TLE4978

User manual

1 Intended audience

This user manual is written for experienced hardware and software engineers involved in the implementation of the device into a system. It is the responsibility of the system integrator to ensure that the product is suitable for the chosen application and that the procedures described within this user manual are correctly followed.

2 Scope and purpose

This is the user manual of the product family XENSIV™ TLE4978. The TLE4978 is a high-bandwidth, high-precision, automotive, coreless current sensor with analog output, over-current detection (OCD) and zero crossing detection (ZCD) features. The current sensor is a fully integrated with the primary conductor where the measurement current flows, differential hall elements & embedded metal coils senses the magnetic field generated primary conductor. Therefore, two signal paths are used: one with Infineon's well-established and robust Hall technology path to capture DC and low-frequency current signal, and an inductive coil path to capture the high-frequency current signal. The current is sensed differentially by two Hall plates and two coils that cancel out interfering common-mode (stray) magnetic fields. These two paths are summed up to to enable a high bandwidth sensing device. The properties of the coil increase SNR as frequency increases, minimizing noise seen at the output. This means that by TLE4978 limiting the bandwidth with external filter is not needed because of extremely low noise despite the high-bandwidth. Therefore the system can benefit from the low delays and high accuracy on higher harmonics.

The internal construction provides galvanic isolation (basic isolation) between the high voltage (input) primary conductor and the low voltage secondary (output) side. The preprogrammed full-scale current ranges and the different output modes enable the feasibility of the integration in the different application scenarios. The TLE4978 sensor is provided in a small 16 pin custom DSO-300mil surface mount package with the conductor leads, which is compatible to SOIC package type current sensors.

The coreless concept without the use of a magnetic flux concentrator enables significant miniaturization, enhanced linearity, high bandwidth and avoids negative effects commonly known from sensors using flux concentration techniques (saturation, hysteresis).

The sensor is developed in compliance to ISO 26262 (second edition 2018), supporting safety requirements rated up to ASIL B and is equipped with internal safety mechanisms. Detailed information are available in the Safety Manual [2]. The user manual describes the output modes, sensing principle in Chapter 1 and the detailed technical aspects of the device required for integration in a system in Chapter 2. The internal block diagram and the typical application circuit are available in the product datasheet [1]. TLE4978 features an integrated EEPROM enabling high flexibility at system level. The document provides a description of the available interface commands in Chapter 3. The full set of programmable settings is explained in Chapter 4.

Initial errors due to soldering at the system level can be canceled out through a single point, room temperature calibration. The device is intrinsically robust against stray fields thanks to the differential sensing technology, and crosstalk between nearby phases in multi-phase systems can be compensated at system level as explained in Chapter 5.

The devices to which this user manual applies are listed in the table "related devices" below.

For the device's specification, please refer to the product datasheet [1].

Table of contents

Table of contents . . . 2
1 Functional description . . . 4
2 System integration . . . 6
2.1 Power-Down Behavior . . . 6
2.2 Application circuit . . . 6
2.2.1 Typical Application Circuit for Single Ended Mode . . . 6
2.3 Schematic recommendations for Different Use Cases . . . 8
2.3.1 Schematic diagrams for different use cases without using DCDI . . . 8
2.3.2 Schematic diagrams for different use cases with using DCDI . . . 11
2.4 Layout recommendations . . . 12
2.4.1 Layout recommendations for power side . . . 14
2.4.2 Layout recommendations for signal side . . . 15
3 Digital Control Diagnostic Interface (DCDI) . . . 17
3.1 DCDI feature set . . . 17
3.2 DCDI activation . . . 17
3.3 System setup . . . 18
3.4 Operation in harsh environment . . . 19
3.5 Internal State Machine (ISM) . . . 19
3.6 Protocol specification . . . 19
3.6.1 Byte fields format . . . 20
3.6.1.1 Command byte . . . 20
3.6.1.2 Address byte . . . 20
3.6.1.3 Data bytes . . . 21
3.6.1.4 Safety byte . . . 21
3.6.2 Command types . . . 22
3.6.2.1 Read command . . . 22
3.6.2.2 Write command . . . 22
3.6.2.3 Broadcast command . . . 23
3.6.3 Protocol timings . . . 24
3.6.3.1 DCDI timing parameters . . . 25
3.7 High level commands . . . 25
3.7.1 UNLOCK command . . . 27
3.7.2 SET ADDRESS command . . . 28
3.7.3 GET DIAGNOSTIC command . . . 28
3.8 Address spaces . . . 28
3.9 Auto-addressing . . . 29
3.10 CRC . . . 30
3.10.1 CRC3 calculation . . . 30
3.10.2 CRC5 calculation . . . 32

Table of contents

3.10.3 Calculation for Master Read frame . . . 36
3.10.4 Calculation for Master Write frame and Slave Response frame . . . 37
4 EEPROM . . . 38
4.1 EEPROM content . . . 38
4.1.1 Configuration registers . . . 39
4.1.2 Other registers . . . 40
4.2 EEPROM programming instructions . . . 41
4.2.1 EEPROM line read . . . 41
4.2.2 Entire EEPROM programming . . . 42
4.3 EEPROM CRC calculation . . . 45
5 Stray fields and crosstalk . . . 49
5.1 Differential measurement principle . . . 49
6 Glossary . . . 50
7 References . . . 51
8 Revision History . . . 52
Disclaimer . . . 53

1 Functional description

TLE4978 is an internal current rail, magnetic, core-less current sensor with intrinsic galvanic isolation, automotive-qualified, supporting 3.3 V and 5 V power supply applications. It provides fast and accurate solution for measuring high-frequency currents in DC/DC converters and other switch mode power supply applications. It is developed as a Safety Element out of Context (SEooC) for safety requirements up to ASIL B. These features make it ideally suited for current transformer & shunt replacement in applications running at high voltages.

Typical applications are high voltage automotive converter applications like on board chargers (OBC), and HV to LV standalone DC-DC converters, as well as industrial applications like photovoltaic converters, general purpose inverters, switch mode power supplies (SMPS), and fast over load and over-current detection.

The device's functional architecture is illustrated in the simplified block diagram above. The hybrid architecture combining Hall effect and coil sensing elements, enables high bandwidth and extremely low noise at the same time, eliminating the need for trade-off between speed and resolution of the current sensing.

The measurement principle is differential, thus ensuring high stray field suppression. The coreless concept without the use of a magnetic flux concentrator enables significant miniaturization, enhanced linearity and avoids negative effects commonly known from sensors using flux concentration techniques (saturation, hysteresis)

Principle of operation: The sensor features two sensing elements, or probes, properly positioned on opposite sides of the current rail to realize a differential sensing principle (see Chapter 5.1 for more information on differential measurement principle). Each of the two sensing elements consists in a combination of a Hall probe and a coil. Both the Hall probes and the coils provide a signal that is proportional to the magnetic field component perpendicular to the sensing elements plane (plane of chip surface).

The Hall sensing, equipped with well-established spinning and ripple compensation techniques, provides DC measurement capability with particularly low offset while the coils provide high signal at high frequency with outstanding signal to noise ratio. The two signals are properly combined over frequency by the analog signal processing so that the Hall signal is prevalent at low frequency and the coil signal is prevalent at high frequency, thus effectively exploiting the complementary characteristics of the two sensing technologies.

The combination of the 2 signals over frequency domain is illustrated in the simplified block diagram above : the signals produced by the Hall and coils sensing elements are both low-pass filtered at the cross-over frequency (f_X) and added together. The original coil signal is proportional to frequency and once exposed to the low-pass filter becomes flat after f_X. On the other end, the Hall signal is originally flat over frequency so that after the low-pass filtering it remains flat only before f_X, going down afterwards. Therefore the combined signal obtained by summing Hall and and coil signals is flat over frequency and its bandwidth is limited only by the bandwidth of the output buffer amplifier that is used to drive the analog output voltage of the sensor. The cross-over frequency f_X is placed at the frequency that minimizes the overall output noise.

In this way, the need for compromising between speed and high resolution in current sensing is practically eliminated because the output bandwidth can be enlarged without any penalty on the output noise. On the contrary, the output rms noise performance will be vastly improved with respect to conventional Hall sensors despite the much higher bandwidth.

Signal processing: the analog signal processing is digitally assisted providing a high bandwidth analog signal path as well as the possibility for digital regulation of gain and offset. Embedded junction temperature and mechanical stress sensors are used to continuously and accurately regulate the gain and the offset of the signal path employing proprietary techniques. The continuous regulation is implemented by embedded firmware and digital processing ensuring high resolution and high flexibility in calibration, to produce superior accuracy at the sensor output. This analog/digital partitioning enables at the same time a high-speed analog interface as well as an extremely high accuracy of the sensor output over the full operating temperature range.

Internal monitoring functions: several internal monitoring checks are continuously performed inside the sensor in order to detect any fault of the internal blocks that could lead to the violation of the top level safety requirement. These safety mechanisms are not interfering in any way with the regular function of the sensor

and only in case of detection of an internal fault, they trigger a safety reaction. This is indicated by the OCD pin brought to a logical LOW state and the ZCD pin to a logical HIGH state.
Please refer to the safety manual for more details about internal monitoring functions.

DCDI Interface: the sensor is equipped with a one-wire digital interface (DCDI) that enables communication between an external micro-controller and the sensor device. This can be used to program custom settings for OCD and ZCD functions. As an example, the user can program the OCD threshold and the OCD deglitching filter as well as the ZCD threshold and hold time. It is also possible to read the sensor's internal junction temperature value and get diagnostic information via the DCDI during normal operation without interfering in any way with the sensing functions.

Over-Current Detection (OCD): the device integrates a dedicated, fast hardware supervision path that compares the measured current against programmable limits without loading or altering the main analog signal path. The OCD indication is provided on the open-drain OCD output pin. Thresholds, hysteresis, and an optional blanking/deglitch time can be configured via the DCDI interface and stored in EEPROM. When an over-current condition is detected, the OCD output is pulled down and the function can be configured as auto-recover (clears when the current returns within limits) or latched until explicitly cleared. The OCD path provides very low-latency indication suitable for protection use cases.

Zero-Current Detection (ZCD): the device provides a Zero Crossing Detection (ZCD) push-pull output on ZCD pin. ZCD output toggles between logic value 0 and 1 every time the input current crosses 0. The ZCD threshold is by default 0A but this value can changed by the DCDI interface. With this the ZCD can be anticipated or delayed with respect to the actual crossing event occurrence.

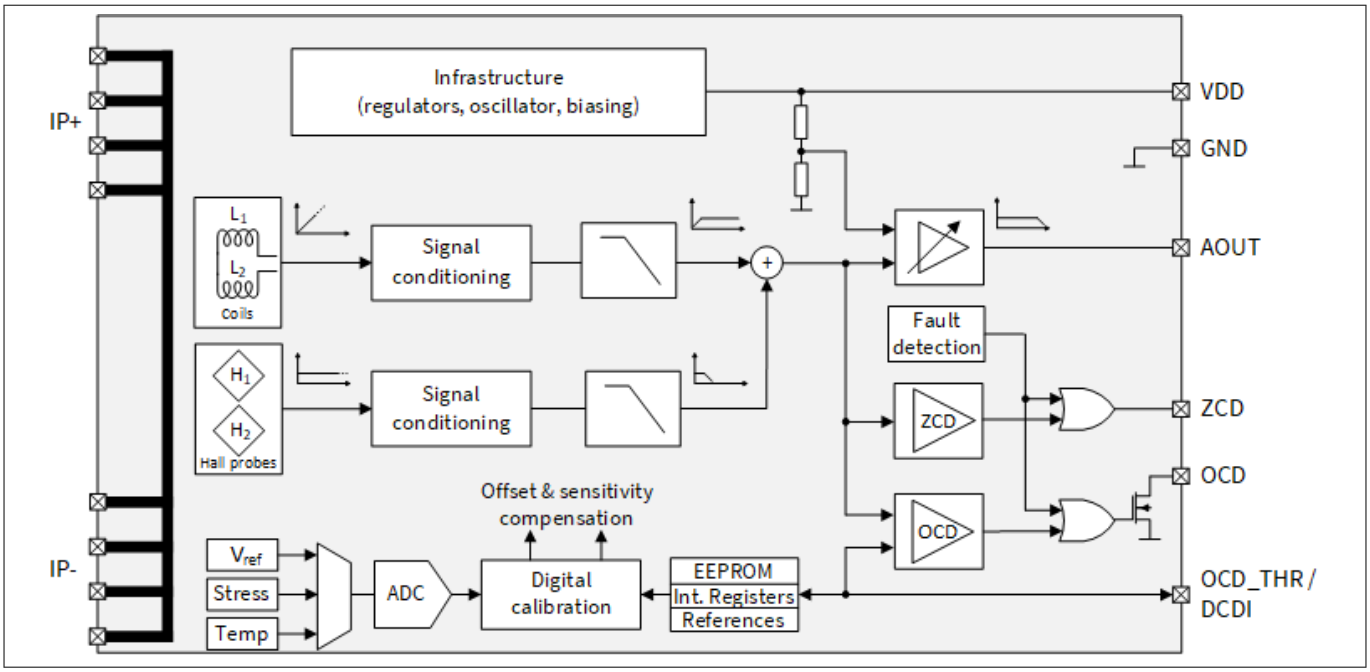


Figure 1 Device functional block diagram

2 System integration

***Note**: This chapter provides guidance on how to connect the TLE4978 in a typical application systems. The information in the following sections is not part of Infineon's component specification, and Infineon does not guarantee its accuracy or completeness. Customers are responsible for determining component suitability and validating and testing the design implementation for system functionality. Adhering to these guidelines can maximize the sensor's potential for a specific use case.*

2.1 Power-Down Behavior

Due to the inherent galvanic isolation of the device, minimal attention is needed to powering down the device, provided that the limits in the Absolute Maximum Ratings table of the respective datasheet [1] are not exceeded on any pins. The isolated current input and the low-voltage signal chain can operate independently; either can be energized while the other is shut down. It is important to ensure that the isolation barrier capabilities are not exceeded. Specifically, the low-voltage power supply can be powered down while the isolated input remains connected to an active high-voltage signal or system.

2.2 Application circuit

The key feature sets of the TLE4978 provide significant advantages in any application where an isolated current measurement is required.

- Low noise and high bandwidth in one sensor eliminate the need for compromising between speed and resolution improving accuracy, reducing phase delay and harmonic distortions.
- Galvanic isolation provides a high isolated working voltage.
- Excellent common mode immunity to input voltage transients.
- Magnetic based measurement simplifies system level solution without the need for a power supply on the high voltage (HV) side.
- DC to ultra-high bandwidth, along with high accuracy, supports diverse position requirements in any application.
- A low impedance input current path minimizes the power dissipation.
- Low temperature & life time drift eliminate the need for end of line calibrations.

These advantages increase system-level performance while minimizing complexity for any application where precision current measurements must be made on isolated currents. Specific examples and design requirements are detailed in the following section.

2.2.1 Typical Application Circuit for Single Ended Mode

The following figure shows application block diagram in a single-ended mode. For the detailed application circuit please refer to the product datasheet [1]. DC to high bandwidth, along with high accuracy, supports diverse position requirements, for example, in the on-board charger (OBC) application, as shown in the below diagrams. The on-board charger (OBC) topology with a resonant converter is a configuration commonly utilized in electric vehicle (EV) charging systems. This topology leverages resonant circuitry to efficiently manage the transfer of power from the grid to the vehicle's battery.

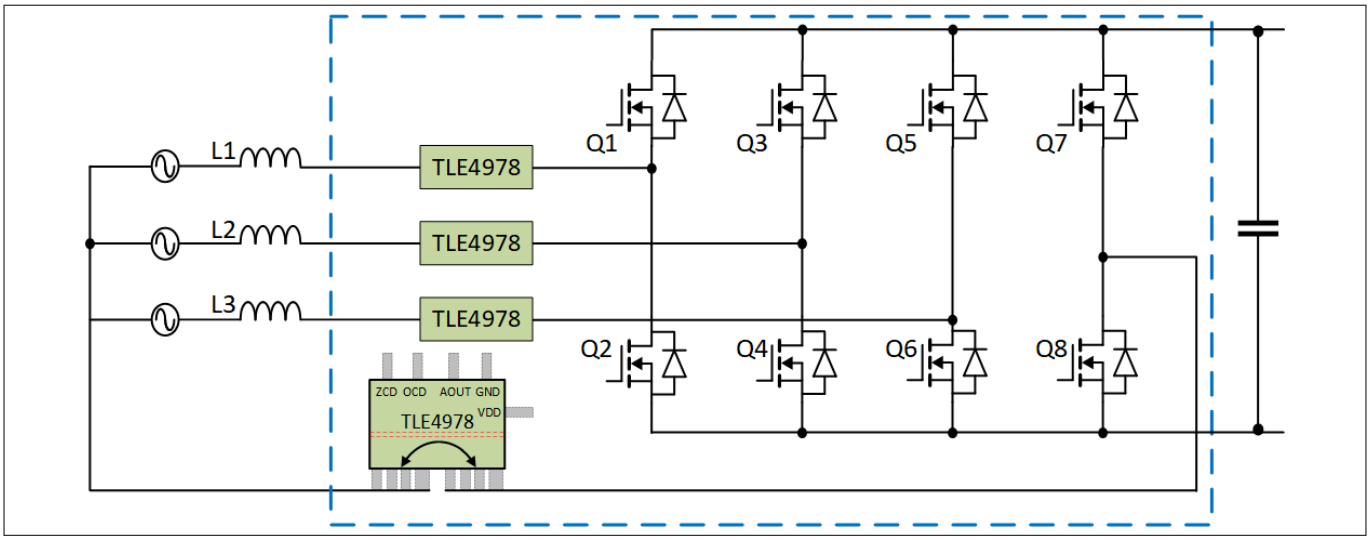


Figure 2 Typical Application Circuit for primary stage of OBC including TLE4978

The primary stage of an OBC is an AC to DC converter. Controlled PFC stages improve power transfer efficiency, and faster power switches in modern PFC stages reduce size and improve efficiency. The PFC stage is typically directly connected to AC power grids, but sensing poses challenges due to voltage spikes and transients. However, the TLE4978 construction provides intrinsic isolation, high CMTI, and high levels of isolation between HV current sensing nodes and low-voltage control circuitry.

