# Features

## Audio

### Inputs

- 1 x USB Audio Class 2.0
- 3 x IEC 60958 (S/PDIF, AES3)
- 1 x External I2S (differential, PSAudio/DSD/SerceI2S)
- 1 x TV ARC
- 1 x Internal I2S

### Outputs

- Main - I2S / DSD
- Auxiliary - IEC 60958 (S/PDIF, AES3)

### Formats

- PCM: up to 768 kHz, up to 32 bits
- DSD: up to DSD512, base 44.1 and 48 kHz, DoP
- MQA[1]: full decoding with rendering

### Processing

- DoP decoding
- MQA[1]decoding and rendering
- Soft mute
- Resolution meter
- FIR upsampling
- Volume control

More algorithms are planned to be added.

### Clocks

Generation of frame sync and serial clock from master clock. Master clock generation, recovery from IEC 60958, or external input. Master clock output with optional divider.

### Signal detection

Signal existence detection on USB, IEC 60958, TV ARC, External I2S.

## Connectivity

### USB

USB 2.0 High Speed (480 Mbps) device with classes:

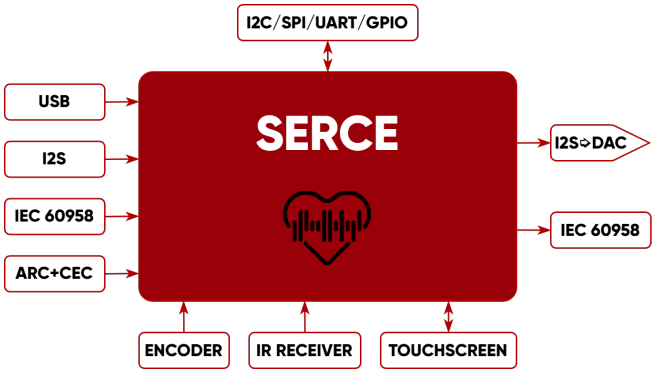
- Audio Class 2.0
- DFU (device firmware upgrade)
- HID (for MQA[1]control)

### CEC

- Power on/off
- Volume and mute control
- Name customization
- ARC configuration

### In-device peripherals

- 3 x I2C
- 1 x SPI
- 1 x UART
- 58 x GPIO (15 I/O, 1 button input, 42 special or I/O)
- 1 x rotary encoder
- 1 x IR receiver




### Display

Parallel 24-bit RGB interface for external display. 3.9" 480 x 128 pixels IPS 24-bit 60 Hz RGB bar display with capacitive touch panel supported.

## Power supply

Single 5 V / 1 A power supply required.
Power consumption (typ/max): 1.5 W / 5 W.

## Mechanics

5 cm x 6 cm board with two board-to-board connectors.

## Firmware update

Safe and secured update process over USB, DFU 1.1 compatible with custom extensions. Tools for update and version management provided.

## Drivers

Driverless operation on Windows 10/11, Linux, macOS, iOS, Android.

## Encryption

AES-256 encryption of firmware and configuration data. Custom key can be used.

## Power loss detection

SERCE can detect power loss and stop operation safely.

## Storage

### Flash

- 1'792 kB for firmware
- 4 MB for filesystem or extended firmware
- Support for external SPI flash for filesystem

### Host board configuration

Support for 64 kB EEPROM on the host board for configuration data. Configuration data contains Vendor and Product IDs, names, etc. EEPROM is required for SERCE to work.

## CPU

ARM Cortex-M7 32-bit 480 MHz.

[1]MQA support needs additional license from MQA.

# Contents

# List of figures



# List of tables

# 1. Introduction

This document provides information on SERCE High-End Digital Audio Platform, such as description, functional overview, pin assignment and definition, electrical and mechanical characteristics as well as design guides, firmware update procedure and customization options.

# 2. Description

SERCE is a high-end digital audio platform designed to be used in high-end audio devices. It has form of a small module, which has to be mounted on a host board. SERCE is designed to be used in devices such as DACs, streamers, DDCs, etc.

It provides easy to use audio subsystem, which includes input and output interfaces, inputs and clocks management, clock generation, recovery and distribution, wide range of sample rates and formats support, audio processing including format decoding, conversion, volume control, etc. and signal detection.

SERCE is designed to be used as a main digital system in a device, so it provides all necessary interfaces to control and configure all the components in a system. SERCE is provided with firmware, which implements all the features mentioned above. Then firmware can be customized to implement additional features, interface with other components in a system like display, DAC, etc. or to adjust it for specific application. There is also a possibility to customize its names and IDs for branding purposes. It also provides firmware update mechanism, which allows to update firmware in a safe and secure way.

SERCE can be easily extended with additional hardware, like additional GPIOs, audio clocks, other audio input interfaces. It features master clock input and additional I2S input, which gives a possibility to connect other audio sources or use SERCE with custom clock generators. It can be also extended with additional software, like additional audio processing algorithms, communication protocols and others. SERCE uses high-performance ARM® Cortex-M7 microcontroller, which provides enough processing power to implement additional features. In a basic configuration, SERCE firmware uses only about 10% of CPU time while core is running at 480 MHz clock.

SERCE can be also used as an audio subsystem in a device, which has other main digital system like SoC or SoM. In this case SERCE can be connected to the main digital system via USB, and then updated and controlled over it. SERCE can be also used as a USB to I2S converter in this case.

# 3. Functional overview

## 3.1 Audio

SERCE provides easy to use audio subsystem, which includes:

- 7 digital inputs and 2 digital outputs
- inputs and clocks management
- clock generation, recovery and distribution
- wide range of sample rates and formats support
- audio processing including format decoding, conversion, volume control, etc.
- signal detection

### 3.1.1 Inputs

There are 7 inputs available on SERCE:

- 1 x USB
  USB Audio Class 2.0 with driverless support on Windows, Linux, macOS, iPadOS, iOS and Android.
- 3 x IEC 60958
  To implement any combination of optical S/PDIF, coaxial S/PDIF or AES3 inputs.
- 1 x External I2S
  Differential LVDS, PS Audio compatible I2S with DSD support or SERCE I2S protocol with reclocking and control commands.
- 1 x TV ARC
  Audio Return Channel from TV, with CEC control.
- 1 x Internal I2S
  To interface SERCE with other sources on the host board.

### 3.1.2 Outputs

There are two outputs available on SERCE:

- Main - I2S / DSD
  Designed to be connected to DAC or other device with I2S input. It is also possible to implement differential LVDS I2S output compatible with PS Audio standard or SERCE I2S protocol with reclocking and control commands.
- Auxiliary - IEC 60958
  To implement optical S/PDIF, coaxial S/PDIF or AES3 output.

### 3.1.3 Formats

- PCM: up to 768 kHz, 32-bit
- DSD: up to DSD512, base 44.1 and 48 kHz, DoP decoding
- MQA[1]: full decoding with rendering

[1]MQA support needs additional license from MQA.

**Format limitations**

Following IEC 60958 standard, S/PDIF, AES3 and TV ARC inputs and Auxiliary output support only PCM up to 192 kHz, 24-bit. In this case DoP can only contain DSD64.

Due to hardware limitations, when S/PDIF, AES3 or TV ARC input is used, maximum PCM output sample rate is limited to 384 kHz.

When External or Internal I2S input is used, maximum output sample rate depends on provided master clock frequency, i.e. for 22.5792 MHz or 24.576 MHz master clock maximum output sample rate is 352.8 kHz or 384 kHz respectively, for 45.1584 MHz or 49.152 MHz master clock maximum output sample rate is 705.6 kHz or 768 kHz respectively.

### 3.1.4 External I2S

SERCE provides LVDS differential lines for I2S data and clocks inputs and optional master clock output. Additionally, there are 4 control lines to use for format selection (PCM/DSD) or to support proprietary SERCE I2S protocol providing control commands like volume, mute and power on/off.

There are many ways of implementing External I2S input, because there is no standard for External I2S connection. One of the most popular is to use HDMI connector, which is our recommendation. By default, data format is compatible with PS Audio standard. Additionally, proper master clock has to be provided by external source on `EI2S_MCKIO_N`/`EI2S_MCKIO_P` lines. Its frequency has to be 22.5792 MHz or 24.576 MHz for sample rates up to 384 kHz or 45.1584 MHz or 49.152 MHz for sample rates up to 768 kHz.

SERCE can also generate master clock internally and provide it on `EI2S_MCKIO_N`/`EI2S_MCKIO_P` lines, but in that case SERCE I2S protocol has to be used to handshake with external source.

Please refer to External I2S connection guide to get more details about HDMI connector pinout.

### 3.1.5 Processing

SERCE provides following processing algorithms:

- DoP decoding
- MQA[2]decoding and rendering
- Soft mute
- Resolution meter
- FIR upsampling
- Volume control

With firmware update, more algorithms can be added in the future. Please contact us if you need more processing algorithms. We can also provide a way to inject your own processing into SERCE.

Processing algorithms can be enabled or disabled and their parameters can be changed. They can be also configured to process data only on some output, e.g. upsampling can be done only on Main output.

### 3.1.6 Signal detection

Signal detection is implemented for all inputs except Internal I2S. It detects presence of signal on the input, but it does not detect if the signal is valid or if it is silent. Monitoring of signal parameters is only available during processing i.e. on active input.

[2]MQA support needs additional license from MQA.

