




DCB1M - UART/SPI/I²C/DMX/RDM Over Powerline Transceiver

1. Overview

The DCB1M is a device for multiplex communication over noisy power lines, at speed up to 1.4Mbit/s. The DCB1M is based on the DC-BUS™ technology for network communication between modules sharing a common DC or AC powerline. The device avoids complex cabling, saves weight, and simplifies installation. The DCB1M supports UART, SPI, DMX512, RDM, and I²C protocols, enabling the user to meet this application protocol. Sleep mode allows low power consumption when the device is not used. A QFN32 5x5 mm package provides a small PCB footprint . The DCB1M is beneficial for many applications ranging from LED, Aerospace, Automotive, Industrial, and even Toys.

Applications

- Battery management (BMS)
- Climate control network
- Sensors/actuators bus
- Robotics control network
- Lighting control
- Truck-Trailer communication
- Multiple vehicle networks sharing the same powerline
- DMX512/RDM Communication over DC power line

Features

- Noise robust UART, SPI, DMX512, RDM, and I²C transceiver over DC powerline.
- Bitrate up to 1.4Mbit/s over the powerline.
- Multiple networks may operate over a single powerline.
- 251 selectable carrier frequencies (5MHz to 30MHz).
- Built-in CSMA/CA (powerline arbitration) mechanism.
- Operates as Multi-master Transceiver in a multiplex network.
- Channel interference detection.
- Communicates over a wide range of DC voltages.
- Power management (Sleep modes) for low power consumption.

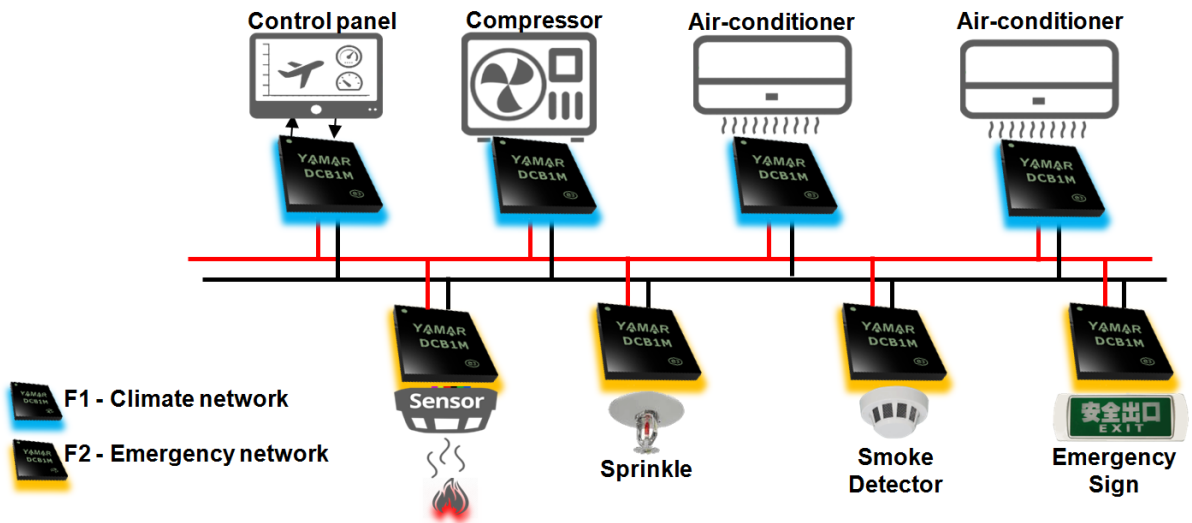


Figure 1 - DCB1M climate control and security networks sharing single powerline

Table of Content

1.	OVERVIEW	1
2.	DESCRIPTION	6
2.1	The DCB1M network	6
2.2	DCB1M channel parameters	6
2.3	Device architecture	6
2.4	Pin configuration and function	7
2.4.1	Pinout diagram	7
2.4.2	Signals and Pinout description	7
2.5	Implementation	9
2.5.1	DCB1M reference schematic	9
2.5.2	External filter (BPF)	10
2.5.3	External Crystal	10
2.5.4	L1 and L2 inductors	10
2.5.5	Optional EMC chip-bead (L5)	11
2.5.6	Powerline coupling interface	11
2.5.7	External protection network	11
2.5.8	Recommended connection to power-supply	11
2.5.9	AC Powerline coupling interface	12
3.	DCB1M OPERATION	12
3.1	DCB1M messages	12
3.1.1	Message structure (SPI/I ² C/UART)	12
3.1.2	Message structure (DMX/RDM)	13
3.1.3	Transmit flow	13
3.1.4	Receive flow	13
3.2	Device configuration	14
3.2.1	Protocol interfaces	14
3.2.2	Codec selection	14
3.2.3	DCB1M TX-FIFO and RX-FIFO handling (UART/SPI/I ² C interface only)	14
3.2.4	Carrier frequency configuration	16
3.2.5	Pins 12-13 IO function control (UART/SPI/I ² C interface only)	17
3.2.6	TXO output level and drive control	17
3.3	<i>TX-Trigger</i> mode (UART/SPI/I ² C interface only)	17
3.3.1	<i>TX-Trigger</i> mode - UART interface handling	17
3.3.2	<i>TX-Trigger</i> mode - SPI & I ² C interface handling	18
3.4	Carrier Sense mode (CS) (UART/SPI/I ² C interface only)	18
3.5	Arbitration mode (ARB) (UART/SPI/I ² C interface only)	19
3.6	Auto Arbitration mode (RDM Discovery in RDM interface only)	19
3.7	DCB1M UUID	19
4.	POWER OPERATION MODES	19
4.1	Normal mode	19
4.2	Standby mode	19
4.3	Sleep modes (power-saving)	19
4.3.1	Wake-up message (<i>WUM</i>)	20
4.3.2	Entering Sleep mode	20
4.3.3	Exiting Sleep mode	20

4.3.4	Sleep modes description	21
4.3.5	Sleep modes Examples	21
5.	DCB1M REGISTERS	23
5.1	REG_0 - 'Device Control 0' (Address 0x00) (UART/SPI/I ² C interface only)	23
5.2	REG_1 - 'Device Control 1' (Address 0x01)	23
5.3	REG_2 - 'Frequency Select' (Address 0x02)	24
5.4	REG_3 - 'Sleep & IO Control' (Address 0x03)	24
5.5	REG_4 - 'Interrupts Enable (Address 0x04) (SPI & I ² C Only)	24
5.6	REG_5 - 'Interrupt RX-FIFO Threshold 1' (Address 0x05) (SPI & I ² C Only)	24
5.7	REG_6 - 'Interrupt RX-FIFO Threshold 2' and Error status (Address 0x06)	24
5.8	Status interrupt byte (Accessed only in SPI& I ² C, through READ-INT command)	25
5.9	REG_3C - 'Interrupt TX-FIFO Thresholds1' (Address 0x3C) (SPI & I ² C Only)	25
5.10	REG_3D - 'Interrupt TX-FIFO Thresholds2' (Address 0x3D) (SPI & I ² C Only)	25
5.11	REG_3E - 'Interrupt TX-FIFO Thresholds 3' (Address 0x3E) (SPI & I ² C Only)	25
5.12	REG_7 - 'Arbitration ID 1' (Address 0x07) (UART/SPI/I ² C interface only)	25
5.13	REG_8 - 'Arbitration ID 2' (Address 0x08) (UART/SPI/I ² C interface only)	26
5.14	REG_59 – UUID[47:40] (Address 0x59)	26
5.15	REG_5A – UUID[39:32] (Address 0x5A)	26
5.16	REG_5B – UUID[31:24] (Address 0x5B)	26
5.17	REG_5C – UUID[23:16] (Address 0x5C)	26
5.18	REG_5D – UUID[15:8] (Address 0x5D)	26
5.19	REG_5E – UUID[7:0] (Address 0x5E)	26
6.	UART PROTOCOL INTERFACE	27
6.1	Interfacing to UART ECU	27
6.2	UART registers configuration (Command mode)	27
6.2.1	WRITE-REG command	27
6.2.2	READ-REG command	27
6.3	UART Bitrate configuration (Bitrate learning)	28
6.4	UART Codec Select configuration	28
6.5	UART RTR pin handling	28
6.6	UART interface typical set-up and operation	28
6.7	UART Examples	29
6.7.1	UART Example 1 - Typical communication flow	29
6.7.2	UART Example 2 - <i>WRITE-REG</i> command	29
6.7.3	UART Example 3 - <i>READ-REG</i> command	29
6.7.4	UART Example 4 – Using RTR (ready-to-receive) pin	29

7.	DMX/RDM PROTOCOL INTERFACE	30
7.1	Interfacing with DMX/RDM Controller	30
7.2	Interfacing to an existing DMX/RDM module	30
7.3	DMX/RDM interface typical set-up and operation	31
7.4	DMX registers configuration (Command mode)	31
7.4.1	WRITE-REG command	31
7.4.2	READ-REG command	31
7.5	DMX Examples	32
7.5.1	DMX Example 1 - Typical communication flow	32
7.5.2	DMX Example 2 - <i>WRITE-REG</i> command	32
7.5.3	DMX Example 3 - <i>READ-REG</i> command	32
8.	SPI PROTOCOL INTERFACE	33
8.1	Interfacing to SPI ECU	33
8.2	SPI Flow control	33
8.3	SPI Message (frame) construction	33
8.4	SPI Status interrupt byte	33
8.5	SPI Commands	34
8.5.1	<i>SPI TX-DATA</i> Command (0x04 to 0x34)	34
8.5.2	<i>SPI FRAME-END</i> Command (0x0F)	35
8.5.3	<i>SPI RX-DATA</i> Command (0x0B)	35
8.5.4	<i>SPI WRITE-REG</i> Command (0xF5)	35
8.5.5	<i>SPI READ-REG</i> Command (0xFD)	35
8.5.6	<i>SPI READ-INT</i> Command (0x01)	35
8.5.7	<i>SPI READ-ERR</i> Command (0x11)	36
8.6	SPI interface typical set-up and operation	36
8.7	SPI Examples	36
8.7.1	SPI Example 1 - <i>WRITE-REG</i> command	36
8.7.2	SPI Example 2 - <i>READ-REG</i> command	36
8.7.3	SPI Example 3 - Data frame transmission	37
8.7.4	SPI Example 4 - Data frame reception after <i>RX-frame-end</i> interrupt	37
8.7.5	SPI Example 5 - Data frame reception after <i>Rx-FIFO-not-empty</i> interrupt	37
8.7.6	SPI Example 6 - Empty frame interrupt event	37
8.7.7	SPI Example 7 - Multiple frames management	38
9.	I ² C PROTOCOL INTERFACE	39
9.1	Interfacing to I ² C ECU	39
9.2	I ² C Flow control	39
9.3	I ² C Status interrupt byte	39
9.4	I ² C Message (frame) construction	40
9.5	I ² C Commands	40
9.5.1	<i>I²C TX-DATA</i> Command (0x04 to 0x34)	41
9.5.2	<i>I²C FRAME-END</i> Command (0x0F)	41
9.5.3	<i>I²C RX-DATA</i> Command	41
9.5.4	<i>I²C WRITE-REG</i> Command (0xF5)	42
9.5.5	<i>I²C READ-REG</i> Command (0xFD)	42
9.5.6	<i>I²C READ-INT</i> Command (0x01)	42
9.5.7	<i>I²C READ-ERR</i> Command (0x11)	42

9.6	I ² C interface typical set-up and operation	43
9.7	I ² C Examples	43
9.7.1	I ² C Example 1 - <i>READ-REG</i> command	43
9.7.2	I ² C Example 2 - <i>WRITE-REG</i> command	43
9.7.3	I ² C Example 3 - <i>TX-DATA</i> command	44
9.7.4	I ² C Example 4 - <i>READ-INT</i> command	44
9.7.5	I ² C Example 5 - <i>RX-DATA</i> command	44
10.	SPECIFICATIONS	45
11.	DCB1M PCB LAYOUT RECOMMENDATION	47
12.	PACKAGE, MECHANICAL	49
12.1	Mechanical Drawing	49
12.2	PCB drawing	49
12.3	Soldering profile	50

2. Description

2.1 The DCB1M network

The DCB1M operates as part of a powerline (DC-BUS) communication network consisting of multiple DCB1M devices. Each device can transmit messages (frames) to other devices over the power lines at four selectable coding strengths and bitrates, ranging from 1.4Mbit/s down to 225Kbit/s for a very noisy channel. The data is phase modulated by a sine wave at a user predefined carrier frequency.

Each DCB1M can communicate with its host controller (ECU) using one of the supported protocols (UART, SPI, DMX512, RDM, and I²C). One ECU using SPI protocol interface can communicate with another ECU interfacing with UART or I²C protocol and vice versa. The DCB1M operates as a gateway between the ECUs' different protocols. All network topologies (e.g. Star, ring, line, tree, etc.) are applicable, as long as the RX signal level at RXI is above minimal RXI_{lev} (see Table 39).

Users may create multiple DCB1M networks operating over a single powerline, where each network communicates using a different carrier frequency (channel).

2.2 DCB1M channel parameters

Carrier frequency:	251 selectable frequencies between 5MHz - 30MHz with 100 kHz spacing.
Powerline bitrate:	1.4Mbit/s, 1Mbit/s, 490Kbit/s, 225Kbit/s
Powerline voltage:	Any, with proper powerline coupling interfacing (see 2.5.6)
Cable length:	Depends on the powerline loads AC signal-attenuation (100m is practicable)
Cable type:	Any cable.

2.3 Device architecture

Figure 2 depicts the DCB1M blocks.

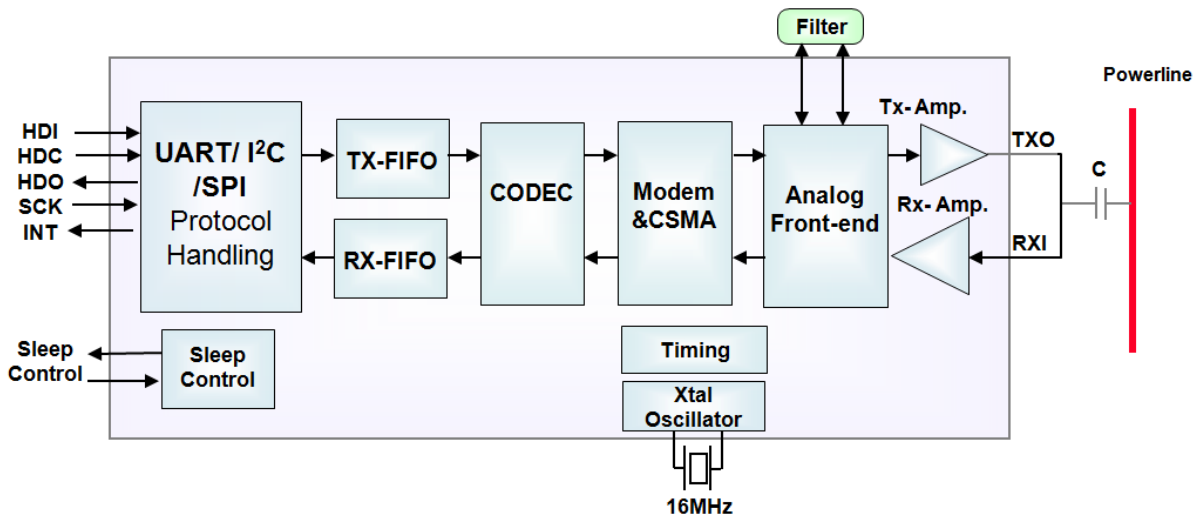


Figure 2 - DCB1M block diagram

The DCB1M main building blocks:

- **Protocol handling** - Interprets the ECU protocol.
- **Transmit and Receive FIFOs** - allows 2 x 1024 bytes data buffering between the ECU and the DCB1M.
- **CODEC** - Encodes/decodes the data according to the selected channel protection code (bitrate).
- **Modem** - Phase modulates and demodulates the data to and from the DC-BUS powerline.
- **CSMA/CA** - Allows Carrier sense and arbitration capabilities to the device
- **Sleep** - Ensures low power consumption during Sleep mode.

2.4 Pin configuration and function

2.4.1 Pinout diagram

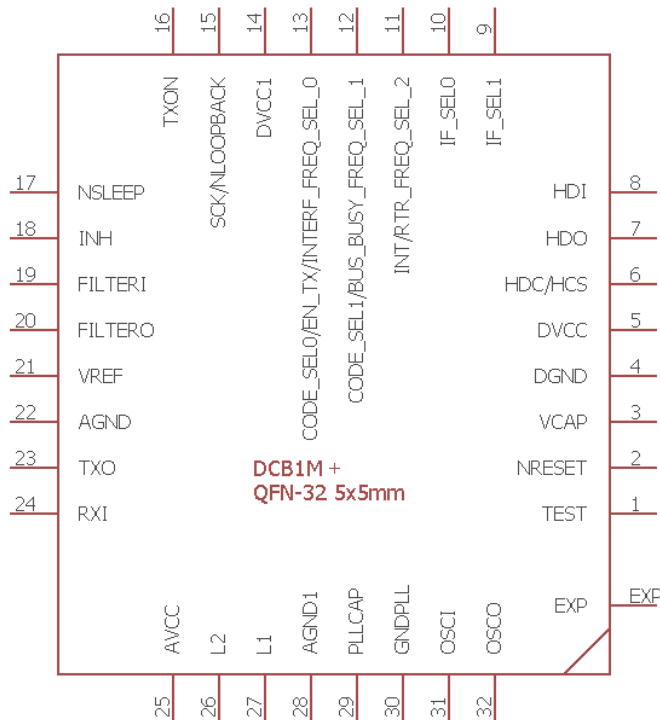


Figure 3 - DCB1M pinout diagram in QFN32 5x5mm package

2.4.2 Signals and Pinout description

Table 1 - Pinout description

Name	Pin #	Pin type	Internal PU/PD	Description										
HDO	7	Digital IO 12mA	PU (SPI, I ² C Only)	Digital IO signal. In SPI and UART interfaces, Outputs the received data from the powerline or from internal registers to the ECU. In the I ² C interface, this pin is IO and should be externally pull-up with a 10kΩ resistor.										
HDI	8	Digital input	PU	Digital data input signal. Transfers data from the ECU to the powerline or the internal registers.										
HDC /HCS	6	Digital input	PU	UART/DMX/RDM - ECU data / command input, enables read and write from/to DCB1M control registers (see section 6.2) SPI - Chip Select input.										
TEST	1	Digital Input	PD	Should be connected to GND.										
NRESET	2	Digital Input	PU	Reset, active low.										
NSLEEP	17	Digital Input		Sleep mode control input (see section 4.3). Should be pull-up to 3.3V when not in use.										
IF_SEL1	9	Digital Input		ECU interface selection inputs. <table border="1" style="width: 100%;"> <thead> <tr> <th>IF_SEL[1:0]</th> <th>Interface</th> </tr> </thead> <tbody> <tr> <td>'00'</td> <td>UART</td> </tr> <tr> <td>'01'</td> <td>DMX/RDM</td> </tr> <tr> <td>'10'</td> <td>SPI</td> </tr> <tr> <td>'11'</td> <td>I²C</td> </tr> </tbody> </table>	IF_SEL[1:0]	Interface	'00'	UART	'01'	DMX/RDM	'10'	SPI	'11'	I ² C
IF_SEL[1:0]	Interface													
'00'	UART													
'01'	DMX/RDM													
'10'	SPI													
'11'	I ² C													
IF_SELO	10	Digital Input												

