OPEN DRIVE

OPEN DRIVE

Canbus

INDEX

OPEN DRIVE line products are compatible with CAN open Communication Profile DS301 of CiA rev 4.02. This document describes the mandatory and the optional functions that complete the implementation.

1 Configuration of the application

1.1 *Configuration of the node*

The drive configuration as CAN node includes the use of the following customer parameters (of conventional use):

Name	Description	Range	Default
P162	ID CAN BUS node	1÷127	1
C48	Configuration CAN BUS baud rate	0 ÷ 7 0 = 1 Mbit/s 1 = 800 Kbit/s 2 = 500 Kbit/s 3 = 250 Kbit/s 4 = 125 Kbit/s 5 = 50 Kbit/s 6 = 20 Kbit/s 7 = 10 Kbit/s	0 = 1 Mbit/s

These parameters must be rightly configured and saved in the permanent memory of the drive (C63=1). At start up these data are considered and become operating.

1.2 Configuration of the communication objects

The configuration of the communication objects CAN OPEN DS301 can uniquely be done via CAN.
At first switch on, the drive is a non-configured node which satisfies the "pre defined connection set" for the identifiers allocation; for this, the following objects are available:

- rx SDO with COB-ID = 600h + ID CAN node (parameter P162)
- tx SDO with COB-ID = 580h + ID CAN node
- an emergency object with COB-ID = 80h + ID CAN node
- NMT objects (Network Management) : in broadcast (COB-ID=0) for Module Control services and COB-ID = 700h + ID CAN node for Error Control.
- The SYNC object in broadcast with COB-ID = 80h

With the SDO available, the drive can be totally configured as CAN node and only after the communication objects can be saved in the permanent memory using the proper command "store parameters" (1010h)" on the Sub-Index 2.
Also the object "restore default parameters (1011h)" Sub-Index 2 is managed to load all the default communication objects and to save them automatically in the permanent memory (switch off and then on the drive to make objects operating).

2 Managed services

2.1 Service data object (SDO)

SDO are used to access the objects dictionary. In our implementation a maximum of 4 server SDO can be available which can be configured with the following objects:

1200h 1st server SDO parameter
1201h 2nd server SDO parameter
1202h 3rd server SDO parameter
1203h 4th server SDO parameter

The transfer mode depends on the length of the data to be transferred : up to 4byte data length, the modality ***expedited*** is used as it is simple and immediate; for bigger size objects the modality ***segmented*** and ***block*** are both supported. See the specific Communication Profile DS301 for having details on the different transmission modes; hereinafter are written only some peculiarities of our implementation:

- a writing access to SDO must indicate the number of significant byte (data set size)
- the writing data by SDO is liable to the same rules (drive state, keys, tolerated range…) seen for the other modalities of parameters modify (serial and keyboard).
- If SDO are structured in more segments, the drive will start writing the data at the indicated address with the first segment, without using a temporary buffer
- A controller is intended to avoid that two SDOs access the same object at the same time.
- With the transmission in block modality, the computation of CRC and the "Protocol Switch Threshold" are not supported.

- It is possible to set the block size of the SDO Block Download service at the address 2000h of the objects dictionary, in the manufacturer specific section.

2.2 Process Data Object (PDO)

PDO are used for the data exchange in real-time in the objects dictionary that supports this function.

2.2.1 Transmit PDO

In our implementation up to a maximum of **4 TPDO** can be configured with the following objects :

1800h 1st Transmit PDO Communication parameter
1801h 2nd Transmit PDO Communication parameter
1802h 3rd Transmit PDO Communication parameter
1803h 4th Transmit PDO Communication parameter

the 5 Sub-Index related to every type of TPDO are all managed : it is possible to set the transmission type (see the following table), the inhibit time with 100µs resolution and the period of the event timer with 1ms resolution.

transmission type	PDO transmission
0	**Synchronous**: data are refreshed and transmitted with every SYNC received.
1-240	**Synchronous and cyclical**: the number indicates how many SYNC are in between two following transmissions
241-251	---------- reserved --------------------------
252	Data are refreshed and sent at the following **RTR** when the SYNC is received
253	Data are refreshed and sent when the **RTR** is received (remote transmission request)
254	**Event timer** : cyclical transmission with a period time settable in ms in the Sub-Index 5
255	**Manufacturer specific** : it is settable time by time

Note: in the transmission type 255, it is possible to choose on which event the TPDO transmission works. The event choice can be effectuated only during the compiling the software code.