### 3.1.7 Clocks

SERCE generates all necessary clocks like bit clock, frame sync, etc. internally. To do so it uses one of the following master clock sources:

- Internal clock generator: 45.1584 MHz or 49.152 MHz depending on sample rate
- Clock recovered from IEC 60958 stream (S/PDIF, AES3, TV ARC): 22.5792 MHz or 24.576 MHz depending on sample rate
- Clock provided by External I2S source on `EI2S_MCKIO_P/N` differential pair lines
- Clock provided by host board source on `MCKI` line

SERCE always uses one master clock source at the same time to serve all audio interfaces. Clock source is selected automatically depending on audio input and application configuration. It is not required to provide any clock to SERCE, but it is possible to replace internal clock with one provided to board-to-board connector pin. In that case SERCE has to be able to configure clock generator on the host board to generate clock with matching frequency. It can be done by using GPIO lines or any other interface. Implementation of this feature is up to the user.

Depending on audio input, SERCE uses different clock sources:

- USB - SERCE uses clock generated internally
- IEC 60958 and TV ARC - SERCE uses clock recovered from the stream. It is always 22.5792 MHz or 24.576 MHz depending on sample rate
- External I2S - SERCE uses clock provided by External I2S source on `EI2S_MCKIO_P/N` differential pair lines. In case of SERCE I2S protocol, it can be also generated internally by SERCE and provided to External I2S source on `EI2S_MCKIO_P/N` differential pair lines
- Internal I2S - SERCE uses internal clock generated by quartz oscillators or clock provided by host board on `MCKI` line

## 3.2 Connectivity

### 3.2.1 USB

SERCE incorporates USB device hardware and its support in firmware. It supports High Speed (480 Mbps) and Full Speed (12 Mbps) modes, but only High Speed mode is supported in the firmware, because it is required for Audio Class 2.0 operation. SERCE is driverless on Windows, Linux, macOS, iPadOS, iOS and Android.

It includes USB Audio Class 2.0 interface, firmware update interface and optional HID interface for MQA[3].

### 3.2.2 CEC

SERCE incorporates CEC hardware and library, which is used among others to control SERCE with TV remote control. It adds following features:

- Power on/off
- Volume control
- Mute
- Name of the device which is displayed on TV
- ARC (Audio Return Channel) configuration[4]

[3] MQA support needs additional license from MQA.
[4] ARC requires configuration through CEC. This functionallity is embedded in SERCE's firmware without any additional configuration.

### 3.2.3 I2C

There are four I2C buses available on SERCE:

- I2C1 is reserved for TV DDC communication, and it is not accessible in the firmware.
- I2C2 and I2C4 are available to be used to communicate with ICs on the host board. Use of them is optional, so there are no pull-up resistors on their lines. Pull-up resistors should be added on the host board if I2C2 or I2C4 will be used.
- I2C3 is used by SERCE internally and addresses 0x70 and 0x50 are reserved. It is also exposed on the board-to-board connector to communicate with EEPROM and can be shared with other ICs. It is suggested to use I2C3 for EEPROM only or for EEPROM and DAC only.
  2.2 kΩ pull-up resistors on I2C3 SDA and SCL lines are already on SERCE's board. If I2C lines will be long it is suggested to put another pair of pull-up resistors near the end of those lines, e.g. at the DAC's end.

All I2C buses are single master buses, SERCE is always a master. I2C buses number 2, 3 and 4 are 3.3V level. I2C bus number 1 is 5V level. All I2C buses works with 100 kHz clock frequency, but during startup SERCE uses 400 kHz clock frequency to communicate with EEPROM on I2C3. Please be aware of this when you design your host board especially if you use I2C3 for other ICs.

SERCE SDK provides easy to use API to communicate with I2C devices.

### 3.2.4 SPI

SERCE provides SPI master interface, which can be used to communicate with ICs on the host board. It provides serial clock, `MOSI` and `MISO` lines. CS line can be implemented in the firmware using GPIO line, there can be multiple CS lines for different ICs, and they can be configured to be active high or active low. Each CS line represents different IC, so it is possible to use different SPI modes and clock frequencies for different ICs.

SERCE SDK provides easy to use API to communicate with SPI devices.

### 3.2.5 UART

SERCE provides UART interface, which can be used to communicate with ICs on the host board or to provide e.g. RS-232 interface (required UART ↔ RS-232 converter IC on the host board) to the user. It provides TX and RX lines and can be configured to use any baud rate, odd or even parity and 1 or 2 stop bits.

SERCE SDK provides easy to use API to communicate over UART.

### 3.2.6 GPIO

SERCE provides 58 programmable lines.

- 15 of them can be used as general purpose input or output.
- 1 is dedicated to force SERCE to enter update mode during startup, but it can be used to serve button on the device as well. Please refer to 7.3 for more details.
- 42 of them are dedicated to specific functions, but they can be used as general purpose input or output as well.

Some of the lines can be used in interrupt mode, i.e. they can generate interrupt on rising or falling edge.

### 3.2.7 Rotary encoder

SERCE provides hardware support for rotary encoder and easy to use API to read rotation.

### 3.2.8 Display

SERCE provides hardware support for 24-bit parallel RGB display with up to 8-bit per color. It can support any display with resolution limitation caused by memory limitations. SERCE can provide up to 400 kB of display buffer (single buffering) or 200 kB of each display buffer (double buffering).

There is a software port of TouchGFX[5] GUI library for SERCE. This port supports following displays:

- YB-TG480128S01A-C-A0 - 3.9" 480 x 128 pixels IPS 24-bit 60 Hz RGB bar display with capacitive touch panel. Port contains both, display driver and touch panel driver.

Software port for display is available at special request.

If you want to use other display connected by RGB interface, please contact us to get more details. There are 3.9" and 5.2" bar displays with 480 x 128 pixels resolution available in the market which can be easily used with SERCE.

It is also possible to connect display to SERCE using SPI or I2C interface, but it is up to the user to communicate with it.

### 3.2.9 Remote control

SERCE provides hardware support for IR remote control receiver, firmware support for NEC format and easy to use API to read received commands.

Please contact us if you need support for other formats.

## 3.3 Firmware update

SERCE provides firmware update interface over USB.

Firmware update has the following features:

- Safety
  It is always possible to recover SERCE from any failure during firmware update.
- Security
  Firmware file is encrypted, and it is not possible to modify it or use in another device.
- Compatibility
  It uses DFU 1.1 protocol.
- Tooling
  We provide SERCE Updater GUI application[6] for Windows and macOS to update firmware in a few clicks. We also provide serce-cli console application for Windows, macOS and Linux.
- Version management
  It is possible to get information about firmware version, including pre-release versions and build number.

Please refer to Firmware update chapter for more details.

## 3.4 Host board configuration storage

External EEPROM memory is REQUIRED on the host board to store configuration data required by SERCE to work. Data stored in EEPROM is encrypted, and it describes device built with SERCE, e.g. product name, serial number, etc. It gives a possibility to easily change SERCE module from one device to another without

[5] TouchGFX is a registered trademark of STMicroelectronics International NV. Please refer to `https://www.touchgfx.com/` for details.
[6] SERCE Updater is already work in progress, please contact us to get to know status of the project. It will be available soon.

a need to reprogram device configuration data. Driver implemented in SERCE's bootloader is compatible with 24C64 EEPROM series only. There is no guarantee that other EEPROMs will work with SERCE. EEPROM has to be connected to I2C3 bus of SERCE and has address 0x50.

To get more details about configuration data stored in EEPROM, please refer to Customization section. EEPROM implementation design guidelines are described in 7.1.

## 3.5 Security

SERCE provides security features to protect your product from unauthorized access:

- Firmware readout protection
  When SERCE is in production, it is locked, so it is not possible to read firmware from it.
- Firmware update encryption
  Firmware file is encrypted, so it is not possible to modify it or use in another device.
- Custom encryption key
  To give you more confidence, you can use your own private key to encrypt firmware, so nobody else can decrypt your firmware or encrypt firmware for your device.
- Secure configuration storage
  SERCE requires EEPROM memory on the host board to store data required by SERCE to work. This includes encryption key for firmware. All configuration data is encrypted, so it is not possible to read it from EEPROM.

## 3.6 Debug

SERCE provides serial debug interface over UART. It is used to print debug messages from the firmware. SERCE SDK provides easy to use API to print debug messages from your application as well.

Additionally, SERCE has JTAG and SWD interfaces, so it is possible to debug firmware using external debugger.

## 3.7 Storage

SERCE incorporates two types of storage flash memory:

- 1'792 kB of flash memory for firmware. It includes SERCE library and user application. This memory is read only when firmware is running.
- 4 MB of additional flash memory. It can be used for file system (read/write) or to extend flash memory for firmware (read only). In the second case, it is possible to use it to store images, fonts or other data which is not modified by firmware.

If additional flash memory is used to extend flash memory for firmware, it is possible to connect additional SPI flash memory and mount it as a filesystem to read/write files.

# 4. Pin descriptions

SERCE has 2 identical **Amphenol 10144517-10180** 100-pin connectors[1], both of the bottom side of the board, at opposite edges. Connectors are marked as A and B, and their pins are marked Axx and Bxx respectively.

Figure 4.1 shows pinout of the connectors placed on the SERCE board. Table 4.2 contains detailed description of each pin, including its type and name, and table 4.1 contains legend for the table 4.2.