Name	Pin #	Pin type	Internal PU/PD	Description																											
INT/RTR/ FREQ_SEL_2	11	Digital IO 8mA		<p>SPI/I²C - Interrupt event indication Output. See section 8.4 and 9.3.</p> <p>UART-RTR (Ready to Receive) Output. When high, ECU can transfer bytes through the HDI pin. When low, ECU should pause its data bytes transfer.</p> <p>DMX/RDM - Pre-defined frequency selection Input pin (see 3.2.4.1).</p>																											
SCK /NLOOPBACK/NC	15	Digital Input	PU (UART only)	<p>SPI/I²C - Serial Clock Input.</p> <p>UART- NLOOPBACK, when high, loopback from HDO to HDI is disabled.</p> <p>DMX/RDM – Should be left floated.</p>																											
INH	18	Digital Output 8mA		<p>When high, DCB1M is in Normal mode</p> <p>When low, DCB1M is in Sleep mode</p>																											
CODE_SELO /EN_TX /INTERF /FREQ_SEL_0	13	Digital IO 12mA		<p>UART</p> <table border="1"> <thead> <tr> <th>REG_3[5:4]</th> <th>Pin Function</th> <th>Pin Type</th> </tr> </thead> <tbody> <tr> <td>'00' (Default)</td> <td>CODE_SELO</td> <td>Input</td> </tr> <tr> <td>'01'</td> <td>EN_TX</td> <td>Input</td> </tr> <tr> <td>'10'</td> <td>INTERF</td> <td>Output</td> </tr> <tr> <td>'11'</td> <td colspan="2">Not Valid</td> </tr> </tbody> </table> <p>SPI/I²C</p> <table border="1"> <thead> <tr> <th>REG_3[5:4]</th> <th>Pin Function</th> <th>Pin Type</th> </tr> </thead> <tbody> <tr> <td>'00' (Default), '01'</td> <td>EN_TX</td> <td>Input</td> </tr> <tr> <td>'10'</td> <td>INTERF</td> <td>Output</td> </tr> <tr> <td>'11'</td> <td colspan="2">Not Valid</td> </tr> </tbody> </table> <p>DMX/RDM</p> <p>Pre-defined frequency selection Input pin (see 3.2.4.1).</p>	REG_3[5:4]	Pin Function	Pin Type	'00' (Default)	CODE_SELO	Input	'01'	EN_TX	Input	'10'	INTERF	Output	'11'	Not Valid		REG_3[5:4]	Pin Function	Pin Type	'00' (Default), '01'	EN_TX	Input	'10'	INTERF	Output	'11'	Not Valid	
REG_3[5:4]	Pin Function	Pin Type																													
'00' (Default)	CODE_SELO	Input																													
'01'	EN_TX	Input																													
'10'	INTERF	Output																													
'11'	Not Valid																														
REG_3[5:4]	Pin Function	Pin Type																													
'00' (Default), '01'	EN_TX	Input																													
'10'	INTERF	Output																													
'11'	Not Valid																														
CODE_SEL1 /BUS_BUSY /FREQ_SEL_1	12	Digital IO 8mA		<p>UART</p> <table border="1"> <thead> <tr> <th>REG_3[5:4]</th> <th>Pin Function</th> <th>Pin Type</th> </tr> </thead> <tbody> <tr> <td>'00' (Default)</td> <td>CODE_SEL1</td> <td>Input</td> </tr> <tr> <td>'01'</td> <td rowspan="2">BUS_BUSY</td> <td rowspan="2">Output</td> </tr> <tr> <td>'10'</td> </tr> <tr> <td>'11'</td> <td colspan="2">Not Valid</td> </tr> </tbody> </table> <p>SPI/I²C</p> <table border="1"> <thead> <tr> <th>REG_3[5:4]</th> <th>Pin Function</th> <th>Pin Type</th> </tr> </thead> <tbody> <tr> <td>'00' (Default)</td> <td rowspan="2">BUS_BUSY</td> <td rowspan="2">Output</td> </tr> <tr> <td>'01'</td> </tr> <tr> <td>'10'</td> <td colspan="2">Not Valid</td> </tr> <tr> <td>'11'</td> <td colspan="2">Not Valid</td> </tr> </tbody> </table> <p>DMX/RDM</p> <p>Pre-defined frequency selection Input pin (see 3.2.4.1).</p>	REG_3[5:4]	Pin Function	Pin Type	'00' (Default)	CODE_SEL1	Input	'01'	BUS_BUSY	Output	'10'	'11'	Not Valid		REG_3[5:4]	Pin Function	Pin Type	'00' (Default)	BUS_BUSY	Output	'01'	'10'	Not Valid		'11'	Not Valid		
REG_3[5:4]	Pin Function	Pin Type																													
'00' (Default)	CODE_SEL1	Input																													
'01'	BUS_BUSY	Output																													
'10'																															
'11'	Not Valid																														
REG_3[5:4]	Pin Function	Pin Type																													
'00' (Default)	BUS_BUSY	Output																													
'01'																															
'10'	Not Valid																														
'11'	Not Valid																														
TXON	16	Output 12mA		TX_ON output - High when transmission onto the powerline is active.																											
TXO	23	Analog Output Max 66 mA		<p>Powerline Transmit signal out</p> <table border="1"> <thead> <tr> <th>TXON State</th> <th>REG_1[3]</th> <th>TX level [V-p-p]</th> <th>Impedance [Ω]</th> </tr> </thead> <tbody> <tr> <td rowspan="2">High</td> <td>'0'</td> <td>1</td> <td rowspan="2">18¹</td> </tr> <tr> <td>'1' (Default)</td> <td>2</td> </tr> <tr> <td>Low</td> <td></td> <td>High Z</td> <td>5.3k²</td> </tr> </tbody> </table> <p>¹Series output impedance</p> <p>²Input impedance referenced to VREF</p>	TXON State	REG_1[3]	TX level [V-p-p]	Impedance [Ω]	High	'0'	1	18 ¹	'1' (Default)	2	Low		High Z	5.3k ²													
TXON State	REG_1[3]	TX level [V-p-p]	Impedance [Ω]																												
High	'0'	1	18 ¹																												
	'1' (Default)	2																													
Low		High Z	5.3k ²																												
RXI	24	Analog Input		Powerline receive Input																											

Name	Pin #	Pin type	Internal PU/PD	Description
VREF	21	Analog Output		Analog out reference VCC/2 for filtering capacitor. Place 1uF between VREF to AGND. The VREF is used as a virtual ground for the external analog circuitry.
FILTERI	19	Analog, Bi-directional		External filter I/O
FILTERO	20	Analog, Bi-directional		External filter I/O
OSCO	32	Analog output		16MHz Crystal Output
OSCI	31	Analog Input		16MHz Crystal Input
L1	27	Analog Input		External inductor L1 (pin capacitance should be maximal 1pF), see 2.5.4.
L2	26	Analog Input		External inductors L2 (optional), see 2.5.4.
AVCC	25	Power		Analog 3.3V supply
AGND	22,28	Power		Analog ground
VCAP	3	Power		1.8V core supply output for filtering capacitor. Place 4.7uF between VCAP and DGND.
DGND	4	Power		Digital Ground
DVCC	5,14	Power		Digital 3.3V supply
GNDPLL	30	Power		Analog Ground
PLLCAP	29	Power		PLL 1.8V output to a filtering capacitor. Place 1uF between PLLCAP and GNDPLL.
EXP	33	Power		Expose pad, should be connected to DGND.

PD – Internal Pull-down resistor 50K ohm +/-%30

PU – Internal Pull-up resistor 50K ohm +/-%30

2.5 Implementation

2.5.1 DCB1M reference schematic

Figure 4 depicts a typical DCB1M schematic.

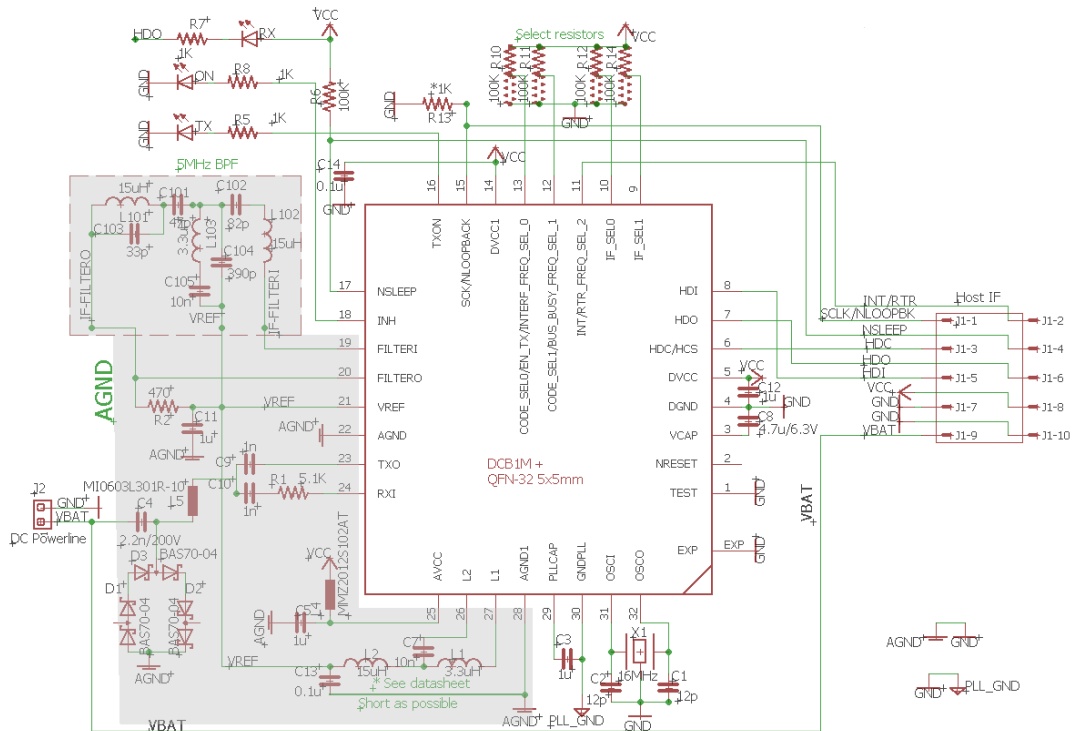


Figure 4 - DCB1M reference schematic

2.5.2 External filter (BPF)

The DCB1M operates using an external 5MHz bandpass filter. The minimum allowable bandwidth of the filters is +/- 700 kHz @ 3dB. Narrower bandwidth limits the maximal bitrate.

Figure 5 depicts a recommended 5MHz discrete passive filter.

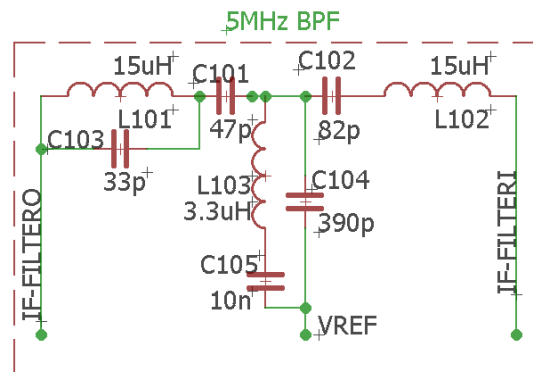


Figure 5- 5MHz bandpass filter

2.5.3 External Crystal

The device operates with a low cost, small size 16MHz crystal connected between OSCI and OSCO pins. Each of these pins should be connected to the DGND via a load capacitor. The load capacitors values should be determined according to the crystal manufacturer's recommendations and the actual PCB layout. The PCB traces should be as short as possible.

The overall frequency tolerance should not exceed ± 50 ppm.

2.5.3.1 Recommended crystals

- NDK - NX2520SA-16MHz, SMD, 2.5x2 mm
- NDK - NX3225SA/GB-16MHz, SMD, 3.2x2.5mm
- NDK - NX2016GC-16MHz, SMD, 2.0x1.6mm
- ECS - ECS-160-12-37B-CTN-TR, SMD, 2.0x1.6mm

2.5.3.2 16MHz clock from an external source

It is possible to operate the device from an external 16MHz clock source that meets the requirements above.

Figure 6 depicts an external 16MHz clock connection to the device.

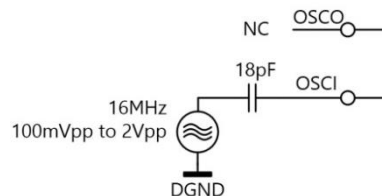


Figure 6 - External 16MHz clock connection

2.5.4 L1 and L2 inductors

The DCB1M requires one or two inductors for its operation, depending on the desired operating frequency.

- For full in-band operation, 5MHz - 30MHz:
 - L1 - 3.3uH
 - L2 - 15uH with 10nF series capacitor between L2 pin and L2 inductor.
- For low in-band operation, 5MHz -12MHz:
 - L1 - 18uH
 - L2 - NC
- For high in-band operation, 12MHz - 30MHz:
 - L1 - 3.3uH
 - L2 - NC

Figure 7 depicts the in-band operation inductors' connection to pins L1 and L2.

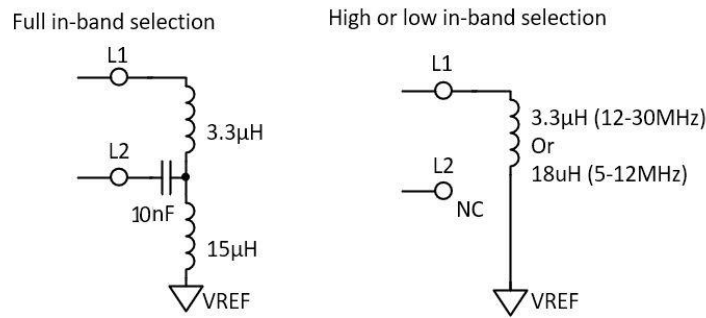


Figure 7 - L1 and L2 inductors connections

2.5.4.1 Recommended L1 & L2 inductors

Table 2 describes the recommended L1 and L2 inductors.

Table 2 - Recommended L1 and L2 manufacturers

Inductor	ABRACON	VISHAY	TDK
L1=3.3uH	815-AIML-0805-3R3K-T	ILSB0805ER3R3K	MLF2012A3R3JT000
L2=15uH	815-AIML-0805-150K-T	ILSB0805ER150K	MLF2012C150KT000
L1=18uH	815-AIML-0805-180K-T	ILSB0805ER180K	MLF2012C180KT000

2.5.5 Optional EMC chip-bead (L5)

For enhanced mitigation of high harmonics above 30MHz conducted over the powerline, it is recommended to add L5 in series to the coupling capacitor C4 (see Figure 4).

Table 3 describes the recommended EMC chip-beads.

Table 3 - Recommended L5 (optional)

LAIRD	MI0603L301R-10
LAIRD	HZ0603A222R-10
TDK	MMZ1608Q

2.5.6 Powerline coupling interface

The DCB1M is coupled to the powerline through a single small footprint coupling capacitor that blocks the DC, typically 2.2nF. The $C_{coupling}$ voltage rating depends on the powerline voltage and its expected impulses.

For high voltage powerline applications (e.g. battery monitoring system in EV or solar panels), it is required to add capacitors to achieve full galvanic isolation.

2.5.7 External protection network

It is recommended to add an external diode protection network before the $C_{coupling}$, for protection from high powerline pulses (above 2 V-P-P). The protection network consists of three Schottky diodes serially connected (for both polarities), with low capacitance (< 10pF) and fast clipping (e.g. BAS70-04).

2.5.8 Recommended connection to power-supply

Power-supplies usually require big filtering capacitors that may attenuate strongly the DCB1M carrier signal. It is recommended to add an inductor (>22uH) or ferrite bead (>100Ω @ 5MHz-30MHz) in series to the power supply connection to the DC powerline to avoid carrier signal attenuation.

Figure 8 depicts a typical DCB1M connection to a DC powerline and its power-supply.

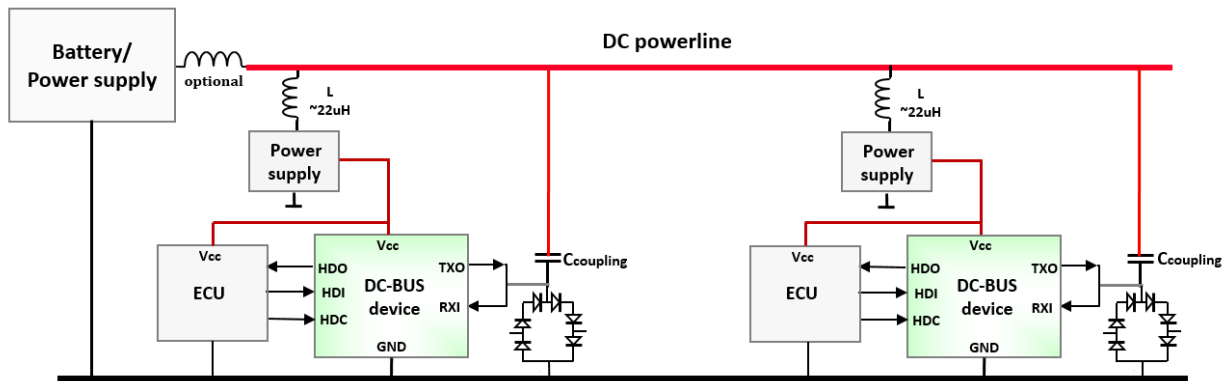


Figure 8 – DCB1M connection to 3.3V power-supply and powerline

2.5.9 AC Powerline coupling interface

Figure 9 presents a possible high voltage AC and DC powerline interface. The DCB1M high-frequency carrier is coupled to the powerline via capacitors C7 and C8. D5, D6, and D7 are external protection networks that clip the residual high voltage impulses that may pass via C7 and C8.

It is the system designer's responsibility to check the local regulations for high voltage coupling.

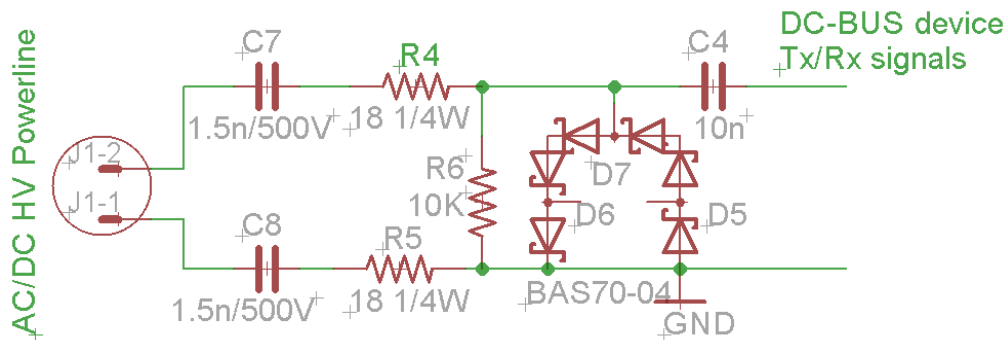


Figure 9 – Possible high voltage interface

3. DCB1M operation

3.1 DCB1M messages

3.1.1 Message structure (SPI/I²C/UART)

The DCB1M is a message-oriented device. The message is divided by the DCB1M into packets that are encoded against errors and constructed into modulated powerline message (frame).

A frame is constructed from a Start-frame consisting of a preamble and optionally an arbitration sequence (if enabled), followed by packet/s of data bytes (at least 1 packet) and terminated with a "Frame-End" indicating the last packet of the frame.

When Arbitration is enabled, unique arbitration patterns are transmitted before the start-frame preamble pattern (see section 3.5).

- Start of frame preamble length - **64usec**
- Arbitration pattern length (if enabled) - **105usec.**
- Packet length - **180usec** (codec 0, 1) / **360usec** (codec 2, 3) (see Table 5)
- Delay time between two consecutive frames - **50usec.**

An Error Correction Code [ECC] protects each data packet. The user selected powerline bitrate defines also the ECC being used. See section 0 for more details about codec selection and its configuration.

3.1.2 Message structure (DMX/RDM)

The DMX/RDM is message-oriented. The message is divided by the DCB1M into packets that are encoded against errors and constructed into modulated powerline message (frame).

A frame is constructed from a Start-frame consisting of a preamble and optionally an arbitration sequence (in RDM discovery process only), followed by packet/s of data bytes (at least 1 packet) and terminated with a "Frame-End" indicating the last packet of the frame.

When the RDM discovery process is activated by the DMX-controller, unique arbitration patterns are transmitted before the start-frame preamble pattern (see 3.6).

- Start of frame preamble length - **64usec**
- Arbitration pattern length (For RDM Discovery only) - **336 usec.**
- Packet length - **180usec**

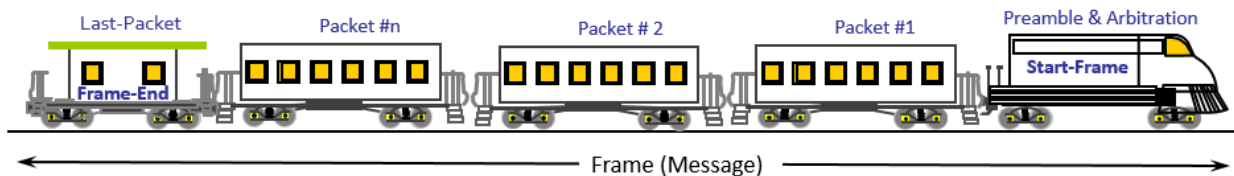


Figure 10 – Frame (Message) structure

3.1.3 Transmit flow

The DCB1M message transmission sequence over powerline depends on the selected interface protocol and codec. Using the **UART** interface, upon receiving the first data byte from ECU, the Start-frame transmission begins. The message will be closed automatically after all data bytes are extracted from the TX-FIFO (TX-FIFO is empty) and with ECU completion of data bytes transfers.

Using **SPI & I²C** interfaces, upon receiving the *TX-DATA* command, a Start-frame transmission begins. The frame is closed upon receiving the *FRAME-END* indication.

It is recommended to use the INT/RTR pin indication for pausing ECU data byte transfers when needed, to prevent TX-FIFO overflow events. For more details, refer to the protocols handling sections.

The *TX-Trigger mode* is a special feature that enables the user to control manually when to start the transmission of frame/s. Refer to section 3.3 for more details.

Using the **DMX/RDM** interface, upon detecting the DMX/RDM start of message on the HDI pin, the DCB1M initiates a new powerline frame containing the DMX/RDM message data bytes. A DMX/RDM frame contains a single DMX/RDM message sent from the DMX-controller.