The TPDO mapping can be dynamically effectuated by rightly configuring the following communication objects:

1A00h 1st Transmit PDO Mapping parameter
1A01h 2nd Transmit PDO Mapping parameter
1A02h 3rd Transmit PDO Mapping parameter
1A03h 4th Transmit PDO Mapping parameter

the PDO mapping must be done by following these instructions:

1. the number of the mapped objects in Sub-Index 0 must be equal to zero
2. the addresses of all mapped objects must be configured
3. the correct number of mapped objects in the Sub-Index 0 must be indicated

2.2.2 Received PDO

In our implementation a maximum of 4 RPDO can be configured with the following objects:

1400h 1st Receive PDO Communication parameter
1401h 2nd Receive PDO Communication parameter
1402h 3rd Receive PDO Communication parameter
1403h 4th Receive PDO Communication parameter

The first 2 Sub-Index related to each RPDO are managed: in this way it is possible to set the transmission type:

transmission type	PDO receiving
0-240	**synchronous**: when the following SYNC is received, the values received on the RPDO will be activated.
241-253	--------------------------- reserved --------------------------
254	**Asynchronous**: the values received in the RPDO are immediately activated.

The RPDO mapping can be dynamically effectuated by rightly configuring the following communication objects:

1600h 1st Receive PDO Mapping parameter
1601h 2nd Receive PDO Mapping parameter
1602h 3rd Receive PDO Mapping parameter
1603h 4th Receive PDO Mapping parameter

RPDO mapping must be executed by following the next directives as well:

3. Set the number of mapped objects in Sub-Index 0 to be equal to zero
4. Configure the addresses of all mapped objects
5. Indicate the correct number of mapped objects in Sub-Index 0

2.3 Emergency Object (EMCY)

The emergency object is transmitted by the drive when a new enabled alarm comes trough or when one or more alarms are reset. The Emergency telegram is made by 8byte as shown in the following table:

Byte	0	1	2	3	4	5	6	7
meaning	**Emergency Error Code**		**Error register**	**Manufacturer specific** alarms LSB –MSB				

In our implementation only two codes of the error code are implemented :

00xx = Error Reset or No Error
10xx = Generic Error

Speaking of the **Error register** (object 1001h), the following bits are managed corresponding to the following alarms:

Bit	Meaning	Corresponding alarms
0	General error	all
1	Current	A3
2	Voltage	A10 - A11 -A13
3	temperature	A4 - A5 - A6

In Manufacturer specific only the bytes 3 and 4 are assigned which contain the state of the various alarms of the drive. Further 3 bytes for the transmission of possible other user's data are available.

The management of 1003h "pre-defined error field " object memorises the chronology of the alarm events (from start up of the drive) up to a maximum of 32 elements.
At every new alarm event 4 bytes are memorised, 2 are mandatory and correspond to the Error Code; the other 2 are Manufacturer specific and in our specific case correspond to the state of all the drive alarms.

MSB LSB

Additional information		**Error code**	
alarms MSB	alarms LSB	Error code MSB	Error code LSB

2.4 Network Management Objects (NMT)

This function allows the NMT master to check and set the state to every NMT slave.
All the services of Module Control and also the Node Guarding Protocol which uses the COB-ID = 700h + ID CAN node are implemented: this allows the slave to communicate that the bootup ended and the pre-operational modality is active, thus the master can interrogate the different slaves with an RTR.

The Life guarding function is implemented as well: the drive (NMT slave) can be set up by the objects:

100Ch **Guard time** in ms

100Dh **Life time factor** (multiplier factor)

} their product yields the **Node life time**

note: node life time is internally saturated in the period time of 32767/fpwm sec.