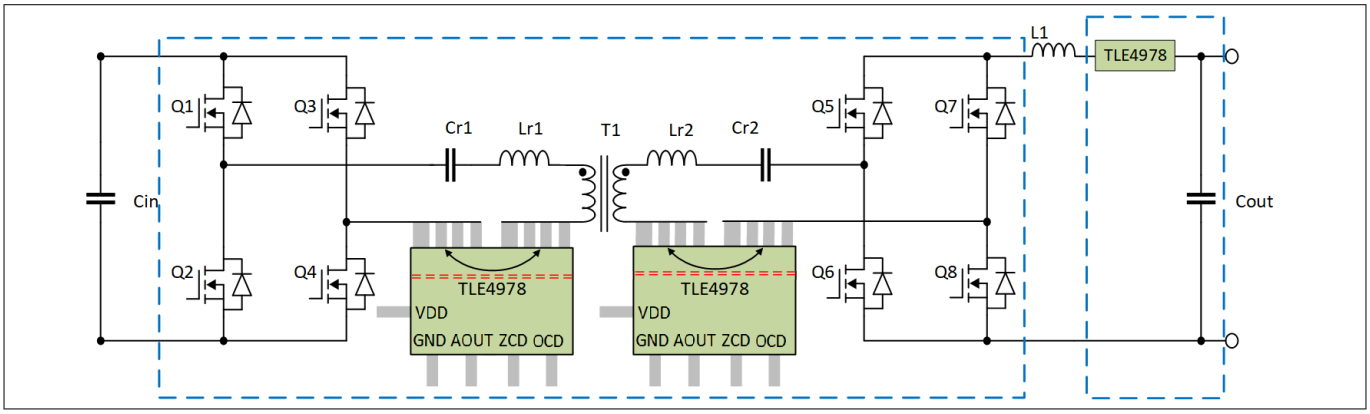


Figure 3 Typical Application Circuit for secondary stage of OBC including TLE4978

High-bandwidth current sensors play a crucial role in resonant DC-DC converters (second stage of an OBC) by enabling accurate and dynamic monitoring of current flow within the system. High bandwidth and high accuracy of TLE4978 allows for precise measurement of current dynamics, enabling the system to maintain optimal performance. The sensors are equipped to swiftly detect and respond to transient changes in current, ensuring that the control system can adapt to dynamic load conditions and maintain stability within the resonant converter. To accurately capture high-frequency current waveforms without distortion, TLE4978 is designed to exhibit low phase shift, preserving the integrity of the measured signals and facilitating precise control and feedback within the resonant converter system.

The following figure shows the input current and respective output voltage waveform, for example, for single-ended output bidirectional current mode.

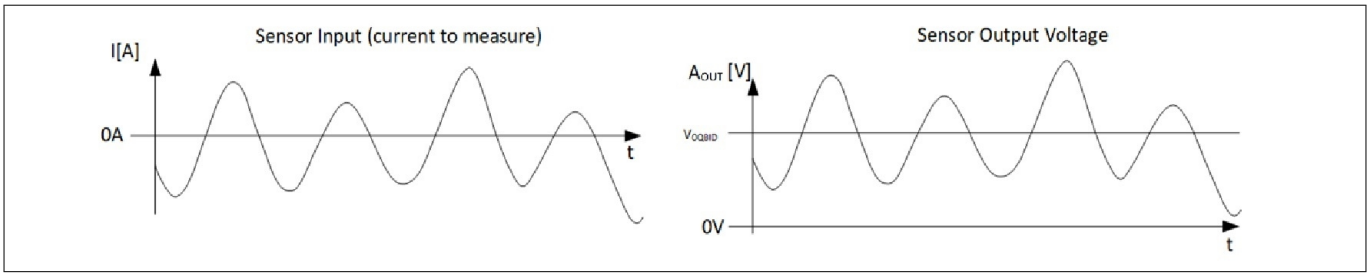


Figure 4 Sensor output for single-ended mode

Where, V_{OQBID} represents the quiescent voltage, while A_{OUT} denotes the sensor's analog output voltage.

2.3 Schematic recommendations for Different Use Cases

This section illustrates how to connect the TLE4978 on the system level. The different electrical schematic diagrams below depict the connections and components involved in measuring electrical current for various use cases. For example, if the customer wishes to utilize certain features exclusively, the diagrams below explain how to connect the sensor to the system.

2.3.1 Schematic diagrams for different use cases without using DCDI

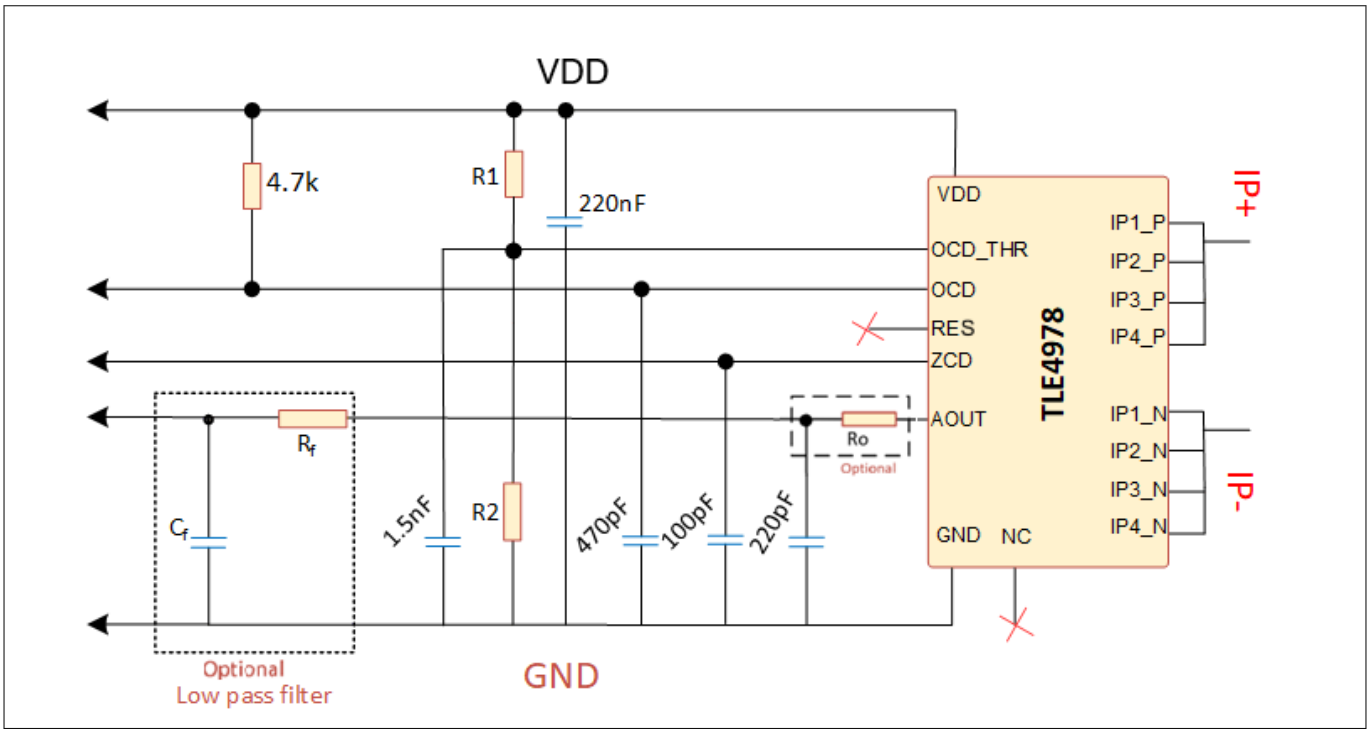


Figure 5 TLE4978 application circuit example, with AOUT, OCD and ZCD functions used. OCD threshold is set by voltage on the OCD_THR pin.

The figure above represents an application circuit example where the AOUT pin is connected and the OCD and ZCD functions of the device are used.

The optional low pass filter depends on what kind of EMC influence is expected in the circuit. If the EMC influence is not known then, since every filter impacts the signal on the AOUT pin, it is recommended to only foresee the placing of the components in the layout. Start by placing: R_f = 0 Ω, C_f = not populated; and adapt if needed. The optional R_0 is needed for a robust dV/dt or dI/dt performance. When a good dV/dt or dI/dt performance is needed the resistance of R_0 can be up to 150 Ω. But be careful with the cutoff frequency, with R_0 = 150 Ω and the capacitor of 220 pF it will be around 4,8MHz. If dV/dt or dI/dt are not of importance the R_0 can be lower or even bridged.

In this example the OCD threshold is set by the voltage on the OCD_THR pin, which is given by the voltage divider *R1/R2*. For information on how to set the OCD threshold by the OCD_THR pin, please refer to the product datasheet [1].

The ZCD is set to the default value. For the default value of the ZCD please refer to the product datasheet [1].

The ZCD can be set via the DCDI interface. For information on how to set the ZCD please refer to Chapter 3 and product datasheet [1].

A fail in the internal safety checks leads to a fault indication that is externally visible on the OCD and ZCD pin. For more information on fault indications please refer to the product safety manual [2].

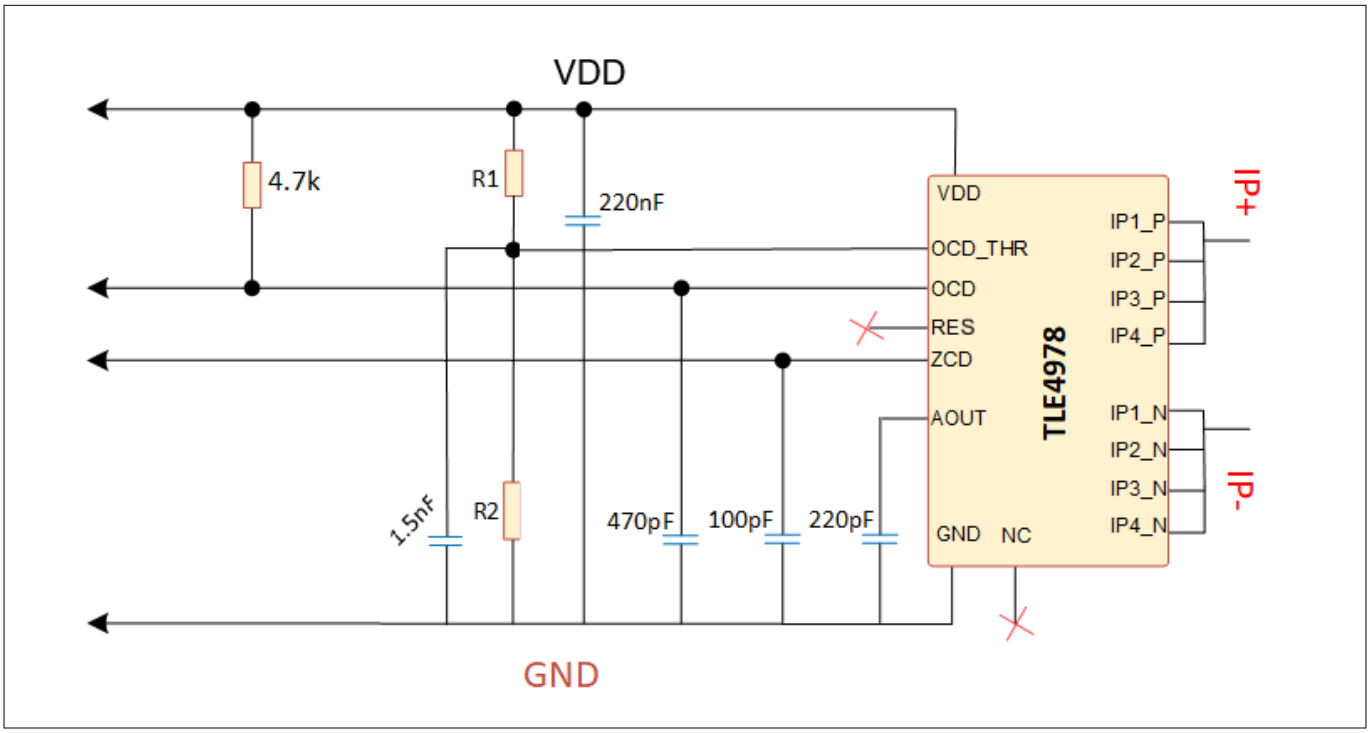


Figure 6 **TLE4978 application circuit example, with OCD and ZCD functions used only. OCD is set by voltage on the OCD_THR pin.**

The figure above represents an application circuit example where the AOUT pin is not connected and only the OCD and ZCD functions of the device are used.

In this example the OCD threshold is set by the voltage on the OCD_THR pin, which is given by the voltage divider *R1/R2*. For information on how to set the OCD threshold by the OCD_THR pin, please refer to the product datasheet [1].

The ZCD is set to the default value. For the default value of the ZCD please refer to the product datasheet [1].

A fail in the internal safety checks leads to a fault indication that is externally visible on the OCD and ZCD pin. For more information on fault indications please refer to the product safety manual [2].

In the system environment where the OCD threshold is set by the voltage on the OCD_THR pin and the voltage on the VDD is unstable it is recommended to implement programmable shunt regulator as show in the figure below.

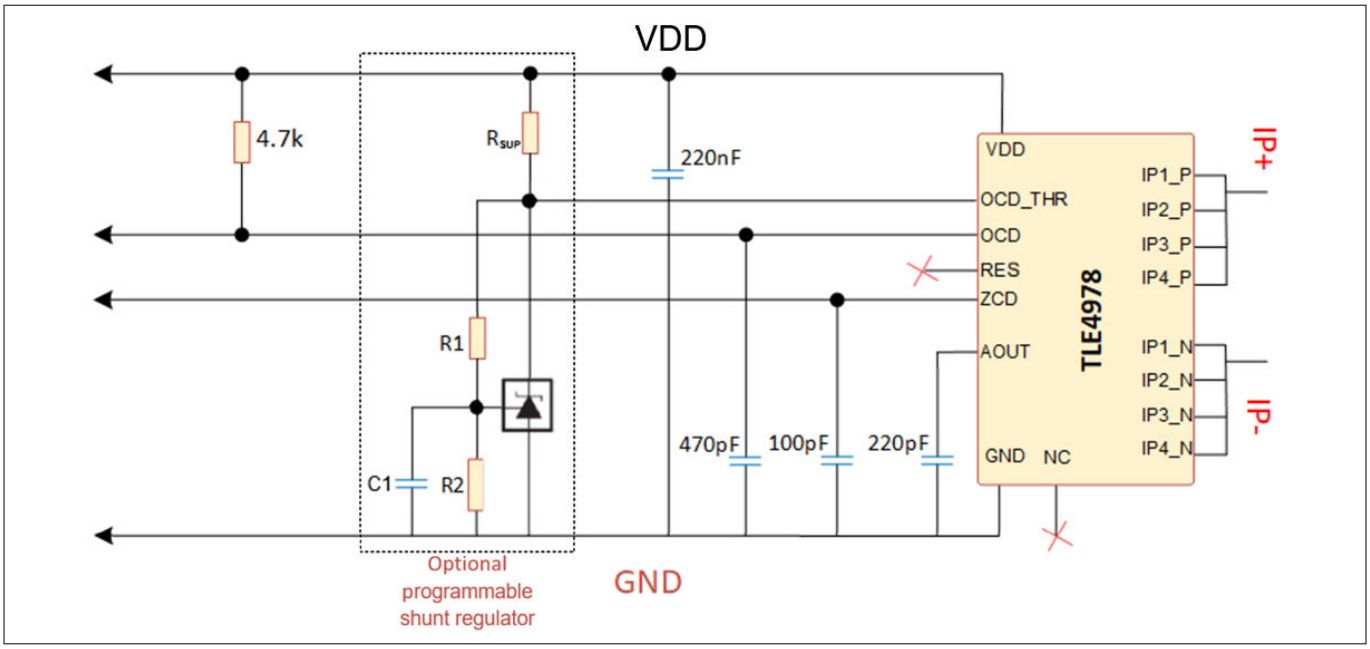


Figure 7 **TLE4978 application circuit example where the OCD is set by voltage on the OCD_THR pin and a programmable shunt regulator is used to stabilize the voltage on the OCD_THR pin.**

The programmable shunt regulator stabilizes the voltage on the OCD_THR pin. For values of R_{SUP}, *R1*, *R2* and *C1* please refer to the documentation of the selected programmable shunt regulator.

For information on how to set the OCD threshold by the OCD_THR pin, please refer to the product datasheet [1].

A fail in the internal safety checks leads to a fault indication that is externally visible on the OCD and ZCD pin. For more information on fault indications please refer to the product safety manual [2].

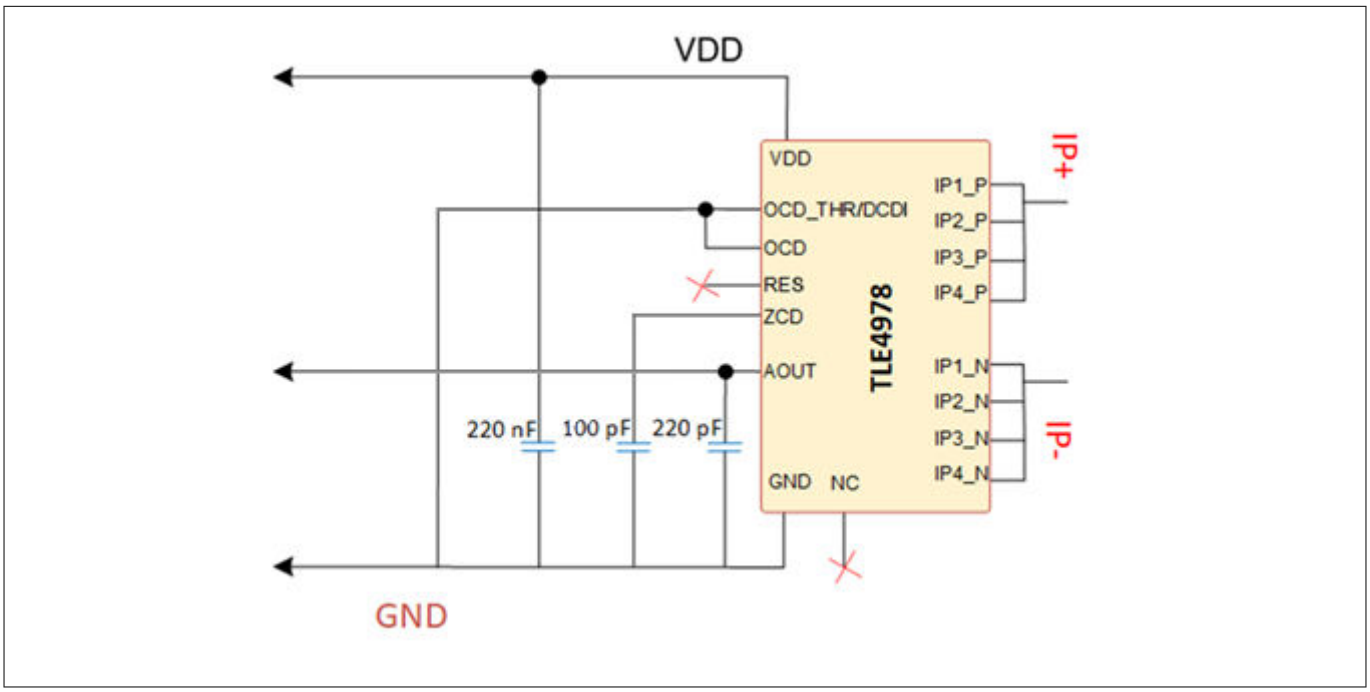


Figure 8 **TLE4978 application circuit example where AOUT only is used.**

The figure above represents an application circuit example where only the AOUT pin is used and the OCD and ZCD functions of the device are not used.

If the OCD function is not used the OCD pin must be connected to GND. If the OCD_THR and DCDI functions are not used the OCD_THR/DCDI pin must be connected to GND.

If ZCD function is not used it must be connected to a capacitor as seen in the circuit above.

Also, in this case the optional low pass filter as well as he optional R_0 (both not depicted in the figure above) can be implemented if needed.