3.1.4 Receive flow

In general, upon detecting a powerline frame, the frame is decoded into data bytes that are inserted into the RX-FIFO.

Using the **UART** interface, each received data byte is inserted into the RX-FIFO and then, automatically transferred to the ECU. The data transfer to the ECU will continue as long as the RX-FIFO is not empty.

Using **SPI & I²C** interfaces, ECU shall use the interrupt to perform an *RX-DATA* command to extract the stored RX-FIFO data bytes. For more details, refer to the protocols handling sections.

Using **DMX/RDM** interface, upon detecting a powerline frame, the frame is decoded back into the DMX/RDM original message and sent sequentially to the DMX-controller through the HDO pin.

3.2 Device configuration

3.2.1 Protocol interfaces

The DCB1M interfaces with UART, SPI, DMX512, RDM, and I²C protocols. The selected interface protocol is set after reset/power-up and stays valid until the next reset/power-cycle event.

DCB1M pins 9 and 10 (*IF_SEL*[1:0]) configure the interface selection as described in Table 1 and Table 4.

Table 4 - Protocols Selection

IF_SEL[1:0]	Selected Protocol
00	UART
01	DMX/RDM
10	SPI
11	I ² C

3.2.2 Codec selection

The DCB1M has four levels of error correction coding (ECC). Code 3 has the strongest data protection against powerline noises with the lowest powerline bitrate. ECU may select different codes for each frame.

The DCB1M codec selection can be set in various methods, depending on the selected interface.

In the UART interface, the user can set the codec selection either by using pins 12 and 13 (see section 2.3) or by configuring REG_0[1:0] bits. Both methods have the same priority and the last configuration change will determine the DCB1M codec selection. For example, after power-up/reset, the DCB1M will read the status of pins 12 and 13, and set the codec selection accordingly. Then, the user may change the codec selection by configuring REG_0[1:0] differently. The codec selection will change accordingly.

Table 5 describes the DCB1M codecs and their max bitrates over powerline.

Table 5 - Codec Selection

Codec Select	Max powerline bitrate (Mbit/s)	CODE_SEL1, CODE_SEL0	REG_0[1:0]
Code 0	1.4	Low, Low	'00'
Code 1	1	Low, High	'01'
Code 2	0.5	High, Low	'10'
Code 3	0.225	High, High	'11'

In SPI and I²C protocol interfaces, the user sets the codec selection inherently in the TX-DATA Command. Please refer to section 8.5.1 or 9.5.1 for more information.

In DMX/RDM interface the codec selection is fixed to level 3.

3.2.3 DCB1M TX-FIFO and RX-FIFO handling (UART/SPI/I²C interface only)

The DCB1M consists of 1024 bytes TX-FIFO and 1024 bytes RX-FIFO to buffer data between the ECU and the DCB1M. The TX-FIFO stores all the data bytes from the ECU HDI pin. Each frame stored in the TX-FIFO, contain also two extra control bytes that are not transmitted over the powerline. This information is relevant when multiple frames are stored and TX-Trigger mode enabled (see 3.3).

The RX-FIFO stores all the data bytes received from the powerline, before its transfer to the ECU via HDO pin. Each frame constructed in the RX-FIFO contains an extra control byte that is not transferred to the ECU.

See below section's examples of FIFOs threshold handling.

3.2.3.1 FIFOs interrupt thresholds configuration

Users may define FIFOs interrupts status thresholds according to the network expected payload. The interrupt is invoked on INT/RTR pin 11 (for **UART** see 6.5, for **SPI** see 8.4, for **I²C** see 9.3).

The FIFOs interrupt threshold control registers are described in Table 6.

Table 6 - FIFOs Interrupts threshold control

FIFO Threshold Description	Default threshold [Data bytes]	Related Control registers	Comments
Rx-FIFO-not-empty [9:0]	256	REG_6[1:0],	Applicable to SPI & I2C Only. Defines how many data bytes are inserted to RX-FIFO before the interrupt

		REG_5[7:0]	raise. Indicate the ECU to start reading the received frame. See <i>Example 1 - Rx-FIFO-not-empty threshold setting</i>
<i>Tx-FIFO-almost-full[9:0]</i>	1012	REG_3D[1:0], REG_3C[7:0]	For SPI & I²C: Defines how many bytes inserted to the TX-FIFO before the interrupt rises, Indicates the ECU to stop transferring data to the DCB1M before TX-FIFO overflow may occur. For UART: The RTR will always drop when TX-FIFO is inserted with >= 1022 bytes. The RTR remains low until TX-FIFO is below the <i>Tx-FIFO-almost-full[9:0]</i> threshold. During RTR low time, ECU must stop transferring data to the DCB1M. See <i>Example 2 - Tx-FIFO-almost-full threshold setting</i>
<i>Tx-FIFO-almost-empty [9:0]</i>	12	REG_3E[5:0], REG_3D[7:4]	Applicable to SPI & I ² C Only. Allow user to re-start data transfer to the TX-FIFO safely, until <i>Tx-FIFO-almost-full</i> event. <i>Example 3 - Tx-FIFO-almost-empty threshold setting</i>

Example 1 - Rx-FIFO-not-empty threshold setting

Configuration *Rx-FIFO-not-empty* threshold to 0x384 results in interrupt triggering when there are at least 900 data bytes are stored in Rx-FIFO:

REG_6 = 0x03
REG_5 = 0x84

Example 2 - Tx-FIFO-almost-full threshold setting

The *Tx-FIFO-almost-full* interrupts indicate when the ECU transmitter needs to pause any data transfer to prevent the Tx-FIFO overflow event.

In general, in the case of an interrupt event on *Tx-FIFO-almost-full*, data transfer should be paused until the interrupt on *Tx-FIFO-almost-empty* is received.

It is recommended to configure the *TX-FIFO_{almost_full_thrs}* as given in Definition of equation (1).

Definition of equation

$$TX-FIFO_{almost_full_thrs} = \left(\frac{1024}{bytes\ per\ frame + 2} \right) * bytes\ per\ frame \quad (1)$$

EXAMPLE A ECU sends 30 data bytes in each frame:

$$TX-FIFO_{almost_full_thrs} = \left(\frac{1024}{30+2} \right) * 30 = 960$$

EXAMPLE B ECU sends 500 data bytes per frame:

$$TX-FIFO_{almost_full_thrs} = \left(\frac{1024}{500+2} \right) * 500 = 1019.9 = 1019$$

Example 3 - Tx-FIFO-almost-empty threshold setting

The *Tx-FIFO-almost-empty* interrupts indicate when the ECU transmitter can re-start its byte transfer after the transmission paused due to *Tx-FIFO-almost-full* previous event.

It is recommended to configure the *TX-FIFO_{almost_empty_thrs}* as given in Definition of equation(2).

Definition of equation

$$TX-FIFO_{almost_empty_thrs} = TX-FIFO_{almost_full_thrs} - Bytes\ per\ frame \quad (2)$$

With EXAMPLE A:

$$TX-FIFO_{almost_empty_thrs} = 960 - 30 = 930$$

With **EXAMPLE B:**

$$TX-FIFO_{almost_empty_thrs} = 1019 - 500 = 519$$

For more details, please refer to the RTR/INT handling in each one of the protocol's sections.

3.2.3.2 FIFOs reset control (Soft-Reset event)

ECU may reset the TX-FIFO and RX-FIFO stored data by activating a DCB1M *Soft-reset* event.

Using the **UART** interface, a *Soft-reset* is activated while the HDC pin 6 is low.

Using **SPI & I²C** interfaces, a *Soft-reset* is activated while performing a WRITE-REG command or READ-REG command.

During *Soft-reset*, the DCB1M performs only write and read to/from DCB1M control registers. Neither transmission nor reception to/from the powerline is available. The TX-FIFO and RX-FIFO are kept in reset.

3.2.4 Carrier frequency configuration

Users can define carrier frequency from 5MHz to 30MHz with a spacing of 100 kHz (Total of 251 selectable carriers). The active carrier frequency selection is made by configuring REG_2 (see 5.3). Upon completion of configuration, the DCB1M will update its operating carrier frequency within 1msec. During this 1msec period, the DCB1M is kept in Soft-reset and will not communicate with its ECU nor detect new frames (messages) from the powerline.

When setting multiple DCB1M networks to operate over a single powerline, it is recommended to select carrier frequencies spaced more than 1.5MHz from each other. ¹

The carrier-selected value is calculated as given in the Definition of equation (3).

Definition of equation

$$REG_2 = (\text{Carrier Freq. [MHz]} - 5) * 10 \quad (3)$$

EXAMPLE 1

- ❖ When setting the frequency to 14.1MHz:

$$REG_2 = (14.1 - 5) * 10 = \mathbf{0x5B}$$

EXAMPLE 2

- ❖ When Setting to 5MHz:

$$REG_2 = (5 - 5) * 10 = \mathbf{0x00}$$

¹ It is recommended to always configure REG_2 last during the DCB1M configuration routine by the user (if the configuration is required).

3.2.4.1 Carrier frequency configuration using **FREQ_SEL[2:0]** pins (**DMX/RDM interface only**)

When using the DMX/RDM interface, after a power-up or hard-reset event, the carrier frequency is set according to FREQ_SEL[2:0] state (see Table 7). Then, the carrier frequency is updated according to either REG_2 configuration or FREQ_SEL[2:0] changed status. The last action prevails².

Table 7 describes FREQ_SEL[2:0] frequency selection allocation.

Table 7 – FREQ_SEL[4:0] frequency selection

FREQ_SEL[2:0]	Carrier Frequency [MHz]
3'b000	REG_2 last configuration (default 13MHz, after power-up/hard-reset event)
3'b001	8
3'b010	13
3'b011	16
3'b100	20
3'b101	23
3'b110	26
3'b111	30

² FREQ_SEL[2:0] PINS MUST BE PULLED UP OR DOWN BY THE USER.

3.2.5 Pins 12-13 IO function control (UART/SPI/I²C interface only)

The DCB1M pins 13 and 12 have several configurable functionalities, as describes in Table 8 and Table 9.

- **CODE_SEL0** and **CODE_SEL1** – digital inputs, for codec selecting in UART only.
- **EN_TX** – digital input - enables manual transmission over the powerline (see section 3.3).
- **INTERF** – digital output - high while an interference signal is being detected in the operating carrier frequency.
- **BUS_BUSY** – digital output, high when DCB1M is transmitting over the powerline or during the reception from the powerline. This function can be used to monitor the status of the powerline channel and act accordingly (wait for the completion of reception and transmission).

Table 8 - UART Interface pins functionality

REG_3[5:4]	Pin 13 Function	Pin 13 Type	Pin 12 Function	Pin 12 Type
'00' (Default)	CODE_SEL0	Input	CODE_SEL1	Input
'01'	EN_TX	Input	BUS_BUSY	Output
'10'	INTERF	Output		
'11'	Not Valid		Not Valid	

Table 9 - SPI & I²C Interface pins functionality

REG_3[5:4]	Pin 13 Function	Pin 13 Type	Pin 12 Function	Pin 12 Type
'00' (Default), '01'	EN_TX	Input	BUS_BUSY	Output
'10'	INTERF	Output		
'11'	Not Valid		Not Valid	

3.2.6 TXO output level and drive control

The TXO pin output level and drive capability to the powerline are controlled by REG_1[3], as described in Table 10.

Table 10 - TXO signal level

TXON State	REG_1[3]	TX level [V-p-p]
High	'0'	1
	'1' (Default)	2
Low (Rx)		High Z

Setting the TXO output drive capability is made by configuring REG_1[0], as described in Table 11.

Table 11- TXO output drive control

TXON State	REG_1[0]	Output drive [A]	Impedance [Ω]
High	'0' (Default)	33mA	18 ¹
	'1'	66mA	
Low (Rx)		Disabled	5.3k ²

¹Series output impedance

²Input impedance referenced to VREF

3.3 TX-Trigger mode (UART/SPI/I²C interface only)

As explained in section 3.1.3, in TX Normal mode, as soon as the TX-FIFO is filled with data byte, a powerline frame transmission begins. The *TX-Trigger* mode disables the auto frame transmission. ECU can manually start a frame transmission over the powerline.

The *TX-Trigger* mode is enabled by setting [REG_0\[5\]](#), along with setting pin 13 to EN_TX and pin 12 to BUS_BUSY functionality (see section 3.2.5).

3.3.1 TX-Trigger mode - UART interface handling

When TX trigger mode is enabled, each data byte is transferred from ECU to the TX-FIFO. The DCB1M will not start automatically a frame transmission over the powerline. To start the frame transmission, ECU shall toggle the EN_TX pin high for at least 100nsec and then pull it low. The new frame transmission starts and will automatically end when TX-FIFO is empty and ECU finishes data bytes transfers.

3.3.2 TX-Trigger mode - SPI & I²C interface handling

When TX trigger mode is enabled, each data byte transferred from ECU is stored in the TX-FIFO. The DCB1M will not start automatically a frame transmission over the powerline. To start a powerline frame transmission, ECU shall toggle the EN_TX pin high for at least 100nsec and then pull it low.

The new frame transmission starts and will automatically end at the first *FRAME-END* command extracted from TX_FIFO. It means that the user may insert multiple frames into the TX-FIFO by sending the TX-DATA command, followed by frame data bytes, followed by the *FRAME-END* command, multiple times. Each one of the frames is transmitted separately according to the EN_TX toggle event.

For example, SPI ECU transfers the following sequence of 2 frames as described in Table 12.

Table 12 - SPI multiple frames example

ECU bytes transfer	Frame	Comment
1. TX-DATA command	A	<i>start-of-frame byte</i> of Frame A
2. Data byte 0x44		Frame A first data byte
3. Data byte 0x10		Frame A second data byte
4. FRAME-END command		<i>end-of-frame byte</i> of Frame A
5. TX-DATA command	B	<i>start-of-frame byte</i> of Frame B
6. Data byte 0x33		Frame B only data byte
7. FRAME-END command		<i>end-of-frame byte</i> of Frame B

At this point, the TX-FIFO holds two separated frames, frame A contains two data bytes (0x44 and 0x10), and frame B contains one data byte (0x33).

At the first EN_TX toggle, frame A is transmitted. With the second EN_TX toggle, frame B is transmitted.

In general, the user may use the BUS_BUSY pin for an indication of whether the DC-BUS is idle to start new frame transmission.

Figure 11 depicts a *TX-Trigger* network example. In this example, one DCB1M network contains five nodes (Assigned with ID 0 to 4). A master node (ID0) transmits a broadcast information request to all four nodes. All other DCB1M nodes will decode the message whereas only node ID1 shall respond first. During the node ID0 response frame, the rest of the nodes can gather their information and transfer it to their DCB1M TX-FIFO. After node ID0 completed its response frame, the BUS_BUSY pin will go low, then node ID2 can trigger its stored frame, then node ID3 can trigger its frame, and so on.

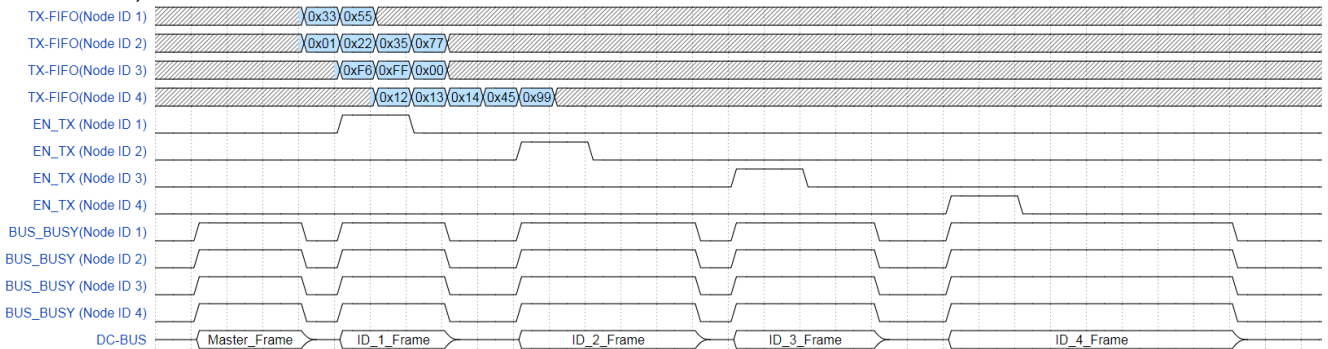


Figure 11 - TX-Trigger example

3.4 Carrier Sense mode (CS) (UART/SPI/I²C interface only)

The Carrier Sense (CS) allows the DCB1M to sense the powerline before starting transmission. It prevents interfering with other DCB1M node's transmission. In this mode, the DCB1M check if the DC-BUS (powerline channel) is idle or if there is currently an ongoing frame transmission. When CS is enabled and the ECU starts to transfer data bytes to the DCB1M, the device will start transmission only if the DC-BUS is idle. If the DC-BUS is not idle, the device will wait until the DC-BUS idle again, and only then, the transmission will start. To enable the CS mode, set [REG_0\[3\]](#) to high (Default CS is disabled).

3.5 Arbitration mode (ARB) (UART/SPI/I²C interface only)

The Arbitration mode is used to prioritize messages over powerline in multiple Master network topology. If two or more devices try to gain access to the DC-BUS at the same time, the arbitration mechanism is used to allow the message with higher priority (lower arbitration ID) to gain access and transmit its message over the DC-BUS. When Arbitration mode is enabled, the message will begin with a carrier sense period followed by the arbitration sequence. If the arbitration has passed successfully then the DCB1M will proceed with data frame transmission, else, the DCB1M will abort the transmission and will wait for the received message from other DCB1M to finish before automatically try to retransmit. The total arbitration period is 105usec. The arbitration ID is user-defined by configuring the *Arbitration ID*[10:0] stored in [REG 7](#) and [REG 8](#). To enable the Arbitration mode, set [REG 0\[4\]](#) high (Default ARB is disabled).

3.6 Auto Arbitration mode (RDM Discovery in RDM interface only)

The Arbitration mode is used to prioritize messages over the powerline when the RDM discovery process is activated by the DMX-controller. The arbitration mechanism is used to allow the message with higher priority (lower arbitration ID) to gain access and transmit its message over the DC-BUS. When an RDM discovery response message from an RDM slave is detected, the powerline frame will begin with an arbitration sequence to gain access to the powerline. In case the arbitration has passed successfully, then the DCB1M will proceed with data frame transmission, else, the DCB1M will abort the transmission and receive the message from another DCB1M device.

3.7 DCB1M UUID

Each DCB1M device is hard-coded with a 48 bit universally unique identifier (UUID[47:0]).

The UUID is stored in REG_59 to REG_5E and can be retrieved using the READ-REG commands (see 5.14 to 5.19).

4. Power operation modes

The DCB1M has three power operation modes; Normal, Standby, and Sleep.

4.1 Normal mode

In Normal mode, the DCB1M is either in RX mode, listening for a powerline message, or in TX mode, transmitting a message over the powerline.

4.2 Standby mode

The DCB1M enters Standby mode upon wake-up from Sleep mode, while the NSLEEP pin is still low. The DCB1M is kept in *Soft-reset*, whereas communication with the ECU is suspended until the NSLEEP pin is set High.

4.3 Sleep modes (power-saving)

The DCB1M has four Sleep modes for best power consumption/performance during Sleep. During this mode, only a small amount of hardware is operational mainly to detect wake-up messages (*WUM*) from the powerline and returning to Normal mode operation.

Table 13 describes the DCB1M sleep modes.

Table 13 - Sleep modes description

Sleep mode	Description	Typical Power consumption [A]	Performance
Enhanced sleep (SLP1)	The device wakes-up every 32ms to sense the powerline for WUM detection.	120μ	Wake-up detection with-in 64mSec. Best detection in a noisy environment.
Fast wake-up (SLP2)	The device continuously monitors the powerline for WUM detection.	1000μ	Fast wake-up detection with-in 250uSec.
Low-power (SLP3)	The device wakes-up every 32ms to sense the powerline for WUM detection.	85μ	Wake-up Detection with-in 64mSec.
Deep Sleep (SLP4)	The device does NOT wake-up to sense for bus activity, staying in deep sleep. Wake-up only locally by the ECU.	65μ	No bus wake-up detection.

The Sleep modes use four interface pins as described in Table 14.

Table 14- Sleep interface pins

NSLEEP	Digital input	High - Normal mode is active. Low - Sleep /Standby mode is active.
INH	Digital output	Output indication to Inhibit ECU. High - Normal mode is active. Low - Sleep mode is active.
HDO	Digital output	Normal mode - data output to ECU. Sleep/Standby mode - asserted low while the wake-up message is being detected/transmitted over the powerline.
HDC	Digital input	Normal mode - ECU Command mode / chip select. Sleep mode - ECU wakes-up the DCB1M locally by toggling the HDC high-low-high. The DCB1M then exit the Sleep mode to Standby mode (NSLEEP still asserted low), or Normal mode (NSLEEP is high).