Life guarding is enabled only if life time Node is different to zero; in this case the check-up starts after having received the first RTR from the NMT master.
The Communication profile DS301 doesn't decide which action it has to start if the time constrain of life guarding hasn't been respected. It's possible to decide how to act, during the firmware compilation step. By default, no action is done.

2.5 Objects dictionary : communication profile area

The following objects of the communication profile are supported:

Index (hex)	Object	Name	Type	Access	Par.
1000	VAR	Device type	UNSIGNED32	Reading	
1001	VAR	Error register	UNSIGNED8	Reading	2.3
1002	VAR	Manufacturer status register	UNSIGNED32	Reading	
1003	ARRAY	Pre-defined error field	UNSIGNED32	Reading	2.3
1005	VAR	COB-ID SYNC	UNSIGNED32	Reading/writing	2.2
1006	VAR	Communication cycle period	UNSIGNED32	Reading/writing	2.2
1008	VAR	Manufacturer device name	Vis-String	constant	
1009	VAR	Manufacturer hardware version	Vis-String	constant	
100A	VAR	Manufacturer software version	Vis-String	constant	
100C	VAR	Guard time	UNSIGNED16	Reading/writing	2.4
100D	VAR	Life time factor	UNSIGNED8	Reading/writing	2.4
1010	ARRAY	Store parameters	UNSIGNED32	Reading/writing	1.2
1011	ARRAY	Restore dafault parameters	UNSIGNED32	Reading/writing	1.2
1014	VAR	COB-ID EMCY	UNSIGNED32	Reading/writing	2.3
1015	VAR	Inhibit Time EMCY	UNSIGNED16	Reading/writing	2.3
1018	RECORD	Identity Object	Identity (23h)	Reading	
Server SDO Parameter					
1200	RECORD	1st Server SDO parameter	SDO parameter	Reading/writing	2.1
1201	RECORD	2nd Server SDO parameter	SDO parameter	Reading/writing	2.1
1202	RECORD	3rd Server SDO parameter	SDO parameter	Reading/writing	2.1
1203	RECORD	4th Server SDO parameter	SDO parameter	Reading/writing	2.1
Receive PDO Communication Parameter					
1400	RECORD	1st receive PDO parameter	PDO CommPar	Reading/writing	2.2.2
1401	RECORD	2nd receive PDO parameter	PDO CommPar	Reading/writing	2.2.2
1402	RECORD	3rd receive PDO parameter	PDO CommPar	Reading/writing	2.2.2
1403	RECORD	4th receive PDO parameter	PDO CommPar	Reading/writing	2.2.2
Receive PDO Mapping Parameter					
1600	RECORD	1st receive PDO mapping	PDO Mapping	Reading/writing	2.2.2
1601	RECORD	2nd receive PDO mapping	PDO Mapping	Reading/writing	2.2.2
1602	RECORD	3rd receive PDO mapping	PDO Mapping	Reading/writing	2.2.2
1603	RECORD	4th receive PDO mapping	PDO Mapping	Reading/writing	2.2.2
Transmit PDO Mapping Parameter					
1800	RECORD	1st transmit PDO parameter	PDO CommPar	Reading/writing	2.2.1
1801	RECORD	2nd receive PDO parameter	PDO CommPar	Reading/writing	2.2.1
1802	RECORD	3rd receive PDO parameter	PDO CommPar	Reading/writing	2.2.1
1803	RECORD	4th receive PDO parameter	PDO CommPar	Reading/writing	2.2.1
Transmit PDO Mapping Parameter					
1A00	RECORD	1st transmit PDO mapping	PDO Mapping	Reading/writing	2.2.1
1A01	RECORD	2nd transmit PDO mapping	PDO Mapping	Reading/writing	2.2.1
1A02	RECORD	3rd transmit PDO mapping	PDO Mapping	Reading/writing	2.2.1
1A03	RECORD	4th transmit PDO mapping	PDO Mapping	Reading/writing	2.2.1

2.6 Objects' dictionary : manufacturer specific profile area

The words reported in bold type can be mapped in PDO.

