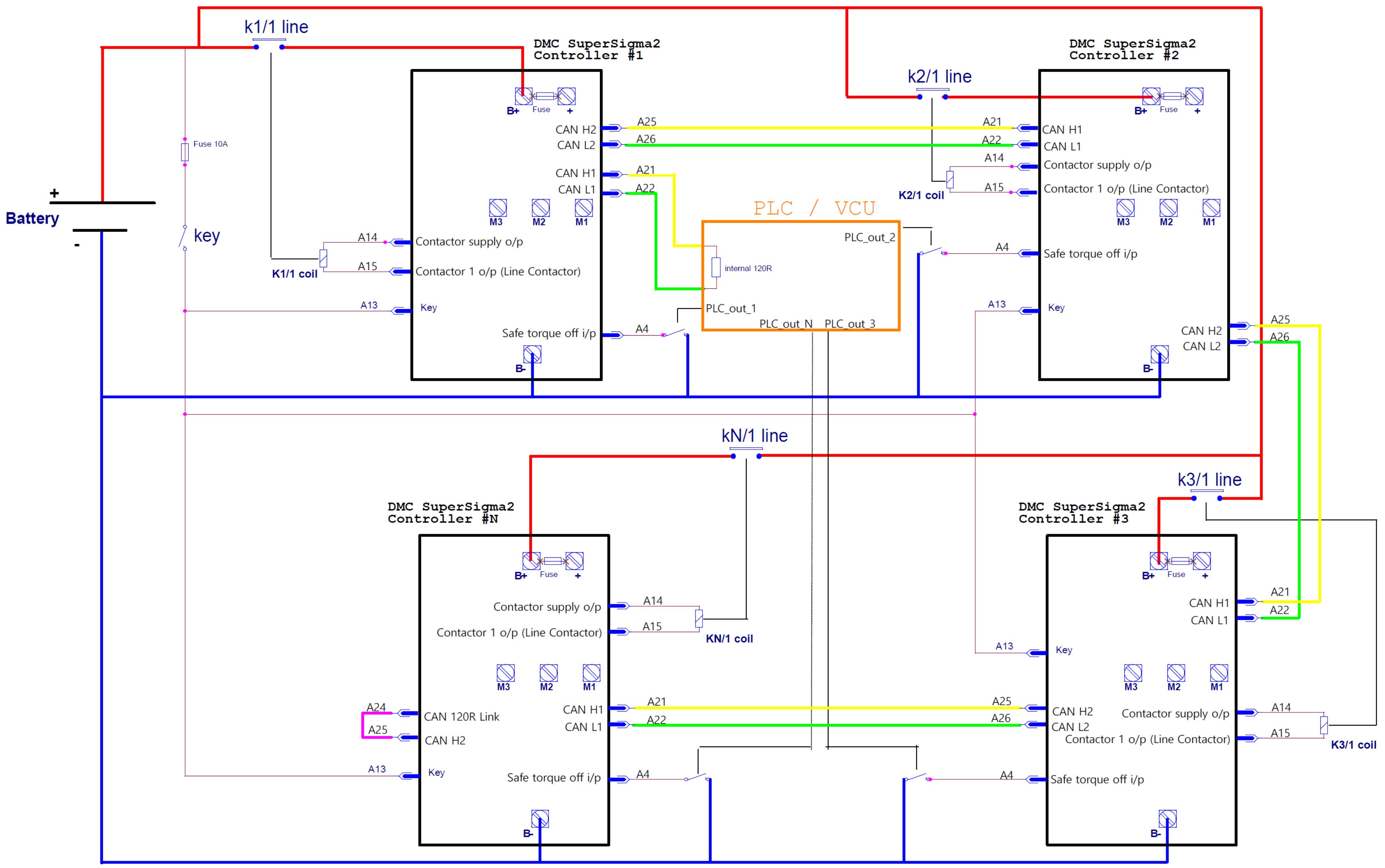
9.1 Shared Line Contactor with non-isolated CAN bus