4.3.1 Wake-up message (WUM)

When *Auto-WUM* is enabled (REG[3]='1'), upon the rise of the NSLEEP pin, the DCB1M transmits a broadcast *WUM* over the powerline, to wake-up all network-connected devices.

ECU can configure the length of the WUM as described in Table 15.

Table 15 - Wake-up message length configuration

REG_3[2]	Wake-up message length
0	SLP2 - 250usec / SLP1, SLP3 - 75msec
1	SLP2 - 1.5msec / SLP1, SLP3 - 150msec

During WUM transmission, the HDO pin is asserted low until WUM transmission is completed, indicating to the ECU the wake-up process status. ECU shall wait for the HDO rise, before initiating new bytes transfer.

4.3.2 Entering Sleep mode

During Sleep mode, the device is kept in a Soft-reset state and will not transfer data bytes from the ECU nor receive data frames from the powerline. When the device enters Sleep mode, the INH pin is asserted low.

There are two ways to enter Sleep mode;

4.3.2.1 Enter Sleep by NSLEEP

By asserting the NSLEEP pin low, the DCB1M will enter Sleep mode.

4.3.2.2 Enter Sleep by register setting

By setting REG_3[7] high, the DCB1M will enter Sleep mode, and reset automatically REG_3[7] to low.

4.3.3 Exiting Sleep mode

There are three ways to exit Sleep mode. When exiting Sleep mode, the INH pin is raised and the device switches to Standby or Normal mode.

4.3.3.1 Exit Sleep by WUM detection

Upon detection of a *WUM*, the device immediately exits Sleep mode, INH pin rises and the device enters Standby mode.

In case the NSLEEP pin is low, the device remains in Standby mode, where the device is kept in *Soft-reset*.

In case the NSLEEP pin is high, the device immediately switches to Normal mode.

During WUM reception, the HDO pin is asserted low until WUM reception is completed, indicating the ECU on the wake-up process status. ECU shall wait for HDO to rise, before initiating new bytes transfer.

4.3.3.2 Exit sleep by NSLEEP pin

Upon detection of NSLEEP pin rise, the device immediately exits Sleep mode, INH pin rises, and enters Normal mode. When *Auto-WUM* is enabled, a WUM is transmitted over the powerline (see 4.3.1).

4.3.3.3 Exit Sleep by toggling HDC

Upon detection of HDC pin toggle high-low-high, the device immediately exits Sleep mode, INH pin rises, and enters Standby mode.

In case the NSLEEP pin is still low, the device remains in Standby mode, where the device is kept in *Soft-reset*.

In case the NSLEEP pin is high, the device immediately switches to Normal mode.

In this case, the WUM will NOT be transmitted over the powerline.

ECU shall use the HDC pin to exit Sleep mode when the NSLEEP pin is not connected.

4.3.4 Sleep modes description

ECU can select between four Sleep modes (see 5.4).

4.3.4.1 Enhanced Sleep mode (SLP1)

By setting REG_3[1:0] = '00', the enhanced Sleep mode (SLP1) is selected.

When entering SLP1, the device wakes-up every 32ms periodically to monitor (sense period) for activity on the powerline. If a WUM is detected, the device exit Sleep modes as described in section 4.3.3.1, otherwise the device return to Sleep mode until the next sense period, and so on...

4.3.4.2 Fast wake-up Sleep mode (SLP2)

By setting REG_3[1:0] = '01', the Fast wake-up Sleep mode (SLP2) is selected. The device continuously monitors the powerline for WUM detection. It allows fast WUM detection within 250usec. When WUM is detected, the device exit Sleep mode as described in section 4.3.3.1.

4.3.4.3 Low-power Sleep mode (SLP3)

By setting REG_3[1:0] = '10', the low-power mode (SLP3) is selected. The device wakes-up every 32msec periodically to monitor (sense period) for activity on the powerline. If a WUM is detected, the device exit Sleep modes as described in section 4.3.3.1, otherwise the device return to Sleep mode until the next sense period.

4.3.4.4 Deep Sleep mode (SLP4)

By setting REG_3[1:0] = '11', the Deep Sleep mode (SLP4) is selected. The device will NOT wake-up to monitor (sense) the powerline for activity, rather than stay in deep sleep, whereas all its analog resources are shut down to maintain the lowest power consumption.

The device can exit Deep Sleep mode locally only, either by the NSLEEP or by HDC pins (see 4.3.3.2 and 4.3.3.3).

4.3.5 Sleep modes Examples

4.3.5.1 Sleep Example 1 - Enter by NSLEEP, Exit Sleep mode by NSLEEP & WUM

Figure 12 depicts entering sleep by NSLEEP and exit sleep by NSLEEP pin (Node A) and WUM detection (Node B).

In this example, the ECU wakes-up device Node A by raising the NSLEEP pin. Upon pull-up the NSLEEP pin, the INH pin is raised and a WUM is transmitted over the powerline (*Auto-WUM* is enabled) to wake-up Node B.

While transmitting the WUM, device Node A asserts HDO pin low. After completion of WUM transmission, the HDO is raised again (can be used as signal/interrupt to ECU). At the Node B side, during its sensing period (e.g. SLP1), the WUM is detected, and the INH rises while switching to Standby mode. Node B HDO pin is asserted low for the remaining duration of WUM reception. Then, ECU Node B raises the NSLEEP pin and the device switches to Normal mode.

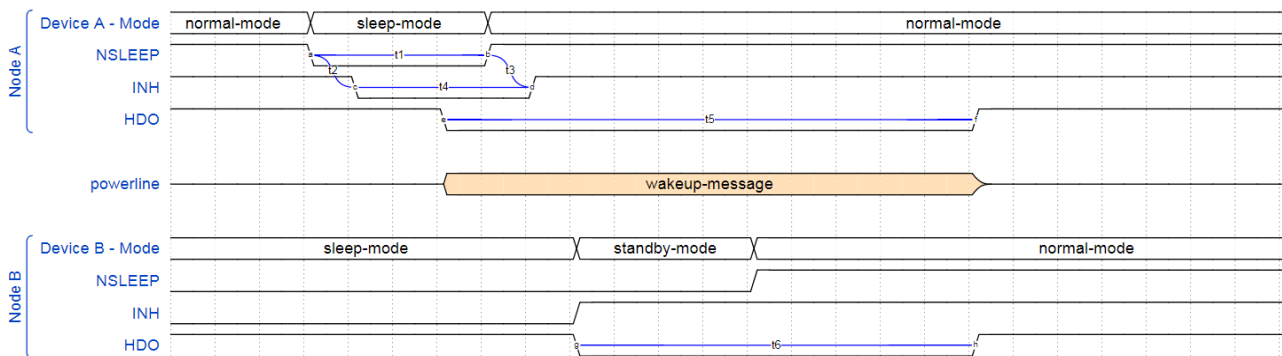


Figure 12 - Enter sleep by NSLEEP, Exit sleep by NSLEEP & WUM

4.3.5.1 Sleep Example 2 - Enter sleep by control register bit, exit sleep by HDC

Figure 13 depicts entering sleep by setting REG_3[7] high and exiting Sleep mode by toggling the HDC pin. In this example, ECU configured REG_3[7] high using Command mode (UART interface), the device enters Sleep mode, and INH pin drops. After a while, ECU toggle HDC pin low to high, and the device exits Sleep mode without transmitting the WUM, raising the INH pin and switching to Normal mode again.

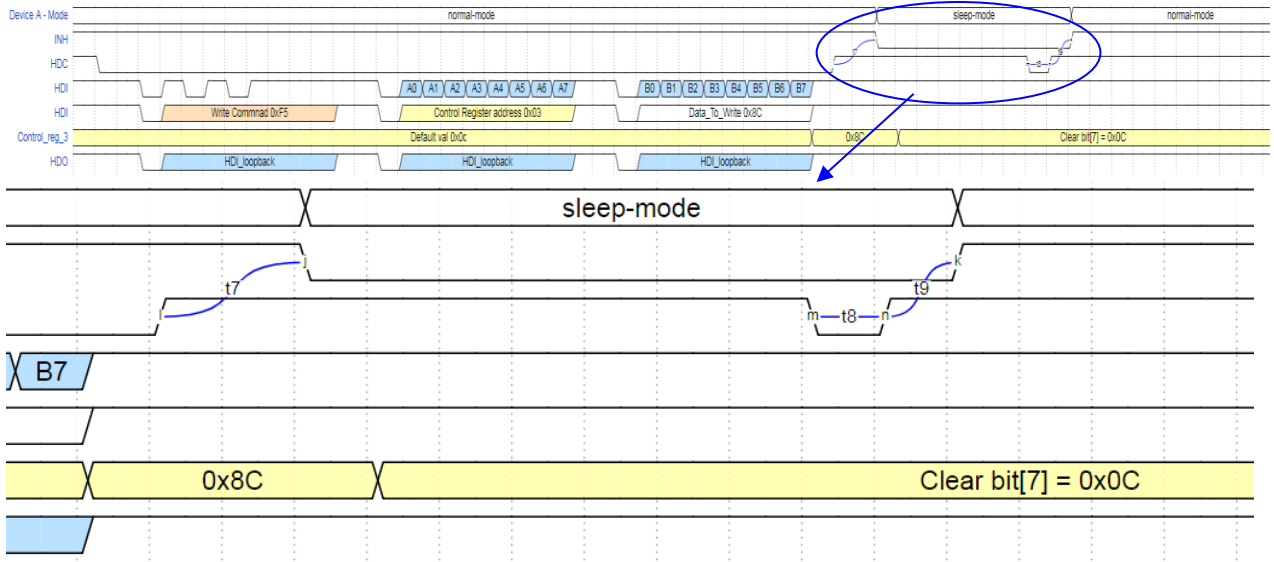


Figure 13 - Enter sleep by control register bit, Exit sleep by HDC

5. DCB1M Registers

The DCB1M contains internal registers for configuration and status checks. Each of these registers is accessible by the ECU for *Read* and *Write* operations. The access to these registers is different for each interface protocol and described in the protocol interfaces sections. This section elaborates on the registers and their default values after power-up/reset. Also, a read-only byte containing the Interrupt status can be fetched with SPI and I²C interfaces.

Table 16 - Registers summary table

Register name	Addr.	Description
REG_0 - 'Device Control 0'	0x00	Codec selection, Loopback, Arbitration, Carrier sense
REG_1 - 'Device Control 1'	0x01	Transmit level control
REG_2 - 'Frequency Select'	0x02	Carrier frequency selection
Reg_3 - 'Sleep & IO Control'	0x03	Sleep modes and IO pins
REG_4 - 'Interrupts Enable'	0x04	Enable Interrupts control
REG_5 - 'Interrupt RX-FIFO Threshold 1'	0x05	RX-FIFO not empty threshold lower nibble
REG_6 - 'Interrupt RX-FIFO Threshold 2', Device error status	0x06	RX-FIFO not empty threshold higher nibble, device errors status
REG_3C - 'Interrupt TX-FIFO Thresholds 1'	0x3C	Tx-FIFO almost full threshold lower nibble
REG_3D - 'Interrupt TX-FIFO Thresholds 2'	0x3D	Tx-FIFO almost full threshold higher nibble, Tx-FIFO almost empty threshold lower nibble
REG_3E - 'Interrupt TX-FIFO Thresholds 3'	0x3E	FIFO almost empty threshold high nibble
REG_7 - 'Arbitration ID 1'	0x07	Arbitration Id low nibble
REG_8 - 'Arbitration ID 2'	0x08	Arbitration Id higher nibble
Status Interrupt byte		Interrupts status (Accessible by <i>READ-INT</i> command during <i>TX-DATA</i> routine, SPI, I ² C only).
REG_59 - DCB1M UUID[47:40]	0x59	Read only - UUID[47:40]
REG_5A - DCB1M UUID[39:32]	0x5A	Read only - UUID[39:32]
REG_5B - DCB1M UUID[31:24]	0x5B	Read only - UUID[31:24]
REG_5C - DCB1M UUID[23:16]	0x5C	Read only - UUID[23:16]
REG_5D - DCB1M UUID[15:8]	0x5D	Read only - UUID[15:8]
REG_5E - DCB1M UUID[7:0]	0x5E	Read only - UUID[7:0]

5.1 REG_0 - 'Device Control 0' (Address 0x00) (UART/SPI/I²C interface only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R/W [1]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [1]	R/W [1]
Reserved	nLoopBack	Enable Tx trigger mode	Enable arbitration	Enable carrier sense	0	Codec_Sel[1]	Codec_Sel[0]

Bit [1:0] - Codec Select - Selects the coding strength (Maximal powerline bitrate) of the transmitted message (UART only, see 0).

Bit [2] - '0'

Bit [3] - Enable Carrier Sense mode (see section 3.4).

Bit [4] - Enable Arbitration mode (see section 3.5).

Bit [5] - Enable TX trigger mode (see section 3.3).

Bit [6] - 'nLoopBack' -Set this bit to disables loopback between HDI to HDO (UART only)

Bit [7] - Reserved

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.2 REG_1 - 'Device Control 1' (Address 0x01)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
[1]	[1]	[1]	[1]	R/W [1]	[0]	[0]	R/W [0]
1	1	1	1	TX signal level	0	0	Enable TXO high power

Bit [0] - Enable TXO high power. Set this bit to enable maximal TXO drive of 66mA, clear this bit for maximal TXO drive of 33mA (see section Table 11).

Bit [1] - '0'

Bit [2] - '0'

Bit [3] - TX signal level control at TXO pin: '0' = 1Vpp, '1' = 2Vpp (see section 3.2.6).

Bit [7:4] - '1111'

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.3 REG_2 - 'Frequency Select' (Address 0x02)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W [0]	R/W [1]	R/W [0]	R/W [1]	R/W [0]	R/W [0]	R/W [0]	R/W [0]
Carrier Frequency Configuration							

Bits [7:0] - Carrier Frequency configuration from in-band. Default configuration is 13MHz ¹

(See section 3.2.4 - Carrier frequency configuration).

¹ It is recommended to always configure REG_2 last during the DCB1M configuration routine by the user (if the configuration is required).

5.4 REG_3 - 'Sleep & IO Control' (Address 0x03)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W [0]	[0]	R/W [0]	R/W [0]	R/W [1]	R/W [1]	R/W [0]	R/W [0]
Enter Sleep mode	0	pin 12 and pin 13 functional selection		Auto WUM	Long WUM	Sleep modes selection	

Bit [1:0] - '00' - Enhanced Sleep mode [SLP1], '01' - Fast wake-up Sleep mode [SLP2], '10' - Low-power sleep mode [SLP3], '11' - Deep Sleep mode [SLP4] (see section 4.3).

Bit [2] - Control powerline wake-up message duration (see Table 15).

Bit [3] - Auto wake-up message (WUM): '0' disables transmission of WUM after wakeup from NSLEEP pin.

Bits [5:4] - Device pin 12 and pin 13 functional selection (see section 3.2.5).

Bit [6] - '0'

Bit [7] - Enter Sleep mode reg. Instead of entering Sleep mode through the NSLEEP pin, the user can activate the Sleep mode selected in bits [1:0], by setting bit[7]. After entering Sleep mode, bit [7] is automatically cleared to '0'.

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.5 REG_4 - 'Interrupts Enable (Address 0x04) (SPI & I²C Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W/R[1]	R/W [0]	R/W [1]	R/W [1]	R/W [1]	R/W [1]	R/W [1]	R/W [1]
SPI & I ² C Interrupts Enable [7:0]							

'1' - Interrupt bit is enabled. '0' - Interrupt bit is disabled.

Bit [0] - Tx-FIFO-almost-full enable bit

Bit [1] - Tx-FIFO-almost-empty enable bit

Bit [2] - Rx-frame-end enable bit – **Must share the same state as Bit [5] (enabled or disabled).**

Bit [3] - Rx-FIFO-not-empty enable bit

Bit [4] - Rx-FIFO-empty enable bit

Bit [5] - End-of-frame (EOF) enable bit – **Must share the same state as Bit [2] (enabled or disabled).**

Bit [6] - Empty-frame enable bit (**Default disabled**)

Bit [7] - Error-flag enable bit

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.6 REG_5 - 'Interrupt RX-FIFO Threshold 1' (Address 0x05) (SPI & I²C Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W/R[0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]
Rx-FIFO-not-empty [7:0] threshold							

Bits [7:0] - Rx-FIFO-not-empty[7:0], eight LSB of Rx-FIFO-not-empty [9:0] threshold bits[9:8] are configured in Interrupt Control 2[1:0].

Rx-FIFO-not-empty [9:0] threshold - default set to 256 data bytes

5.7 REG_6 - 'Interrupt RX-FIFO Threshold 2' and Error status (Address 0x06)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	[0]	[0]	R/W [0]	R/W [1]
DCB1M Error indication				0	0	Rx-FIFO-not-empty[9:8] threshold	

Bits [1:0] - Rx-FIFO-not-empty [9:8], Two MSB of Rx-FIFO-not-empty [9:0] threshold (SPI & I²C Only)
 Bits [3:2] - '00'.
 Bits [7:4] - DCB1M Error status indication: (In SPI & I²C, these bits are fetched using the READ-ERR command).
 Bit[4] - Reserved
 Bit[5] - Rx-FIFO overflow error indication.
 Bit[6] - Tx-FIFO overflow error indication.
 Bit[7] -Invalid command error in SPI / I²C interfaces.
R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.8 Status interrupt byte (Accessed only in SPI& I²C, through READ-INT command)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
Error-flag	Empty-frame	EOF	Rx-FIFO-empty	Rx-FIFO-not-empty	Rx-frame-end	Tx-FIFO-almost-empty	Tx-FIFO-almost-full

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

This register is accessed only in SPI& I²C via READ-INT command, see section 8.4 or 9.3.

5.9 REG_3C - 'Interrupt TX-FIFO Thresholds1' (Address 0x3C) (SPI & I²C Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W/R[1]	R/W [1]	R/W [1]	R/W [1]	R/W [0]	R/W [1]	R/W [0]	R/W [0]
<i>Tx-FIFO-almost-full [7:0] threshold</i>							

bit[7:0] - *Tx-FIFO-almost-full [7:0] threshold* - eight LSB of *Tx-FIFO-almost-full [9:0] threshold*.
 bit[9:8] - are configured in Interrupt TX-FIFO Threshold 2 register[1:0].
 The default value is set to 1012 data bytes.

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.10 REG_3D - 'Interrupt TX-FIFO Thresholds2' (Address 0x3D) (SPI & I²C Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W/R[1]	R/W [1]	R/W [0]	R/W [0]	[0]	[0]	R/W [1]	R/W [1]
<i>Tx-FIFO-almost-empty[3:0] threshold</i>				0	0	<i>Tx-FIFO-almost-full [9:8] threshold</i>	

bit[1:0] - *Tx-FIFO-almost-full [9:8] threshold* –Two MSB of *Tx-FIFO-almost-full [9:0] threshold*.
 bit[3:2] - '00'.
 bit[7:4] - *Tx-FIFO-almost-empty [3:0] threshold* - four LSB of *Tx-FIFO-almost-empty [9:0] threshold*.
 Bits [9:4] are configured in Interrupt TX-FIFO Threshold 3 register [5:0].
 Default threshold set to 12 bytes.

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.11 REG_3E - 'Interrupt TX-FIFO Thresholds 3' (Address 0x3E) (SPI & I²C Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
[0]	[0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]
0	0	<i>Tx-FIFO-almost-empty[9:4] threshold</i>					

bit[5:0] - *Tx-FIFO-almost-empty[9:4] threshold* - Six MSB of *Tx-FIFO-almost-empty[9:0] threshold*
 bit[7:6] - '00'.

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.12 REG_7 - 'Arbitration ID 1' (Address 0x07) (UART/SPI/I²C interface only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]	R/W [0]
<i>Arbitration ID [7:0]</i>							

Bits [7:0] - *Arbitration ID [7:0]* - Eight LSB of *Arbitration ID[10:0]*. See section 3.5

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.13 REG_8 - 'Arbitration ID 2' (Address 0x08) (UART/SPI/I²C interface only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
[0]	[0]	[0]	[0]	[0]	R/W [0]	R/W [0]	R/W [0]
0	0	0	0	0	Arbitration ID [10:8]		

Bits [2:0] - Arbitration ID [10:8] - Three MSB of - Arbitration ID[10:0]. See section 3.5

Bits [7:3] - '00000'.

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.14 REG_59 – UUID[47:40] (Address 0x59)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[47:40]							

Bits [7:0] - UUID[47:40]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.15 REG_5A – UUID[39:32] (Address 0x5A)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[39:32]							

Bits [7:0] - UUID[39:32]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.16 REG_5B – UUID[31:24] (Address 0x5B)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[31:24]							

Bits [7:0] - UUID[31:24]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.17 REG_5C – UUID[23:16] (Address 0x5C)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[23:16]							

Bits [7:0] - UUID[23:16]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.18 REG_5D – UUID[15:8] (Address 0x5D)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[15:8]							

Bits [7:0] - UUID[15:8]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

5.19 REG_5E – UUID[7:0] (Address 0x5E)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
UUID[7:0]							

Bits [7:0] - UUID[7:0]

R - Readable bit, *W* - Writeable bit *[x]* - Value on power-up. '1' - bit is set; '0' - bit is cleared

6. UART protocol interface

To set DCB1M to the UART protocol interface, set IF_SEL[1:0] to '00' (See section 3.2).