2.3.2 Schematic diagrams for different use cases with using DCDI

Note: *If DCDI is not active by default ("eep_ocd_thr_dis" bit set to 0) the DCDI activation sequence must be followed (Chapter 3.2). Otherwise, if a pull-up to VDD is present on the DCDI pin, unwanted oscillations are present on the sensor output pin (AOUT).*

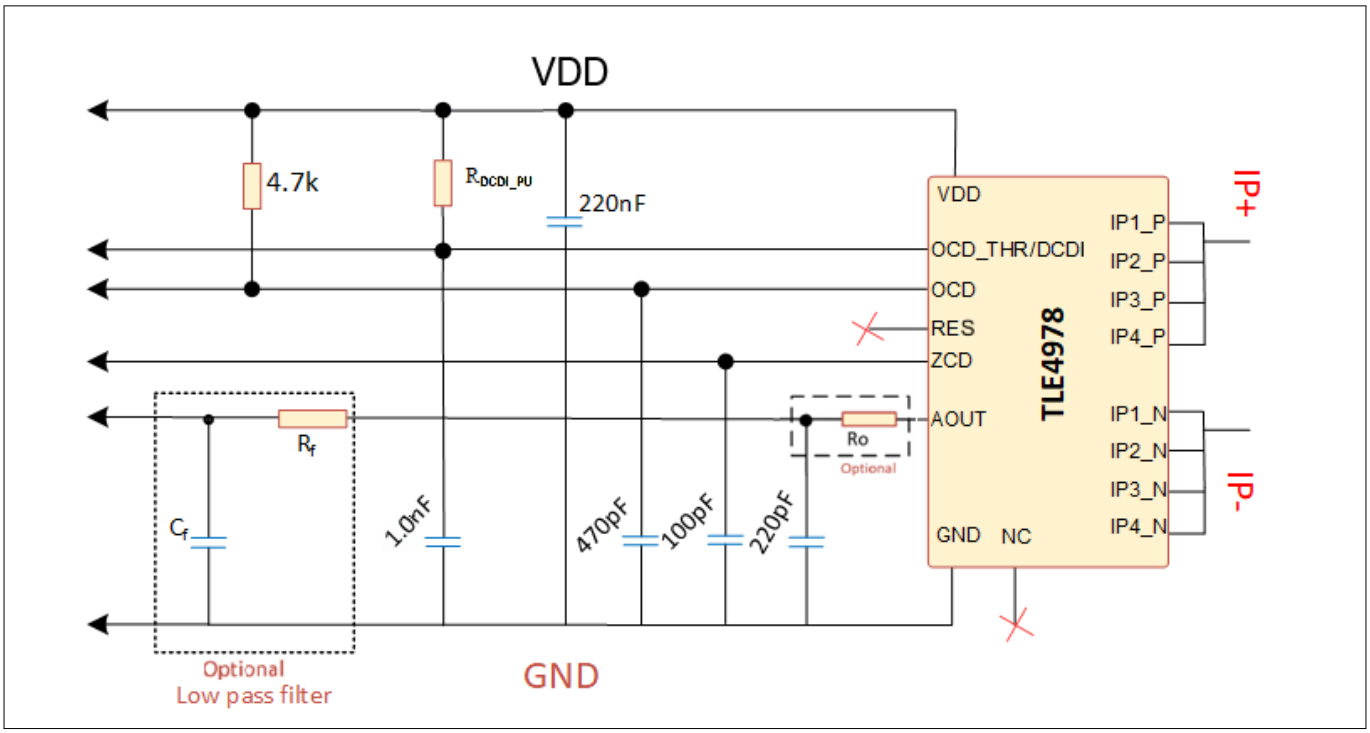


Figure 9 TLE4978 application circuit example, with AOUT, OCD and ZCD functions used. OCD threshold and additional ZCD features are set by programming over the DCDI.

The figure above represents an application circuit example where the AOUT pin is connected and the OCD and ZCD functions of the device are used. OCD and ZCD functions are both set via the DCDI interface.

The optional low pass filter depends on what kind of EMC influence is expected in the circuit. If the EMC influence is not known then, since every filter impacts the signal on the AOUT pin, it is recommended to only foresee the placing of the components in the layout. Start by placing: R_f = 0 Ω, C_f = not populated; and adapt if needed. The optional R_0 is needed for a supreme dV/dt or dI/dt performance. When a good dV/dt or dI/dt performance is needed the resistance of R_0 can be up to 150 Ω. But be careful with the cutoff frequency, with R_0 = 150 Ω and the capacitor of 220 pF it will be around 4,8MHz. If dV/dt or dI/dt are not of importance the R_0 can be lower or even bridged.

In this example the OCD threshold is set by programming over the DCDI. The suggested value for the pull-up resistor on the DCDI pin (R_{DCDI_PU}) is 1 kΩ. For information on how to set the OCD threshold by the DCDI please refer to Chapter 3 and product datasheet [1].

The ZCD can be set via the DCDI interface. For information on how to set the ZCD please refer to Chapter 3 and product datasheet [1].

A fail in the internal safety checks leads to a fault indication that is externally visible on the OCD and ZCD pin. For more information on fault indications please refer to the product safety manual [2].

If the system is only using the ZCD function the application circuit example below represents this use case. The suggested value for the pull-up resistor on the DCDI pin (R_{DCDI_PU}) is 1 kΩ.

The ZCD can be set via the DCDI interface. For information on how to set the ZCD please refer to Chapter 3 and product datasheet [1].

A fail in the internal safety checks leads to a fault indication that is externally visible on the ZCD pin. For more information on fault indications please refer to the product safety manual [2].

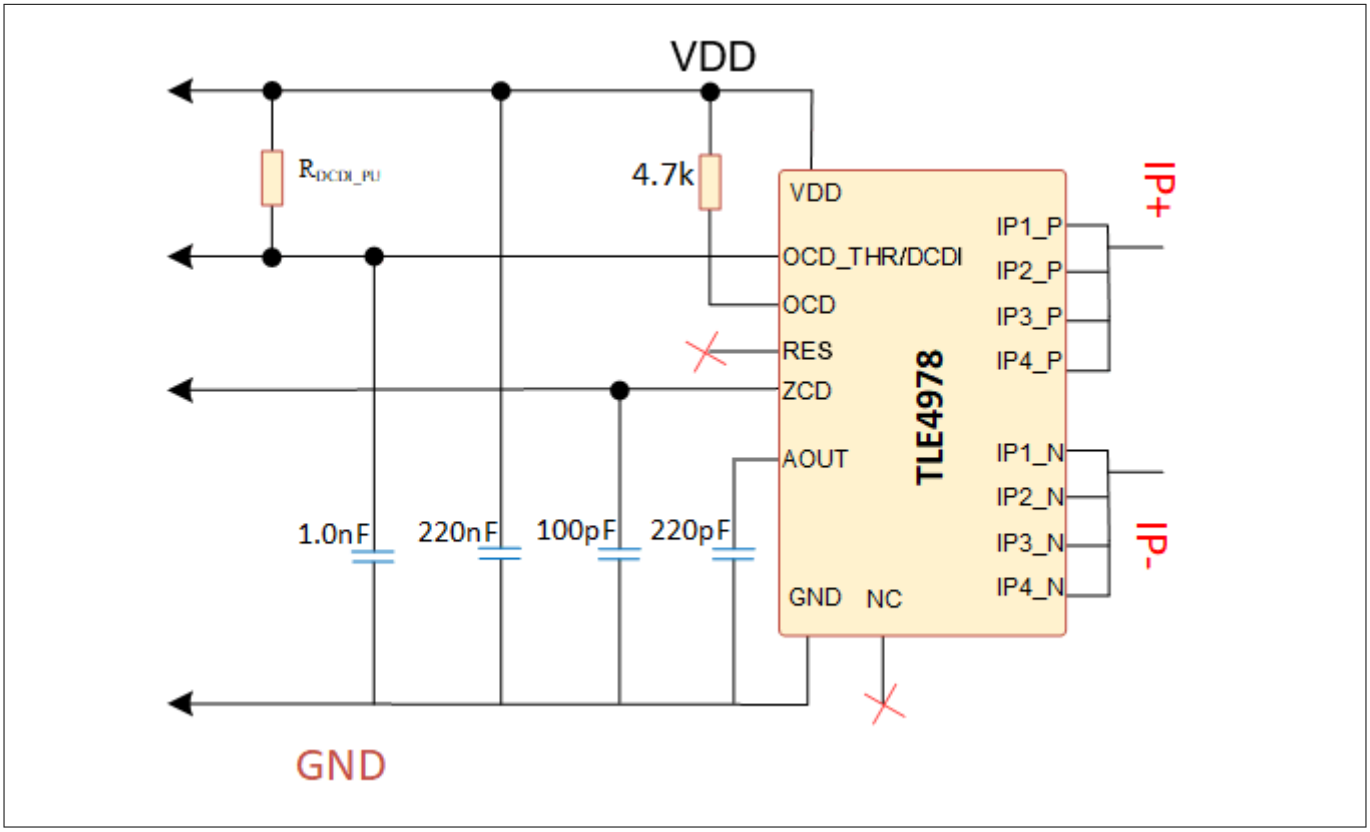


Figure 10 **TLE4978 application circuit example with only ZCD function used. Additional ZCD features are set by programming over the DCDI.**

2.4 Layout recommendations

This section illustrates how to create the layout for the TLE4978 on the system level. The layout instructions in the following sections provide all the information to enhance the sensor performance in the system. The layout presented here can also be found on the TLE4978 evaluation board seen on the pictures below.

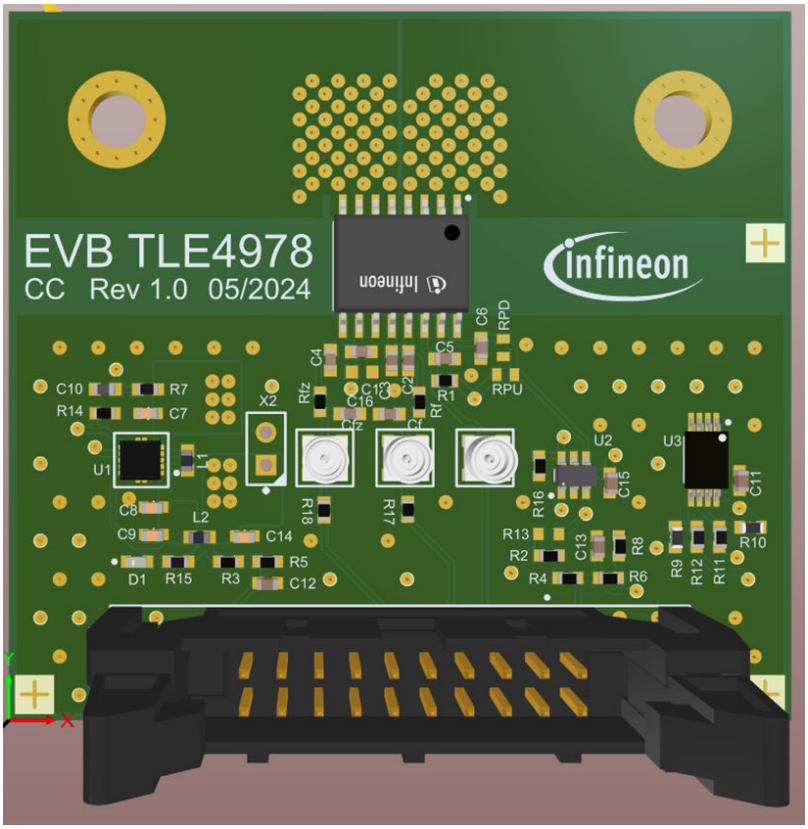


Figure 11 TLE4978 evaluation board

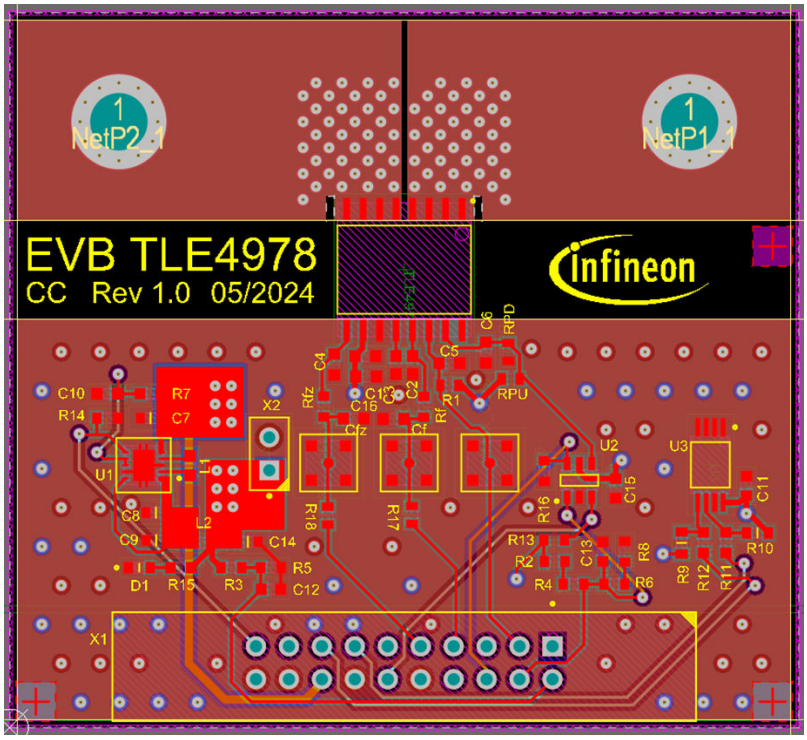


Figure 12 TLE4978 evaluation board top layer

2.4.1 Layout recommendations for power side

The picture below represents the recommendation for the connectors (in violet circles) and current rail. For improving thermal dissipation the connectors must be large and the current rails must be wide.

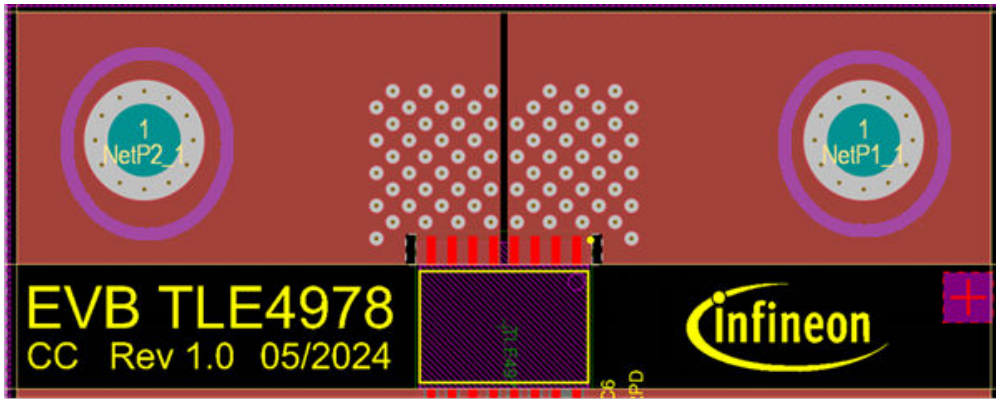


Figure 13 TLE4978 layout connectors and current rail recommendation

To bring current to the top layer from other power layers include vias in your design as represented in the figure below.

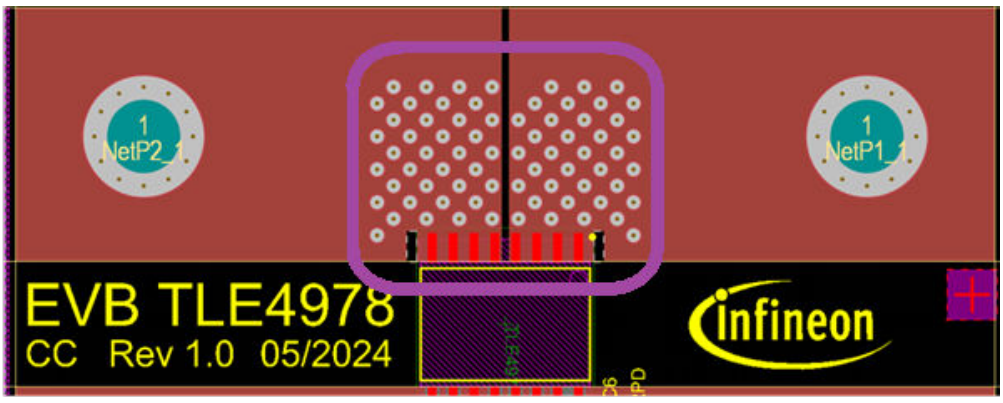


Figure 14 TLE4978 layout vias recommendation

The two slits / keep-outs on each side of the sensor (see violet circles on the figure below) block the current flow straight into the sensor and enable a uniform current distribution across sensor pins. In general the current must be directed in the pin direction inside the pin as depicted with the orange arrows in the figure below.

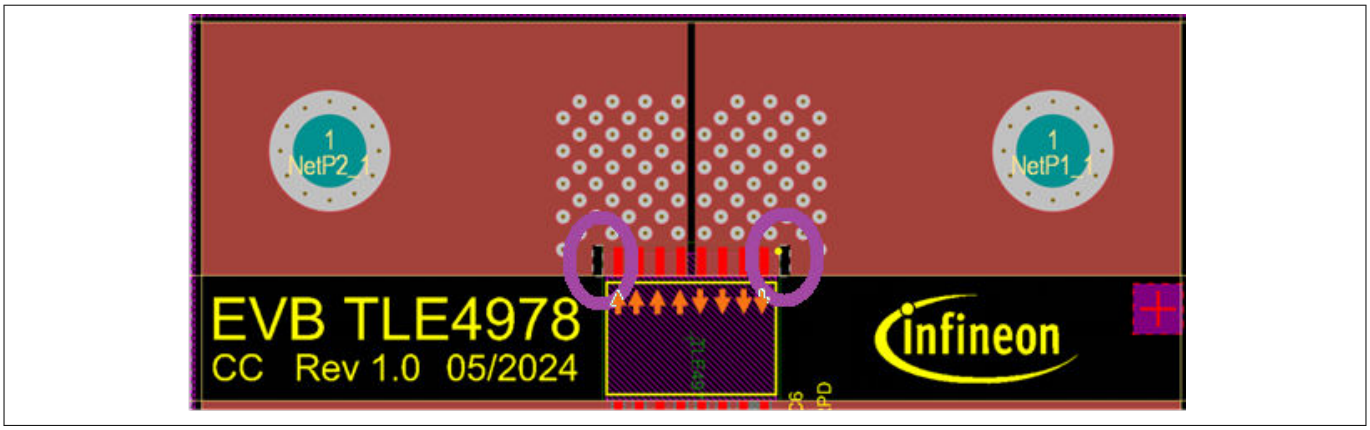


Figure 15 TLE4978 layout slits and current direction recommendation

Include 8mm reinforced isolation clearance and creepage between the HV and the LV side in all layers of your design to fulfill the isolation requirements.