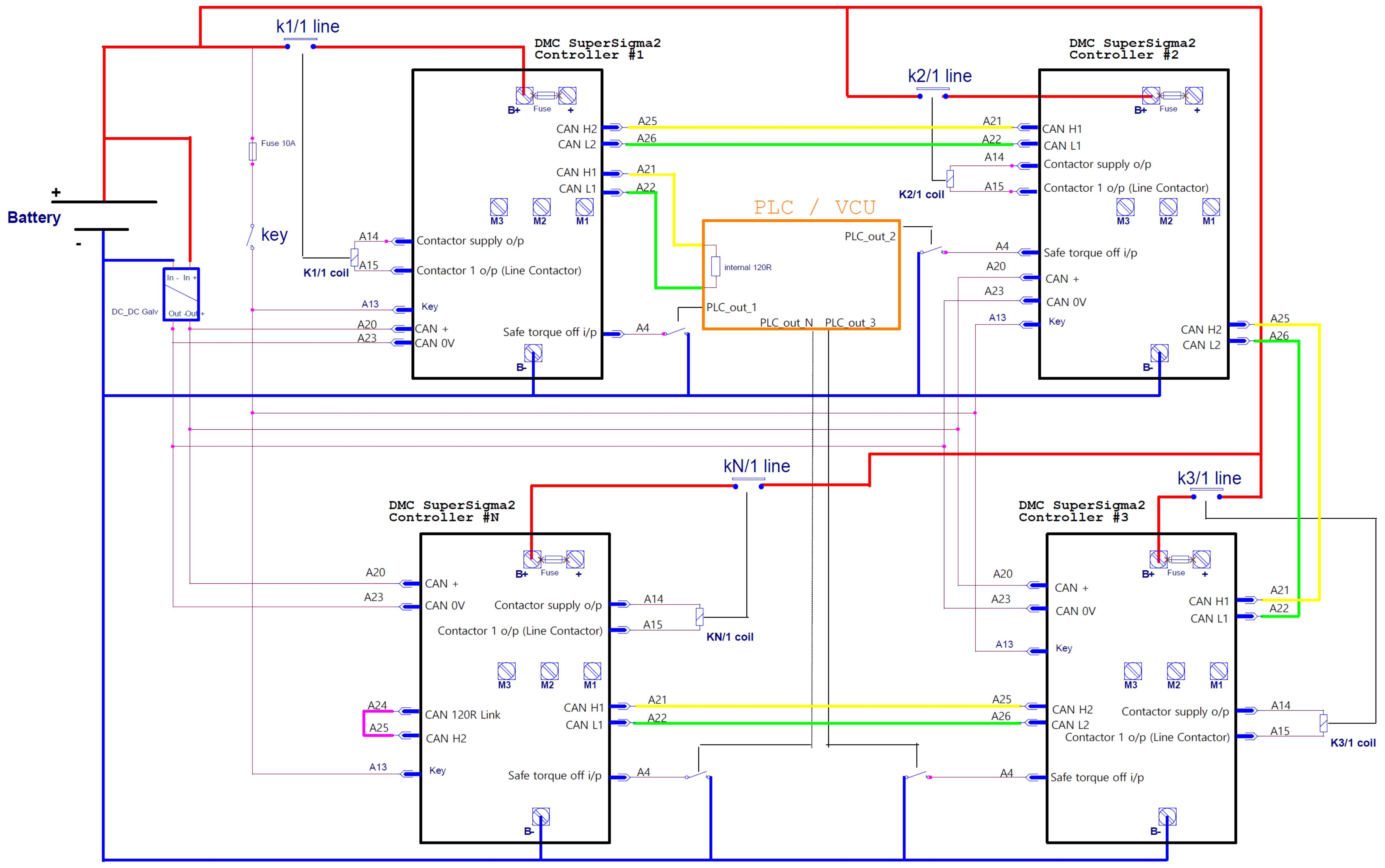
9.2 Shared Line Contactor with isolated CAN bus


k1/1 line
DMC SuperSigma2 Controller #1 (Shared LC Master)
DMC SuperSigma2 Controller #2 (Shared LC Slave)
DMC SuperSigma2 Controller #N (Shared LC Slave)
DMC SuperSigma2 Controller #3 (Shared LC Slave)
Battery
+
-
Fuse 10A
key
K1/1 coil
DC_DC Galv
In - In +
Out - Out +
B+
Fuse
+
B-
M3
M2
M1
CAN H1
CAN L1
CAN H2
CAN L2
CAN +
CAN 0V
Key
Contactor supply o/p
Contactor 1 o/p (Line Contactor)
CAN 120R Link
A13
A14
A15
A20
A21
A22
A23
A24
A25
A26