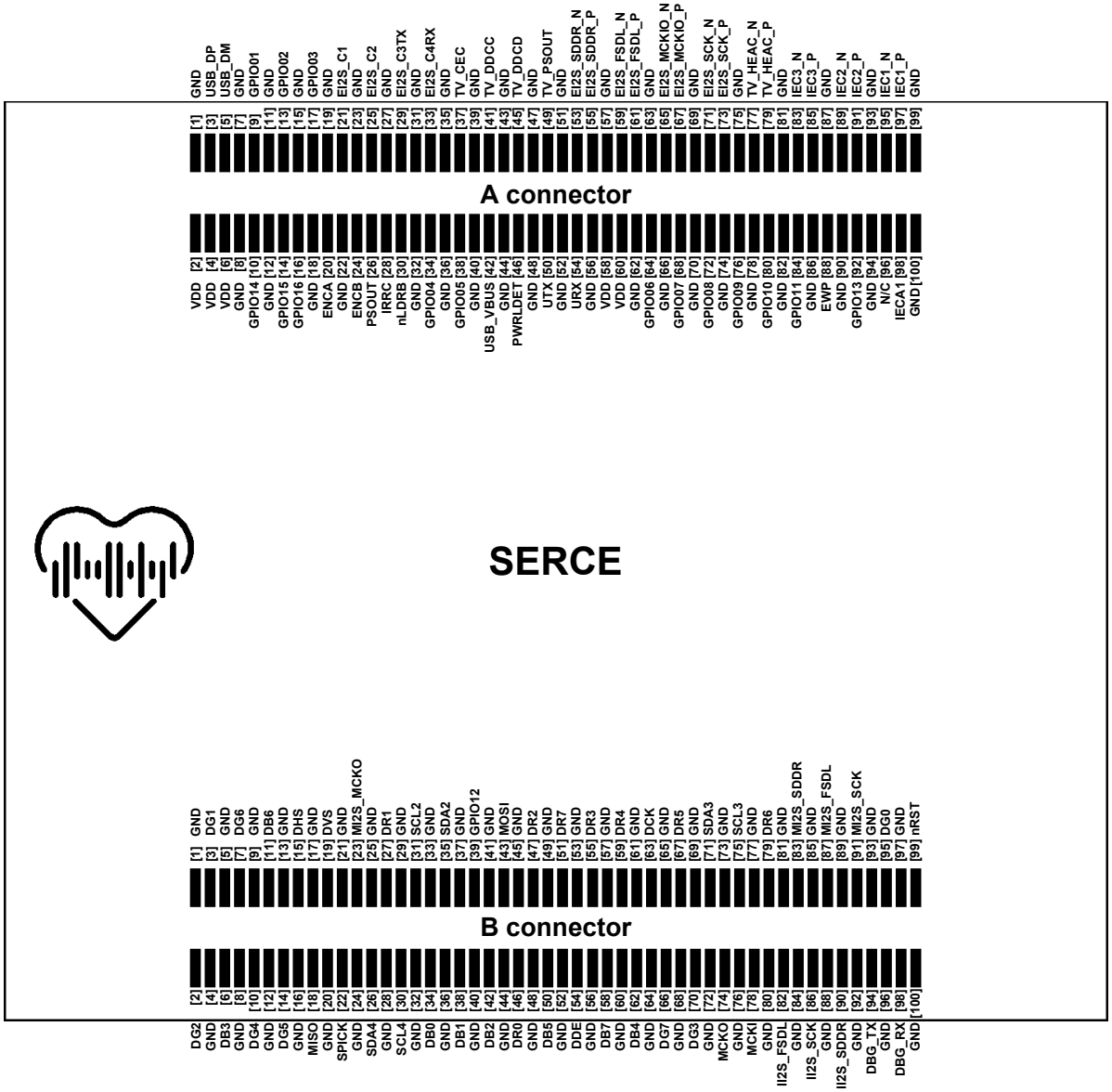



Figure 4.1: SERCE pinout (top view)
Connectors are on the bottom side of the board

[1]Check out **Amphenol 10144517-10180** datasheet for more details.

Table 4.1: Legend/abbreviations used in SERCE pinout table

| | **Abbreviation** | **Description** |
|---|---|---|
| **Name** | **prefixes - ports** | |
| | `USB_` | USB high-speed input port |
| | `EI2S_` | External I2S input port |
| | `II2S_` | Internal I2S input port |
| | `TV_` | TV input port |
| | `MI2S_` | Main (I2S) output port |
| | `IECx` | IEC 60958 x (1, 2 or 3) input port |
| | `IECAx` | Auxiliary (IEC 60958) output port |
| | `DBG_` | debug serial port |
| | **suffixes** | |
| | `_P` | differential pair positive line |
| | `_N` | differential pair negative line |
| | **some naming conventions** | |
| | `SCLx` | I2C bus x (2, 3 or 4) |
| | `SDAx` | |
| | `MISO` | SPI bus |
| | `MOSI` | |
| | `SPICK` | |
| | `ENCx` | rotary encoder input x (A or B) |
| | `Dxy` | display parallel port output xy (RGB bit) |
| **Type** | `S` | power supply |
| | `SO` | power supply output |
| | `G` | general purpose input/output |
| | `B` | button input |
| | `I` | function input |
| | `O` | function output |
| | `IO` | function input/output |
| | `A` | analog |
| | `RST` | reset input |
| | **Option for function lines** | |
| | `_l` | LVDS high-speed differential |
| | `_u` | USB high-speed differential |
| | `_d` | other high-speed differential |
| | `_c` | clock |
| | `_h` | high-speed single |
| | `_e` | ESD protected |
| | `_T` | 3.3 V level, but 5 V tolerant |
| | `_F` | 5 V level |

Table 4.2: SERCE pinout

| Pin | Name | Type | Description |
|---|---|---|---|
| A1 | GND | S | ground |
| A2 | VDD | S | power supply (+5 V) |
| A3 | USB_DP | IO_ue | USB high speed data (+) |
| A4 | VDD | S | power Supply (+5 V) |
| A5 | USB_DM | IO_ue | USB high speed data (-) |
| A6 | VDD | S | power supply (+5 V) |
| A7 | GND | S | ground |
| A8 | GND | S | ground |
| A9 | GPIO01 | G_T | general purpose I/O |
| A10 | GPIO14 | G_T | general purpose I/O |
| A11 | GND | S | ground |
| A12 | GND | S | ground |
| A13 | GPIO02 | G_T | general purpose I/O |
| A14 | GPIO15 | G_T | general purpose I/O |
| A15 | GND | S | ground |
| A16 | GPIO16 | G_T | general purpose I/O |
| A17 | GPIO03 | G_T | general purpose I/O |
| A18 | GND | S | ground |
| A19 | GND | S | ground |
| A20 | ENCA | IG_T | rotary encoder input A |
| A21 | EI2S_C1 | IO_eT | External I2S additional control line 1 |
| A22 | GND | S | ground |
| A23 | GND | S | ground |
| A24 | ENCB | IG_T | rotary encoder input B |
| A25 | EI2S_C2 | IO_eT | External I2S additional control line 2 |
| A26 | PSOUT | SO | power supply output (+3.3 V) |
| A27 | GND | S | ground |
| A28 | IRRC | IG_T | IR remote control receiver or general purpose I/O |
| A29 | EI2S_C3TX | IO_eT | External I2S additional control line 3 with SERCE protocol (TX) |
| A30 | nLDRB | B | force update mode button input (active low) |
| A31 | GND | S | ground |
| A32 | GND | S | ground |
| A33 | EI2S_C4RX | IO_eT | External I2S additional control line 4 with SERCE protocol (RX) |
| A34 | GPIO04 | G_T | general purpose I/O |
| A35 | GND | S | ground |
| A36 | GND | S | ground |
| A37 | TV_CEC | IO_e | CEC line |
| A38 | GPIO05 | G_T | general purpose I/O |
| A39 | GND | S | ground |
| A40 | GND | S | ground |

| Pin | Name | Type | Description |
|---|---|---|---|
| A41 | TV_DDCC | O_ecF | DDC clock |
| A42 | USB_VBUS | I_eF | USB high speed VBUS |
| A43 | GND | S | ground |
| A44 | GND | S | ground |
| A45 | TV_DDCD | IO_eF | DDC data |
| A46 | PWRLDET | AG | power loss detection |
| A47 | GND | S | ground |
| A48 | GND | S | ground |
| A49 | TV_PSOUT | SO | DDC power supply output (+5 V) |
| A50 | UTX | OG_T | UART transmitter or general purpose I/O |
| A51 | GND | S | ground |
| A52 | GND | S | ground |
| A53 | EI2S_SDDR_N | I_el | External I2S serial data or DSD right input (-) |
| A54 | URX | IG_T | UART receiver or general purpose I/O |
| A55 | EI2S_SDDR_P | I_el | External I2S serial data or DSD right input (+) |
| A56 | GND | S | ground |
| A57 | GND | S | ground |
| A58 | VDD | S | power supply (+5 V) |
| A59 | EI2S_FSDL_N | I_el | External I2S frame sync or DSD left input (-) |
| A60 | VDD | S | power supply (+5 V) |
| A61 | EI2S_FSDL_P | I_el | External I2S frame sync or DSD left input (+) |
| A62 | GND | S | ground |
| A63 | GND | S | ground |
| A64 | GPIO06 | G_T | general purpose I/O |
| A65 | EI2S_MCKIO_N | IO_ecl | External I2S master clock input or output (-) |
| A66 | GND | S | ground |
| A67 | EI2S_MCKIO_P | IO_ecl | External I2S master clock input or output (+) |
| A68 | GPIO07 | G_T | general purpose I/O |
| A69 | GND | S | ground |
| A70 | GND | S | ground |
| A71 | EI2S_SCK_N | I_ecl | External I2S serial clock input (-) |
| A72 | GPIO08 | G_T | general purpose I/O |
| A73 | EI2S_SCK_P | I_ecl | External I2S serial clock input (+) |
| A74 | GND | S | ground |
| A75 | GND | S | ground |
| A76 | GPIO09 | G_T | general purpose I/O |
| A77 | TV_HEAC_N | I_ed | hot plug detect/HEAC input (-) |
| A78 | GND | S | ground |
| A79 | TV_HEAC_P | I_ed | utility/HEAC input (+) |
| A80 | GPIO10 | G_T | general purpose I/O |
| A81 | GND | S | ground |