Index (hex)	Object	Type	Name	Description	Access
2000	VAR	INTEGER16	Block size	SDO Block size Block Download	Reading/writing
2001	VAR	DOMAIN	Tab_formati	Formats of the 200 parameters	reading
2002	VAR	DOMAIN	Tab_con_formati	Formats of the 100 connections	Reading
2003	VAR	DOMAIN	Tab_exp_int	Formats of the 64 internal values	reading
2004	VAR	DOMAIN	Tab_exp_osc	Formats of the 64 monitor's sizes	Reading
2005	VAR	DOMAIN	Tab_par_def	Values of the default parameters	Reading
2006	VAR	DOMAIN	Tab_con_def	Values of the default connections	Reading
2007	**VAR**	**INTEGER16**	**hw_software**	Sensor managed by the firmware	Reading
2008	**VAR**	**INTEGER16**	**hw_sensore**	Sensor managed by electronic card	Reading
2009	VAR	INTEGER16	K_zz	Monitor counter	Reading
200A	VAR	INTEGER16	Via_alla_conta	Monitor trigger	Reading
200B	VAR	DOMAIN	Tab_monitor_A	Data saved in the channel A of the monitor	Reading
200C	VAR	DOMAIN	Tab_monitor_B	Data saved in the channel B of the monitor	Reading
200D	ARRAY	INTEGER16	Tab_par [200]	Actual values of the parameters	Reading/writing
200E	ARRAY	INTEGER16	Tab_con [100]	Actual values of the connection	Reading/writing
200F	**ARRAY**	**INTEGER16**	**Tab_int [64]**	Actual values of the internal words	Reading
2010	**ARRAY**	**INTEGER16**	**Tab_inp_dig [32]**	Actual values of the logical input's functions	Reading
2011	**ARRAY**	**INTEGER16**	**Tab_out_dig [32]**	Actual values of the logical output's functions	Reading
2012	**ARRAY**	**INTEGER16**	**Tab_osc [64]**	Actual values of the checked words	Reading
2013	**VAR**	**UNSIGNED16**	**ingressi**	Logical status of the 8 inputs of the terminal board	Reading
2014	**VAR**	**UNSIGNED16**	**ingressi_hw**	Logical status of the 3 inputs from the power	Reading
2015	**VAR**	**UNSIGNED16**	**uscite_hw**	Logical status of the 4 digit outputs	Reading
2016	**ARRAY**	**INTEGER16**	**Tab_inp_dig_field [32]**	Values set by CAN of the output logical function	Reading/writing
2017	**VAR**	**UNSIGNED16**	**stato**	Variable of the drive's status	Reading
2018	**VAR**	**UNSIGNED16**	**allarmi**	Drive alarms' status	Reading
2019	**VAR**	**UNSIGNED16**	**abilitazione_allarmi**	Word for enabling drive's alarms	Reading
201A	**VAR**	**INTEGER16**	**f_fieldbus**	Speed reference in % of n_{MAX} in 16384	Reading/writing
201B	**VAR**	**INTEGER16**	**limit_fieldbus**	torque limit in % di Tnom in 4095	Reading/writing
201C	**VAR**	**INTEGER16**	**trif_fieldbus**	torque reference in % di Tnom in 4095	Reading/writing
201D	**VAR**	**INTEGER16**	**theta_fieldbus**	Speed reference in electr. pulses x Tpwm	Reading/writing
201E	**ARRAY**	**INTEGER16**	**Tab_dati_applicazione [100]**	Data Area available for the application	Reading/writing
201F	**VAR**	**UNSIGNED32**	**Ingressi_standard_wr**	Writing standard logical inputs	Reading/writing
2020	**VAR**	**UNSIGNED32**	**Ingressi_appl_wr**	Writing application logical inputs	Reading/writing
2021	**VAR**	**UNSIGNED32**	**Ingressi_standard_rd**	Reading standard inputs	Reading
2022	**VAR**	**UNSIGNED32**	**Ingressi_appl_rd**	Reading application inputs	Reading
2023	**VAR**	**UNSIGNED32**	**Uscite logiche_rd**	Reading logical outputs	Reading
2024	**VAR**	**UNSIGNED16**	**word_vuota**	Unused Word	Reading/writing
2025	**VAR**	**UNSIGNED32**	**double_vuota**	Unused Double word	Reading/writing
2026	VAR	DOMAIN	Tab_formati_extra	Formats of extra parameters	Reading

