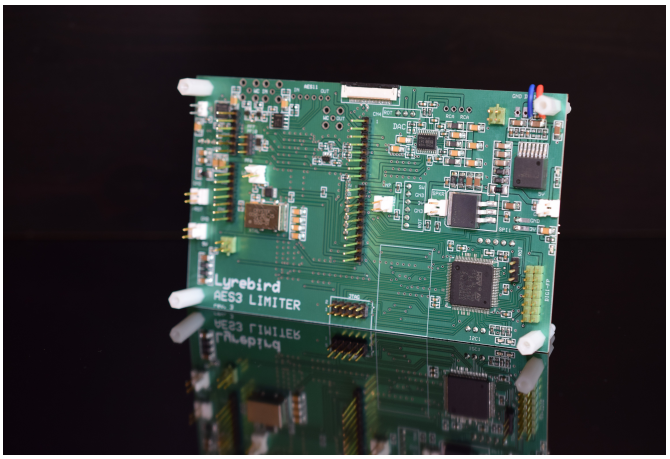
Lyrebird APP

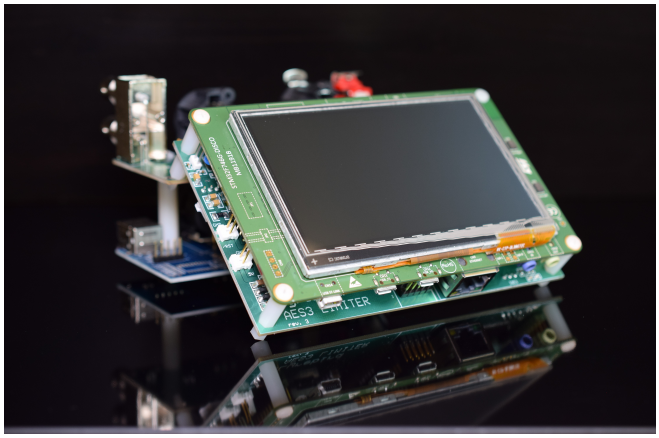
DualMCU

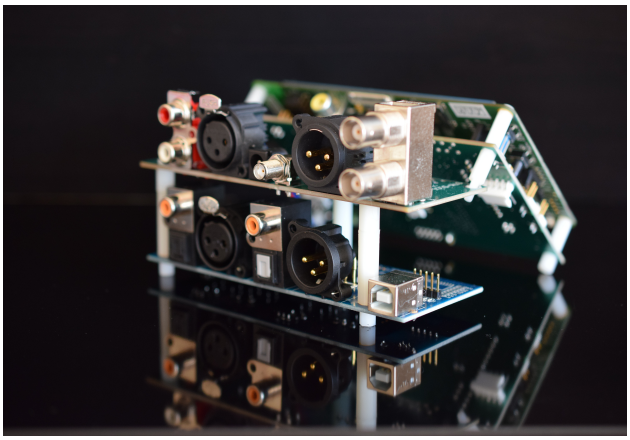
Torsten Jaekel
March 2016

Audio Processing Platform with Cortex-M7 and multiple processing engines (MCUs)



DualMCU with STM32F767


DualMCU plus STM32F7 Discovery


DIGI-FP as digital audio in and out

Overview

The “Lyrebird Audio Processing Platform” (APP) is used to do high-performance, professional audio processing.

In order to provide a flexible solution for any kind of “Audio Processing Pipeline” a hardware module was designed, als DualMCU board. It can be used stand-alone for audio-processing or in combination with other system components such as STM32F7 Discovery, Raspberry Pi, PINE64 etc.

The DualMCU consists of the following main components:

- **STM32F767 Cortex-M7** with DSP and **Double Precision** (DP) Floating Point Unit (FPU), 216 MHz nominal system clock – overclocking with **260 MHz**
- Connected to a DIGI-FP board for all kinds of **digital audio interfaces**, e.g. AES3, extended by a **Sample Rate Conversion** (SRC) chip
- High-accurate clock generation and **PLL for low-jitter audio** performance, **Word Clock** (WC) inputs and outputs
- High-quality **on-board DAC** for 2Vrms analog output for external power amplifiers
- Up to **192 KHz audio** sample rate, internal **32bit** audio processing with DSP and hardware DP FPU support

The DualMCU board is intended to be connected directly with a STM32F7 Discovery board. Such board is used for UI, user control via LCD display, touch and rotary encoder.