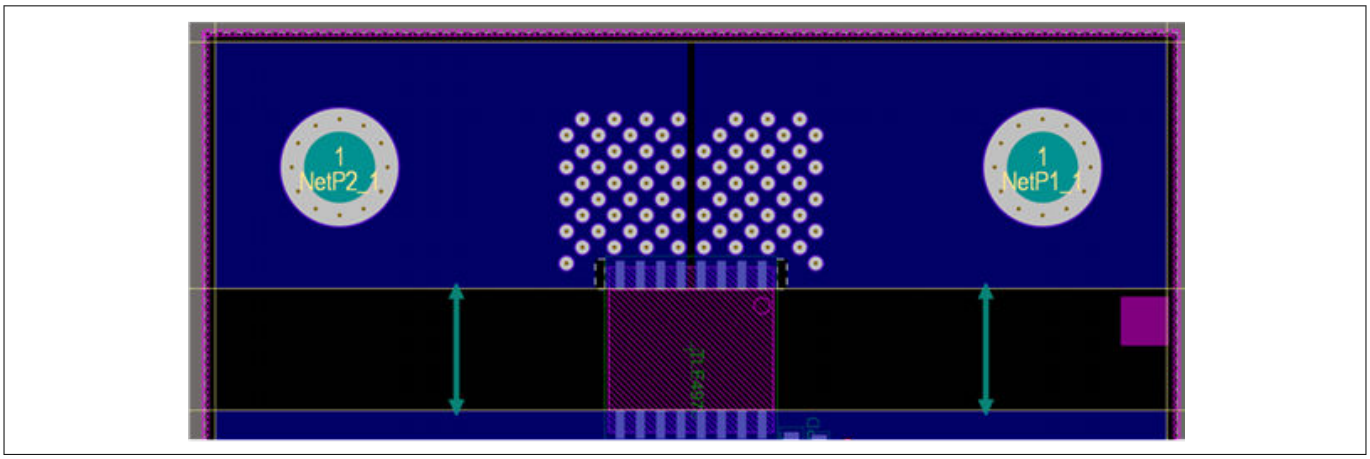


Figure 16 8 mm clearance and creepage for isolation purposes

Additionally make sure that under and near the sensor there is a magnetically free environment:

- No ground or power layers under the sensor. The current flowing underneath the sensor could effect the sensing elements of the sensor.
- No current tracks below the sensor. This reduces all possible magnetic noise to a minimum.

2.4.2 Layout recommendations for signal side

For the signal side layout, when placing components, follow the general rule: "The nearest to the sensor, the better."

The figure below shows VDD and GND pins stabilized by the nearby couple of capacitors (violet circle) and in the case of AOUT, OCD, ZCD and DCDI, the capacitor nearby is as near as possible to the sensor pin.

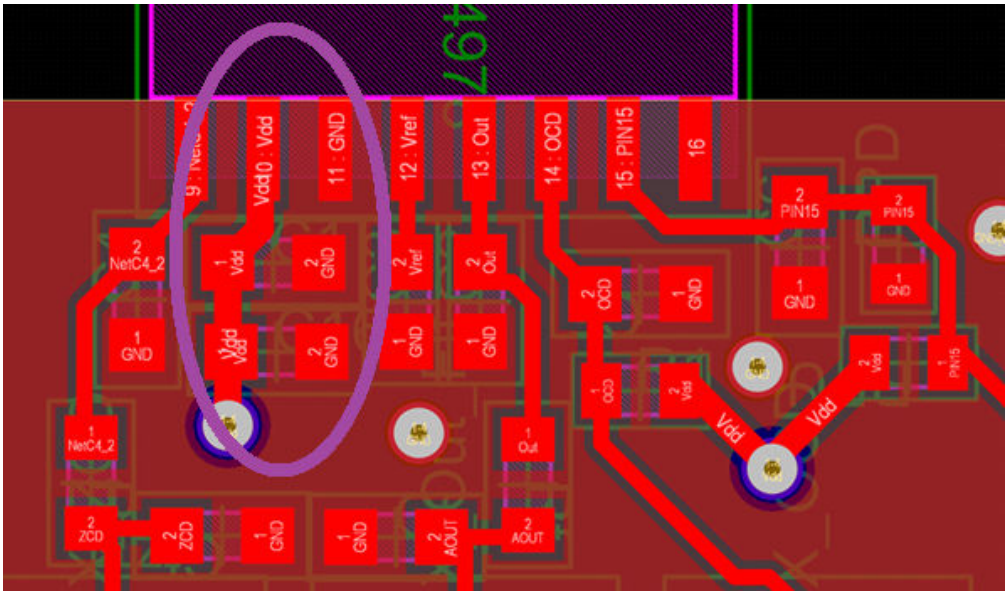


Figure 17 TLE4978 layout of signal side

3 Digital Control Diagnostic Interface (DCDI)

3.1 DCDI feature set

The Digital Control Diagnostic Interface (DCDI) is a high-speed digital interface for high-performance applications where reliability and fault tolerance are critical. The interface's data link layer is based on standard Universal Asynchronous Receiver Transmitter (UART) specifications. The protocol is designed to support direct communication between a microcontroller and a sensor IC to implement extensive, robust diagnostic monitoring and configurability of the sensor, allowing for the reading of the real-time temperature and internal monitoring mechanism status during operation. Moreover, DCDI enables easy programming of the sensor's internal EEPROM at the microcontroller level, without the need for additional external programming circuitry. This simplifies the system design and reduces the overall bill of materials.

The DCDI interface offers several features that enhance its functionality and reliability. The DCDI interface protocol layer provides internal error status information to the master node within each communication frame, and the DCDI end-to-end frame protection provides a failure detection rate of at least 99%. Data transmission and reception are ensuring the integrity of the data and the reliability of the system. The device provides an auto-addressing functionality, allowing the master in the system to address different sensors connected on the bus; DCDI supports a single master, multiple slave nodes network configuration, allowing for the connection of up to 8 sensors per bus. The interface uses a one-wire configuration, reducing the complexity of the system design and improving reliability. The auto-addressing procedure starts with the the slave receiving the auto-addressing "broadcast" command. The slave answers by providing its "dynamic" address (modulated in a voltage level) to the master via the AOUT pin. The master performs the auto-addressing "broadcast" command as long as all slaves provide a unique "dynamic" address on the dedicated AOUT pin. After each iteration, the master addresses the slave using the generated (unique) dynamic slave address and reassign a static address to each slave. The static address can be stored in the device EEPROM at the end of the auto-addressing procedure.

3.2 DCDI activation

Note: *The DCDI activation sequence can be disabled by setting the appropriate EEPROM configuration registers* Table 11*. In this case the DCDI function is always activated at sensor power-on. This setting disables the possibility of setting the threshold with voltage on OCD_THR pin and the over current threshold will always be the one set in the EEPROM, even if the DCDI function is not used.*

The OCD_THR pin is a dual-purpose input pin. The default function of the pin is to adjust OCD threshold with an analog voltage. It can also serve as DCDI communication pin, which provides the user with much more flexibility in programming OCD/ZCD threshold, adjusting hold time or turning on/off specific functions. To make use of the DCDI function, a proper activation sequence has to be performed. After the activation sequence, OCD_THR pin functions exclusively as DCDI communication pin. To activate the DCDI, the user must ensure that the OCD_THR pin is pulled low and the activation sequence (see description below) is completely sent before the end of the activation time window. This will initiate the DCDI activation procedure.

Note: *OCD_THR pin can be pulled low immediately with power-on. The activation is then recognized after ~100us.*

There is a timeout window started after OCD_THR rising edge (start of the DCDI activation sequence) in which the OCD_THR pin shall be kept first high for at least 100us, then pulled low and kept low for at least 100us. If the timeout is reached, the regular function is activated automatically. Otherwise the DCDI mode is activated and the OCD_THR pin is now the DCDI data pin.

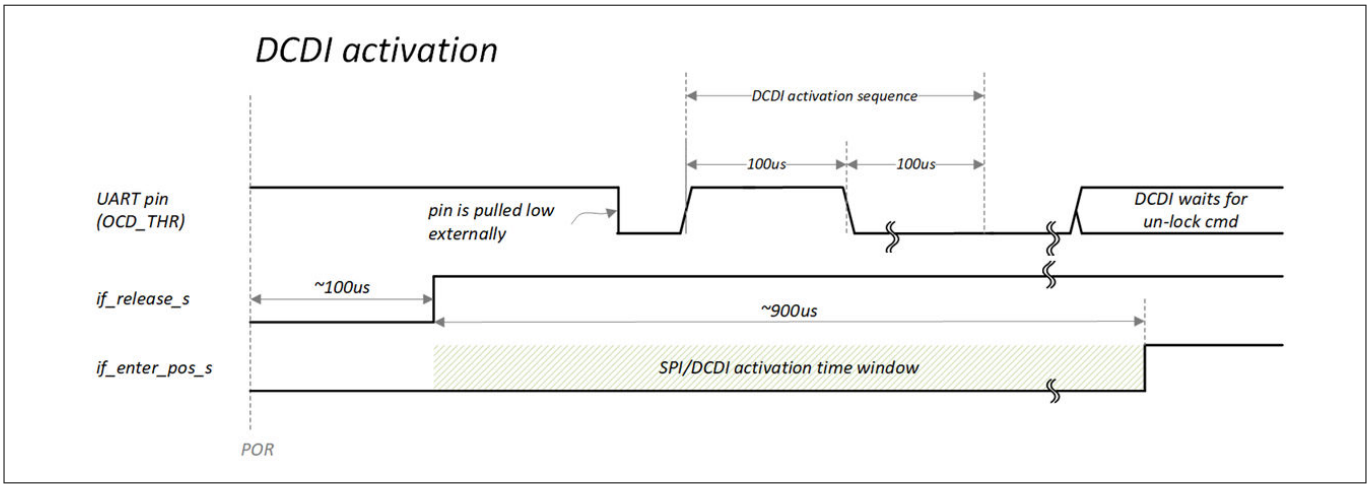


Figure 18 DCDI activation sequence

3.3 System setup

The below figure shows the typical system setup for using multiple current sensor ICs connected to the microcontroller unit through DCDI interface and dedicated analog output signal connections.

The interface enables:

- Connection up to eight sensors to serial bus
- High speed configuration of multiple sensors
- High speed readout of multiple sensor registers

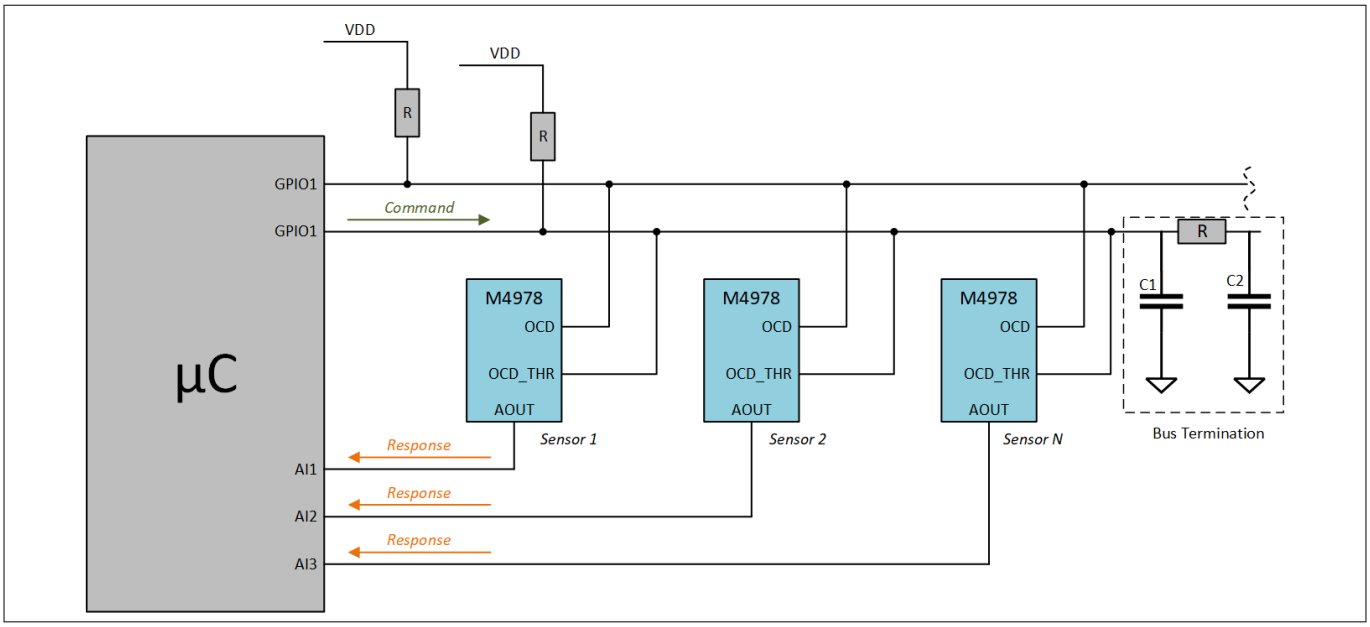


Figure 19 Typical system block diagram for multi sensor interface to microcontroller

In cases where multiple sensors are connected to the same serial bus, two or more slave nodes share a common connection to the microcontroller. As a result, the line must be time-multiplexed to prevent conflicts on the communication bus. To enable this, the outputs of the sensors' DCDI are configured to stay in tristate mode when they are not actively transmitting data. This enables the use of a pull-up resistor connected to the DCDI bus to maintain a high level when the output driver is not actively driving the line, and thus prevents a floating DCDI input at the microcontroller. Furthermore, a unique sensor address must be assigned to all sensors sharing the same DCDI node. These sensor addresses can be assigned to the individual sensor ICs during an auto-addressing procedure triggered by the MCU. The auto-addressing procedure is explained in Chapter 3.9.

3.4 Operation in harsh environment

The DCDI is specifically designed to operate reliably in harsh electromagnetic environments. To mitigate voltage spikes on the DCDI bus resulting from the capacitive coupling of fast voltage transients (e.g., PWM switching events in SiC inverters) a majority vote principle is implemented in the sensor.

Each bit is sampled three times at points S1, S2, and S3, as depicted in the figure below, during the middle of the bit period. The value of the sampled bit is determined using a majority vote mechanism. For instance, if S1 = 0, S2 = 1, and S3 = 0, the bit value is considered to be 0. The sampling interval is given by the selected communication speed. For specific details, Please refer to Chapter 4. The master of the communication must implement similar majority vote concept in order to correctly interpret sensor response in the harsh environment.

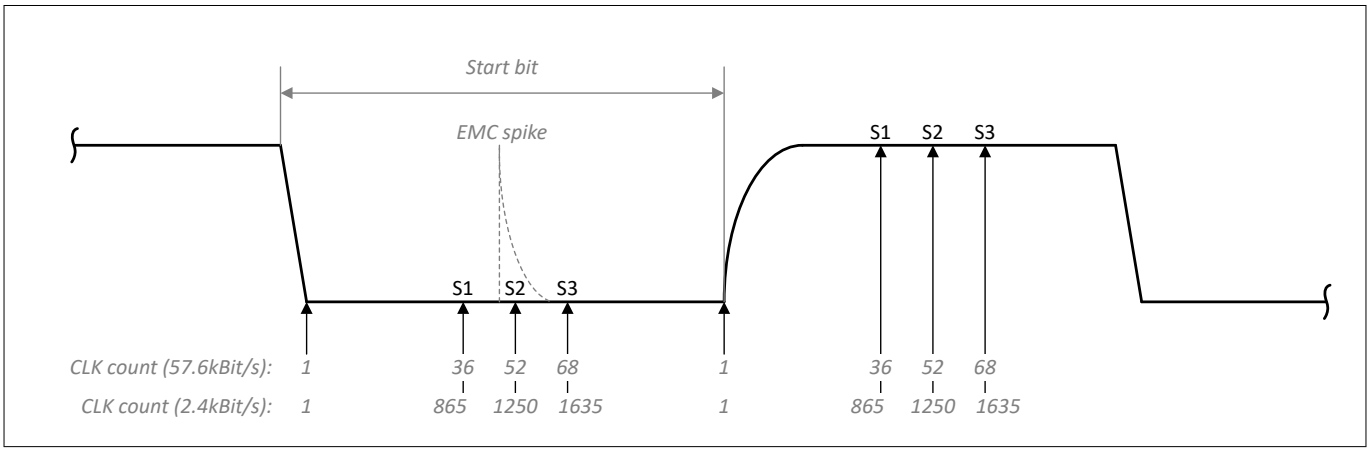


Figure 20 Majority vote concept in DCDI

3.5 Internal State Machine (ISM)

The sensor features a digital signal conditioning concept. The Internal State Machine (ISM) optimizes in real time the sensing performance, providing stable sensor behavior over temperature and lifetime. The ISM has to be disabled during programming, read/write of EEPROM content and writing to internal registers, in order to avoid internal register access conflicts. The reactivation of the ISM happens through power cycle of the device, or by sending the corresponding DCDI command.

3.6 Protocol specification

The figure below shows a standard UART byte field which is the basis for data transfer between sensor and MCU. The LSB of data is transmitted first. The start bit is encoded as low and the stop bit is encoded as high. All byte fields described below follow this format.

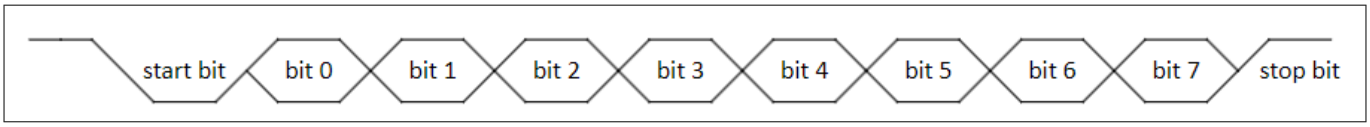


Figure 21 UART byte field in DCDI

The figure below shows the structure of the frame. The communication is always initiated by the master node. Data sent by master node are followed by correspondent slave node response. Each master node command or slave node response is composed by multiple byte fields.

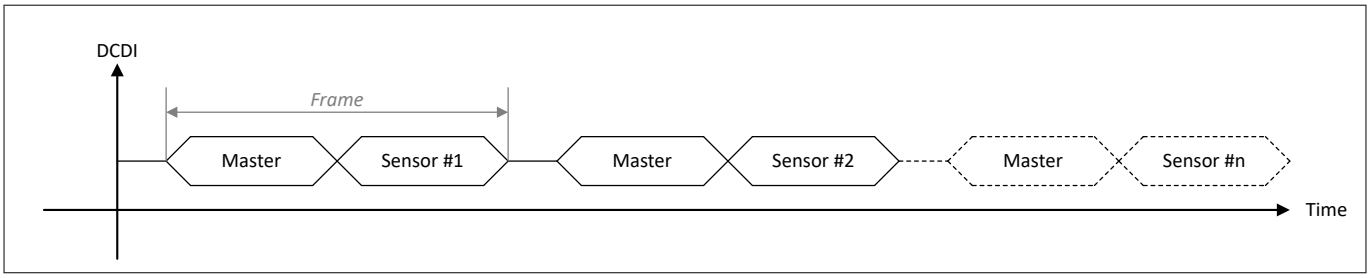


Figure 22 UART frame in DCDI

In case that the start bit is not confirmed or end bit is not present, received byte is ignored.

3.6.1 Byte fields format

This section describes the typical byte fields that form a DCDI communication frame; there are four types of byte fields:

- Command byte (CMD);
- Address byte (ADDRESS);
- Data bytes (DATA);
- Safety byte (SAFETY).

3.6.1.1 Command byte

The command byte is a header part of each frame (of any type). It consists of the fields described in table below.

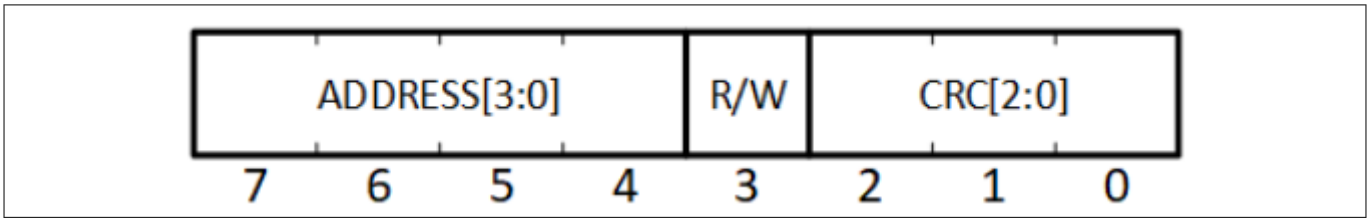


Figure 23 Command byte in DCDI

Table 1 Command byte description

Field name	Bits	Description
SENSOR ADDRESS	CMD[7:4]	Slave node address: • 0_{hex} = Broadcast command; • Other = Slave node address.
R/W	CMD[3]	Read/Write flag: • 0 = Read; • 1 = Write.
CRC	CMD[2:0]	CRC-3-GSM. Calculated for CMD[7:3].