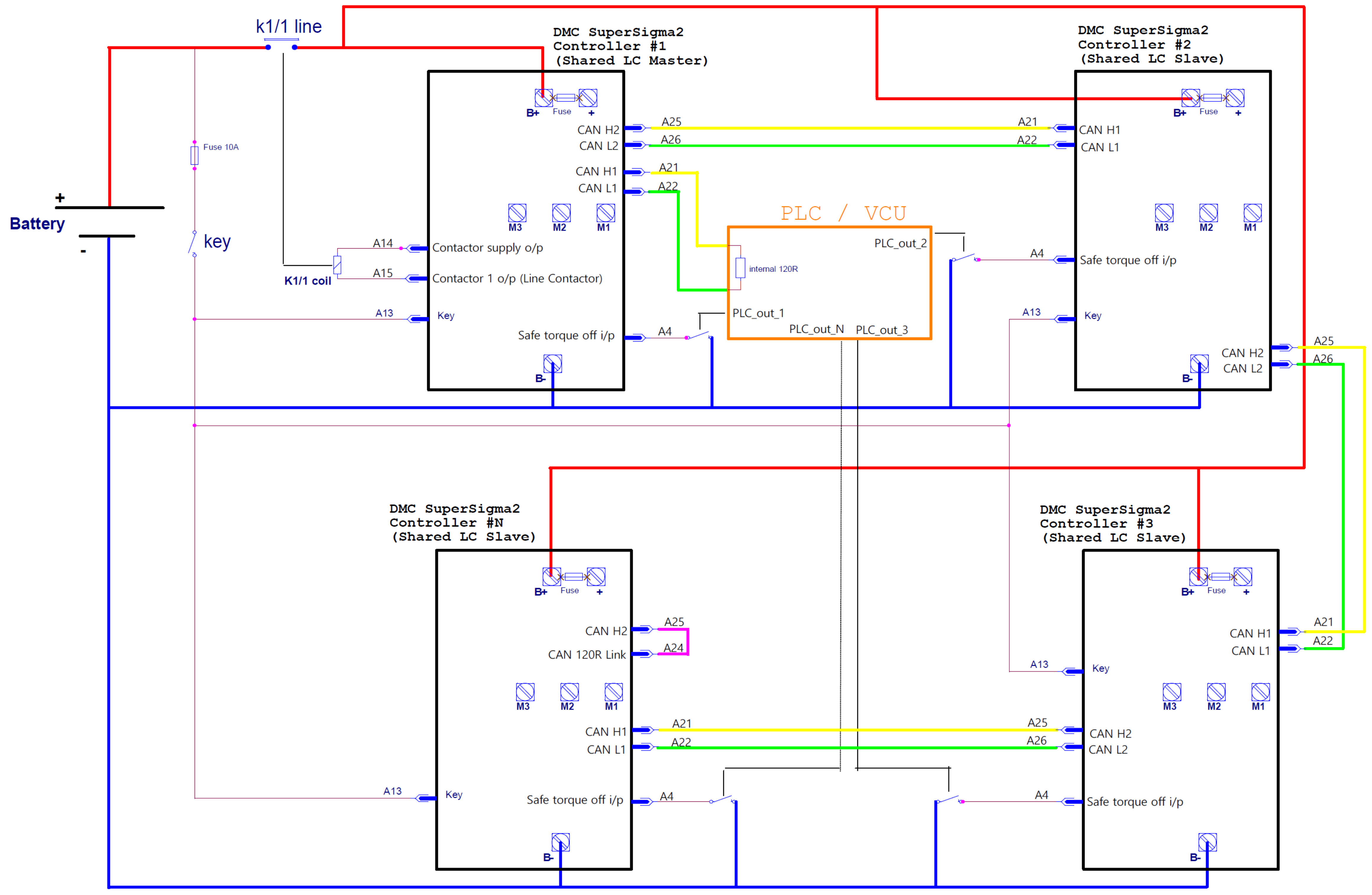
9.3 Control Via CAN with non-isolated CAN bus