| Pin | Name | Type | Description |
|---|---|---|---|
| A82 | GND | S | ground |
| A83 | IEC3_N | I_d | IEC 60958 (S/PDIF or AES3) differential input 3 (-) |
| A84 | GPIO11 | G_T | general purpose I/O |
| A85 | IEC3_P | I_d | IEC 60958 (S/PDIF or AES3) differential input 3 (+) |
| A86 | GND | S | ground |
| A87 | GND | S | ground |
| A88[2] | EWP | O | host EEPROM write protect |
| A89 | IEC2_N | I_d | IEC 60958 (S/PDIF or AES3) differential input 2 (-) |
| A90 | GND | S | ground |
| A91 | IEC2_P | I_d | IEC 60958 (S/PDIF or AES3) differential input 2 (+) |
| A92 | GPIO13 | G_T | general purpose I/O |
| A93 | GND | S | ground |
| A94 | GND | S | ground |
| A95 | IEC1_N | I_d | IEC 60958 (S/PDIF or AES3) differential input 1 (-) |
| A96 | IECA1 | O_d | IEC 60958 (Auxiliary) output 1 |
| A97 | IEC1_P | I_d | IEC 60958 (S/PDIF or AES3) differential input 1 (+) |
| A98 | N/C | | leave unconnected |
| A99 | GND | S | ground |
| A100 | GND | S | ground |
| B1 | GND | S | ground |
| B2 | DG2 | OG_T | display parallel interface green line 2 or general purpose I/O |
| B3 | DG1 | OG_T | display parallel interface green line 1 or general purpose I/O |
| B4 | GND | S | ground |
| B5 | GND | S | ground |
| B6 | DB3 | OG_T | display parallel interface blue line 3 or general purpose I/O |
| B7 | DG6 | OG_T | display parallel interface green line 6 or general purpose I/O |
| B8 | GND | S | ground |
| B9 | GND | S | ground |
| B10 | DG4 | OG_T | display parallel interface green line 4 or general purpose I/O |
| B11 | DB6 | OG_T | display parallel interface blue line 6 or general purpose I/O |
| B12 | GND | S | ground |
| B13 | GND | S | ground |
| B14 | DG5 | OG_T | display parallel interface green line 5 or general purpose I/O |
| B15 | DHS | OG_T | display parallel interface horizontal sync line or general purpose I/O |
| B16 | GND | S | ground |
| B17 | GND | S | ground |
| B18 | MISO | IG_hT | SPI data input line or general purpose I/O |
| B19 | DVS | OG_T | display parallel interface vertical sync line or general purpose I/O |
| B20 | GND | S | ground |
| B21 | GND | S | ground |
| B22 | SPICK | OG_hcT | SPI clock output line or general purpose I/O |

| Pin | Name | Type | Description |
|---|---|---|---|
| B23 | MI2S_MCKO | O_hc | Main I2S master clock output |
| B24 | GND | S | ground |
| B25 | GND | S | ground |
| B26 | SDA4 | IOG_T | serial data of I2C4 or general purpose I/O |
| B27 | DR1 | OG_T | display parallel interface red line 1 or general purpose I/O |
| B28 | GND | S | ground |
| B29 | GND | S | ground |
| B30 | SCL4 | OG_cT | serial clock of I2C4 or general purpose I/O |
| B31 | SCL2 | OG_cT | serial clock of I2C2 or general purpose I/O |
| B32 | GND | S | ground |
| B33 | GND | S | ground |
| B34 | DB0 | OG_T | display parallel interface blue line 0 or general purpose I/O |
| B35 | SDA2 | IOG_T | serial data of I2C2 or general purpose I/O |
| B36 | GND | S | ground |
| B37 | GND | S | ground |
| B38 | DB1 | OG_T | display parallel interface blue line 1 or general purpose I/O |
| B39[3] | GPIO12 | G | general purpose I/O |
| B40 | GND | S | ground |
| B41 | GND | S | ground |
| B42 | DB2 | OG_T | display parallel interface blue line 2 or general purpose I/O |
| B43 | MOSI | OG | SPI data output line or general purpose I/O |
| B44 | GND | S | ground |
| B45 | GND | S | ground |
| B46 | DR0 | OG_T | display parallel interface red line 0 or general purpose I/O |
| B47 | DR2 | OG_T | display parallel interface red line 2 or general purpose I/O |
| B48 | GND | S | ground |
| B49 | GND | S | ground |
| B50 | DB5 | OG_T | display parallel interface blue line 5 or general purpose I/O |
| B51 | DR7 | OG_T | display parallel interface red line 7 or general purpose I/O |
| B52 | GND | S | ground |
| B53 | GND | S | ground |
| B54 | DDE | OG_T | display parallel interface data enable line or general purpose I/O |
| B55 | DR3 | OG_T | display parallel interface red line 3 or general purpose I/O |
| B56 | GND | S | ground |
| B57 | GND | S | ground |
| B58 | DB7 | OG_T | display parallel interface blue line 7 or general purpose I/O |
| B59 | DR4 | OG_T | display parallel interface red line 4 or general purpose I/O |
| B60 | GND | S | ground |
| B61 | GND | S | ground |
| B62 | DB4 | OG_T | display parallel interface blue line 4 or general purpose I/O |
| B63 | DCK | OG_cT | display parallel interface clock line or general purpose I/O |

| Pin | Name | Type | Description |
|---|---|---|---|
| B64 | GND | S | ground |
| B65 | GND | S | ground |
| B66 | DG7 | OG_T | display parallel interface green line 7 or general purpose I/O |
| B67 | DR5 | OG_T | display parallel interface red line 5 or general purpose I/O |
| B68 | GND | S | ground |
| B69 | GND | S | ground |
| B70 | DG3 | OG_T | display parallel interface green line 3 or general purpose I/O |
| B71 | SDA3 | IO | serial data of I2C3 |
| B72 | GND | S | ground |
| B73 | GND | S | ground |
| B74 | MCKO | O_hc | master clock output |
| B75 | SCL3 | O_c | serial clock of I2C3 |
| B76 | GND | S | ground |
| B77 | GND | S | ground |
| B78 | MCKI | I_hc | master clock input |
| B79 | DR6 | OG_T | display parallel interface red line 6 or general purpose I/O |
| B80 | GND | S | ground |
| B81 | GND | S | ground |
| B82 | II2S_FSDL | I_h | Internal I2S frame sync or DSD left input |
| B83 | MI2S_SDDR | O_h | Main I2S serial data or DSD right output |
| B84 | GND | S | ground |
| B85 | GND | S | ground |
| B86 | II2S_SCK | I_hc | Internal I2S serial clock input |
| B87 | MI2S_FSDL | O_h | Main I2S frame sync or DSD left output |
| B88 | GND | S | ground |
| B89 | GND | S | ground |
| B90 | II2S_SDDR | I_h | Internal I2S serial data or DSD right input |
| B91 | MI2S_SCK | O_hc | Main I2S serial clock |
| B92 | GND | S | ground |
| B93 | GND | S | ground |
| B94 | DBG_TX | O | debug console serial port output (TX) |
| B95 | DG0 | OG_T | display parallel interface green line 0 or general purpose I/O |
| B96 | GND | S | ground |
| B97 | GND | S | ground |
| B98 | DBG_RX | I | debug console serial port input (RX) |
| B99 | nRST | RST | system reset (active low) |
| B100 | GND | S | ground |

[2]Pin **A88** was formerly called **GPIO 12**. It has special function and can't be used as GPIO so it was renamed to **EWP**.
[3]Pin **B39** was formerly called **SPI NSS**. It has not special function so it was renamed to **GPIO12**.

# 5. Electrical characteristics

## 5.1 Parameters conditions

All voltages are referenced to the ground (GND). Typical data are based on $T_A = 25°C$ and $V_{DD} = +5\,V$.