3.6.1.2 Address byte

The address byte contain address of the target sensor internal register.

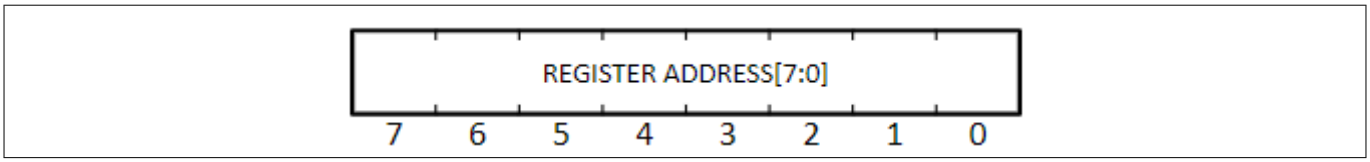


Figure 24 Address byte in DCDI

3.6.1.3 Data bytes

The data transmission consists of two data bytes, resulting in a communication data length of 16 bits. These data bytes are transferred sequentially, with the less significant byte being sent first.

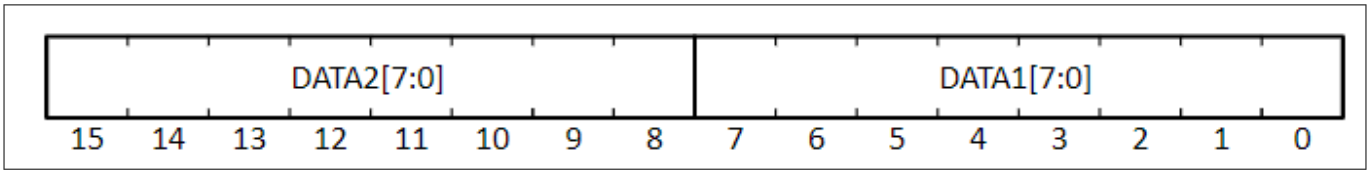


Figure 25 Data bytes in DCDI

3.6.1.4 Safety byte

The safety byte is used for data consistency protection and diagnostic information exchange. It consists of the fields described in table below.

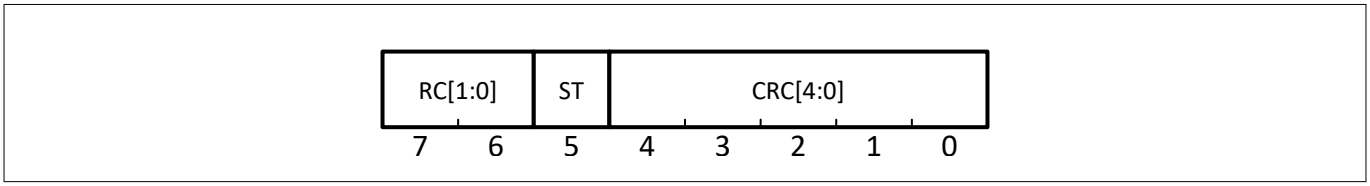


Figure 26 Safety byte in DCDI

Table 2 Safety byte description

Field name	Bits	Description
RC	SAFETY[7:6]	Rolling Counter. Incremented by slave by the number of bytes transmitted in the response. E.g.: • Read command: the device answers with 3 bytes (data bytes and Safety byte), RC incremented by 3; • Write command: the device answers with 1 byte (Safety byte), RC incremented by 1. The master can keep the RC constant for each communication or increment it after each sent frame as additional protection (the RC is part of the CRC calculation therefore a slave answer that refers to an old RC count would be detected).
ST	SAFETY[5]	Status Bit: • 0 = Normal operation; • 1 = Internal error detected.
CRC	SAFETY [4:0]	CRC-5-USB. Calculated for {REGISTER_ADDRESS,DATA1, DATA2, SAFETY[7:5]}.

3.6.2 Command types

This section describes the typical commands that can be implemented in DCDI communication. There are three types of commands:

- Read command;
- Write command;
- Broadcast command.

Note: *Read and Write commands require the user to unlock the DCDI interface using the UNLOCK command and stop the ISM with the DISABLE ISM command in order to avoid access conflicts to the internal register banks, unless otherwise specified.*

3.6.2.1 Read command

The read command is issued by master MCU and is accepted by addressed sensor only. It is used for data readout from the dedicated slave node. The master node sends the correspondent command byte followed by slave node address and safety byte. The slave node answers with a 16 bits data word followed by a safety byte. DATA1 byte is sent first.

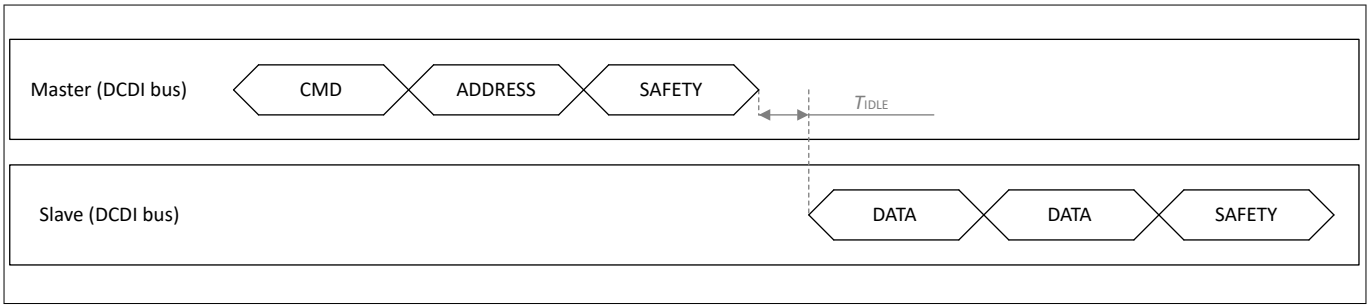


Figure 27 Read command in DCDI

3.6.2.2 Write command

The write command is issued by the master MCU and is accepted by the addressed sensor only. It is used for writing data to the dedicated slave node. The master node sends the correspondent command byte followed by the slave node address, 16 bits data word and safety byte. DATA1 byte is sent first. The slave node responses with safety byte.

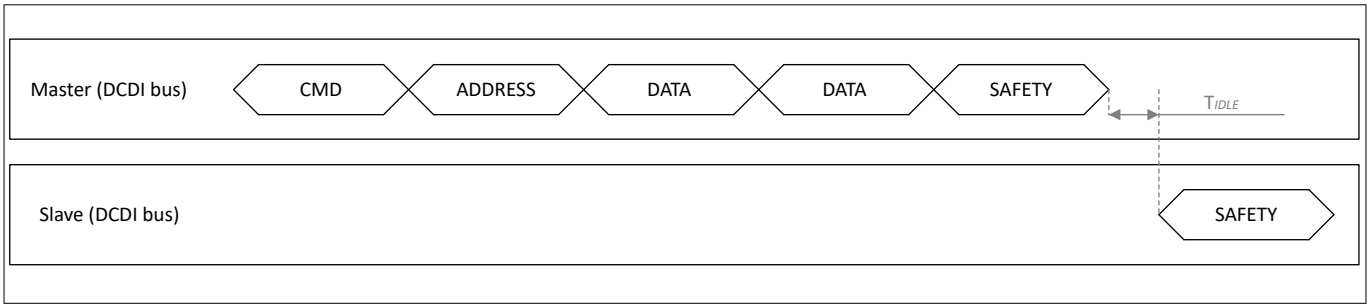


Figure 28 Write command in DCDI

3.6.2.3 Broadcast command

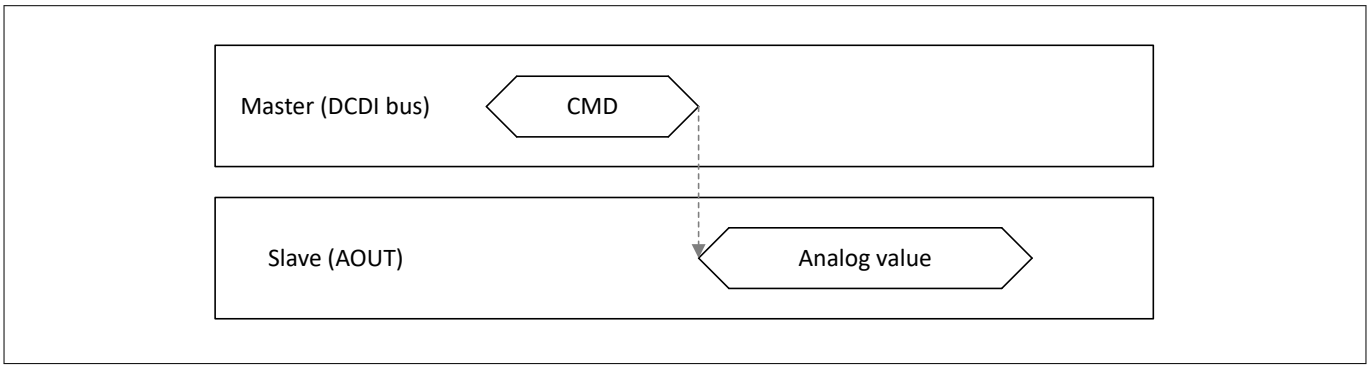


Figure 29 Broadcast command in DCDI

The BROADCAST SET command requires all slave devices to update their dynamic addresses and set analog output voltage to the value proportional to its current address. If the bit [3] of the sensor address is set (refer to Dynamic/Static flag in Figure 36), the BROADCAST SET command does not lead to address change.

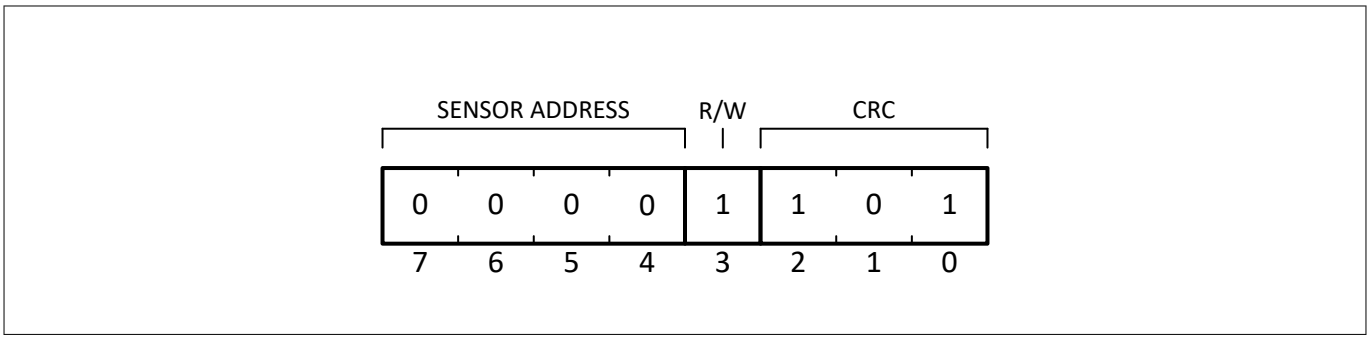


Figure 30 BROADCAST SET command frame in DCDI

The BROADCAST GET command requires all slave devices to set analog output voltage to the value proportional to its current address.

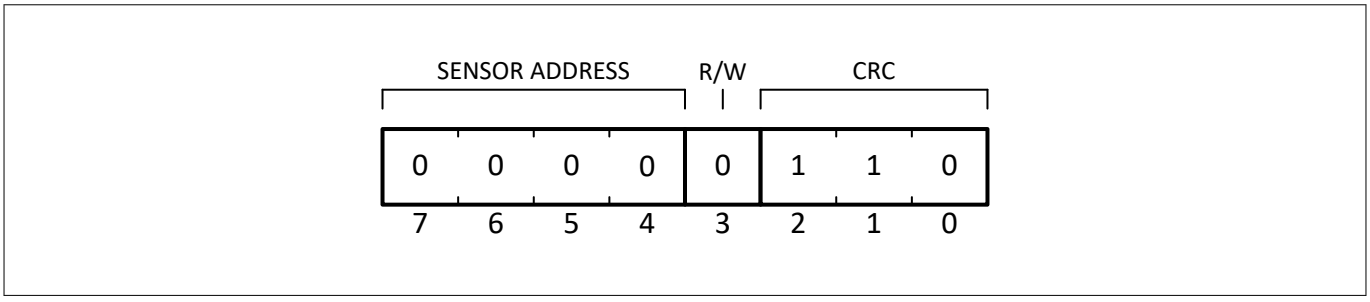


Figure 31 BROADCAST GET command frame in DCDI

The broadcast command is issued by the master MCU and is accepted by all sensors connected to the same bus (independently of their addresses). It is used during the auto-addressing procedure either to request the slave node to indicate its current address or to trigger random address generation for each slave node. The slave node responds by setting voltage level at AOUT which is proportional to the digital value according to the formula:

$$V\left(AOUT_{\mathrm{i}}\right)=\frac{V_{\mathrm{DD5}}}{2}+(ADDRESS_{\mathrm{i}}-8)\times 0.218 \quad for \quad 1\le ADDRESS_{\mathrm{i}}\le 15; \quad for \quad 5 \quad V \quad product \quad variants$$
$$V\left(AOUT_{\mathrm{i}}\right)=\frac{V_{\mathrm{DD3}}}{2}+\left(ADDRESS_{\mathrm{i}}-8\right)\times 0.134 \quad for \quad 1\le ADDRESS_{\mathrm{i}}\le 15; \quad for \quad 3.3 \quad V \quad product \quad variants \tag{1}$$

The analog value at AOUT pin remains at its level until the next frame of any type (the value persists until a new frame is initiated). Additionally, it's important to note that the slave does not provide a reply to broadcast commands with a SAFETY byte, distinguishing its behavior from other command responses.

3.6.3 Protocol timings

This section describes the timing specifications of the DCDI communication.

The DCDI supports multiple communication speeds. The value is programmable in the EEPROM.

The communication implements a timeout condition. The sensor will detect a broken frame in case of:

- Program error in the MCU software;
- Loss of synchronization between MCU and sensor;
- EMC interference;
- Collision on the bus.

The sensor expects each frame byte been received within timeout window equal to 10.5 * T_{BIT}. If there is no data received until timeout, sensor resets its state to idle and waits for a new frame.

The DCDI is based on an 8 bit standard UART protocol, including one start and one stop bit. Therefore each sub-frame of 8 bits information requires in total 10 bits. The total length of the command frames is:

- T_{CMD} = 10 * T_{BIT};
- T_{ADDR} = 10 * T_{BIT};
- T_{DATA} = 10 * 2 * T_{BIT};
- T_{SAFETY} = 10 * T_{BIT}.

Where T_{BIT} is the nominal time required to transmit a bit.

In case of a Broadcast frame T_{FRAME} = T_{CMD} + $T_{DAC_SETTLING}$, where $T_{DAC_SETTLING}$ is the settling time of the internal DAC, which is used for the address value conversion to voltage during the auto-addressing procedure. In case of Read and Write frames T_{FRAME} = T_{CMD} + T_{ADDR} + T_{DATA} + 2 * T_{SAFETY} ≈ 60 * T_{BIT} . Therefore, Read/Write commands require 60 bits in total to be communicated.

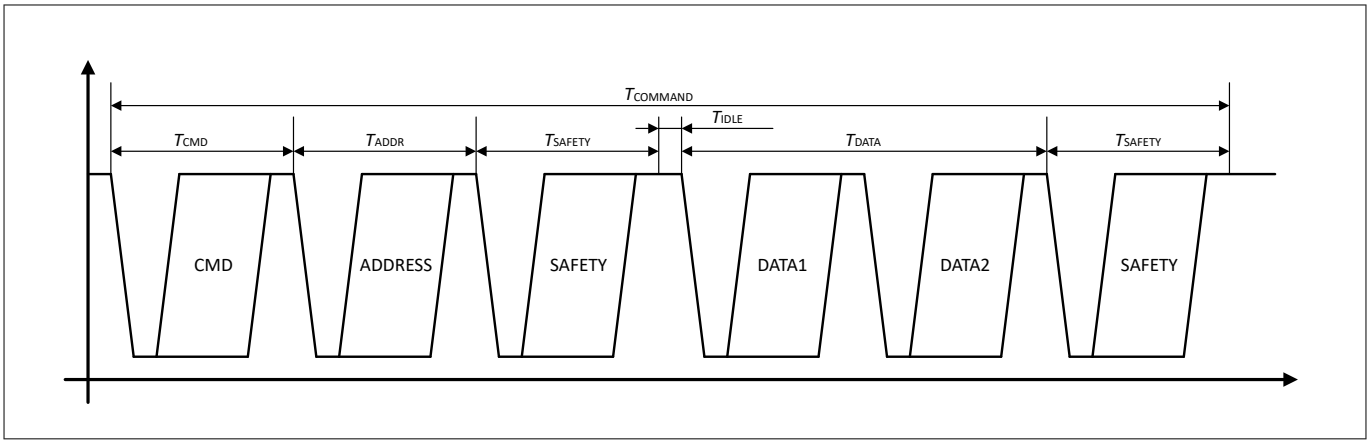


Figure 32 Example of Read Command timing in DCDI

The sensor is able to handle invalid command frames, that will be marked as invalid in case of:

- Command frame that addresses a different sensor;
- Program error in the MCU software;
- Distorted communication (EMC interference);
- Loss of synchronization between MCU and sensor causing wrong interpretation of received data;
- Collision on the bus.

A sensor receiving an invalid command frame will switch into a silent mode for a fixed time. During this time the sensor will neither send any data to the bus nor receive any master request frames. This time is named "mask-out time" and it is calculated as $T_{MASK} = T_{ADDR} + T_{DATA} + 2 * T_{SAFETY} + T_{IDLE} \approx 50 * T_{BIT}$. Corrupted data in the ADDRESS or SAFETY sub-frames are ignored by the sensor.