6.1 Interfacing to UART ECU

The UART communication protocol uses four pins as described in Table 17.

Table 17- UART interface pins

HDI	Data Input from the ECU.	
HDC	Data/Command select input. When pulled down, the DCB1M enters command mode, enabling access to DCB1M Control registers.	Shared function with HCS
HDO	Data output to the ECU	
RTR	Ready to Receive output. indicating that the device is ready to receive new data bytes from the ECU (i.e. TX-FIFO is not full)	Used to control the data flow between the ECU and the DCB1M. Shared function with INT pin. When Low ECU shall stop transferring data to the DCB1M.

6.2 UART registers configuration (Command mode)

The Command mode allows the ECU to access the DCB1M internal registers for write and read operations.

Enter the Command mode by lowering the HDC pin. When the HDC pin is high, the device is in Normal mode.

During Command mode, the DCB1M is in a *Soft-reset* state, TX-FIFO and RX-FIFO are reset and all data in the FIFOs is erased, and the device cannot send or receive a message to/from the powerline.

6.2.1 WRITE-REG command

Write register command consists of three bytes as described in Table 18.

Table 18 - WRITE-REG command structure

1 st Byte	2 nd Byte	3 rd Byte
0xF5	Control register address	Data to write

1st byte is the write command byte.

2nd byte is the designated control register address to write to.

3rd byte is the data byte value to write.

For example, writing 0x34 to REG_3 (address 0x03) performed as follows:

1. Lower the HDC pin (Enter Command mode).
2. Wait at least 100nsec
3. Transfer 3 bytes: [0xF5][0x03][0x34]
4. The value 0x34 is written to REG_3.
5. Wait for at least 100ns.
6. Raise the HDC pin (Exit Command mode to Normal mode).

6.2.2 READ-REG command

A READ-REG command consists of 2 bytes as described in Table 19.

Table 19 - READ-REG command structure

1 st Byte	2 nd Byte
0xFD	Control register address

1st byte is the Read command byte.

2nd byte is the designated register address to read from.

Following the second byte, the DCB1M outputs the register value to ECU.

For example, reading from REG_5 (address 0x05) is performed as follows:

1. Lower the HDC pin (Enter Command mode).
2. Wait at least 100nsec
3. Transfer 2 bytes: [0xFD][0x05]
4. Wait for the DCB1M to output the value of REG_5.
5. Wait for at least 100ns.
6. Raise the HDC pin (Exit Command mode to Normal mode).

6.3 UART Bitrate configuration (Bitrate learning)

The default UART bitrate is set to 921.6Kbit/s.

The maximum UART bitrate allowed is 2Mbit/s.

The DCB1M will automatically learn the ECU UART bitrate during Command mode (see sections 6.2.1 and 6.2.2).

An alternative way for bitrate learning is proposed when ECU does not enter the *Command mode* for register configuration;

1. Lower the HDC pin.
2. Wait for at least 200nsec.
3. Write single-byte **0xF5**.
4. Wait for at least 200nsec.
5. Raise the HDC pin.

The DCB1M will then learn the ECU bitrate during 0xF5 transfer on the HDI pin.

6.4 UART Codec Select configuration

Refer to section 3.2.2.

6.5 UART RTR pin handling

The Ready to Receive (RTR, pin 11) output allows the ECU to control its data bytes transfer to the DCB1M.

When RTR is high, ECU can transfer bytes via HDI pin. When low, ECU should pause its data bytes transfer.

The RTR output state is subject to two conditions:

1. **TX-FIFO overflow protection** - In case the TX-FIFO is full (i.e. TX-FIFO is inserted with ≥ 1022 data bytes), the RTR will drop and remains low until the TX-FIFO bytes count is below the *Tx-FIFO-almost-full* threshold. The default *Tx-FIFO-almost-full* threshold is 1012 data bytes. For more details, see section 3.2.3.1.
2. **Loopback enabled mode** - When loopback is enabled (HDI is loopback to the HDO), the priority is given to ECU transfers to TX-FIFO over bytes transfers from RX-FIFO to ECU.
When ECU is not transferring data, the RTR will drop when at least one byte is waiting in the RX-FIFO to be transferred to the ECU. The RTR remains low until RX-FIFO is empty again.
When ECU transfers data to the TX-FIFO, while there are awaiting data bytes in RX-FIFO, the DCB1M will notify the ECU to stop transferring data bytes by dropping the RTR signal. In this case, the RTR drops after ECU data byte transfer completion.

In a high payload network, it is recommended to sample the RTR state before a new data byte transfer. See RTR handling example in 6.7.4.

6.6 UART interface typical set-up and operation

1. Set IF_SEL[1:0] pins to '00' (see section 3.2)
2. Interface HDI, HDO, and HDC pins.
3. Set pins 12, 13 according to the selected functionality (see section 3.2.5).
4. Set codec selection (see section 0).
5. Enable/disable loopback of HDI pin to HDO pin (see section 0 and pin 15 in section 2.4).
6. Select a carrier frequency (default 13MHz) (see section 3.2.4 - Carrier frequency configuration).
7. In the case of steps, 3 and 6 are not performed, use the Command mode to learn the ECU bitrate (see section 6.3).
8. Raise HDC to Normal mode.
9. Transmit data bytes via HDI pin to the powerline.
10. Receive data bytes from the powerline via HDO pin.

6.7 UART Examples

6.7.1 UART Example 1 - Typical communication flow

Figure 14 depicts a typical communication with enabled loopback. The first two bytes at the HDI pin are data bytes sent by the ECU and the last two bytes are received data bytes from the DC-BUS. The RTR (ready-to-receive) pin is dropped 1usec before the bytes received from the powerline are transferred to the ECU and raised again after completing the transfer to the ECU (see also section 6.7.4).

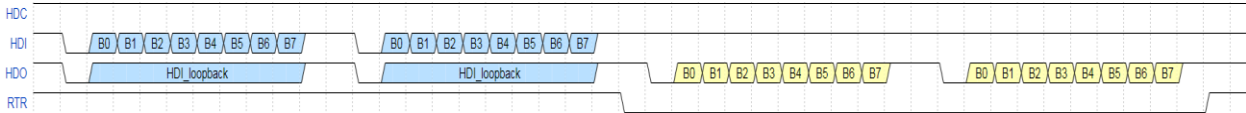


Figure 14 - UART typical communication sequence

6.7.2 UART Example 2 - WRITE-REG command

Figure 15 depicts a *WRITE-REG* command sequence. First, the HDC is pulled low and the device enters the Command mode. The ECU sends the write command with the 1st byte of 0xF5, followed by the control register address byte (A[7:0]), and then the data byte to be written (B[7:0]). After completing the write sequence, the HDC pin is pulled high and the device returns to Normal mode.

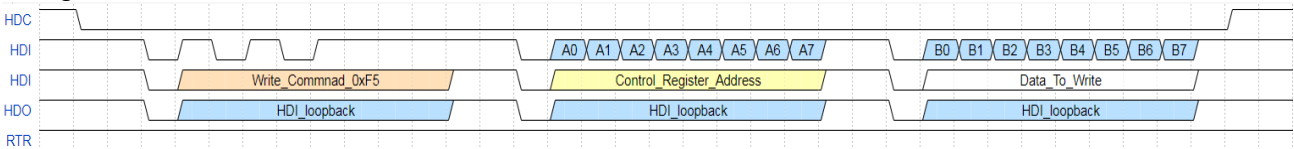


Figure 15- UART Write Register command sequence

6.7.3 UART Example 3 - READ-REG command

Figure 16 depicts a *READ-REG* command sequence. First, the HDC is pulled low and the device enters the Command mode. The user sends the read command with the 1st byte of 0xFD, followed by the control register address byte (A[7:0]). Then the ECU receives the register internal value (B[7:0]). The HDC pulled back to high and the device returns to Normal mode.

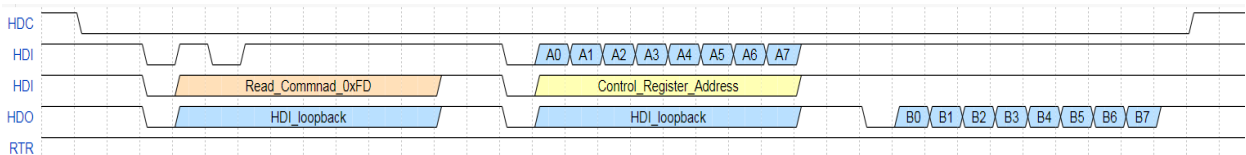


Figure 16 - UART Read Register command sequence

6.7.4 UART Example 4 – Using RTR (ready-to-receive) pin

The RTR pin indicates the ECU whether the DCB1M is ready to receive bytes from ECU for transmission over the powerline (loopback enabled mode only). Before each byte transfer, the ECU should sample the RTR pin. If the RTR pin is low, the ECU shall not transfer to the DCB1M and shall wait until the RTR is raised again.

The RTR pin drops in case that the TX-FIFO is full during data transfer from ECU to DCB1M, or in case the DCB1M starts to transfer data from Rx-FIFO to ECU.

Figure 17 depicts an RTR handling example. First, the ECU starts with bit learning, the HDC pin is pulled down and 0xF5 byte is transferred into the device for bitrate learning. Then the HDC is pulled back high and the device returns to normal Data mode. The ECU samples the RTR pin before transferring each data byte. After the 0xCF data byte is transferred, the RTR is dropped indicating the TX-FIFO is full and the ECU should wait until the RTR back to high before continue with the data bytes transfers.

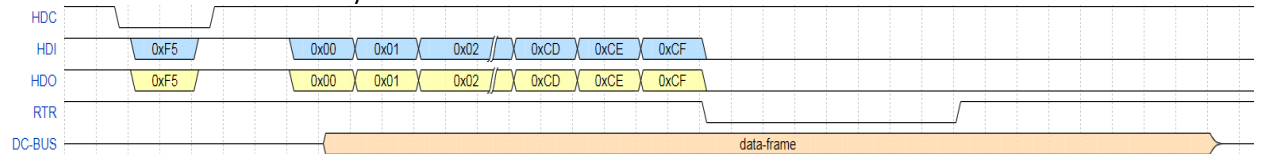


Figure 17 - RTR drop during TX-FIFO full event.

7. DMX/RDM Protocol interface

To set DCB1M to operate with DMX512/RDM protocol interface, set IF_SEL[1:0] to '01' (See section 3.2).

7.1 Interfacing with DMX/RDM Controller

The DMX/RDM communication protocol uses 3 pins as described in Table 20.

Table 20 - DMX/RDM interface pins

HDI	Data Input from the DMX/RDM-controller.
HDC	Data/Command select input. When pulled down, the DCB1M enters command mode, enabling access to DCB1M Control registers.
HDO	Data output to the DMX/RDM-controller

Figure 18 depicts a typical DCB1M to DMX/RDM controller interface connection.

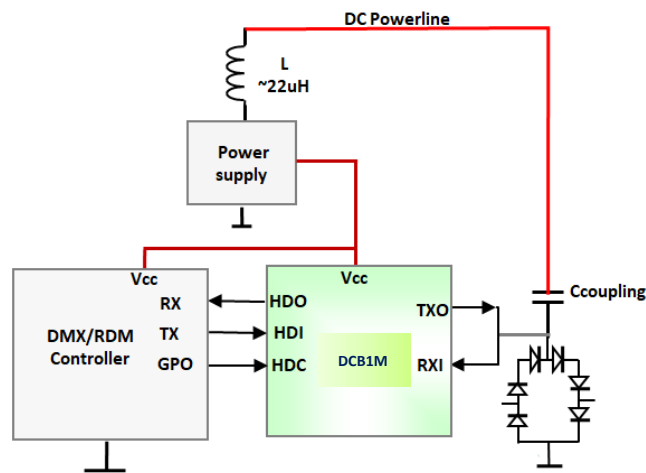


Figure 18 – Typical DCB1M to DMX/RDM controller interface

7.2 Interfacing to an existing DMX/RDM module

When interfacing to an existing DMX/RDM module that has already a built-in RS485 transceiver, an additional RS485 transceiver is required to translate the signals to Tx and Rx 3.3V logic.

Figure 19 depicts a typical DCB1M to RS485 transceiver interface connection.

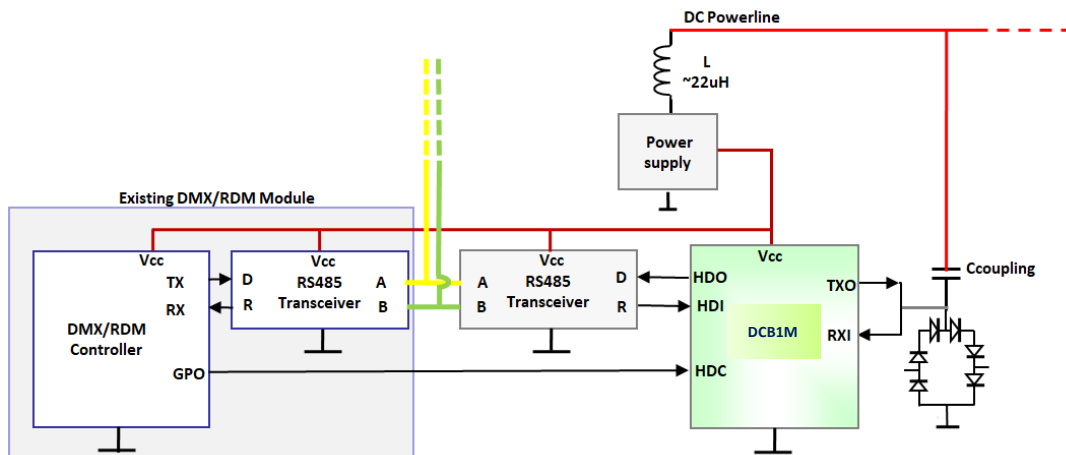


Figure 19 – Typical DCB1M to RS485 transceiver interface

7.3 DMX/RDM interface typical set-up and operation

1. Set IF_SEL[1:0] pins to '01' (see section 3.2)
2. Interface HDI, HDO, and HDC (Optional) pins.
3. Select a carrier frequency (default 13MHz) (see 3.2.4).
4. Transmit data bytes via HDI pin to the powerline.
5. Receive data bytes from the powerline via HDO pin.

7.4 DMX registers configuration (Command mode)

The Command mode allows the DMX-controller to access the DCB1M internal registers for write and read operations.

Enter the Command mode by lowering the HDC pin. When the HDC pin is high, the device is in Normal mode.

During Command mode, the DCB1M is in a *Soft-reset* state, TX-FIFO and RX-FIFO are reset and all data in the FIFOs is erased, and the device cannot send or receive a message to/from the powerline.

7.4.1 WRITE-REG command

Write register command consists of three bytes as described in Table 18.

Table 21 - WRITE-REG command structure

1 st Byte	2 nd Byte	3 rd Byte
0xF5	Control register address	Data to write

1st byte is the write command byte.

2nd byte is the designated control register address to write to.

3rd byte is the data byte value to write.

For example, writing 0x34 to REG_3 (address 0x03) performed as follows:

1. Lower the HDC pin (Enter Command mode).
2. Wait at least 100nsec
3. Transfer 3 bytes: [0xF5][0x03][0x34]
4. The value 0x34 is written to REG_3.
5. Wait for at least 100ns.
6. Raise the HDC pin (Exit Command mode to Normal mode).

7.4.2 READ-REG command

A READ-REG command consists of 2 bytes as described in Table 19.

Table 22 - READ-REG command structure

1 st Byte	2 nd Byte
0xFD	Control register address

1st byte is the Read command byte.

2nd byte is the designated register address to read from.

Following the second byte, the DCB1M outputs the register value to DMX-controller.

For example, reading from REG_2 (address 0x02) is performed as follows:

1. Lower the HDC pin (Enter Command mode).
2. Wait at least 100nsec
3. Transfer 2 bytes: [0xFD][0x02]
4. Wait for the DCB1M to output the value of REG_2.
5. Wait for at least 100ns.
6. Raise the HDC pin (Exit Command mode to Normal mode).

7.5 DMX Examples

7.5.1 DMX Example 1 - Typical communication flow

Figure 14 depicts a typical TX-RX DMX message flow. The DMX-controller starts a new message with a break signal, followed by start code and data bytes on DCB1M HDI. Upon detection of the break signal, the DCB1M initiates the DMX frame over the powerline. After a fixed latency of ~100usec, the DCB1M RX device output on HDO the break signal, followed by the start code and message received data bytes.

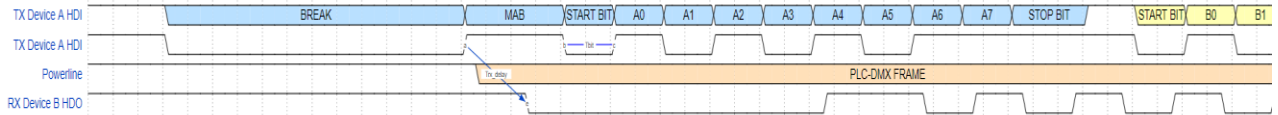


Figure 20 - DMX typical communication sequence

7.5.2 DMX Example 2 - WRITE-REG command

Figure 15 depicts a *WRITE-REG* command sequence. First, the HDC is pulled low and the device enters the Command mode. The DMX-controller sends the write command with the 1st byte of 0xF5, followed by the control register address byte (A[7:0]), and then the data byte to be written (B[7:0]). After completing the write sequence, the HDC pin is pulled high and the device returns to Normal mode.

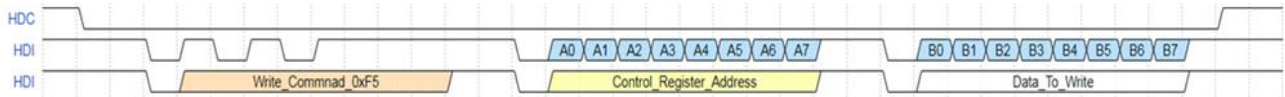


Figure 21- DMX Write Register command sequence

7.5.3 DMX Example 3 - READ-REG command

Figure 16 depicts a *READ-REG* command sequence. First, the HDC is pulled low and the device enters the Command mode. The user sends the read command with the 1st byte of 0xFD, followed by the control register address byte (A[7:0]). Then the DMX-controller receives the register internal value (B[7:0]). The HDC pulled back to high and the device returns to Normal mode.

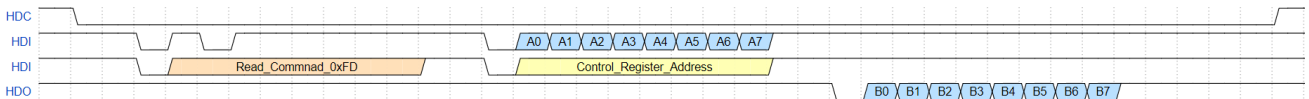


Figure 22 - DMX Read Register command sequence

8. SPI Protocol interface

To set DCB1M to operate with SPI protocol interface, set IF_SEL[1:0] to '10' (See section 3.2).

8.1 Interfacing to SPI ECU

The SPI protocol uses five pins, as described in Table 23.

Table 23 - SPI protocol interface pins

HDI	Data Input from the ECU	
HCS	Chip select input.	Shared function with HDC
HDO	Data output to the ECU	
SCK	Serial clock input.	
INT	Interrupt output (Active High)	Shared function with RTR

The device acts as SPI slave (CPOL = 1, CPHA = 1). Data is sampled at the rising edge of SCK and changed at the SCK falling edge. The SCK pin default state is HIGH. The data is MSB first. The maximal SCK rate is 8MHz.

When the ECU lowers the HCS pin, the DCB1M expects a command to be received. Based on the received command, the ECU can transfer data to DCB1M or start receiving new data from the DCB1M. For example, to transfer a frame to DCB1M, the ECU lowers the HCS pin and sends a one-byte *TX-DATA* command to the device, then starts sending the frame data bytes to be transmitted over the powerline. To close the transferred frame, the ECU has to raise the HCS pin, lower it again, and send the *FRAME-END* command.

8.2 SPI Flow control

Three pins (HDI, HDO, and SCK) are required for the SPI interface. Additionally, the INT (interrupt) pin is used to manage the SPI data transfer flow.

It is recommended to read the status of the INT pin continuously (ECU polling or on-change methods) to determine the status of the frame transmission/reception.

ECU shall use the SPI commands as specified in section 8.5, for proper interfacing with DCB1M.

8.3 SPI Message (frame) construction

SPI message starts with *TX-DATA* command followed by a stream of bytes transferred from ECU. Frame message terminates by *FRAME-END* command.