## 5.2 Absolute maximum ratings

Table 5.1: Voltage characteristics

| Symbol | Ratings | | Min | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | External main power supply voltage | | -0.3 | 5.5 | V |
| $V_{IN}$ | 3.3 V and LVDS pins (pin type without _T suffix) | | -0.3 | 3.9 | V |
| | 5 V tolerant pins (pin type with _T suffix) | | -0.3 | 7.2[a] | V |
| | 5 V pins (pin type with _F suffix) | | -0.3 | 6.0 | V |
| $V_{USBD}$ | USB data line | | -0.5 | 6.0 | V |
| $V_{IEC}$ | IEC 60958 data line | | -0.3 | 3.6 | V |
| $V_{PP\_TV}$ | TV interface ESD | Contact discharge[b] | | 8 | kV |
| | | Air discharge[b] | | 15 | |
| $V_{PP\_EI2S}$ | EI2S interface ESD | Human-body model (HBM) | | 4.5 | kV |
| $V_{PP\_USB}$ | USB interface ESD | Human-body model (HBM) | | 8 | kV |
| | | Contact Discharge[b] | | 8 | |
| | | Air discharge[b] | | 15 | |
| $V_{PP\_IEC}$ | IEC 60958 interface ESD | Human-body model (HBM)[c] | | 2.5 | kV |
| | | Charged-device model (CDM)[d] | | 1.5 | |

[a] Absolute maximum rating for 5 V tolerant pins is 7.2 V only when pull-up and pull-down resistors are disabled. Otherwise, the maximum voltage is 4.0 V.
[b] EN/IEC 61000-4-2
[c] ANSI/ESDA/JEDEC JS-001
[d] JEDEC specification JESD22-C101

Table 5.2: Current characteristics

| Symbol | Ratings | Max | Unit |
|---|---|---|---|
| $I_{IN}$ | Power supply current on all VDD pins | 1.5 | A |
| | Power supply current on each VDD pin[e] | 0.5 | A |
| $I_{PIN}$ | Output current of single-ended data pins | 20 | mA |
| $I_{PSOUT}$ | Output current of PSOUT pin | 0.5 | A |

[e] Each VDD pin current limitation is due to connector specification. It is important to connect power supply to all VDD pins.

Table 5.3: Temperature characteristics

| Symbol | Ratings | Min | Max | Unit |
|---|---|---|---|---|
| $T_A$ | Ambient working temperature | 0 | 70 | $°C$ |
| $T_{STG}$ | Recommended storage temperature[f] | 18 | 28 | $°C$ |

[f] Relative humidity between 30% and 60%.

## 5.3 Operating conditions

Table 5.4: General operating conditions

| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | Power supply voltage | 4.75 | 5.0 | 5.25 | V |
| $I_{DD}$ | Power supply current | 180 + $I_{PSOUT}$ | 250 + $I_{PSOUT}$ | 650 + $I_{PSOUT}$ | mA |
| $I_{PSOUT}$ | Output current PSOUT | 0 | | 200 | mA |
| $I_{TV\_PSOUT}$ | DDC power out current | 0 | | 50 | mA |
| $V_{IN}$ | 3.3 V pins (type without _T suffix) | 0 | 3.3 | 3.6 | V |
| | 5 V tolerant pins (type with _T suffix) | 0 | 3.3 | 5.5 | |
| | 5 V pins (type with _F suffix) | 0 | 5.0 | 5.5 | |
| $f_{MCU}$ | MCU clock frequency | | 480 | | MHz |
| $V_{REF}$ | Reference voltage for Power Loss Detection | 1.180 | 1.216 | 1.225 | V |

# 6. Mechanical characteristics

## 6.1 Dimensions

SERCE is a module with dimensions 50 mm x 60 mm. Minimum distance between SERCE PCB and host PCB is at least 5 mm depending on the connector used on the host PCB. Above SERCE should be always at least 15 mm of free space for proper cooling.

## 6.2 Connectors

There are 2 `Amphenol 10144517-10180` board-to-board connectors[1] on the bottom side of the PCB. Additionally, SERCE has 14-pin 0.1" raster standard ARM debugger socket mounted on the top side of the PCB.

## 6.3 Mounting holes

SERCE has 1 mounting hole in the middle of the PCB. It is 3.2 mm in diameter, and it is designed to be used with M3 screw.

## 6.4 PCB

Figure 6.1 shows dimensions of the PCB and placement of the connectors and mounting hole. All dimensions are in millimeters. Dimensions related to the connectors are measured from the center of each connector.

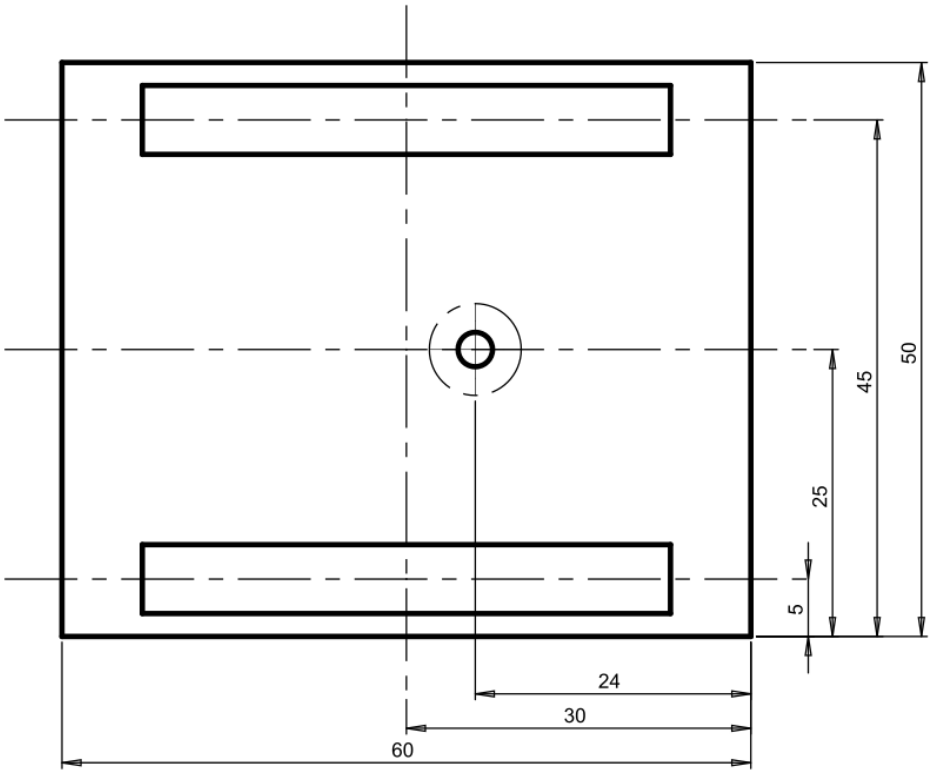



Figure 6.1: PCB dimensions and important mounting points (top view)

[1]Check out `Amphenol 10144517-10180` datasheet for more details.

## 6.5 Host board mounting

SERCE is designed to be mounted on a host board. Figure 6.2 shows how host board should be designed to mount SERCE.

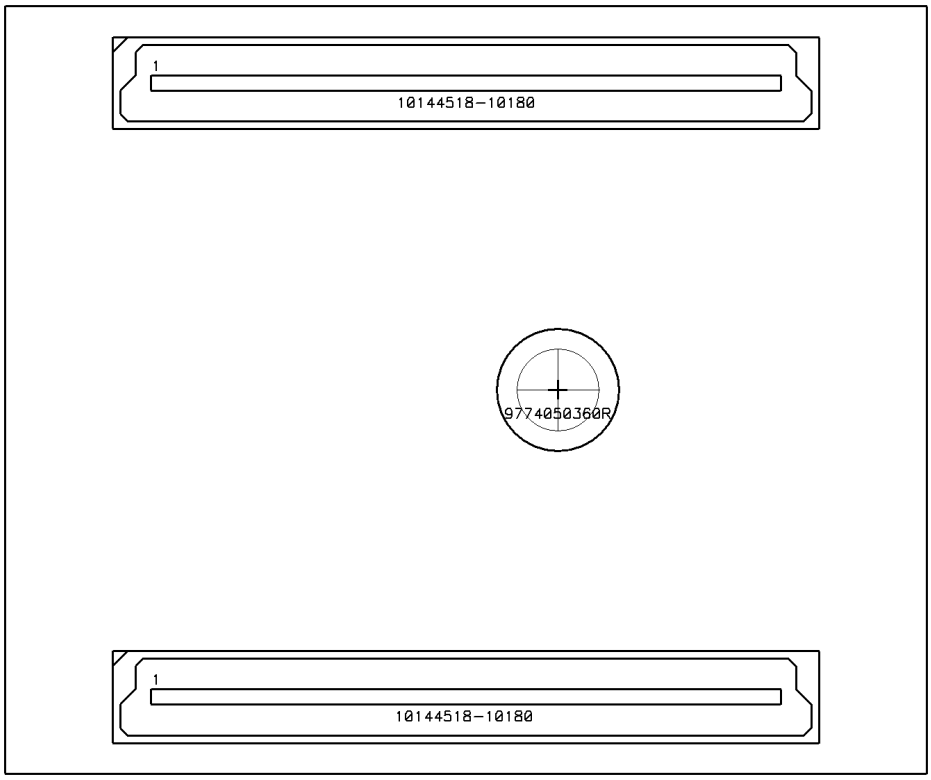



Figure 6.2: SERCE mounting on a host board (top view)

Host PCB board-to-board connectors should be `Amphenol 10144518-10` series or compatible `Amphenol 61083-10` series. Typically, it can be `Amphenol 10144518-10180` connector, which gives 5 mm distance between boards (PCB to PCB). Other connectors from `Amphenol 10144518-10` series can be used to increase distance between boards. Considering that, please remember, that SERCE PCB is always provided with `Amphenol 10144517-10180`.