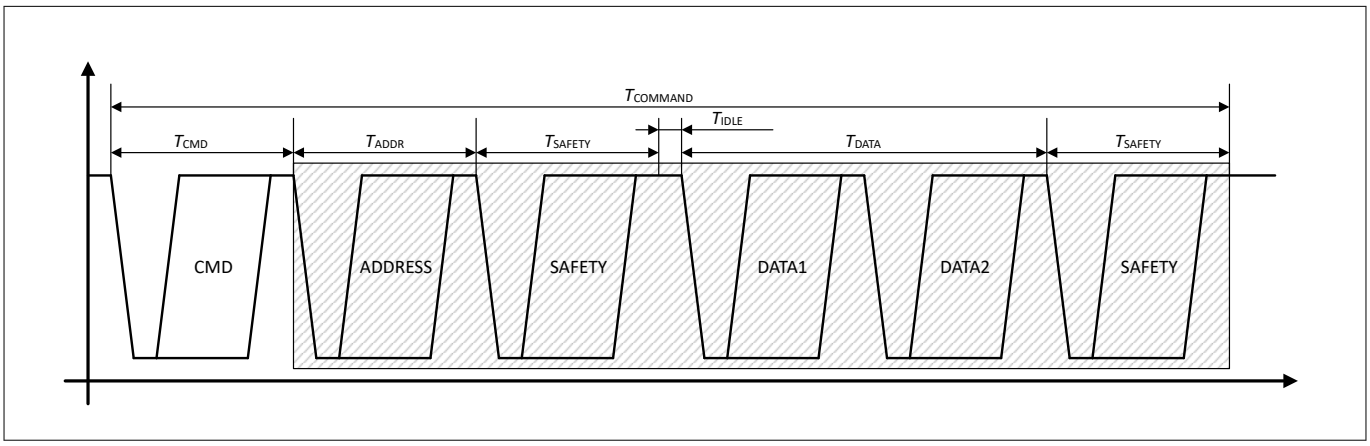


Figure 33 Example of mask-out time for a Read frame in DCDI

3.6.3.1 DCDI timing parameters

Table 3 DCDI timing parameters

Parameter	Symbol	Min	Typ	Max	Unit	Notes
Nominal bit time	T_{BIT}	17.36	–	416.66	µs	Linked to DCDI communication speed.
Idle time	T_{IDLE}	–	–	1.1 X T_{BIT}	µs	Idle time after incoming DCDI frame is received.
DCDI communication speed	–	2400	–	57600	Bit/s	Default value is 57600 Bit/s.
DAC settling time	$T_{DAC_SETTLING}$	–	–	100	µs	Settling time after random voltage generation at AOUT. With 220pF capacitor on AOUT pin.
ISM settling time	$T_{ISM_SETTLING}$	–	–	10	ms	Time needed after ISM reactivation for AOUT voltage to settle.

3.7 High level commands

This section describes the high level commands available in DCDI. Write commands require the user to unlock the DCDI interface using the UNLOCK command and stop the ISM with the DISABLE ISM command in order to avoid access conflicts to the internal register banks, unless otherwise specified.

In case a not supported command is received by the sensor, it will be ignored.

For example, in case of write access to the address 0x82 (get diagnostic) sensor does not respond to the master node.

Table 4 DCDI high level commands overview

Command name	CMD type	ADDR	DATA1 LSBs	DATA2 MSBs	Description
SET ADDRESS	Write	0x81	ADDR[7:0]	0x00	Sets sensors' address to ADDR[3:0]. Does not require DCDI unlock or ISM disable.
GET DIAGNOSTIC	Read	0x82	0x00	0x00	Read sensor, OCD status, diagnostic mode status and temperature. Refer to Chapter 3.7.3
UNLOCK	Write	0xA5	0xAB	0x49	Unlocks write access to the sensor internal registers banks.
DISABLE ISM	Write	0x22	0x00	0x80	Disable the ISM to prevent register access conflicts.
ENABLE ISM	Write	0x22	0x00	0x00	Enable the ISM to restore normal operation; reactivation and return to normal firmware execution can take up to $T_{ISM_SETTLING}$. Please refer to the timing specifications in Chapter 3.6.
READ	Read	REG_ADDR[7:0]	0x00	0x00	Read data from REG_ADDR register.
WRITE	Write	REG_ADDR[7:0]	DATA[7:0]	DATA[15:8]	Write DATA[15:0] REG_ADDR register.
RESET	Write	0x01	0x01	0x00	Hardware reset activation. The sensor does not answer with a safety byte. The reset procedure completes after the power-on delay time t_{POR} specified in the product datasheet [1].
EEPROM ON	Write	0x25	0x00	0x00	Activate EEPROM.
EEPROM UNLOCK	Write	0x4D	0x80	0x00	Enable writing to data buffer registers.
EEPROM ECC OFF	Write	0x4C	0xFF	0x00	Switches off EEPROM ECC protection.
EEPROM ECC ON	Write	0x4C	0x00	0x00	Switches on EEPROM ECC protection.

(table continues...)

Table 4 (continued) DCDI high level commands overview

Command name	CMD type	ADDR	DATA1 LSBs	DATA2 MSBs	Description
EEPROM STATUS GET	Read	0x45	0x00	0x00	Get EEPROM status.
EEPROM PROG ONES ADAPTIVE (i)	Write	0x44	(i<<4) + 9_{dec}	0x00	"i" is the EEPROM page number. The command programs the "ones" currently present in the data buffer registers into the page "i" of the EEPROM (adaptive programming). Wait for $t_{EEPPROG}$ after the command execution.
EEPROM PROG ONES NON ADAPTIVE (i)	Write	0x44	(i<<4) + 13_{dec}	0x00	"i" is the EEPROM page number. The command programs the "ones" currently present in the data buffer registers into the page "i" of the EEPROM (non-adaptive programming). Wait for $t_{EEPPROG}$ after the command execution.
EEPROM PROG DATA (i)	Write	0x44	(i<<4) + 8_{dec}	0x00	"i" is the EEPROM page number. The command programs the data currently present in the data buffer registers into the page "i" of the EEPROM. Wait for $t_{EEPPROG}$ after the command execution.
EEPROM REFRESH PAGE (i)	Write	0x44	(i<<4) + 1_{dec}	0x00	"i" is the EEPROM page number. The command loads the page "i" into 4 data buffer registers. Needed during programming procedure. Wait for $t_{EEPWAIT}$ after the command execution.

3.7.1 UNLOCK command

The unlock command has to be executed in order to obtain write privilege to the protected part of the sensor internal register banks. The command performs write of the security key to the internal register banks of the device. If unlock command is not executed, write access to the internal registers have no effect. Unlock status is cleared by writing any data to the unlock register which is different from correct security key.

The unlock command enables write access only if the EEPROM lock-bit is programmed to '0' and only to:

- the shadow registers and EEPROM fields corresponding to the first 4 lines of the EEPROM:

- eep_shadow_0_reg
- eep_shadow_1_reg

- eep_shadow_2_reg
- eep_shadow_3_reg

- The registers needed to program the EEPROM:

- eep_data_0_reg
- eep_data_1_reg
- eep_data_2_reg
- eep_data_3_reg
- eep_ctrl_0_reg
- eep_test_0_reg

3.7.2 SET ADDRESS command

The SET ADDRESS command is issued by the master MCU in order to change current address of the sensor attached to the serial bus. It does not change address value stored in the EEPROM. It is not needed to unlock device and disable the ISM in order to execute this command. Description of the addressing principle is described in Chapter 3.8.

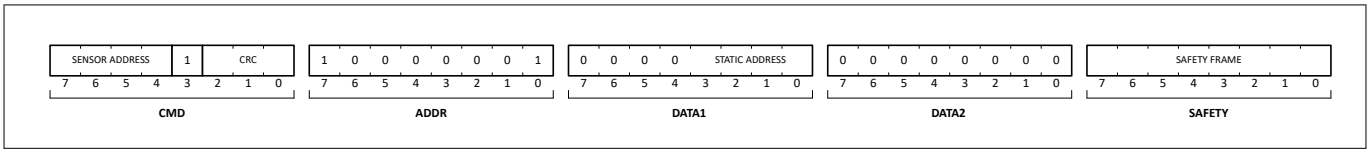


Figure 34 SET ADDRESS command in DCDI

3.7.3 GET DIAGNOSTIC command

The read diagnostic command requests the sensor to send its internally measured temperature, OCD status and diagnostic mode status to the master. In order to provide additional safety measure next status bits are included in the response frame:

- OCD status (1 = OCD triggered, 0 = OCD not triggered)
- Diag: Activation status of the diagnostic mode.

The 14 bit temperature can be translated to °C using the following formula:

$$T[°C] = \frac{[DATA[13:0]]}{32} + 65 \quad (2)$$

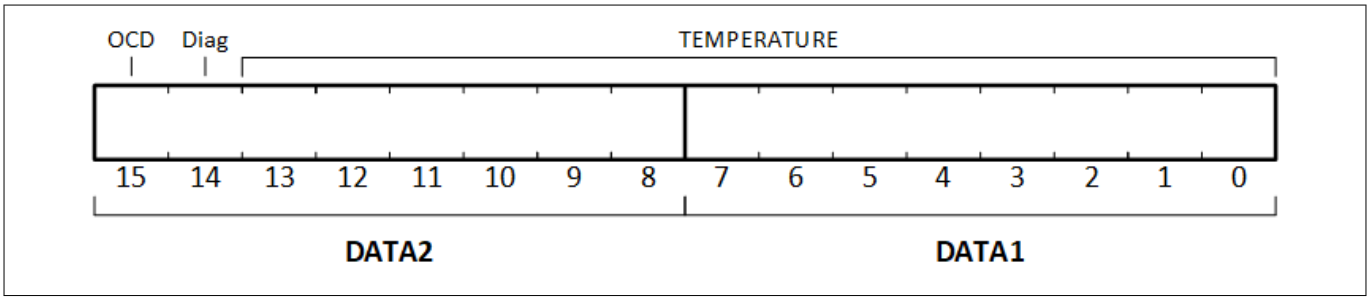


Figure 35

3.8 Address spaces

The DCDI command byte consists of a 4-bit field dedicated to slave node addressing. The available address space is partitioned into two areas: dynamic and static. Addresses with the dynamic/static flag set to 1 are classified as static, while those with the dynamic/static flag set to 0 are considered dynamic. Initially, all devices are pre-programmed with the same dynamic address, typically set to 1hex.

The goal of the auto-addressing procedure is to assign a unique static address to each sensor on the DCDI bus. After completion of the auto-addressing process, sensor addresses are transitioned to the static range, ranging from 8hex to Fhex. It is important to note that address 0hex is reserved for "broadcast" type commands, which are part of the auto-addressing process.

After the auto-addressing procedure, the resolved static addresses can be programmed into the EEPROM, in order to skip the auto-addressing procedure after each startup.

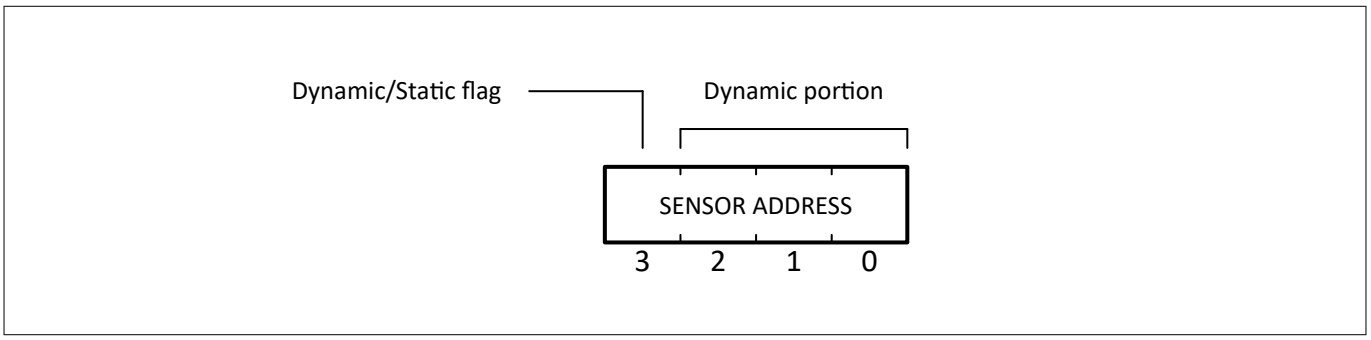


Figure 36 Sensor address in DCDI

3.9 Auto-addressing

To ensure collision-free communication on a one-wire serial bus, each sensor connected to the bus must have a unique address. Since all devices come pre-programmed with the same dynamic address, a collision-free address resolution process is essential to facilitate in-system sensor configuration. This process involves the sequential execution of the BROADCAST SET command and the observation of analog voltages at the corresponding analog nodes (AOUT pins) using the MCU's internal A/D converter. Additionally, the system-integrator has the flexibility to modify sensor configuration during the EEPROM programming procedure, as explained in the Chapter 4 section, by altering the content of the system-integrator configuration registers if necessary.

Upon the completion of the auto-addressing procedure, all sensors on the DCDI bus have a static address. However, it's important to note that the address change is not permanent. To make the address change permanent, the user is required to perform EEPROM programming. The flowchart of the auto-addressing process is described in the figure below.

***Note**: After a SET ADDRESS command, the sensor will immediately consider the new address for CRC calculations.*

***Note**: When the BROADCAST SET command is received by the sensor, the status bit in the safety byte of the DCDI frame is set to 1, indicating an error status. The user can ignore the error indication. The status bit is set back to no error indication after a hardware reset.*

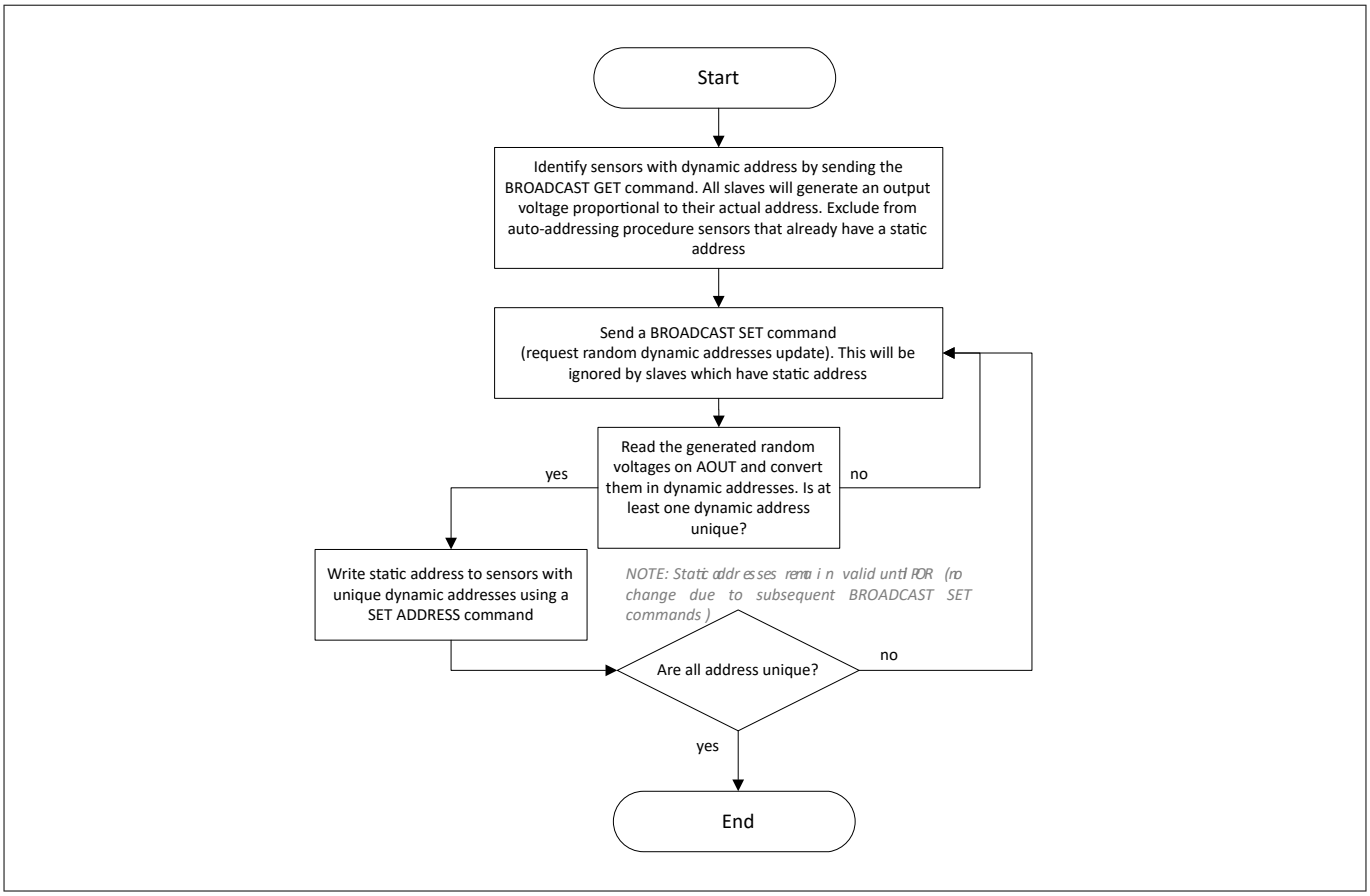


Figure 37 DCDI auto-addressing procedure algorithm

3.10 CRC

There are two CRC types involved in the data consistency protection in DCDI protocol. The CRC-3-GSM is used to protect data within CMD frame and CRC-5-USB is used for the remaining data protection.

3.10.1 CRC3 calculation

The CRC3 calculation is performed according to the CRC-3-GSM standard. It is calculated using the 4 bits sensor address and the R/W bit of CMD byte. The generator polynomial is specified as x^3+x+1; the seed is specified as 0x5. Examples of CRC-3 calculation are given in the following paragraphs, included in the command byte bits [2:0]. Please refer to the code example below for detailed implementation.

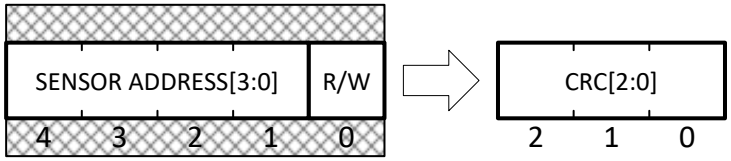


Figure 38 CRC3 coverage in DCDI

Table 5 Example of CRC-3 computation implementation

```
#include <iostream>
#include <cstdint> // For uint8_t

// Function to calculate CRC-3
uint8_t calculateCRC3(uint8_t data, uint8_t seed, uint8_t polynomial) {
    uint8_t crc = seed; // Initialize CRC with the seed value

    // Process each of the 5 bits in the data (4-bit address + 1-bit R/W)
    for (int i = 4; i >= 0; --i) { // Start from the MSB of the 5-bit data
        // Extract the current data bit (0 or 1)
        uint8_t dataBit = (data >> i) & 0x1;

        // XOR the data bit into the MSB of the CRC
        crc ^= (dataBit << 2); // XOR into bit 2 of the CRC (MSB of 3-bit CRC)

        // Shift the CRC left by 1 bit
        if (crc & 0x4) { // If the MSB (bit 2) is set, apply the polynomial
            crc = (crc << 1) ^ polynomial;
        } else {
            crc <<= 1;
        }

        // Keep only the lower 3 bits of the CRC (mask off extra bits)
        crc &= 0x7;
    }

    return crc; // Return the final CRC result
}

int main() {
    // Define constants
    const uint8_t polynomial = 0xB; // Polynomial x³ + x + 1
    const uint8_t seed = 0x5;       // Seed value

    // Test Case 1: Address 0x8, R/W = 0
    uint8_t address = 0x8; // 4-bit address
    uint8_t data = (address << 1) | 0x0; // Append R/W = 0 to form 5-bit data
    uint8_t crc = calculateCRC3(data, seed, polynomial);
    std::cout << "Test Case 1: Address 0x8, R/W = 0 -> CRC3: 0x"
              << std::hex << +crc << std::endl;

    // Test Case 2: Address 0x9, R/W = 0
    address = 0x9; // 4-bit address
    data = (address << 1) | 0x0; // Append R/W = 0 to form 5-bit data
    crc = calculateCRC3(data, seed, polynomial);
    std::cout << "Test Case 2: Address 0x9, R/W = 0 -> CRC3: 0x"
              << std::hex << +crc << std::endl;
```

(table continues...)