Note: as long as the frame is not terminated, the DCB1M will continue to transmit empty packets over the powerline keeping it occupied.

8.4 SPI Status interrupt byte

The Status interrupt byte has eight interrupt events. By default, all interrupts are enabled aside from *Empty-frame* interrupt (see REG_4 in 5.5). The byte is automatically extracted when the Tx-Data command is used.

In the case of an interrupt event, the INT pin will rise and remains high, until *READ-INT/READ-ERR* command is executed (see 8.5.6 and 8.5.7).

Each one of the interrupts is triggered once at the occurrence of its event condition change. The next same Interrupt trigger will take place only with the next change condition event. For example, assuming setting *Rx-FIFO-not-Empty* threshold to 0x01, with Rx powerline frame of 5 bytes. That is, the condition change is having a minimum of one byte in RX-FIFO. As soon as the first Rx data byte is inserted to Rx-FIFO, the *Rx-FIFO-not-Empty* (Bit[3]) interrupt is raised. After *READ-INT*, the interrupt drops. At this point, the *Rx-FIFO-not-empty* interrupt will not rise again, for every RX data byte read, but only when the condition change is triggered again - that is, when RX-FIFO is empty and at least one byte is inserted to Rx-FIFO.

Table 24 describes the read-only Status interrupt byte.

Table 24 - Status interrupt byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
<i>Error-flag</i>	<i>Empty-frame</i>	<i>EOF</i>	<i>Rx-FIFO-empty</i>	<i>Rx-FIFO-not-empty</i>	<i>Rx-frame-end</i>	<i>Tx-FIFO-almost-empty</i>	<i>Tx-FIFO-almost-full</i>

R - Readable bit

- ***Tx-FIFO-almost-full*** *Tx-FIFO-almost-full interrupt* rise when the TX-FIFO data byte count is equal or bigger than the *Tx-FIFO-almost-full threshold* (see 3.2.3.1)
- ***Tx-FIFO-almost-empty*** *Tx-FIFO-almost-empty interrupt* will rise when the TX-FIFO data byte count is equal to or smaller than the *Tx-FIFO-almost-empty threshold* (see 3.2.3.1)
- ***Rx-frame-end*** *Rx-frame-end interrupt* will rise when a full-frame is inserted into the RX-FIFO.
- ***Rx-FIFO-not-empty*** *Rx-FIFO-not-empty interrupt* will rise when the RX-FIFO data byte count is equal to or bigger than the *Rx-FIFO-not-empty threshold*(see 3.2.3.1)
- ***Rx-FIFO-empty*** *Rx-FIFO- -empty interrupt* will rise when the RX-FIFO data byte count is zero. Any further read operation would output an invalid byte. The Rx-FIFO-empty bit shows the real-time status of the RX-FIFO for any status interrupt byte read.
- ***EOF*** EOF (End of frame) interrupt will rise when a full-frame is extracted from the RX-FIFO.
- ***Empty-frame*** The Empty-frame interrupt will rise when an empty frame is received from the DC-BUS (i.e. frame without data bytes). By default, this interrupt is disabled (see 5.5).
- ***Error-flag*** The error-flag interrupt will rise in case of an error event. When triggered, ECU shall read the error-register using the ***READ-INT*** command (see 8.5.7).

8.5 SPI Commands

The DCB1M is in an idle state as long as HCS (ECU Chip Select) is high.

Upon HCS drop, the DCB1M expects the first-byte transfer from ECU to be one of the SPI commands byte.

Any other byte's transfer is overlooked (decode as *invalid-command*) until raising the HCS.

Any new SPI command must start with an HCS drop event.

Table 25 - SPI Commands summary

Command name	Command byte	Description
<i>TX-DATA</i>	0x04	Start new frame with Codec 0
<i>TX-DATA</i>	0x14	Start new frame with Codec 1
<i>TX-DATA</i>	0x24	Start new frame with Codec 2
<i>TX-DATA</i>	0x34	Start new frame with Codec 3
<i>FRAME-END</i>	0x0F	Close the current open frame
<i>RX-DATA</i>	0x0B	Extract received bytes from Rx-FIFO to ECU
<i>WRITE-REG</i>	0xF5	Write from ECU to internal registers
<i>READ-REG</i>	0xFD	ECU read from internal registers
<i>READ-INT</i>	0x01	ECU read from Status interrupt byte
<i>READ-ERR</i>	0x11	ECU read from Error status byte

8.5.1 SPI TX-DATA Command (0x04 to 0x34)

This *TX-DATA* command initiates a new frame construction over the powerline.

The command determines also the codec selection as described in Table 26.

During the TX data transfer routine, the Status Interrupt byte is automatically output on HDO for each data byte transfer. It allows the ECU to get the TX-FIFO status without the need for *READ-INT* command execution.

Table 26 - TX-DATA command

<i>TX-DATA</i> Command	Codec Select	Max powerline bitrate (Mbit/s)
0x04	Code 0	1.4
0x14	Code 1	1
0x24	Code 2	0.5
0x34	Code 3	0.225

Following the *TX-DATA* command, while HCS is kept low, ECU shall transfer its frame data bytes to the device.

In case *TX-Trigger mode* is not enabled (i.e. default *normal-TX* mode), new frame transmission starts immediately over the powerline.

In case of *TX-Trigger mode* is enabled, new frame transmission starts manually when ECU toggles the EN_TX pin (see 3.3).

The new frame is stored in TX-FIFO until ECU performs a *FRAME-END* command.

8.5.1.1 SPI Empty frame transmission

An empty frame is defined as a powerline frame with zero payloads. An empty frame transmission starts with the *TX-DATA* command, followed by the *FRAME-END* command. As long as the *FRAME-END* command is not executed, the empty frame is being transmitted over the powerline, with no data bytes.

For example, ECU may use an empty frame as part of the ACK/NACK feature. ECU may transfer a short (1 packet) empty frame that will not overload the RX-FIFO with data bytes, and will only trigger the Empty-frame interrupt at RX nodes as an ACK message response.

By default, the *Empty-frame* interrupt is disabled (see 5.5).

8.5.2 SPI FRAME-END Command (0x0F)

The *FRAME-END* command closes the last opened frame (i.e. close the frame related to the last executed *TX-DATA* command). ECU shall drop the HCS and send the *FRAME-END* command byte **0x0F** and then raise the HCS.

8.5.3 SPI RX-DATA Command (0x0B)

The *RX-DATA* command transfers received data bytes from the RX-FIFO.

ECU shall drop the HCS and send the *RX-DATA* command byte **0x0B**. Then, ECU shall provide eight SCK cycles to extract one RX data byte on HDO pin (i.e. transfer dummy 0xFF byte on HDI pin).

The DCB1M will continue to output RX data bytes from the RX-FIFO for every eight SCK cycles provided by the ECU. It is recommended to use the INT pin for a proper reading process. In case RX-FIFO is empty and the *RX-DATA* command is still active; the DCB1M will continue to extract dummy bytes on the HDO pin.

When required, ECU shall raise the HCS to terminate the *RX-DATA* command.

8.5.4 SPI WRITE-REG Command (0xF5)

The *WRITE-REG* command enables ECU to configure the DCB1M Registers (see 5).

ECU shall drop the HCS and send the *WRITE-REG* command byte **0xF5**, then write additional two bytes as described in Table 27.

Table 27 - *WRITE-REG* command structure

1 st Byte	2 nd Byte	3 rd Byte
0xF5	Control register address	Data to write

To terminate the *WRITE-REG* command, ECU shall raise the HCS.

8.5.5 SPI READ-REG Command (0xFD)

The *READ-REG* command enables ECU to read the DCB1M Registers (see 5).

ECU shall drop the HCS pin and send the *READ-REG* command byte **0xFD**, then write an additional one byte as described in Table 28.

Table 28 - *READ-REG* command structure

1 st Byte	2 nd Byte
0xFD	Control register address

Following the second byte, ECU shall provide eight SCK cycles (i.e. transfer dummy 0xFF byte on HDI pin) to allow the DCB1M to output on HDO the value of the designated control register.

To terminate the *READ-REG* command, ECU shall raise the HCS.

8.5.6 SPI READ-INT Command (0x01)

The *READ-INT* command enables ECU to read the Status interrupt byte (see Table 24).

Upon the rise of the INT pin, ECU shall perform the *READ-INT* command to read the Status interrupt byte and clear the INT pin.

ECU shall drop the HCS, and send the *READ-INT* command byte **0x01**, then ECU shall provide eight SCK cycles (i.e. transfer dummy 0xFF byte on HDI pin) to allow the DCB1M to output on HDO the value of the Status interrupt byte.

To terminate the *READ-INT* command, ECU shall raise the HCS.

The INT pin will be automatically cleared) after the *READ-INT* command is executed. (*Error-flag* is cleared by *READ-ERR* command).

8.5.7 SPI READ-ERR Command (0x11)

Upon reading the *Error-flag* high (after *READ-INT* command), ECU shall perform the *READ-ERR* command to read the device's internal Error status byte (see Table 29) and for clearing the INT pin.

Table 29 – Error status byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				R	R	R	R
<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Illegal-command</i>	<i>Tx-FIFO-overflow</i>	<i>Rx-FIFO-overflow</i>	<i>Reserved</i>

R - Readable bit

Bit[1] - Rx-FIFO is overflowed

Bit[2] - Tx-FIFO is overflowed

Bit[3] - ECU send illegal (SPI / I²C) command

ECU shall drop the HCS, and send the *READ-ERR* command byte **0x11**, then, ECU shall provide eight SCK cycles (i.e. transfer dummy 0xFF byte on HDI pin) to the DCB1M to output on HDO the value of the Error status byte.

To terminate the *READ-ERR* command, ECU shall raise the HCS.

The INT (triggered from *Error-flag*) pin will be automatically cleared after executing the *READ-ERR* command.

8.6 SPI interface typical set-up and operation

- ✓ Set IF_SEL[1:0] pins to '10' (see section 3.2)
- ✓ Interface ECU with HDI, HDO, SCK, INT, and HCS pins.
- ✓ Set pins 13,12 according to the selected IOs functionality (see section 3.2.5)
- ✓ Select a carrier frequency (default 13MHz) (see section 3.2.4)
- ✓ Open and close frames using the *TX-DATA* and *FRAME-END* commands, along with Interrupt handling.
- ✓ Receive data bytes from the powerline using *RX-DATA* command, along with interrupt handling

8.7 SPI Examples

8.7.1 SPI Example 1 - WRITE-REG command

Figure 23 depicts an example of a *WRITE-REG* command sequence to REG_4 (register address 0x04) with a data value of 0x55.

After HCS is dropped, a *WRITE-REG* command, 0xF5 byte is transferred followed by REG_4 address 0x04 and data byte 0x55 to the designated register. After the write sequence is completed, the HCS pin is pulled high and the device returns to idle mode.

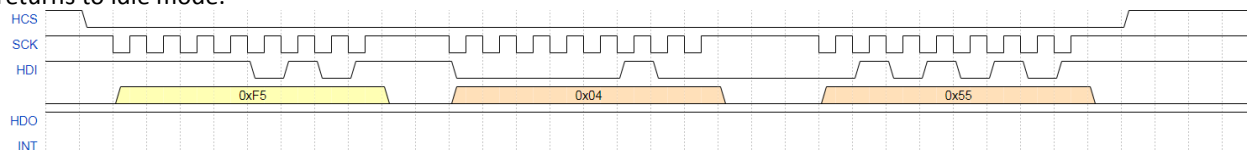


Figure 23 - SPI WRITE-REG Command

8.7.2 SPI Example 2 - READ-REG command

Figure 24 depicts a *READ-REG* command sequence from REG_2.

After dropping the HCS to low, a *READ-REG* command 0xFD is sent, followed by REG_2 address 0x02 and with dummy byte 0xFF to allow 8 SCK clock cycles, shifting out the data value of REG_2. When the read sequence is completed, the HCS pin is pulled high and the device returns to idle mode.

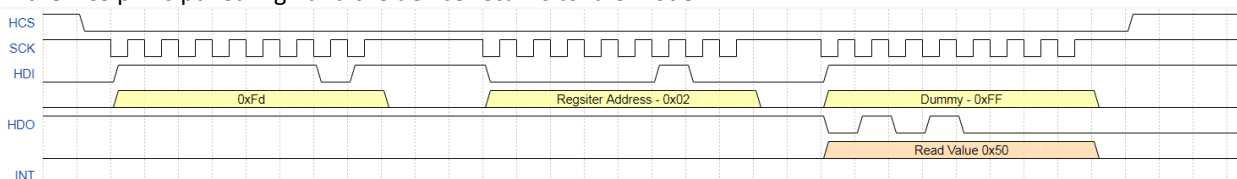


Figure 24 - SPI READ-REG Command

8.7.3 SPI Example 3 - Data frame transmission

Figure 25 depicts a transmission sequence of a data frame with one data byte. First, the HCS is dropped and the TX-DATA command is sent, in this case, using the 0x04 command byte. Then data byte is transferred (e.g.0x55). To close the frame, the HCS is toggled high to low, and the FRAME-END command is transferred (0x0F byte).

In parallel to each data byte transfer, the ECU will receive the updated Status interrupt byte at the HDO pin, regardless if an interrupt event occurred. In this example, the status is 0x10 indicating the RX-FIFO is empty.

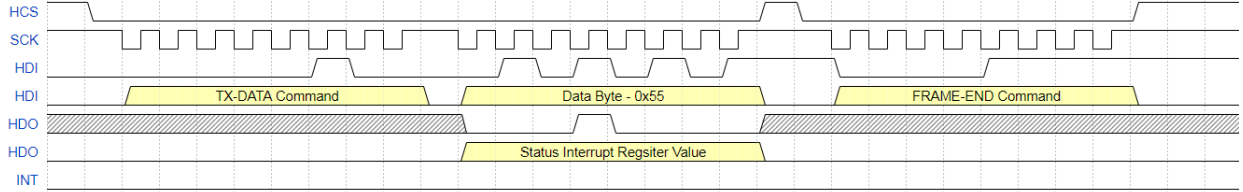


Figure 25 - SPI Transmit data sequence

8.7.4 SPI Example 4 - Data frame reception after Rx-frame-end interrupt

Figure 26 depicts a frame reception sequence of a single data byte. The reception starts with *Rx-frame-end interrupt*, meaning a full-frame is inserted to the Rx-FIFO (bit[2] of Status interrupt byte). The ECU should read the internal Status interrupt byte using *READ-INT(0x01)* command before start fetching the RX data bytes. The Status interrupt byte value is 0x04, meaning an RX data frame is fully received (bit[2] is high) and the Rx-FIFO is not empty (bit[4] is low). After the interrupt read routine is completed, the INT drops and the ECU starts the reception routine from the Rx-FIFO using *RX-DATA (0x0B)* command. The INT pin is raised after the data byte extraction. Then, ECU shall stop the frame reception, toggle the HCS, and read the Status interrupt byte. This time, the Status interrupt byte value is 0x30, meaning, the ECU has reached the end of the data frame (bit[5] is high) and the Rx-FIFO is empty (bit[4] is high).

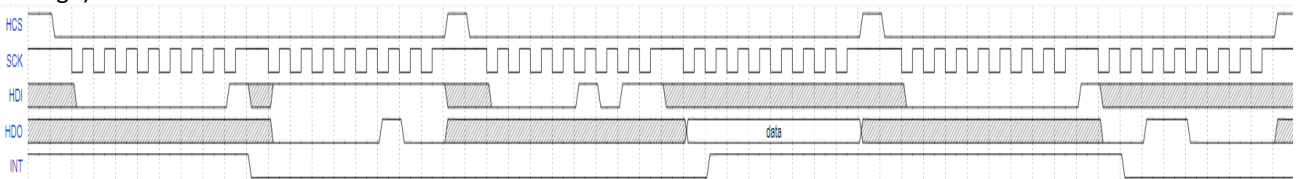


Figure 26- SPI Receive data sequence after *Rx-frame-end* interrupt event

8.7.5 SPI Example 5 - Data frame reception after Rx-FIFO-not-empty interrupt

Figure 27 depicts a data frame reception routine triggered by the *Rx-FIFO-not-empty* interrupt event (Status interrupt byte bit[3]), meaning there are at least 256 data bytes in the RX-FIFO.

After the raise of the INT pin, ECU shall read the interrupt register using *READ-INT(0x01)* command before start fetching the RX data bytes. The Status interrupt byte value is 0x08, meaning, the Rx-FIFO is not empty and it contains at least 256 data bytes. Also, bit[2] is low indicating the current data frame reception is not completed. Following to INT drop, the ECU can start the reception routine until the *Rx-FIFO-empty* and *EOF* interrupts events triggered.

Please note that in the middle of the reception routine, the ECU will also be triggered on interrupt events on *Rx-frame-end*, which indicates that a full data frame is inserted into the RX-FIFO.

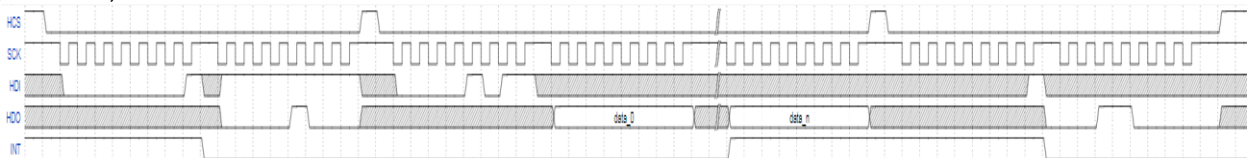


Figure 27- SPI Receive data sequence after *Rx-FIFO-not-empty* interrupt event

8.7.6 SPI Example 6 - Empty frame interrupt event

The DCB1M will notify the ECU if an empty frame is received from the powerline. To transmit an empty frame on the powerline, the ECU shall transfer the TX-DATA command and then close the frame using the FRAME-END (0x0F) command. If such a frame is transmitted, the receiving device will notify its ECU using the *empty-frame* interrupt event in Status interrupt byte bit[6], (ECU shall enable the *empty-frame* interrupt, see 5.5).

Figure 28 depicts the transmission and reception of an empty frame. At the transmitting node, the ECU pulls down the HCS pin and sends the TX-DATA command (e.g. 0x04). Then, the ECU toggles the HCS pin and sends the FRAME-END (0x0F) command. When the empty frame is received at the receiver node, the INT pin will raise indicating an

Empty-frame interrupt event. The ECU reads Status interrupt byte (value is 0x50), indicating an empty frame has been received (bit[6] is high) and the RX-FIFO is empty (bit[4] is high).

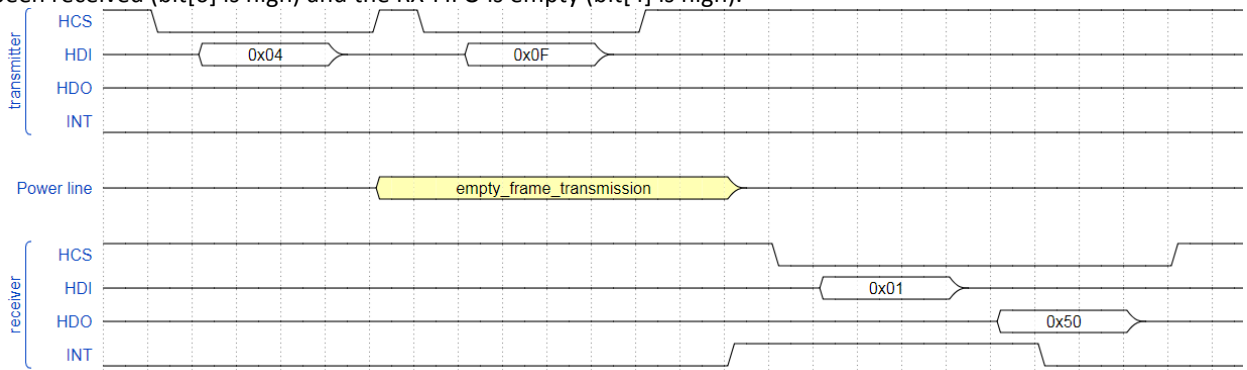


Figure 28 - Empty-frame transmission and reception

8.7.7 SPI Example 7 - Multiple frames management

The DCB1M using the SPI interface allows the ECU to manage multiple frames reception. The device notifies the ECU on each *Rx-frame-end* interrupt event. When ECU completes a full-frame reading, and another frame is received, the device will raise interrupt on *Rx-frame-end* event following the first frame *EOF* interrupt event. This way, ECU can manage multiple data frames from one or several nodes in the network.

Figure 29 depicts the operation of two nodes. Node A transmits two consecutive data frames. Node B receives the first interrupt event. The ECU reads the Status interrupt byte before starting the reception routine. The Status interrupts byte value is 0x04 indicating the RX-FIFO filled with a full data frame A. Then, ECU starts the reception routine until a second interrupt event occurs. The ECU again reads the interrupt register; its value is 0x20 indicating *EOF*. As frame B is also fully loaded to the RX-FIFO, a third interrupt event is raised with a value of 0x04 (*Rx-frame-end*). ECU shall read the Status interrupt byte before starting the reception routine. The ECU then starts the reception routine, and the fourth interrupt event is raised when the *EOF* of frame B is transferred from Rx-FIFO to ECU.