On the host PCB it is recommended to use mounted spacer (for SERCE mounting screw) with its length appropriate to the distance between SERCE and host boards. If the distance is 5 mm then `Würth Elektronik 9774050360R` 5 mm surface mounted steel spacer with internal M3 thread can be used for M3 mounting screw. Using mount screw is recommended to prevent disconnection of the connectors due to shock.

# 7. Implementation guides

## 7.1 EEPROM

Schematic below shows how to connect 24C64 EEPROM to the SERCE. EEPROM has to be connected to I2C3 bus of SERCE and has the address 0x50 (0xA0 when R/W bit is included).

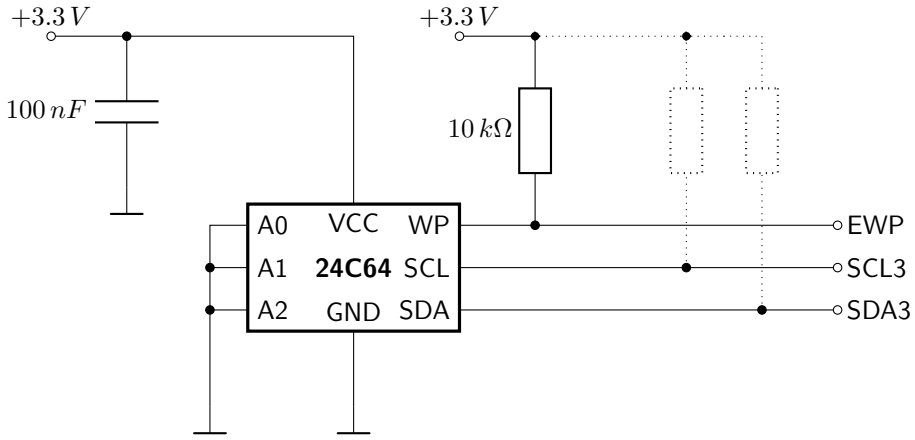



Figure 7.1: Schematic how to connect 24C64 EEPROM to SERCE

I2C3 is a shared bus, please refer to I2C section for more details. If I2C3 lines are long, additional pull-up resistors may be needed at the end of them.

It is required to connect **WP** (write protect) pin of EEPROM to **EWP** pin of SERCE. It is hard-coded in the driver to use **EWP** pin to enable write data to the EEPROM.

Table 7.1: SERCE line ↔ EEPROM line pinout

| SERCE pin | EEPROM pin | Description |
|---|---|---|
| B71 (SDA3) | 5 | SDA |
| B75 (SCL3) | 6 | SCL |
| A88 (EWP) | 7 | WP |

If you want, you can use **PSOUT** pin to power EEPROM.

## 7.2 Power loss detection

SERCE supports power loss detection by monitoring voltage level on **PWRLDET** pin. Host board should provide voltage higher than $V_{REF}$ on **PWRLDET** pin to indicate that power is present. SERCE will detect power loss when voltage on **PWRLDET** pin will be lower than $V_{REF}$. Power loss detection is not enabled by default, it has to be enabled in the firmware by calling proper function from SERCE SDK.

Figure 7.2 shows how to connect **PWRLDET** pin on the host board.

$V_{PS\_det}$ represents detectable voltage. It should be connected to the main power supply input of the device. Depending on your design you may need to adjust schematic to your needs. It is recommended to get $V_{PS\_det}$ before power supply filter to detect power loss before power supply filter will discharge.

When you use transformer to power your device, you can use $V_{PS\_det}$ from the secondary side of the transformer and add half-wave rectifier ($D_1$) with smoothing capacitor $C_{det}$ as shown on figure 7.3.

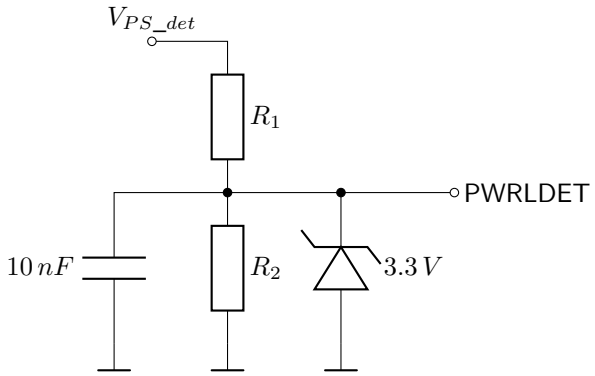



Figure 7.2: Schematic how to connect **PWRLDET** pin to host board

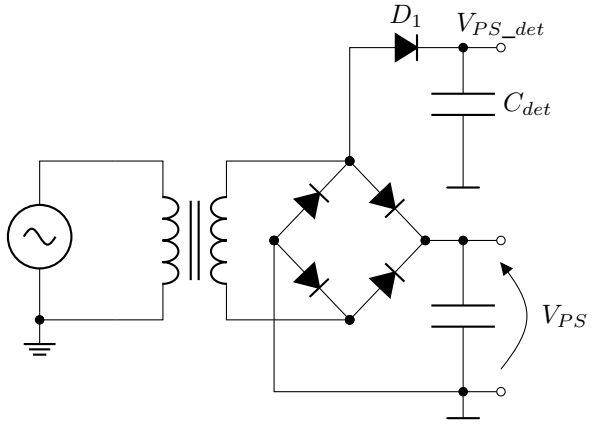



Figure 7.3: Schematic how to connect power detection circuit when transformer is used

In case of external DC power supply you should use $V_{PS\_det}$ directly from the power supply input and optionally add diode ($D_r$) to separate it from the rest of the circuit as shown on figure 7.4. You can also avoid using $D_r$ diode when your $C_f$ capacitor is small enough to discharge quickly when power is lost. But then, $V_{PS}$ has to be provided to other power supply circuits like linear regulators or DC-DC converters and their output capacitors have to discharge slower than $C_f$ capacitor, so $V_{PS\_det}$ goes down before other power supply circuits are discharged.

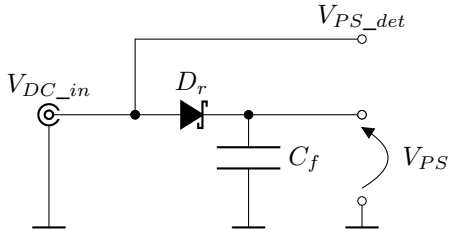



Figure 7.4: Schematic how to connect power detection circuit when DC power supply is used

Resistors $R_1$ and $R_2$ have to be selected to provide voltage higher than $V_{REF}$ on **PWRLDET** pin when $V_{PS}$ is correct. You can use following formulas to calculate values of $R_1$ and $R_2$:

$$\frac{R_2}{R_1 + R_2} = \frac{V_{REF}}{V_{PS\_min}}$$

$$R_1 || R_2 \leq 20\,k\Omega$$

If you use transformer to power your device, $V_{PS\_min}$ should be calculated including the fact that $V_{PS\_det}$ is rectified, so peak voltage is $\sqrt{2}$ times higher than RMS voltage. Voltage drop on $D_1$ and voltage drop on bridge and transformer when it is loaded with maximum current should be also taken into account. Additionaly, half-wave rectifier with smoothing capacitor detecting supply voltage has to discharge much quicker than $V_{PS}$. Otherwise power loss detection will not be operating properly. Therefore, values of $C_{det}$, $R_1$ and $R_2$ must be picked in the way their RC time constant is not too high. Time constant below 0.1 s is usually sufficient, but some cases might require lower values:

$$(R_1 + R_2) \cdot C_{det} \leq 0.1\,s$$

$C_{det} = 10\,\mu F$ is advisable as a first value and $(R_1 + R_2) = 10\,k\Omega$ accordingly. Then calculating values of $R_1$ and $R_2$ is straightforward:

$$R_2 = \frac{V_{REF}}{V_{PS\_min}} \cdot (R_1 + R_2) = \frac{V_{REF}}{V_{PS\_min}} \cdot 10\,k\Omega$$

$$R_1 = 10\,k\Omega - R_2$$

However, there is ripple on the output of the half-wave rectifier with smoothing capacitor and the voltage drop of $D_1$. They must be taken into account, so they will not cause false power loss detection. For shown above values of $C_{det}$ and $R_1 + R_2$ they can be approximated by following formula:

$$V_{PS\_min} = (U_{PS\_minAC} - 1\,V) \cdot \sqrt{2} - 3\,V$$

$U_{PS\_minAC}$ is minimum RMS voltage predicted on the input of the full-bridge rectifier, when device still should be operating. There is $1\,V$ subtracted from this value for margin purposes and those $3\,V$ are representing $D_1$ voltage drop and ripple voltage. Final formula for $R_2$ value looks like this:

$$R_2 = \frac{V_{REF}}{(U_{PS\_minAC} - 1\,V) \cdot \sqrt{2} - 3\,V} \cdot 10\,k\Omega$$

It should be remembered this formula is only an approximation working well in vicinity of proposed here values of $C_{det}$ and $R_1 + R_2$. Verification in simulator is always advisable, especially when $C_{det}$ and $R_1 + R_2$ values are completely different from those proposed here.