Table 5 Example of CRC-3 computation implementation

```
    // Test Case 3: Address 0xA, R/W = 0
    address = 0xA; // 4-bit address
    data = (address << 1) | 0x0; // Append R/W = 0 to form 5-bit data
    crc = calculateCRC3(data, seed, polynomial);
    std::cout << "Test Case 3: Address 0xA, R/W = 0 -> CRC3: 0x"
              << std::hex << +crc << std::endl;

    return 0;
}
```

3.10.2 CRC5 calculation

The calculation includes the data fields, the rolling counter and the status bit. In order to prevent masquerading errors CRC5 calculation also involves sensor address. Since most of the microcontrollers perform 8 bits calculations, data length used for CRC calculation is 32 bits. The CRC-5 calculation is stored in the safety byte bits [4:0].

The calculation is performed according to the CRC-5-USB standard. Refin and refout have to be false and xorout must be 0x00. The most significant byte has to be handled first. The generator polynomial is specified as 0x25 (x^5+x^2+1) and shifted by 3 bits for alignment. The seed is defined based on sensor address as shown in the following figures.

In case of a read frame DATA[15:0] corresponds to the value of the sensor's internal register and in case of a write frame DATA[15:0] corresponds to the value received from the Master. Please refer to the figures and example below for detailed implementation.

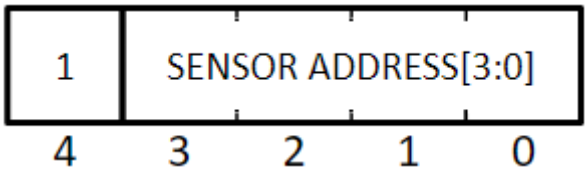


Figure 39 Initial value (seed) for CRC5 calculation in DCDI

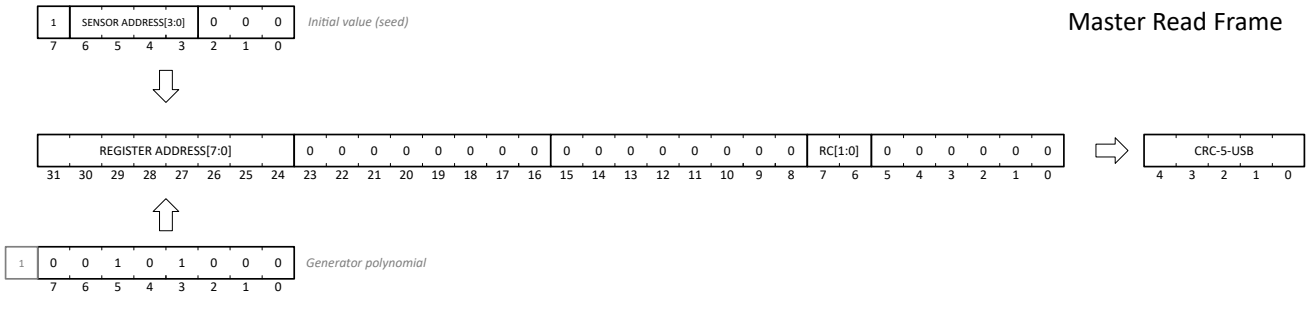


Figure 40 CRC5 calculation for Master Read frame in DCDI

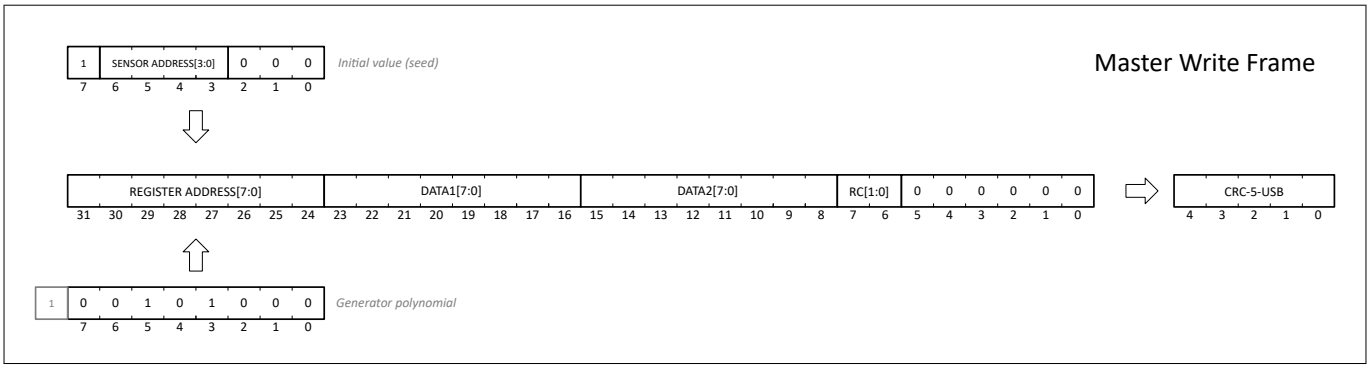


Figure 41 CRC5 calculation for Master Write frame in DCDI

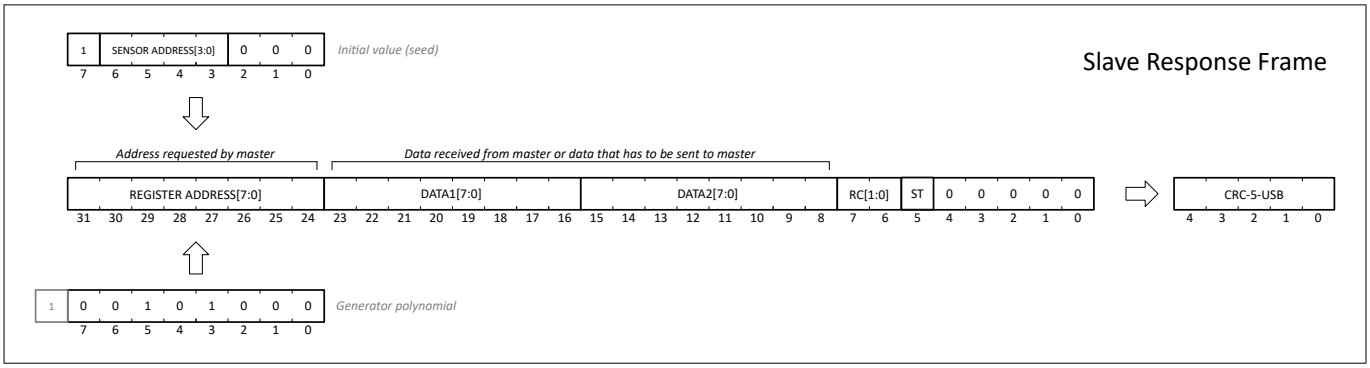


Figure 42 CRC5 calculation for Slave Response frame in DCDI

Table 6 Example of CRC-5 computation implementation

```
#include <iostream>
#include <cstdint> // For uint8_t, uint16_t, uint64_t

// Function to calculate CRC-5
uint8_t calculateCRC5(uint8_t regAddr, uint16_t data, uint8_t counter, uint8_t status, uint8_t
sensorAddr) {
    // Define constants
    const uint8_t polynomial = 0x25; // Polynomial x^5 + x^2 + 1

    // Step 1: Create seed from sensor address
    // Add leading 1 to 4-bit sensor address to make it 5 bits, then shift left by 3 bits
    uint8_t seed5bit = ((sensorAddr & 0x0F) | 0x10); // Convert to 5-bit with leading 1
    uint8_t seed = seed5bit << 3; // Shift left by 3 bits for alignment

    // Step 2: XOR register address with seed
    uint8_t modifiedRegAddr = regAddr ^ seed;

    // Step 3: Pack all data into 64-bit dividend for polynomial division
    // Data layout (37 bits total):
    // [36:29] = Modified register address (8 bits)
    // [28:13] = Data field (16 bits)
    // [12:11] = Rolling counter (2 bits)
    // [10]    = Status bit (1 bit)
    // [9:0]   = Zeros (space for CRC calculation)
    uint64_t dividend = 0;
    dividend |= (static_cast<uint64_t>(modifiedRegAddr)) << 29; // Place at bits 36:29
    dividend |= (static_cast<uint64_t>(data)) << 13;         // Place at bits 28:13
    dividend |= (static_cast<uint64_t>(counter)) << 11;      // Place at bits 12:11
    dividend |= (static_cast<uint64_t>(status)) << 10;       // Place at bit 10

    // Step 4: Perform polynomial long division
    // Process bits from position 36 down to position 5 (leaving 5 bits for remainder)
    for (int i = 31; i >= 0; --i) {
        // Check if the bit at position (i + 5) is set
        if (dividend & (1ULL << (i + 5))) {
            // XOR with polynomial at position i
            dividend ^= (static_cast<uint64_t>(polynomial) << i);
        }
    }

    // Step 5: Extract and return the 5-bit remainder
    return static_cast<uint8_t>(dividend & 0x1F); // Mask to get lower 5 bits
}

int main() {
    // Define constants
    const uint8_t polynomial = 0x25; // Polynomial x^5 + x^2 + 1
```

(table continues...)

Table 6 Example of CRC-5 computation implementation

```
// Test Case 1
uint8_t sensorAddr = 0x0A;  // 4-bit sensor address
uint8_t regAddr = 0x25;     // 8-bit register address
uint16_t data = 0x0080;     // 16-bit data
uint8_t counter = 0x0;      // 2-bit rolling counter
uint8_t status = 0x0;       // 1-bit status

uint8_t crc = calculateCRC5(regAddr, data, counter, status, sensorAddr);

std::cout << "Test Case 1:" << std::endl;
std::cout << "Sensor Address: 0x" << std::hex << +sensorAddr << std::endl;
std::cout << "Register Address: 0x" << std::hex << +regAddr << std::endl;
std::cout << "Data: 0x" << std::hex << +data << std::endl;
std::cout << "Counter: 0x" << std::hex << +counter << std::endl;
std::cout << "Status: 0x" << std::hex << +status << std::endl;
std::cout << "CRC-5 Result: 0x" << std::hex << +crc << std::endl;

// Test Case 2
sensorAddr = 0x0A;  // 4-bit sensor address
regAddr = 0x1B;     // 8-bit register address
data = 0x240A;      // 16-bit data
counter = 0x0;      // 2-bit rolling counter
status = 0x0;       // 1-bit status

crc = calculateCRC5(regAddr, data, counter, status, sensorAddr);

std::cout << std::endl;
std::cout << "Test Case 2:" << std::endl;
std::cout << "Sensor Address: 0x" << std::hex << +sensorAddr << std::endl;
std::cout << "Register Address: 0x" << std::hex << +regAddr << std::endl;
std::cout << "Data: 0x" << std::hex << +data << std::endl;
std::cout << "Counter: 0x" << std::hex << +counter << std::endl;
std::cout << "Status: 0x" << std::hex << +status << std::endl;
std::cout << "CRC-5 Result: 0x" << std::hex << +crc << std::endl;

// Test Case 3
sensorAddr = 0x0A;  // 4-bit sensor address
regAddr = 0x1B;     // 8-bit register address
data = 0x180C;      // 16-bit data
counter = 0x0;      // 2-bit rolling counter
status = 0x0;       // 1-bit status

crc = calculateCRC5(regAddr, data, counter, status, sensorAddr);

std::cout << std::endl;
std::cout << "Test Case 3:" << std::endl;
std::cout << "Sensor Address: 0x" << std::hex << +sensorAddr << std::endl;
std::cout << "Register Address: 0x" << std::hex << +regAddr << std::endl;
```

(table continues...)

Table 6 Example of CRC-5 computation implementation

```
    std::cout << "Data: 0x" << std::hex << +data << std::endl;
    std::cout << "Counter: 0x" << std::hex << +counter << std::endl;
    std::cout << "Status: 0x" << std::hex << +status << std::endl;
    std::cout << "CRC-5 Result: 0x" << std::hex << +crc << std::endl;

    return 0;
}
```

3.10.3 Calculation for Master Read frame

Table 7 CRC-3 and CRC-5 calculation in Master Read frame, example 1

	MSB									LSB
Master Read frame	**Stop**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Start**
read command, DCDI address 8 (includes CRC-3)	1	1	0	0	0	0	1	1	1	0
register address 82hex	1	1	0	0	0	0	0	1	0	0
Tx safety byte (includes CRC-5)	1	0	0	0	0	0	0	1	0	0

Table 8 CRC-3 and CRC-5 calculation in Master Read frame, example 2

	MSB									LSB
Master Read frame	**Stop**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Start**
read command, DCDI address 9 (includes CRC-3)	1	1	0	0	1	0	0	0	1	0
register address 82hex	1	1	0	0	0	0	0	1	0	0
Tx safety byte (includes CRC-5)	1	0	0	0	0	0	0	0	0	0

Table 9 CRC-3 and CRC-5 calculation in Master Read frame, example 3

	MSB									LSB
Master Read frame	**Stop**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Start**
read command, DCDI address 10 (includes CRC-3)	1	1	0	1	0	0	0	0	0	0
register address 82hex	1	1	0	0	0	0	0	1	0	0
Tx safety byte (includes CRC-5)	1	0	0	0	0	0	1	1	0	0

3.10.4 Calculation for Master Write frame and Slave Response frame

Table 10 Example of CRC-3 and CRC-5 calculation for write frame and slave response frame

	MSB									LSB
Master Write frame (Power Down ISM)	**Stop**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Start**
write command, DCDI address 10 (includes CRC-3)	1	1	0	1	0	1	0	1	1	0
register address 22hex	1	0	0	1	0	0	0	1	0	0
data [7:0]	1	0	0	0	0	0	0	0	0	0
data [15:8]	1	1	0	0	0	0	0	0	0	0
Tx safety byte (includes CRC-5)	1	0	0	0	0	0	1	1	1	0
Slave Response frame										
Rx safety byte (including CRC-5)	1	0	1	0	0	0	0	0	0	0

4 EEPROM

4.1 EEPROM content

***Note**: It is important to note that certain bits marked as reserved cannot be overwritten with a different value. These bits must be left at the value originally written by Infineon. Therefore, before making any changes, the system-integrator must first read the EEPROM lines and only modify the needed bits, leaving the reserved bits unchanged.*

This chapter provides an overview of the programmable content present in the current sensor. The sensor's nonvolatile memory (EEPROM) is organized into an 8-page, 4-lines-per-page configuration, with each line being 16-bits long.

The storage space is divided into two distinct areas:

- The users area, which stores user-configurable contents
- IFX reserved area, which stores IFX factory-trimming contents.

As shown in the figure below, EEPROM line 0 to line 3 store users accessible content. EEPROM line 4 to line 28 are IFX reserved area, and line 29 to line 31 store chip ID. When the chip is activated, the respective EEPROM content is downloaded to respective register banks.

Address Line	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	UART address settings								Reserved							
1	OCD threshold settings								OCD deglitch and hysteresis settings							
2	ZCD threshold settings								ZCD hold time settings							
3	DCDI activation setting								EEPROM CRC							
4 to 28	Reserved								Reserved							
29	Reserved								Chip ID (10bits)							
30 to 31	Chip ID								Chip ID							

Figure 43 EEPROM Overview

The EEPROM is programmed based on a per-page basis. To program the EEPROM, desired data is written to data buffer registers, then the desired page is set, and a programming command is sent to transfer data from the data buffer registers into the selected EEPROM page. Also, it is read in a per-page basis. To read the EEPROM, the desired page is set, then the read command is sent to transfer data from EEPROM into the data buffer registers from which the content can be read.

When reading EEPROM registers, it is essential to unlock the DCDI interface using the UNLOCK command and disable the ISM using the DISABLE ISM command. Reprogramming the EEPROM content necessitates updating the CRC field of the EEPROM based on the new content to be programmed. The system-integrator is responsible for updating the CRC value by reading the entire EEPROM content prior to any modification, calculating the new CRC based on the new content (including the Chip Tracking ID) and programming the new EEPROM content containing the correct CRC value. An incorrect CRC value triggers an error detection by an internal monitoring mechanism of the device. For a comprehensive understanding of the system-integrator-accessible content and the CRC calculation procedure, detailed explanations can be found in the following paragraphs.

4.1.1 Configuration registers

Table 11 EEPROM register description for line 0

Bit num	Field Name	Description
<15:13>	eep_uart_speed[2:0]	DCDI data rate: 0 - 2.4[kbit/s], 1 - 4.8[kbit/s], 2 - 9.6[kbit/s], 3 - 19.2[kbit/s], 4 - 38.4[kbit/s], 5 - 57.6[kbit/s]. Default is 0x5.
<12:9>	eep_uart_addr[3:0]	DCDI bus address. If eep_uart_addr[3] = 1 - address is static otherwise - dynamic. Default is 0x1.
<8:0>	Reserved	–

Table 12 EEPROM register description for line 1

Bit num	Field Name	Description
<15:8>	eep_ocd_thrsh[7:0]	OCD threshold configuration. Code at the input of the corresponding DAC. Default is highest threshold, otherwise the following formula applies: $$eep_ocd_thrsh[7:0] = I_{THR}(A) \times 1.755 \quad (3)$$
<7:6>	eep_ocd_hyst[1:0]	OCD threshold hysteresis configuration: - 00: 0% - 01: 6.25% - 10: 12.5% - 11: 25% The default value is reported in the product datasheet [1].
<5:3>	eep_ocd_deglitch[2:0]	OCD deglitching configuration. 0 is default with no additional delay. $$t_{DEGLITCH} = 300ns \times eep_ocd_deglitch[2:0] \quad (4)$$
<2:0>	Reserved	

Table 13 EEPROM register description for line 2

Bit num	Field Name	Description
<15:8>	eep_zcd_thrsh[7:0]	ZCD threshold programming to set the I_{THR_ZCD} by: $$eep_zcd_thrsh\left[7:0\right] = 127 - \frac{86 \times I_{THR_ZCD}(A)}{10} \quad (5)$$ The default value is reported in the product datasheet [1].

(table continues...)

Table 13 (continued) EEPROM register description for line 2

Bit num	Field Name	Description
<7:0>	eep_zcd_hold[7:0]	ZCD hold time programming to set the ZCD indication minimum time (t_{ZCD_High}) by: $$t_{ZCD_High} = (4 + (4 \times eep_ZCD_hold_time)) \times t_{ZCD_HOLD_RES} \quad (6)$$; whereas, $t_{ZCD_HOLD_RES}$(typ.) = 100ns and the default value of eep_zcd_hold[7:0] is 0x6.

Table 14 EEPROM register description for line 3

Bit num	Field Name	Description
<15>	Reserved	
<14>	eep_ocd_thr_dis[0]	0: Default value. DCDI function always needs to be reactivated after sensor power-on. For activation sequence see Figure 18. This setting enables the possibility of setting the threshold with voltage on OCD_THR pin 1: DCDI function is always activated at sensor power-on. This setting disables the possibility of setting the threshold with voltage on OCD_THR pin. In this case the over current threshold will always be the one set in the EEPROM (EEPROM register line 1, eep_ocd_thrsh[7:0]), even if the DCDI function is not used.
<13:8>	Reserved	
<7:0>	eep_config_crc[7:0]	EEPROM level CRC. Default value depends on initial device configuration.