2.6.1 Format parameters table (Tab_format 2001h)

This table is made by 800word (200*4) 4 words for each parameter :

1st word : it defines the parameter typology, its internal representation and the number of decimal and integer digits which are shown up on the display. Each nibble has the following meaning:

0x 0 0 0 0 (in hexadecimal)

- number of digits visualised in decimal
- number of digits visualised in integer
- internal representation :

0	Direct value
1	Percent of the base (100/base)
2	Proportional to the base (1/base)
3	Direct value unsigned

- Type of parameter:

0	Not managed
1	free (changeable on-line)
2	Reserved (changeable off-line + key P60)
4	TDE (changeable off-line + key P99)

For example:

0x1231 → free parameter proportional to the base: the real value is = internal representation/base (4th word).

2nd word : it defines the min. value admitted in the internal representation of the parameter
3rd word : it defines the max value admitted in the internal representation of the parameter
4th word : it defines the representation base of the parameter

example 1: (hexadecimal if leaded by '0x…'):

1st word = 0x1131
2nd word = 0000 free parameter in percent of the base: the real value is = (internal
3rd word = 8190 representation divided by the base)*100
4th word = 4095

if the current value is 1000→ (1000/4095)*100 = 24,4%
the variation range is included between 0 and 200%

example 2 : (hexadecimal if leaded by '0x…'):

1st word = 0x2231
2nd word = 5 reserved parameter proportional to the base : the real value is
3rd word = 1000 internal representation divided by the base
4th word = 10

if the current value is 400→ (400/10) = 40,0%
the variation range is included between 0,5 and 100%

2.6.2 Format connections table (tab_with_formats 2002h)

This table is composed by 400 words (100x4), 4words for each connection:

1st word : it defines the type of connection ,its internal representation and the number of integer and decimal digits that will show up on the display. Each nibble has the following meaning:

0x 0 0 0 0 (hexadecimal)

- Number of digits visualised in decimal
- Number of digits visualised in integer
- Internal representation :

0	Direct value
1	Percent of the base (100/base)
2	Proportional to the base (1/base)

- Parameter type:

0	Not managed
1	free (changeable on-line)
2	Reserved (change off-line + key P60)
4	TDE (change off-line + key P99)

2nd word : it defines the min admitted value in the internal representation of the connection
3rd word : it defines the max admitted value in the internal representation of the connection
4th word : it defines the base of the representation of the connection (always 1)

the internal representation is always the direct value.

Example (hexadecimal if leaded by ‘0x…’) :

1st word = 0x2020
2nd word = 0 reserved connection : its value is included between 0 and 18
3rd word = 18
4th word = 1

2.6.3 Format Extra parameters table (Tab_format 2026h)

This table is made by 1000word (200*5) 5 words for each parameter :

1st word : it defines the parameter typology, its internal representation and the number of decimal and integer digits which are shown up on the display. Each nibble has the following meaning:

0x 0 0 0 0 (in hexadecimal)

- number of digits visualised in decimal
- number of digits visualised in integer
- internal representation :

0	Direct value
1	Percent of the base (100/base)
2	Proportional to the base (1/base)
3	Direct value unsigned

- Type of parameter:

0	Not managed
1	free (changeable on-line)
2	Reserved (changeable off-line + key P60)
4	TDE (changeable off-line + key P99)

For example:

0x1231 → free parameter proportional to the base: the real value is = internal representation/base (4th word).