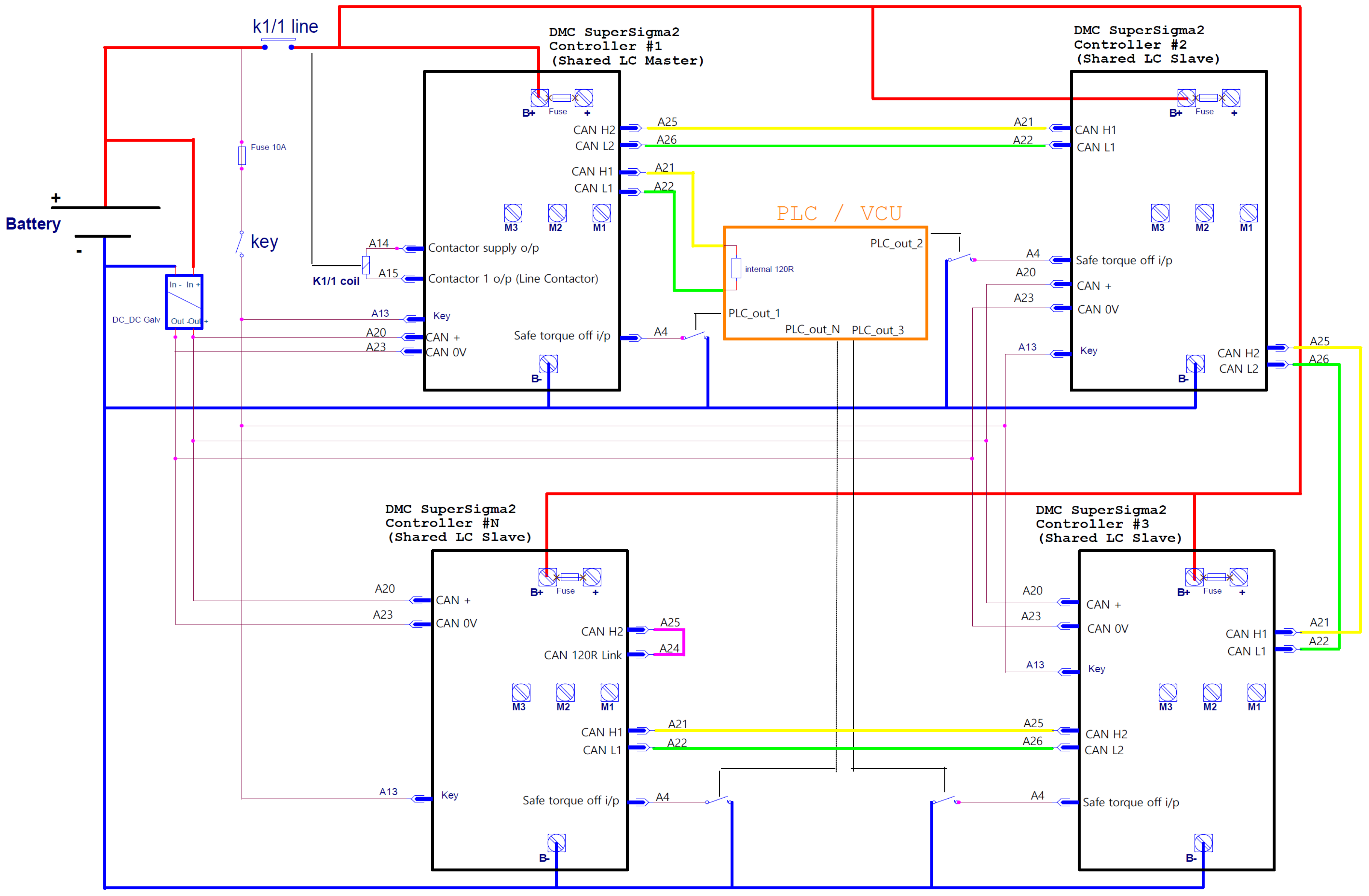
9.4 Control Via CAN with isolated CAN bus



9.5 Control Via CAN and Shared Line Contactor with non-isolated CAN bus


k1/1 line
DMC SuperSigma2 Controller #1 (Shared LC Master)
DMC SuperSigma2 Controller #2 (Shared LC Slave)
DMC SuperSigma2 Controller #N (Shared LC Slave)
DMC SuperSigma2 Controller #3 (Shared LC Slave)
Battery
+
-
Fuse 10A
key
K1/1 coil
A14
A15
A13
A4
A21
A22
A24
A25
A26
B+
Fuse
+
B-
M1
M2
M3
CAN H1
CAN L1
CAN H2
CAN L2
CAN 120R Link
Contactor supply o/p
Contactor 1 o/p (Line Contactor)
Key
Safe torque off i/p
PLC / VCU
internal 120R
PLC_out_1
PLC_out_2
PLC_out_3
PLC_out_N

9.6 Control Via CAN and Shared Line Contactor with isolated CAN bus


k1/1 line
DMC SuperSigma2 Controller #1 (Shared LC Master)
DMC SuperSigma2 Controller #2 (Shared LC Slave)
DMC SuperSigma2 Controller #N (Shared LC Slave)
DMC SuperSigma2 Controller #3 (Shared LC Slave)
Battery
+
-
Fuse 10A
key
K1/1 coil
DC_DC Galv
In - In +
Out -Out +
B+
Fuse
+
B-
M3
M2
M1
CAN H2
CAN L2
CAN H1
CAN L1
CAN 120R Link
Contactor supply o/p
Contactor 1 o/p (Line Contactor)
Key
CAN +
CAN 0V
Safe torque off i/p
A4
A13
A14
A15
A20
A21
A22
A23
A24
A25
A26
PLC / VCU
internal 120R
PLC_out_1
PLC_out_2
PLC_out_3
PLC_out_N