4.1.2 Other registers

Table 15 EEPROM register description for line 29

Bit num	Field Name	Description
<15:10>	Reserved	
<9:0>	eep_ifx_id1[9:0]	Tracking ID1

Table 16 EEPROM register description for line 30

Bit num	Field Name	Description
<15:0>	eep_ifx_id2[15:0]	Tracking ID2

Table 17 EEPROM register description for line 31

Bit num	Field Name	Description
<15:0>	eep_ifx_id3[15:0]	Tracking ID3

4.2 EEPROM programming instructions

This chapter gives instructions about how to modify the EPPROM content accessible to the system-integrator. The tables below will guide the system-integrator through a complete programming sequence by following the listed commands line by line.

***Note**: Be careful when calculating EEPROM CRC. If a wrong EEPROM CRC is written, followed by POR, the device will not load the EEPROM at startup and DCDI interface will be not responsive.*

Do not power off the device during the programming procedure. In case a POR happens after partially writing the EEPROM content, the EEPROM CRC check at startup will fail and DCDI communication will be not responsive.

Table 18 EEPROM programming parameters

Parameter	Symbol	Min	Typ	Max	Unit	Notes
EEPROM programming time	$t_{EEPPROG}$	32	–	–	ms	Time to wait during effective EEPROM programming.
EEPROM wait time	$t_{EEPWAIT}$	5	–	–	ms	Wait time after EEPROM REFRESH command.
Number of programming cycles	N_{PROG}	–	–	100	–	Maximum number of programming cycles.
Programming temperature	T_{PROG}	10	–	85	°C	Temperature limits during programming.

4.2.1 EEPROM line read

Table 19 EEPROM line read instructions

Step	Command name	Command type	ADDR	DATA1 (LSBs)	DATA2 (MSBs)	Description
1	UNLOCK	Write	0xA5	0xAB	0x49	Refer to Chapter 3.7.
2	DISABLE ISM	Write	0x22	0x00	0x80	Refer to Chapter 3.7.
3	EEPROM ON	Write	0x25	0x00	0x00	Refer to Chapter 3.7.
4	EEPROM REFRESH PAGE (i)	Write	0x44	(i<<4) + 1_{dec}	0x00	Refer to Chapter 3.7.
5	Wait for $t_{EEPWAIT}$ after the command execution.					
6	After loading the EEPROM data page "i" into the 4 data buffer registers, read the content of the 4 registers one by one (following 4 commands).					

(table continues...)

Table 19 (continued) EEPROM line read instructions

7	READ	Read	0x40	0x00	0x00	Read data from first data buffer register (0x40)
8	READ	Read	0x41	0x00	0x00	Read data from first data buffer register (0x41)
9	READ	Read	0x42	0x00	0x00	Read data from first data buffer register (0x42)
10	READ	Read	0x43	0x00	0x00	Read data from first data buffer register (0x43)
11	Each EEPROM page contains 4 EEPROM lines, hence a total of 4 registers (16 bits each) has been read. Repeat starting from step 4 for all the EEPROM pages to be read.					
12	ENABLE ISM	Write	0x22	0x00	0x00	Refer to Chapter 3.7.
End of EEPROM line read sequence.						

4.2.2 Entire EEPROM programming

Table 20 EEPROM programming instructions

Step	Command name	Command type	ADDR	DATA1 (LSBs)	DATA2 (MSBs)	Description
1	UNLOCK	Write	0xA5	0xAB	0x49	Refer to Chapter 3.7.
2	DISABLE ISM	Write	0x22	0x00	0x80	Refer to Chapter 3.7.
3	EEPROM ON	Write	0x25	0x00	0x00	Refer to Chapter 3.7.
4	EEPROM REFRESH PAGE (i)	Write	0x44	(i<<4) + 1_{dec}	0x00	Refer to Chapter 3.7.
5	Wait for $t_{EEPWAIT}$ after the command execution.					
6	After loading the EEPROM data page "i" into the 4 data buffer registers, read the content of the 4 registers one by one (following 4 commands).					

(table continues...)

Table 20 (continued) EEPROM programming instructions

7	READ	Read	0x40	0x00	0x00	Read data from first data buffer register (0x40).
8	READ	Read	0x41	0x00	0x00	Read data from first data buffer register (0x41).
9	READ	Read	0x42	0x00	0x00	Read data from first data buffer register (0x42).
10	READ	Read	0x43	0x00	0x00	Read data from first data buffer register (0x43).
11	Repeat starting from step 4 for i = 0 ... 7. A total of 32 registers (16 bits each) has been read, which corresponds to the entire EEPROM content. Store the EEPROM content registers.					
12	Calculate EEPROM CRC in order to verify correctness of the stored EEPROM content - refer to Chapter 4.3.					
13	Modify the stored EEPROM content based on desired changes.					
14	Calculate new EEPROM CRC and update the corresponding bits in the stored EEPROM content at EEPROM line 3 - refer to Chapter 4.3.					
15	Write 0xFFFF into the 4 data buffer registers (following 4 commands).					
16	WRITE	Write	0x40	0xFF	0xFF	Read data from first data buffer register (0x40).
17	WRITE	Write	0x41	0xFF	0xFF	Read data from first data buffer register (0x41).
18	WRITE	Write	0x42	0xFF	0xFF	Read data from first data buffer register (0x42).
19	WRITE	Write	0x43	0xFF	0xFF	Read data from first data buffer register (0x43).
20	EEPROM ECC OFF	Write	0x4C	0xFF	0x00	Refer to Chapter 3.7.

(table continues...)

Table 20 (continued) EEPROM programming instructions

21	EEPROM PROG ONES ADAPTIVE	Write	0x44	9_{dec}	0x00	Refer to Chapter 3.7.
22	Wait for $t_{EEPPROG}$ after the command execution.					
23	EEPROM STATUS GET	Read	0x45	0x00	0x00	Refer to Chapter 3.7.
24	Check and continue only if read content is 0x81.					
25	Write the 4 registers from line [0] to [3] of the stored EEPROM content, which correspond to EEPROM page "0", into the 4 data buffer registers (following 4 commands).					
26	WRITE	Write	0x40	EEPROM line [0]		Read data from first data buffer register (0x40).
27	WRITE	Write	0x41	EEPROM line [1]		Read data from first data buffer register (0x41).
28	WRITE	Write	0x42	EEPROM line [2]		Read data from first data buffer register (0x42).
29	WRITE	Write	0x43	EEPROM line [3]		Read data from first data buffer register (0x43).
30	EEPROM ECC ON	Write	0x4C	0x00	0x00	Refer to Chapter 3.7.
31	EEPROM PROG DATA	Write	0x44	8_{dec}	0x00	Refer to Chapter 3.7.
32	Wait for $t_{EEPPROG}$ after the command execution.					
33	EEPROM STATUS GET	Read	0x45	0x00	0x00	Refer to Chapter 3.7.
34	Check and continue only if read content is 0x81.					
35	EEPROM PROG ONES NON ADAPTIVE	Write	0x44	13_{dec}	0x00	Refer to Chapter 3.7.
36	Wait for $t_{EEPPROG}$ after the command execution.					

(table continues...)

Table 20 (continued) EEPROM programming instructions

37	EEPROM STATUS GET	Read	0x45	0x00	0x00	Refer to Chapter 3.7.
38	Check and continue only if read content is 0x81.					
39	EEPROM REFRESH PAGE	Write	0x44	1_{dec}	0x00	Refer to Chapter 3.7.
40	Wait for $t_{EEPWAIT}$ after the command execution.					
41	ENABLE ISM	Write	0x22	0x00	0x00	Refer to Chapter 3.7.
End of programming sequence.						

4.3 EEPROM CRC calculation

To detect accidental changes of the EEPROM content, the data in the EEPROM are protected with a cyclic redundancy check (CRC). The CRC calculation is based on the polynomial $x^8+x^4+x^3+x^2+1$. The seed word is defined as AA_{hex}. The CRC calculation of the EEPROM is performed byte by byte; starting from the EEPROM line 4. After reaching line 31 the calculation continues with the high byte of line 0 till the high byte of line 3 and is completed with the inversion of the CRC result to get the final CRC checksum.

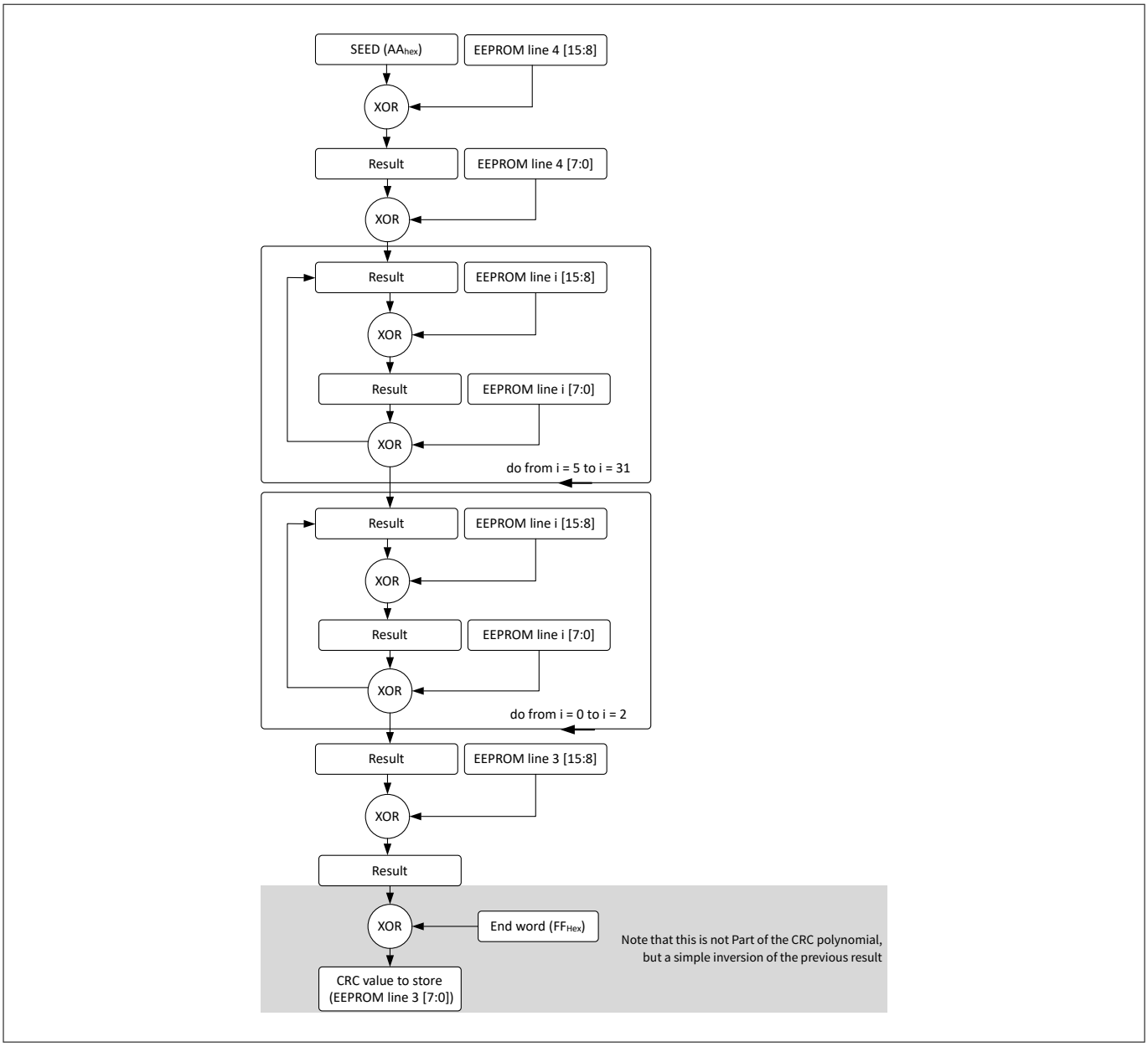


Figure 44 EEPROM CRC Calculation Flowchart

Table 21 Example of EEPROM CRC computation implementation

```
#define CRC_POLYNOMIAL 0x1D
#define CRC_SEED 0xAA

uint8_t crc8(uint8_t *data, uint8_t length)
{
    uint32_t crc;
    int16_t i, bit;

    crc = CRC_SEED;

    for (i = 0; i < length; i++)
    {
        crc ^= data[i];

        for (bit = 0; bit < 8; bit++)
        {
            if ((crc & 0x80) != 0)
            {
                crc <<= 1;
                crc ^= CRC_POLYNOMIAL;
            }
            else
            {
                crc <<= 1;
            }
        }
    }
    return ~crc = crc ^ 0xFF;
}

bool checkCRC (uint16_t* data, int len)
{
    uint8_t checkSum = data[3] & 0xFF; // CRC lower byte in EEPROM line 3
    return checkSum == crcCalc(data, len);
}

// Read data beginning in EEPROM line 4 to line 31, append line 0 to line 3
uint8_t crcCalc(uint16_t* data, int len)
{
    uint8_t crcData8[len * 2];

    for (int i = 0; i < len; i++)
    {
        crcData8[i * 2] = (data[(i + 4) % 32] >> 8) & 0xFF;    // Read upper 8 bit
        crcData8[i * 2 + 1] = (data[(i + 4) % 32]) & 0xFF;     // Read lower 8 bit
    }
```

(table continues...)

Table 21 Example of EEPROM CRC computation implementation

```
    return crc8(crcData8, len * 2 - 1); // Do not include last byte (line 3 lower byte)
}
```

5 Stray fields and crosstalk

The sensor features a differential sensing principle, as explained in Chapter 1. This ensures a high suppression to homogeneous stray fields (*BSR*).

5.1 Differential measurement principle

A differential measurement of the magnetic field caused by the current in the conductor is shown below. The two Hall probes are placed close to the current rail. Assuming that the current flows from the front side to the backside the left Hall probe detects a positive magnetic field and the right Hall probe measures a negative field.

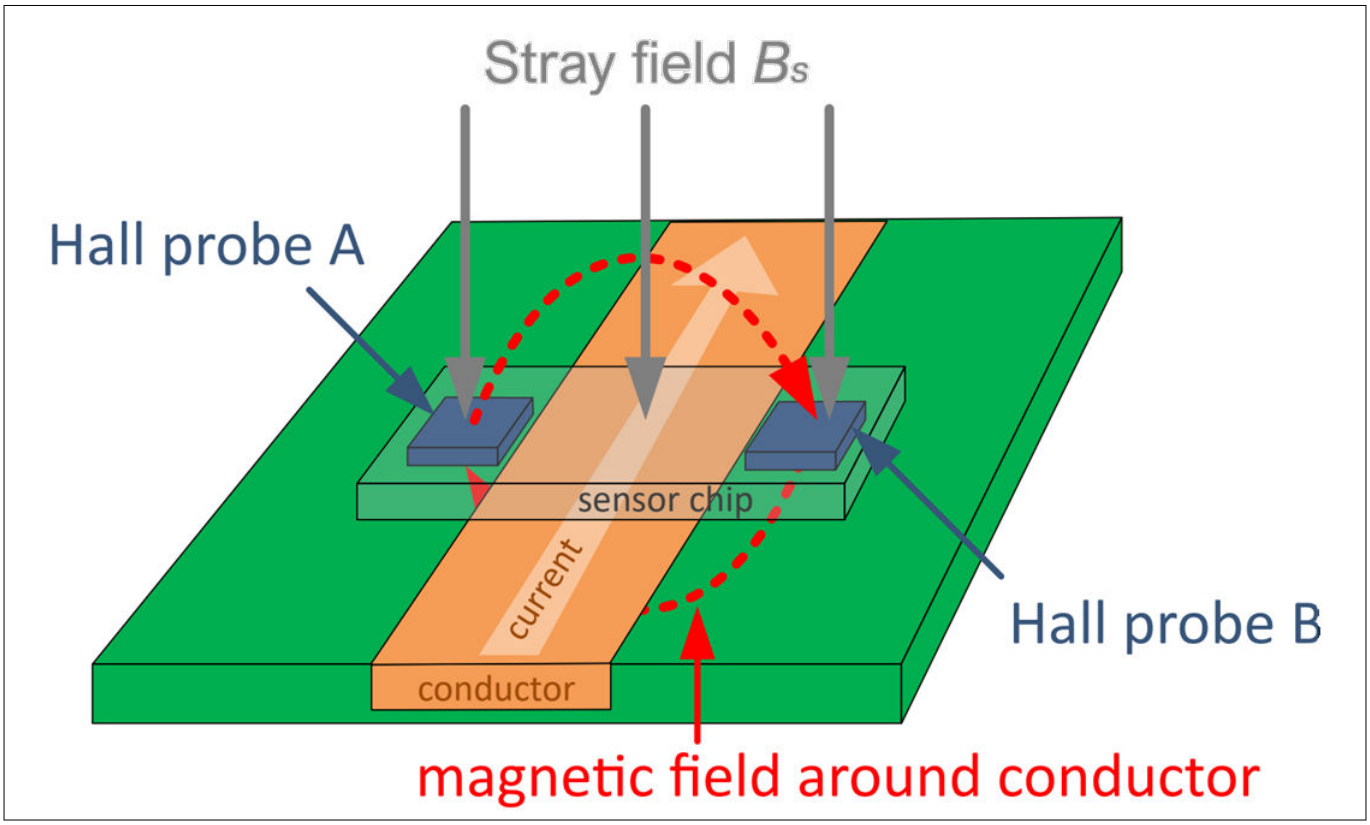


Figure 45 Differential measurement with stray field cancellation

The following formula describes the differential signal calculation obtained by subtracting the right signal (B_B + B_S) from the left signal (B_A + B_S). Here, B_A and B_B represent the magnetic flux density in the left and right Hall probes respectively and B_S is the superimposed stray field.

$$V_O = K \cdot [B_A + B_S] - K \cdot [B_B + B_S] = K \cdot (B_A - B_B) \quad (7)$$

The superimposed stray field is canceled out since it has the same polarity on both Hall probes. A perfect cancellation of the homogeneous magnetic field is the result of the differential measurement principle.

6 Glossary

Notation	Description
TF	Transfer Factor.
FS	Full Scale.
EEPROM	Electrically Erasable Programmable Read-Only Memory.
CRC	Cyclic Redundancy Check.
VDD	Supply voltage.
A/D	Analog Digital Converter Input.
ADC	Analog to Digital Converter.
ISM	Internal State Machine.
IFX	Infineon Technologies.
µC	Micro Controller.
LSB	Least Significant Bit.
MSB	Most Significant Bit.
PWM	Pulse Width Modulation.
GND	Ground.
OCD	Over Current Detection.
DCDI	Digital Control Diagnostic Interface.
EMC	Electro-Magnetic Compatibility.
UART	Universal Asynchronous Receiver-Transmitter.
CLK	Clock signal.
POR	Power-on reset, equivalent to power cycle.
MCU	Micro-Controller Unit.
ZCD	Zero Crossing Detection.

7 References

[1] Latest product datasheet: Infineon-TLE4978-5VRatON-DS_vx_xx-EN.pdf and Infineon-TLE4978-3V3RatON-DS_vx_xx-EN.pdf

[2] Latest product safety manual: Infineon-TLE4978-RatON-SM_vx_xx-EN.pdf

8 Revision History

Revision number	Date of release	Description of changes
5.00	2026-06-17	Editorial changes
4.00	2026-03-13	Integrated the 3.3 V variant, added the layout suggestions chapter and editorial changes
3.00	2025-12-09	Update for product release
2.00	2025-04-16	Reworked the DCDI and EEPROM chapters
1.00	2024-04-15	Initial release

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-06-17

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-jobid__20260616220517557_LastLeaf2

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.