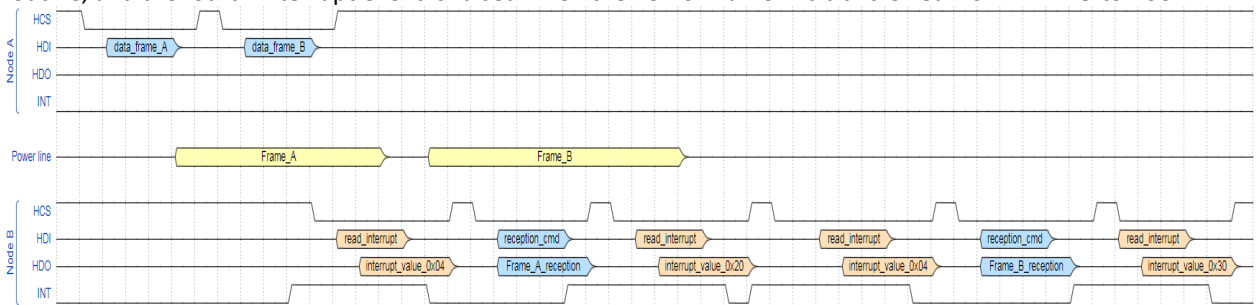


Figure 29 - multiple frame management example

9. I²C Protocol interface

To set DCB1M to I²C protocol interfacing, set IF_SEL[1:0] to '11' (See section 3.2).

9.1 Interfacing to I²C ECU

The I²C communication protocol uses three pins, as described in Table 30.

Table 30 - I²C protocol interface pins

HDO	I ² C Digital IO	This pin should be externally pull-up with a 10kΩ resistor.
SCK	Serial clock input.	
INT	Interrupt output (Active High)	Shared function with RTR

The HDO data is sampled at the rising edge of SCK and shifted out to HDO on the falling edge of SCK, the default state of the SCK pin is HIGH. The expected data is MSB first. The maximal SCK rate is 1MHz.

The DCB1M I²C write address is **0x44**.

The DCB1M I²C read address is **0x45**.

Any I²C write operation from ECU to the DCB1M begins on START condition, followed by 0x44 (The DCB1M I²C write address with R/~W = 0). After DCB1M acknowledge the I²C write address, ECU sends an I²C write address and transfers data bytes according to the I²C command specification, until ECU executes the STOP or REPEATED START conditions (See TX-DATA / FRAME-END / WRITE-REG commands).

Any ECU I²C read operation from DCB1M starts with an I²C write operation (address 0x44), followed by one of the I²C read commands (see RX-DATA / READ-REG/ READ-INT / READ-ERR commands). Then, ECU sends a read request by executing a REPEATED START condition followed by 0x45 (The DCB1M I²C address with R/~W = 1). Then DCB1M outputs the corresponding RX bytes to ECU on every eight SCK clock cycles followed by 9th SCK for ACK/NACK slot until the ECU executes a STOP or REPEATED START condition.

In general, any NACK generated by ECU will result that the DCB1M will be back to the idle state, expecting a new I²C command to be executed.

Figure 30 depicts a typical I²C flow control.

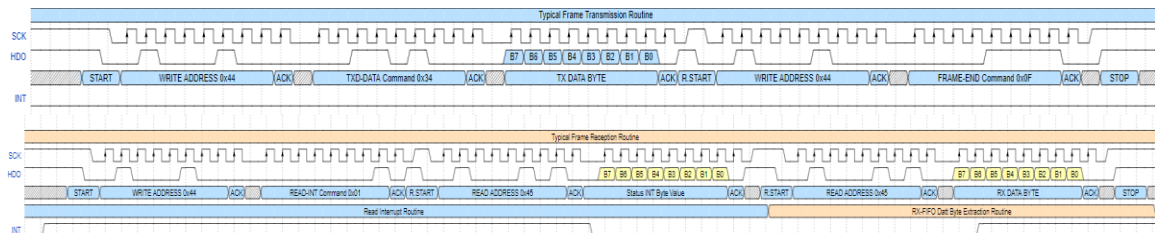


Figure 30 – Typical I²C flow control

9.2 I²C Flow control

In addition to the two pins required (HDO and SCK), the INT (interrupt) pin is used to manage the I²C data transfer flow.

It is recommended to read the status of the INT continuously (ECU polling or on-change methods) to determine the status of the frame transmission /reception.

The I²C commands as specified in section 9.5, are used for proper interfacing with the DCB1M device.

9.3 I²C Status interrupt byte

The Status interrupt byte consists of eight interrupt events (identical to SPI Status interrupt byte). By default, all interrupt events are enabled aside from the *Empty-frame* interrupt event (see REG_4 in 5.5).

When an interrupt event happened, the INT pin rise and remains high until READ-INT / READ-ERR command is executed (see 9.5.6 and 9.5.7).

Each one of the interrupts is triggered once at the occurrence of its event condition change. The next same Interrupt trigger will take place only with the next change condition event. For example, assuming setting *Rx-FIFO-not-Empty* threshold to 0x01, with Rx powerline frame of 5 bytes. That is, the condition change is having a minimum of one byte in RX-FIFO. As soon as the first Rx data byte is inserted to Rx-FIFO, the *Rx-FIFO-not-Empty* (Bit[3]) interrupt is raised. After READ-INT, the interrupt drops. At this point, the *Rx-FIFO-not-empty interrupt* will not rise

again, for every RX data byte read, but only when the condition change is triggered again - that is, when RX-FIFO is empty and at least one byte is inserted to Rx-FIFO.

Table 31 describes the read-only Status interrupt byte.

Table 31 - Status interrupt byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	R	R	R
<i>Error-flag</i>	<i>Empty-frame</i>	<i>EOF</i>	<i>Rx-FIFO-empty</i>	<i>Rx-FIFO-not-empty</i>	<i>Rx-frame-end</i>	<i>Tx-FIFO-almost-empty</i>	<i>Tx-FIFO-almost-full</i>

R - Readable bit

- ***Tx-FIFO-almost-full*** *Tx-FIFO-almost-full interrupt* rise when the TX-FIFO data byte count is equal or bigger than the *Tx-FIFO-almost-full threshold* (see 3.2.3.1)
- ***Tx-FIFO-almost-empty*** *Tx-FIFO-almost-empty interrupt* will rise when the TX-FIFO data byte count is equal to or smaller than the *Tx-FIFO-almost-empty threshold* (see 3.2.3.1)
- ***Rx-frame-end*** *Rx-frame-end interrupt* will rise when a full-frame is inserted into the RX-FIFO.
- ***Rx-FIFO-not-empty*** *Rx-FIFO-not-empty interrupt* will rise when the RX-FIFO data byte count is equal or bigger than the *Rx-FIFO-not-empty threshold* (see 3.2.3.1).
- ***Rx-FIFO-empty*** *Rx-FIFO-not-empty interrupt* will rise when the RX-FIFO data byte count is zero. Any further read operation would output an invalid byte. The Rx-FIFO-empty bit shows the real-time status of the RX-FIFO for any status interrupt byte read.
- ***EOF*** EOF (End of frame) interrupt will rise when a full-frame is transferred from the RX-FIFO.
- ***Empty-frame*** The Empty-frame interrupt will rise when an empty frame is received from the DC-BUS (i.e. frame without data bytes). By default this interrupt is disabled (see 5.5)
- ***Error-flag*** The error-flag interrupt will rise in case of an error event. When triggered, ECU shall read the error-register using the ***READ-INT*** command (see 9.5.6)

9.4 I²C Message (frame) construction

I²C message starts with TX-Data command followed by a stream of bytes transferred from ECU. Frame message terminates by FRAME-END command.

Note: as long as the frame is not terminated, the DCB1M will continue to transmit empty packets over the powerline keeping it occupied.

9.5 I²C Commands

The DCB1M is in an idle state until ECU executes an I²C command.

Before any I²C command, ECU shall perform an I²C write request operation with a START condition followed by 0x44 (DCB1M I²C write address with R/~W = 0). After DCB1M will acknowledge the I²C write address, ECU shall send an I²C command as specified in the below sub-sections (see Figure 30).

Table 32 - I²C Commands summary

Command Name	Command Byte	Description
<i>TX-DATA</i>	0x04	Start new frame with Codec 0
<i>TX-DATA</i>	0x14	Start new frame with Codec 1
<i>TX-DATA</i>	0x24	Start new frame with Codec 2
<i>TX-DATA</i>	0x34	Start new frame with Codec 3
<i>FRAME-END</i>	0x0F	Close the current open frame
<i>RX-DATA</i>	---	Extract received bytes from RX-FIFO to ECU
<i>WRITE-REG</i>	0xF5	Write from ECU to internal registers
<i>READ-REG</i>	0xFD	ECU read from internal registers
<i>READ-INT</i>	0x01	ECU read from Status interrupt byte
<i>READ-ERR</i>	0x11	ECU read from Error status byte

9.5.1 I²C TX-DATA Command (0x04 to 0x34)

This TX-DATA command initiates a new frame construction over the powerline. The command determines also the codec selection as described in Table 33.

Table 33 - TX-DATA command

TX-DATA Command	Codec Select	Max powerline bitrate (Mbit/s)
0x04	Code 0	1.4
0x14	Code 1	1
0x24	Code 2	0.5
0x34	Code 3	0.225

9.5.1.1 I²C TX-DATA sequence

1. Send START condition with write address 0x44.
2. Send TX-DATA command byte.
3. Send data bytes payload.
4. Send STOP condition.

In case of TX-Trigger mode is not enabled (i.e. default normal-TX mode), the new frame transmission immediately begins onto the powerline.

In case of TX-Trigger mode is enabled, the new frame transmission begins manually when ECU toggles the EN_TX pin (see 3.3).

The new frame is kept open until ECU performs a FRAME-END command.

9.5.1.2 I²C Empty frame transmission

An empty frame is a powerline frame with zero payloads. An empty frame transmission starts with the TX-DATA command, followed by the FRAME-END command. As long as the FRAME-END command is not executed, the empty frame is being transmitted over the powerline, with zero payloads (no data bytes).

For example, ECU may use an empty frame as part of the ACK/NACK routine. ECU may transfer a short (1 packet) empty frame that will not overload the RX-FIFO with data bytes and will only trigger the Empty-frame interrupt at RX nodes as an ACK message response.

By default, the Empty-frame interrupt is disabled (see 5.5).

9.5.2 I²C FRAME-END Command (0x0F)

The FRAME-END command closes the frame started with the TX-DATA command.

9.5.2.1 I²C TX-DATA sequence

1. Send START condition with write address 0x44.
2. Send FRAME-END command byte 0x0F.
3. Send STOP condition.

9.5.3 I²C RX-DATA Command

Unlike the rest of the I²C commands, the RX-DATA command does not require a write address 0x44, followed by a specific command byte. Instead, to transfer to ECU the received data bytes (stored in RX-FIFO), ECU sends a read address 0x45¹, and then generate nine SCK cycles for each read data byte (including ACK/NACK slot).

¹ In case of INT pin is already high, while initiating read sequence, then the read address 0x45 transfer will be NACK by the DCB1M. As such, host must first read the interrupt status register (see 9.5.6) and only then initiate the read sequence (see 9.5.3.1).

9.5.3.1 I²C RX-DATA sequence

1. Check INT state. If high, then read interrupt reg status (see 9.5.6).
2. Start read sequence by sending START condition with read address 0x45.
3. Generate nine SCK clock cycles for the DCB1M to extract RX data bytes stored in RX-FIFO.
4. Repeat step 2 as needed. It is recommended to use the INT pin for a proper reading process.
Note - In case RX-FIFO is empty, and the RX-DATA command is still active, the DCB1M will continue the transfer of dummy bytes on the HDO pin.
5. When required, ECU shall send a STOP condition to terminate the RX-DATA command.

9.5.4 I²C WRITE-REG Command (0xF5)

The WRITE-REG command enables ECU to perform the configuration of the DCB1M Registers (see 5).

Table 34 describes the WRITE-REG command bytes.

Table 34 - WRITE-REG command structure

1 st Byte	2 nd Byte	3 rd Byte
0xF5	Control register address	Data to write

9.5.4.1 I²C WRITE-REG sequence

1. Send START condition with write address 0x44.
2. Send WRITE-REG command byte 0xF5.
3. Send Control register address byte.
4. Send the data to write to the designated register.
5. Send STOP condition.

9.5.5 I²C READ-REG Command (0xFD)

The WRITE-REG command enables ECU to perform a read of the DCB1M registers (see 5).

Table 35 describes the READ-REG command bytes.

Table 35 - READ-REG command structure

1 st Byte	2 nd Byte
0xFD	Control register address

9.5.5.1 I²C READ-REG sequence

1. Send START condition with write address 0x44.
2. Send READ-REG command byte 0xFD.
3. Send Control register address byte.
4. Send REPEATED START condition with read address 0x45.
5. Generate nine SCK clock cycles to DCB1M extracting the read register value.
6. Send STOP condition.

9.5.6 I²C READ-INT Command (0x01)

The READ-INT command enables ECU to read the Status interrupt byte (see Table 29).

Upon the rise of the INT pin, ECU shall perform the READ-INT command to read the Status interrupt byte and clear the INT pin.

9.5.6.1 I²C READ-INT sequence

1. Send START condition with write address 0x44.
2. Send READ-INT command byte 0x01.
3. Send REPEATED START condition by reading address 0x45.
4. Generate nine SCK clock cycles to DCB1M extracting the Status interrupt byte value.
5. Send STOP condition.

The INT pin automatically cleared (*Error-flag* is cleared by READ-ERR command) after executing the READ-INT command.

9.5.7 I²C READ-ERR Command (0x11)

Upon reading the *Error-flag* high (after READ-INT command), ECU shall perform the READ-ERR command to fetch the Error status byte (see Table 36) and clear the INT pin.

Table 36 – Error status byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				R	R	R	R
Reserved	Reserved	Reserved	Reserved	Illegal-command	Tx-FIFO-overflow	Rx-FIFO-overflow	Reserved

R - Readable bit

Bit[1] - Rx-FIFO is overflowed

Bit[2] - Tx-FIFO is overflowed

Bit[3] - ECU send illegal (SPI / I²C) command

9.5.7.1 I²C READ-ERR sequence

1. Send START condition with write address 0x44.
2. Send READ-ERR command byte 0x11.
3. Send REPEATED START condition with read address 0x45.
4. Generate nine SCK clock cycles for the DCB1M to extract the Error status byte value.
5. Send STOP condition.

The INT (triggered from *Error-flag*) pin automatically clears after executing the READ-ERR command.

9.6 I²C interface typical set-up and operation

1. Set IF_SEL[1:0] pins to '11' (see section 3.2)
2. Interface HDO, SCK, and INT pins.
3. Set pins 12, 13 according to the selected IOs functionality (see section 3.2.5)
4. Select a carrier frequency (default 13MHz) (see section 3.2.4)
5. Open and close frames using the TX-DATA and FRAME-END commands, along with Interrupt handling.
6. Extract data bytes from RX-FIFO to ECU using RX-DATA command, along with interrupt handling.

9.7 I²C Examples

9.7.1 I²C Example 1 - READ-REG command

Figure 31 depicts a reading sequence from internal REG_1.

The READ-REG command starts with a START condition (a drop of HDO when the SCK is high). Then, ECU sends the DCB1M I²C to write address 0x44. The device responds with ACK at the 9th clock edge. The ECU sends the read command using 0xFd. The device responds with ACK at the 9th clock edge. Then, the ECU sends the REG_1 address byte 0x01 followed by the REPEATED START condition, with read address 0x45. The ECU answers with ACK. Then, the ECU generates dummy clocks to receive the REG_0 value and send it at the 9th clock edge NACK before closing the reading routing with a STOP condition.

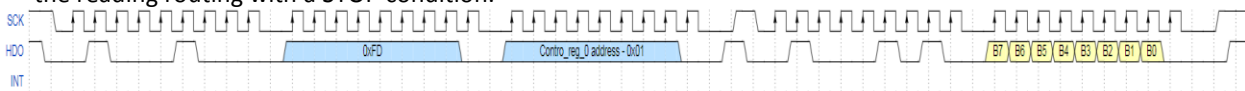


Figure 31 - Read register sequence with I²C interface

Blue color – Data from ECU to DCB1M

Yellow color – Data from DCB1M to ECU

9.7.2 I²C Example 2 - WRITE-REG command

Figure 32 depicts a write sequence to internal REG_0.

The WRITE-REG command begins with a START condition. The ECU then sends 0x44 which is the I²C write address of the DCB1M and responds with ACK at the 9th clock edge. Then, the ECU sends 0xF5, followed by REG_0 address 0x00, and then the data to write to REG_1. The device answers with ACKs and the ECU stops the write routine with a STOP condition.

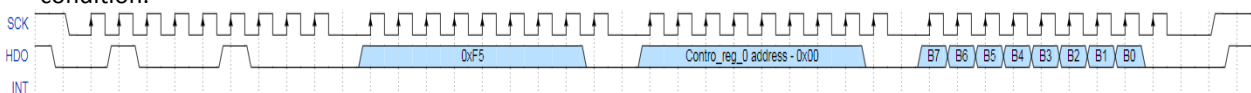


Figure 32 - Write register sequence with I²C interface

Blue color – Data from ECU to DCB1M

9.7.3 I²C Example 3 - TX-DATA command

Figure 33 depicts a powerline frame transmission with single data byte B[7:0] sequence with 1/3 speed codec selection (TX command byte is **0x34**),

The transmission sequence begins with a START condition. Then, the ECU sends the I²C write address **0x44**. The device answers with ACK at the 9th clock edge. Then, ECU sends a TX-DATA command using **0x34** byte and the device answers with ACK at the 9th clock edge, followed by a data byte B[7:0]. Then, the ECU generates the REPEATED START condition and sends again the device write address **0x44**. The device acknowledges the address. Then, ECU sends FRAME-END command **0x0F** byte. The device acknowledges, and the ECU closes the TX sequence with a STOP condition.

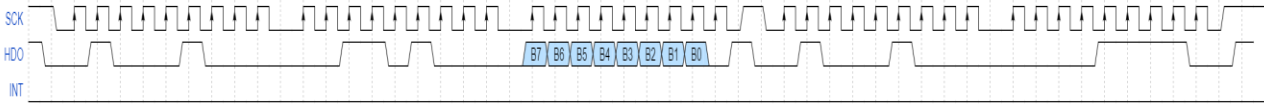


Figure 33 - Frame transmission sequence with I²C interface

Blue color – Data from ECU to DCB1M

9.7.4 I²C Example 4 - READ-INT command

Figure 34 depicts a Status interrupt read routine. The READ-INT command begins with the rise of the INT pin. The ECU sends a START condition followed by I²C write address **0x44**, then READ-INT command **0x01**. After the device acknowledges the **0x01** command, the ECU generates REPEATED START conditions by sending DCB1M I2C to read address **0x45**. The device acknowledges the address. Then the ECU generates eight dummy clock edges to fetch the Status interrupt byte value, followed by NACK at the 9th clock edge. The ECU stops the read interrupt sequence with a STOP condition.

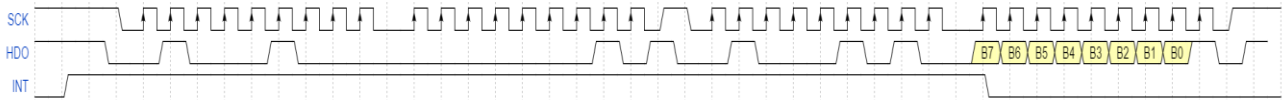


Figure 34 - read interrupt sequence with I²C interface

Yellow color – Data from DCB1M to ECU

9.7.5 I²C Example 5 - RX-DATA command

Figure 35 depicts the frame reception routine. The reception starts with the INT pin pulled high indicating an interrupt event. The ECU generates a START condition followed by I²C write address **0x44** and acknowledges from the device. The ECU sends the READ-INT command (**0x01**) and gets the device acknowledge followed by a REPEATED START condition before sending the I²C read address (**0x45**). Then, the ECU generates eight clock edges to fetch the Status interrupt byte value and sends NACK at the 9th clock edge before ending the interrupt read routine. In this example, the interrupt value is 0x04, indicating a fully received data frame. Then, the ECU generates a REPEATED START condition before sending the I²C read address (**0x45**). The DCB1M I²C read address is sent to activate the RX-DATA command to fetch data bytes from the RX-FIFO. The ECU generates nine clock edges for each data byte. On each ninth clock edge of each data byte fetching, the ECU sends ACK when the INT pin is low and NACK if INT is high. In this example, after the second data byte, the INT is high so the ECU sends NACK at the 9th clock edge. Then, the ECU starts again the READ-INT routine. The DCB1M replies with a Status interrupt byte value of 0x30, indicating EOF and the RX-FIFO is empty. Then the ECU ends the reception routine with a STOP condition.