2nd word : it defines the min. value admitted in the internal representation of the parameter
3rd word : it defines the max value admitted in the internal representation of the parameter
4th word : it defines the representation base of the parameter
5th word : it defines the default value of the parameter

example: (hexadecimal if leaded by ‘0x…’):

1st word = 0x1131
2nd word = 0000 free parameter in percent of the base: the real value is = (internal
3rd word = 8190 representation divided by the base)*100
4th word = 4095
5th word = 4095

if the current value is 1000→ (1000/4095)*100 = 24,4%
the variation range is included between 0 and 200%
the default value is 100%

2.6.4 Format of internal values table (tab_exp_int 2003h)

This table is composed by 64 words, one word for each internal value :

1st word : it defines the representation of the internal values

0x 0 0 0 0 (hexadecimal)

internal representation :

1	Direct value /2 to the power of…	
2	Percent with base 4095	
3	Percent with base 32767	
4	Percent with base 16383	

example 1 (hexadecimal if leaded by '0x…')

0x0002 internal representation of the value : percent of 4095.
For example if its value is 2040 → (2040/4095)*100 = 49,8%

Example 2 (hexadecimal if leaded by '0x…')

0x0041 internal representation of the size : direct value divided by 2^4
For example if its value is 120 → $(120/2^4)$ = 7,5

2.6.5 Format of monitor values table (tab_exp_osc 2004h)

This table is composed by 64 words, one word for each monitor value.

1st word : it defines the representation of internal values :

0x 0 0 0 0 (hexadecimal)

internal representation :

2	Percent with base 4095
3	Percent with base 32767
4	Percent with base 16383

example 1 (hexadecimal if leaded by '0x…'):

0x0003 internal representation of the internal value: percent of 32767.
For example if its value is 5000 → (5000/32767)*100 = 15,2%

2.6.6 Management of the speed sensor (hw_software 2007h and hw_sensor 2008h)

The two variables hw_software and hw_sensor can assume the following values :

value	Corresponding sensor
0	--- none ---
1	Incremental encoder
2	Incremental encoder + Hall probes
4	Resolver
8	Sinuisoidal encoder Sin/Cos analog
9	Sinuisoidal encoder Sin/Cos absolute analog
10	Endat

hw_software represents the managed sensor of the version of the drive firmware.
hw_sensor represents the sensor managed by the feedback board mounted in the drive.

2.6.7 Management of the monitor (objects from 2009h to 200Ch +2012h)

These objects are related to the monitor of the drive internal values.
K_zz (2009h) is the internal counter of the 2000 points circular buffer.
Start_count If ≠0 it indicates that the trigger event set with C14 went off
Tab_monitor_A (200Bh) and **Tab_monitor_B (200Ch)** are circular buffer where the internal values selected by C15 and C16 are stored
Moreover parameter P54,P55 and P56 are involved. P54 sets the sample time of the monitor(units = PWM period); P55 sets the post-trigger points; P56 sets the trigger level if this is effectuated on the monitored internal values
See the product documentation for detailing of the monitored internal values
The object **Tab_osc (2012h)** is an array of 64 internal values with the most recent values of all the monitoring variables. In this way the single objects can be mapped as PDOs to keep under control the internal values of the drive.

2.6.8 Input logic functions (objects 2010h, 2013h, 2014h, 2016h, 201Fh, 2020h, 2021h, 2022h)

The management of the input logic functions is totally controlled via CAN.
In the variable **inputs (2013h)** it is possible to read the status of the 8 input available in the terminal-box in the less significant bit. The 8 logic input are configured by the C1-C8 connections, each one checking a particular input logic function.

Standard input logic functions (I00 ÷ I28)

The status of the 32 input logic functions is available in two different dictionary objects:
the array **Tab_inp_dig (2010h)** in which it’s possible to read function by function using sub-index (logic state 0 = low ; 32767 = high) and the 32 bit variable **Ingressi_standard_rd (2021h)** in which every bit is related to the state of corresponding function.
Via CAN it’s possible to set the status of the input logic functions: writing function by function with the array **Tab_inp_dig_field (2016h)** (0=low, 32767=high) or setting the state of all 32 logic functions writing the 32bit variable **Ingressi_standard_wr (201Fh)**.