If you use DC power supply, $V_{PS\_min}$ is just minimum voltage of $V_{PS}$ when power loss should be detected, however adding some margin to this value is always beneficial. Additionally, values of $R_1$ and $R_2$ can be higher for lower power loss in those resistors, but still their values should not be higher than $100\,k\Omega$.

## 7.3 Force update mode button

The line **nLDRB** is dedicated to connect push button, which can be used by end user to force SERCE to enter update mode. Please refer to Firmware update chapter for more details.

This button should be connected in the way that when it is pressed, it connects the line **nLDRB** to the ground. It is recommended to use pull-up resistor to keep **nLDRB** line high when button is not pressed. Figure 7.5 shows how to connect **nLDRB** pin on the host board.

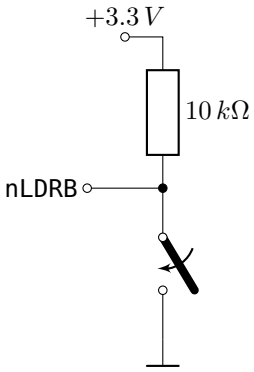



Figure 7.5: Schematic how to connect **nLDRB** pin on the host board

You can use **PSOUT** pin as $+3.3\,V$.

## 7.4 USB connection

SERCE provides USB high-speed interface, which works on 480 Mbps. It means that it needs special care to work properly.

All pins of USB connector should be connected to SERCE on the host board, and it doesn't need any additional components. Nevertheless, it requires care about layout and impedance matching.

It is recommended to use at least 2-layer PCB with impedance controlled traces for USB lines. It is also recommended to place USB connector as close as possible to SERCE in order to reduce length of USB traces.

USB data lines should be routed as differential pairs with $90\,\Omega \pm 15\%$ differential impedance. You have to calculate trace width and distance between traces to get required impedance for your specific stackup and PCB manufacturer. The high-speed signal pair should be trace length matched. Max trace length mismatch between high speed USB signal pairs should be no greater than 150 mils.

Connector ground pin should be connected to the same ground as SERCE (**GND**). It is recommended to use ground plane on the host board, preferably on the layer below USB traces. Plane below USB traces should be left unbroken under the connector and all the way to the SERCE.

Connector shield should be connected to the chassis of the device or to the ground through filter. $V_{BUS}$ line can be routed as single ended trace without special care.

## 7.5 IEC 60958 connections

SERCE provides IEC 60958 interface with up to 3 inputs and 1 output. Different types of connectors may be used. External circuitry that is required by the most common types of connectors is described below.

All IEC 60958 inputs and output should be connected to SERCE on the host board using carefully routed differential pair lines. The traces should be impedance controlled with the differential impedance matching the requirements of the particular input/output type.

### 7.5.1 S/PDIF coaxial connection

RCA or BNC connectors are typically used for S/PDIF coaxial input and output. It is recommended to use a high frequency transformer to isolate the SERCE from the external circuitry. The the minimal implementation is shown on figure 7.6, while recommended implementation of the S/PDIF coaxial input is shown on figure 7.7.

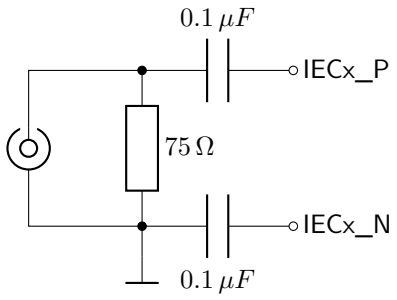



Figure 7.6: Minimal implementations of the coaxial S/PDIF input

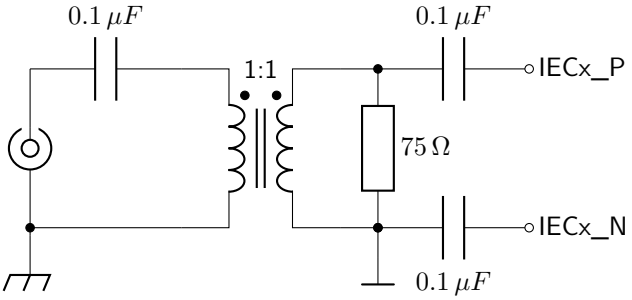



Figure 7.7: Recommended implementations of the coaxial S/PDIF input

Figures 7.8 and 7.9 show minimal and recommended implementation of the S/PDIF coaxial output. In case of the output, the voltage divider should provide desired signal level while ensuring the required output impedance of 75 Ω.

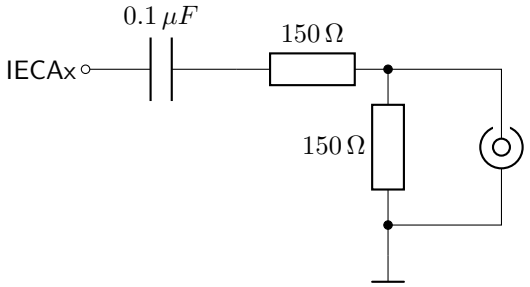



Figure 7.8: Minimal implementation of the coaxial S/PDIF output

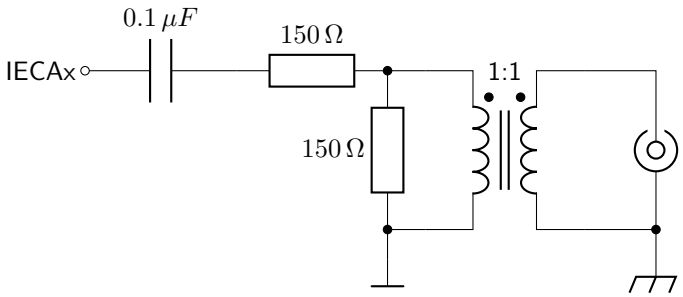



Figure 7.9: Recommended implementation of the coaxial S/PDIF output

### 7.5.2 S/PDIF optical connection

Optical S/PDIF input is implemented using TOSLINK receiver. The receiver may be connected to the SERCE using minimal implementation shown on figure 7.10, however, it is recommended to provide as clean as possible power supply for the receiver and to use filter on the input as shown on figure 7.11.

In some cases it may be also needed to use additional load ($R_{LOAD}$) on the receiver's output in order to achieve stable operation above 96 kHz. The value of the load resistor depends on the receiver used. The PLR135/T10 receiver along with $R_{LOAD} = 33\,\Omega$ is known to work well at 192 kHz.

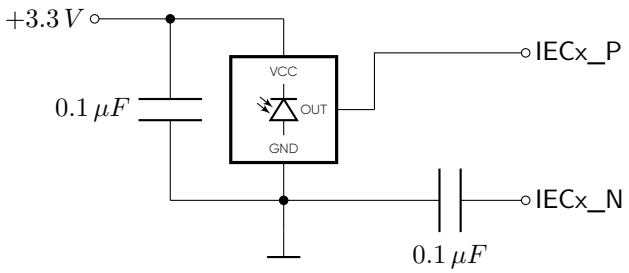



Figure 7.10: Minimal implementation of the optical S/PDIF input

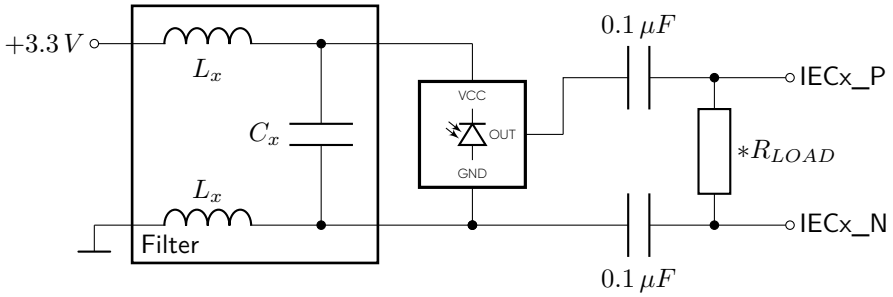



Figure 7.11: Recommended implementation of the optical S/PDIF input

### 7.5.3 AES3 connection

AES3 input and output typically are implemented using XLR connectors. This type of connection requires a transformers to isolate the balanced transmission line both on the transmitter and receiver side. The recommended implementation of the AES3 input is shown on figure 7.12.

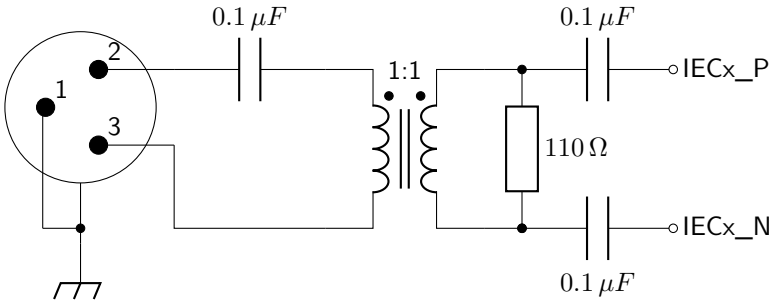



Figure 7.12: Recommended implementation of the AES3 input

SERCE provides only positive signal on the IEC 60958 output. This means that the AES3 output should be implemented using a differential line driver, as shown on figure 7.13. The recommended line driver is the AM26LV31E, however, any other driver with similar characteristics may be used. When selecting a different driver, please ensure that it provides at least 25 MHz of bandwidth.