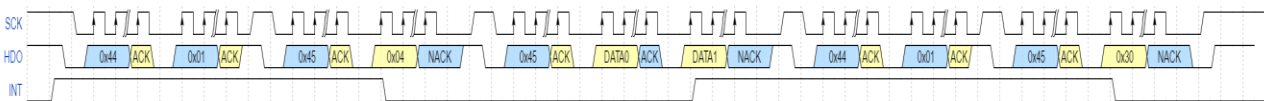


Figure 35 - reception sequence with I²C interface

Blue color – Data from ECU to DCB1M

Yellow color – Data from DCB1M to ECU

10. Specifications**Table 37 - Absolute maximal rating**

Parameter	Symbol	Comments	Min.	Typ.	Max.	Unit
Input voltage, DC	V_{im}		-0.6	3.3	3.9	V
Output voltage, DC	V_{om}		-0.6	3.3	3.9	V
Ambient temperature	T_{am}		-40		125	°C
Storage temperature	T_{sm}		-55		150	°C

Table 38 - Recommended operation conditions

Parameter	Symbol	Comments	Min.	Typ.	Max.	Unit
Supply Voltage	V_{DVCC} V_{AVCC}		3.0	3.3	3.6	V
Supply Voltage Ripple	V_{CC_RIP} A_{VCC_RIP}	Max 2.5MHz, waveform type of triangular		50m		V-p-p
Ambient operating temperature range	T_A		-40		105	°C
Minimum high-level input voltage	V_{IH}		2			V
Maximum low-level input voltage	V_{IL}				0.8	V
Minimum high-level output voltage	V_{OH}		2.4			V
Maximum low-level output voltage	V_{OL}				0.4	V
Maximal output current	I_{out}	see Table 1				
Maximum input current	I_{IN}		-1		1	µA

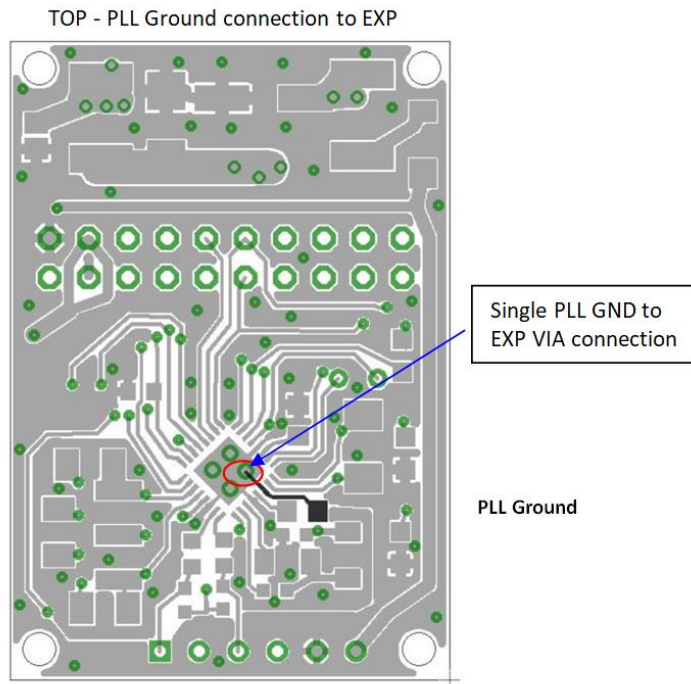
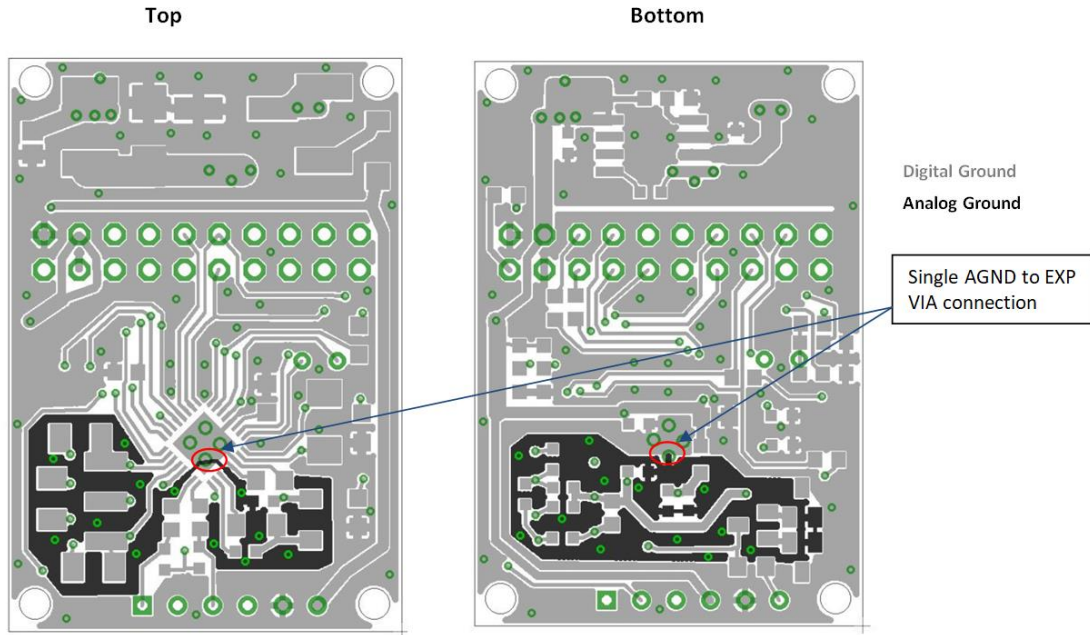
Table 39 - Device Characteristics

Parameter	Symbol	Comments	Min.	Typ.	Max.	Unit
External components requierments						
Powerline coupling capacitor	$C_{coupling}$	Capacitor rate should be selected with respect to powerline voltage		2.2		nF
Protection diodes capacitance	D_{protec}			10		pF
Capacitor at VCAP	V_{cap}		1	4.7		µF
Capacitor at PLLCAP	PLL_{cap}		1			µF
Capacitor at VREF	$VREF_{cap}$		1			µF
Inductor at L1	L1	see 2.5.4		3.3 / 18		µH
Inductor at L2	L2			15		µH
L1 pin input capacitance					1	pF
Crystal frequency	$Xtal_{freq}$	see 2.5.3		16		MHz
Crystal frequency tolerance	$Xtal_{ppm}$				50	±ppm
AC signals characteristics						
Tx signal at TXO	TXO_{lev_1}	TXON high (transmission is active) see 3.2.6.		1		V-p-p
	TXO_{lev_2}			2		V-p-p
TXO input impedance	TXO_{in}	TXON low (transmission is not active)	5.3k			Ω
TXO output impedance	TXO_{out}	TXON high (transmission is active)		18		Ω
TXO driving strength	I_{TXO}	TXON high (transmission is active)	33		66	mA
Rx signal at RXI	RXI_{lev}		10m		3.3	V-p-p
RXI input impedance	RXI_{in}	An external 5.1kΩ series resistor to the RXI pin must be installed.	5.1k			Ω

Parameter	Symbol	Comments	Min.	Typ.	Max.	Unit
Carrier Frequency in-band (channels selection)	F_c	Selection resolution is 100kHz, a total of 251 carrier frequencies, see 3.2.4	5		30	MHz
Adjacent channels spacing	F_{adj}	The space between two adjacent channels operating over the same powerline.	1.5			MHz
Maximal Powerline bitrate (see 0)	PLC _{br0}	Code 0 is selected			1.4	Mbit/s
	PLC _{br1}	Code 1 is selected			1	Mbit/s
	PLC _{br2}	Code 2 is selected			0.49	Mbit/s
	PLC _{br3}	Code 3 is selected			0.255	Mbit/s
Timing requirements of UART/SPI/I²C protocols interfaces						
UART bitrate	UART _{br}	ECU UART bitrate	115.2k		2M	bit/s
SPI SCK	SCK _{spi}	ECU SPI SCK speed			8	Mbit/s
I ² C SCK	SCK _{i2c}	ECU I ² C SCK speed			1	Mbit/s
FIFOs size		see 3.2.3			1024	Byte
The minimal delay between two consecutive PLC frames	T_{cons_delay}			50		μ s
Timing requirements of DMX/RDM protocols interfaces						
DMX bitrate	T_{bit}	Output on HDO		4		μ s
Break signal	T_{break}			200		μ s
MAB	T_{mab}			20		μ s
Stop bit length	T_{sb}			8		μ s
Timing of device operation modes						
Power-cycle/ hard-reset	T_{init}	Initialization time after power-cycle or hard-reset event.		2		ms
Carrier frequency setting	T_{freq_cng}	Carrier frequency change process time		1		ms
Current Consumption @ 3.3V						
Normal TX mode – low power	I_{Tx_lp}	TXON high (transmission is active)		77		mA
Normal TX mode – high power	I_{Tx_hp}			95		mA
Normal RX mode	I_{RX}	TXON low (transmission is not active)		46		mA
Enhanced sleep (SLP1)	I_{slp1}	see 4.3		120		μ A
Fast wake-up (SLP2)	I_{slp2}	see 4.3		1000		μ A
Low-power (SLP3)	I_{slp3}	see 4.3		85		μ A
Deep Sleep (SLP4)	I_{slp4}	see 4.3		65		μ A

11. DCB1M PCB layout recommendation

Note: Analog ground layer and GND PLL should be connected to the digital ground near the Exp pad.

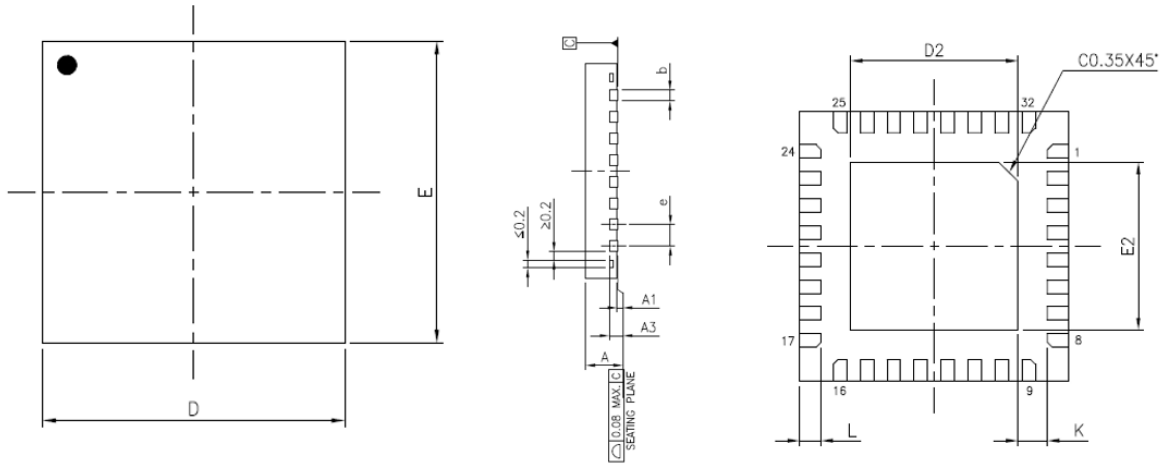


- ✓ VCC and DGND layout traces should be as wide as possible. Connect a 0.1uF capacitor between each VCC and DGND pin, as close as possible to the pins.
- ✓ It is recommended to keep the traces connecting the 3.3V power supply to VCC pins as short as possible with wide PCB traces.
- ✓ Connect AGND to EXP with a single short trace.
- ✓ Connect PLL_GND to EXP with a single short trace.
- ✓ Connect L1, L2, C13, and C3, C5, C7, C8, C11, and C12 as close as possible to their pins.
- ✓ Connect R1 as close as possible to the RXI pin.
- ✓ Connect all filtering caps as close as possible to their pins.
- ✓ Connect crystal and its capacitors close to OSCI and OSCO pins. Keep DGND plan around them.

12. Package, Mechanical

The device package is QFN 32 5mm x 5mm.

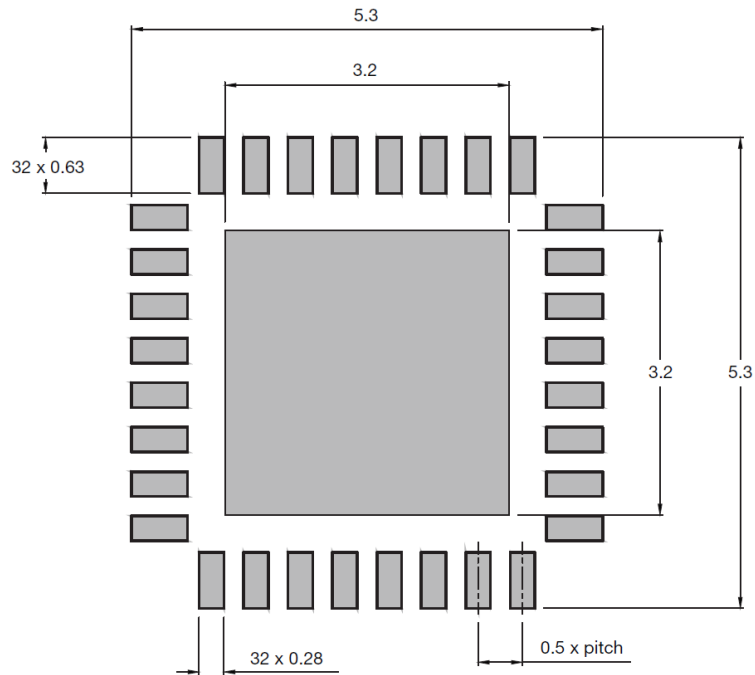
12.1 Mechanical Drawing



D2			E2		
MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
3.15	3.20	3.25	3.15	3.20	3.25

SYMBOLS	MIN.	NOM.	MAX.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF.		
b	0.18	0.25	0.30
D	5.00 BSC		
E	5.00 BSC		
e	0.50 BSC		
L	0.35	0.40	0.45
K	0.20	—	—

12.2 PCB drawing



12.3 Soldering profile

The soldering reflow profile is according to IPC/JEDEC J-STD-020 (MSL3).

- The peak temperature (TP) is 260°C.
- Holding time is between 60 sec to 120 sec between TH min 150°C to TH max 200°C.
- Liquidus temperature (TL) is 217 °C. Liquidus time is between 60 sec to 150 sec.
- TL to TP max ramp-up is 3°C/sec.
- TP to TL max cooldown rate is 6°C/sec.
- Max time above 255°C (Tp) is 30 sec.

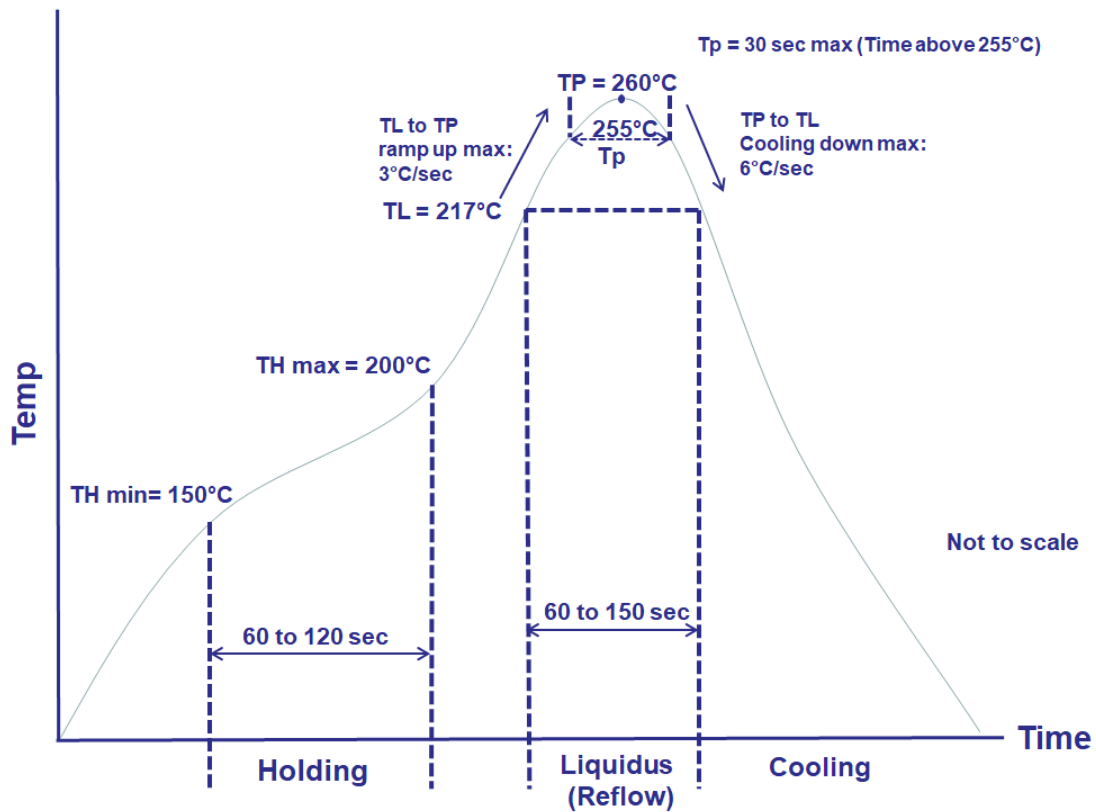


Figure 36 - Representation of IPC/JEDEC J-STD-020 (MSL3) profile

Test Environment

Figure 37 depicts the DC-BUS Test environment that allows testing the DCB1M devices in an emulated lab DC powerline environment.

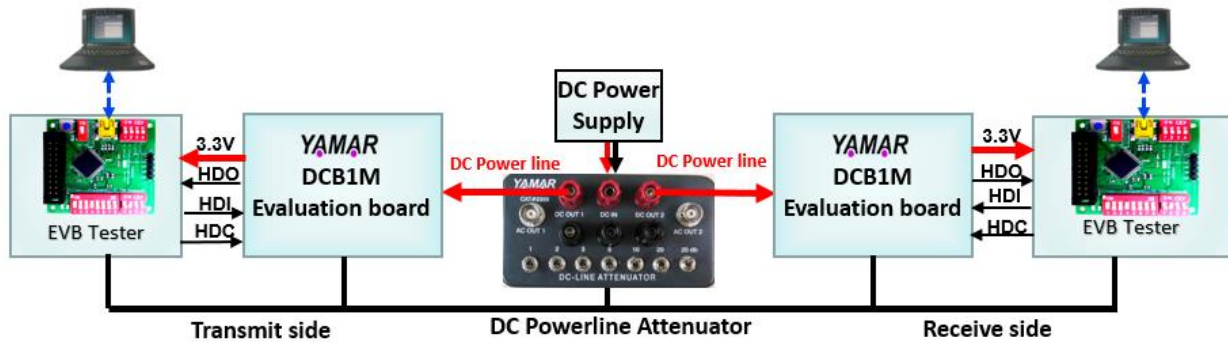


Figure 37 - DC-BUS Test environment

This test environment consists of two DCB1M evaluation boards (EVB), two USB interfaces, and PC test software. The DC-powerline attenuator is optional.

On the transmitting side, the USB interface generates repeatedly a predefined test message [a b c ...x y z] saving the need for a second PC. At the receiver side, the test message is transferred from the EVB through the USB interface to a PC. The Test program analyzes the received predefined messages and performs BER statistics including error byte, miss byte, and noise byte counting indications.

The DC-powerline attenuator is used to test the communication in variable attenuation levels (0-61dB), emulating a DC powerline environment. When connecting the EVB directly to a power supply, it is recommended to add in serial to the power supply an inductor (>22uH) to avoid strong attenuation due to the power supply input filtering capacitors.

Revision History

Rev.	Date	Description
0.7	31/03/2019	Initial preliminary revision.
0.72		formatting
0.73	2/8/2019	Rearrange paragraphs
0.74	2/9/2019	Update typical schematic, Table 34.
0.75	23/9/2019	Editing.
0.76	02/10/2019	Update Table 2 and Figure 7.
0.77	14/11/2019	Update Figure 4 and NSLEEP pin description.
0.78	19/01/2020	Update Table 2, sections 5.5, 7.4, and 8.3.
0.79	17/02/2020	Update Table 24, Table 31, and Table 2. Update 2.5.3.1. Add section 3.7 and UUID[47:0] registers in section 5.
0.80	22/03/2020	Add soldering profile in section 11.3
0.81	16/06/2020	Update RTR handling description.
0.82	01/08/2020	Update 2.5.3.1, 2.5.5, and Table 35.
0.83	11/01/2021	Editing.
0.84	16/01/2022	Update Table 2.
0.85	15/05/2022	Update $T_{\text{cons_delay}}$ in Table 35.
0.86	12/06/2022	Edit description steps in 8.5.5.1.
0.87	17/04/2023	Add DMX/RDM interface section and update relevant sections accordingly.