The implemented logic provides that:

- The 0 logic input function (drive switch on/off) is given by the logic AND of the different input channels : terminal board, field-bus and serial line
- All the other logic functions can be set high by the logic OR of the different channels.

At start up, Tab_inp_dig_field [0]=high : in this way if this value is never over-written, the drive can be controlled via terminal-board.

Application input logic functions (I29 ÷ I63)

The status of the 32 application input logic functions is available in the 32 bit variable **Ingressi_appl_rd (2022h)** in which every bit is related to the state of corresponding function.
Via CAN it's possible to set the status of all application input logic functions writing the 32bit variable **Ingressi_appl_wr (2020h)**.

The implemented logic provides that:

- The 32 application input logic functions can be set via CAN
- If one application input logic function is configured to a connector logic input, the physical state imposes the state of corresponding logic function.

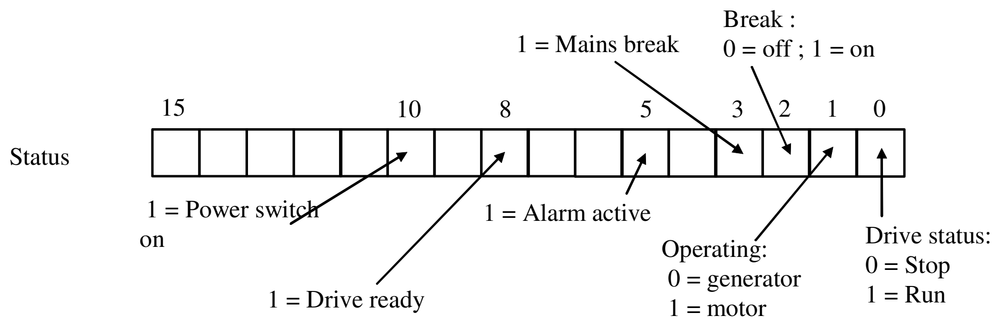
2.6.9 Output logic functions (objects 2011h, 2015h, 2023h)

Via CAN bus ,it is possible the monitoring the state of :

- the status of the 4 logic outputs in the 4 less significant bits of the variable **output (2015h)**
- the status of the 32 logic output functions in the array **Tab_out_dig (2011h)** using the sub-index. Like the inputs logic levels are: 0=low and 32767=high
- the status of all 32 output logic functions in the 32 bit variable **Uscite_logiche_rd (2023h)** in which every bit is related to the corresponding function

2.6.10 Status words (objects 2017h, 2018 and 2019h)

the object **2017h** is available as **status word** of the drive with the following meaning:



The object **2018h** is available as the status of the different **alarms** of the drive bit by bit; for example, the status of A8 alarm is shown by the bit n.8 of the word.
The object **2019h** is the alarm enabling mask. Again the meaning is bit by bit. This variable is available as read only access ; see parameter P163 for read and write access.

2.6.11 Control reference via CAN BUS (objects 201Ah,201Bh,201Ch and 201Dh)

These objects can be used to give: speed-reference, torque-reference, torque-limit to the drive. For doing this it is necessary to enable their management, setting **C52=1**.

f_fieldbus (201A) = speed reference in percent of the max speed set. Base representation is equal to16384; thus 16384 is equal to 100%.

Theta_fieldbus (201D) = speed reference in electric pulses per period of PWM, considering that there are 65536 pulses per revolution and that the term 'electric' means they must be multiplied by the number of polar pairs of the motor.

Trif_fieldbus (201C) = couple reference in percent of the nominal torque of the motor. Base of Representation = 4095 : thus 4095 is = 100%

Limit_fieldbus (201A) = torque limit in percent of the nominal torque of the motor (it is in alternative to the other existing limits, the most restricted is the one that values). Representation base is 4095 : thus 4095 = 100%