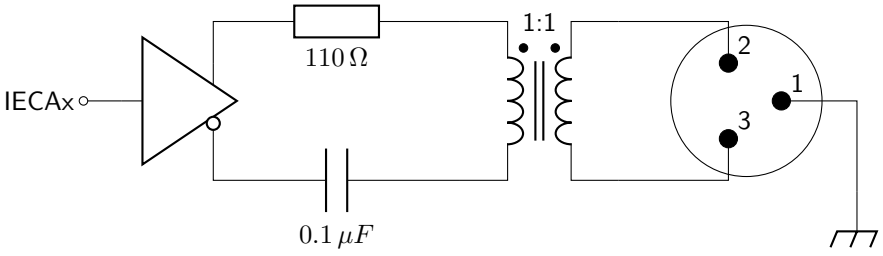



Figure 7.13: Recommended implementation of the AES3 output

## 7.6 External I2S connection

Recommended connection of External I2S is to use HDMI connector. Because SERCE contains all necessary hardware including ESD, it is only necessary to connect specified lines to SERCE. Table 7.2 shows pinout of HDMI connector and how to connect it to SERCE.

Table 7.2: Pinout of External I2S connection using HDMI connector

| **HDMI connector pin** | **Connection** |
|---|---|
| 1 | `EI2S_SDDR_N` |
| 2 | `GND` |
| 3 | `EI2S_SDDR_P` |
| 4 | `EI2S_SCK_P` |
| 5 | `GND` |
| 6 | `EI2S_SCK_N` |
| 7 | `EI2S_FSDL_N` |
| 8 | `GND` |
| 9 | `EI2S_FSDL_P` |
| 10 | `EI2S_MCKIO_P` |
| 11 | `GND` |
| 12 | `EI2S_MCKIO_N` |
| 13 | `EI2S_C1` |
| 14 | `EI2S_C2` |
| 15 | `EI2S_C3TX` |
| 16 | `EI2S_C4RX` |
| 17 | `GND` |
| 18 | unconnected |
| 19 | unconnected |
| `SHIELD` | connect to GND through filter or connect to chassis |

Lines `EI2S_SDDR_N/P`, `EI2S_SCK_N/P`, `EI2S_FSDL_N/P`, `EI2S_MCKIO_N/P` should be routed as differential pairs with $100\,\Omega \pm 5\%$ differential impedance. You have to calculate trace width and distance between traces to get required impedance for your specific stackup and PCB manufacturer.

`SHIELD` pin should be connected to the chassis of the device or to the ground through filter.

## 7.7 TV connection

TV connection uses HDMI connector, which pinout is shown in table 7.3. Please take special care about `TV_HEAC_P` and `TV_HEAC_N` lines, which are used for audio communication (ARC). Because all necessary hardware including ESD protection is already on SERCE, it is only necessary to connect these lines to HDMI connector, it doesn't need any additional components.

Table 7.3: Pinout of TV connection using HDMI connector

| **HDMI connector pin** | **Connection** |
|---|---|
| 1 | unconnected |
| 2 | GND |
| 3 | unconnected |
| 4 | unconnected |
| 5 | GND |
| 6 | unconnected |
| 7 | unconnected |
| 8 | GND |
| 9 | unconnected |
| 10 | unconnected |
| 11 | GND |
| 12 | unconnected |
| 13 | TV_CEC |
| 14 | TV_HEAC_P |
| 15 | TV_DDCC |
| 16 | TV_DDCD |
| 17 | GND |
| 18 | TV_PSOUT |
| 19 | TV_HEAC_N |
| SHIELD | connect to GND through filter or connect to chassis |

`SHIELD` pin should be connected to the chassis of the device or to the ground through the filter.

## 7.8 DAC connection

Main I2S output is designed to be connected to DAC. It also provides master clock signal. Lines `MI2S_FSDL` and `MI2S_SDDR` provides frame sync and serial data signals of PCM (I2S) or left and right data of DSD. If you need line to select DSD or PCM mode, you can use any GPIO line and then configure firmware to use it as DSD select line. You can use any GPIO line to control your DAC. You can also use any I2C bus to control your DAC. It is recommended to use separate I2C bus for DAC and other chips dealing with analog signals, but if you don't have enough I2C buses, you can use I2C3 bus which is shared with EEPROM and IEC 60958 receiver on SERCE. Please refer to I2C section for more details. Figure 7.14 shows how to connect MI2S to DAC.

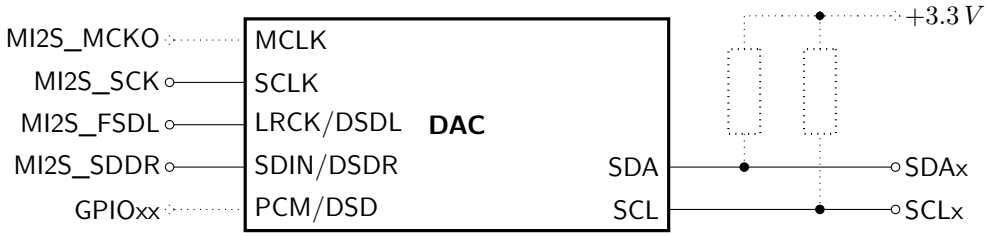



Figure 7.14: Schematic how to connect SERCE to DAC

# 8. Firmware update

SERCE provides firmware update interface over USB.

Update process is safe. It means that even if update process is interrupted, e.g. by power loss, SERCE will not be damaged. In case of any failure, SERCE will automatically enter update mode during startup, and it will wait for new firmware. It is also possible to force SERCE to enter update mode by pressing button connected to the `nLDRB` line, which may be useful in case of any bug in the firmware, which prevents SERCE from starting normally and entering update mode automatically.

Every firmware is encrypted, so it is not possible to modify it or use in another device. Firmware update feature incorporates also product validation, so it is not possible to install firmware for different product. Additionally, firmware update feature incorporates version management, so it is possible to get information about firmware version, including pre-release versions and build number.

Firmware update is available on Windows and macOS through SERCE Updater application[1]. SERCE Updater provides easy to use GUI for end user, which allows to update firmware in a few clicks. Additionally, it automatically checks firmware version with the server and, if newer firmware is available, it notifies the user. Then the user can update firmware in one click, without a need to download firmware manually, because SERCE Updater downloads firmware from the server automatically. SERCE Updater also provides a way to install firmware from the file, check firmware version and get information about firmware version, serial number and other information about SERCE and device built with it.

SERCE Updater is going to be available in Microsoft Store and Apple Mac App Store, so it will be easy to install and update it. Because it is fully managed by HEM, it will be updated automatically with new features and bug fixes, and there is no need to struggle with customizing, signing and distributing it. It is also free for end users. There will be available also a way to load some brand customization, e.g. logo, which will be displayed in SERCE Updater at your device section.

## 8.1 Forcing update mode

The line `nLDRB` is dedicated to connect push button, which can be used by end user to force SERCE to enter update mode. Please refer to 7.3 section for host board implementation details.

When SERCE is powered on, it checks the state of this line. If it is connected to the ground, SERCE enters update mode, otherwise it starts normally. It is highly recommended to implement this button on the host board. It gives a possibility to rescue SERCE module and device built with it in case of firmware update failure. If you have buttons on the device (e.g. on front panel), you can use one of them to enter update mode. To do this, connect the button to the line `nLDRB`. In the firmware, you can use this line as any other GPIO line, so this button can be used for other purposes as well (e.g. to switch inputs of the device). It is checked only during startup, so it is possible to use it for other purposes in the firmware.

Then you have to describe that button as "update mode button" in your device's user manual. In the other case, if you don't have any buttons on the device, you can add button to the host board to be accessible by the user, e.g. on the back panel of the device or inside the device's case, e.g. by a hole.

[1] SERCE Updater is already work in progress, please contact us to get to know status of the project. It will be available soon.

# 9. Customization

SERCE is designed to be customized by the user, which includes:

- Vendor and product IDs for USB
- Vendor and product names for USB
- Encryption key for firmware
- Audio Class 2.0 terminal type
- CEC name shown on TV
- Maximum supported sample rates and formats

Firmware includes SERCE Library (OS, drivers, audio processing, etc.) and user application, which can be written using SERCE SDK[1].

Vendor and product IDs and names for USB as well as encryption key for firmware are stored in EEPROM on the host board, and they have to be programmed during production. Additionally, vendor and product IDs and key has to be configured in the build process to correctly build firmware binary. All the parameters, expect key, has to be set in the firmware by using API provided by SERCE SDK.

SERCE SDK gives you ability to write your own firmware for SERCE, so you can support your host board hardware in the firmware, e.g. you can configure your DAC chip, read buttons, rotary encoder, remote control, light up LEDs, show information on display, etc. By using SERCE library you can easily use all the features of SERCE and you don't have to care about low level details.

SERCE SDK contains also example firmware, which can be used as a starting point for your own firmware.

If you don't want to develop firmware by yourself, we can customize our evaluation board firmware to your needs. Please contact us to get more details.

[1]SERCE SDK is available after signing NDA, please contact us to proceed.

# Revision history

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | 2023-12-25 | PG, JJ, AG, MM | First version of datasheet document.<br>It replaces "SERCE design guide". |