The audio processing load can be shared between both boards, both MCUs – therefore DualMCU. Some filters can be running on Discovery board, the others on DualMCU. The load sharing is very flexible and results in a much better processing performance, flexibility and space for future feature extensions, such as network based audio (DANTE, AES67).

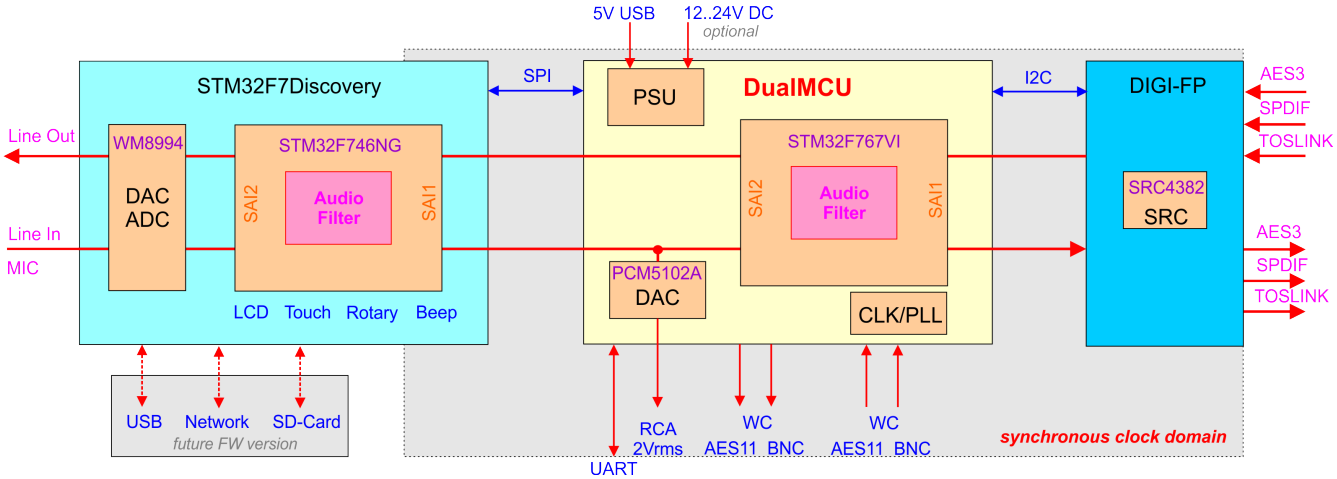


Figure 1: System overview – DualMCU in combination with STM32F7 Discovery

The DualMCU provides the following features and use cases:

- select between common, professional digital audio interfaces, such as:
  AES3 (AES/EBU) – XLR balanced

SPDIF – RCA unbalanced
TOSLINK – optical

- output digital audio on:
  AES3 (AES/EBU) – XLR balanced
  SPDIF (RCA unbalanced
  TOSLINK – optical
  Remark: all outputs are active in parallel
- have Sample Rate Conversion done by hardware (no processing power needed on MCU), used to upconvert from 48 KHz to 96 or even 192 Khz (also possible for the 44.1 based audio)
- high-accurate (low-jitter) clock generation and running all audio interfaces as a single synchronous clock domain
- external Word Clock (LRCK) inputs and outputs, synchronize with other devices in an audio network or let it act as an audio clock master (to generate Word Clock)
- convert audio coming from any digital interface into high-quality analog output for power amplification, e.g. used as a monitor in audio systems
- if DualMCU is used in standalone mode (without Discovery or other processing engines) then the system can be controlled, configured, monitored via UART, with Discovery board connected – this will "talk" to the DualMCU and all configruation done via UI (LCD, Touch) includes also the DualMCU
- direct connection via Arduino header for the STM32F7 Discovery (no wiring, direct plug-and-play)
- secondary I2S input for any other I2S input or output, e.g. to connect a Raspberry Pi, a USB-to-I2S bridge, other I2S audio DACs etc.
- (optional) on-board Power Supply Unit (PSU) capable to power the entire system, including Discovery, Raspberry Pi (12..24V, 2A total on 5V rail, 1.5V on 3.3V rail)
- entire system can be powered just via USB-B device (used also as USB-based UART)
- high-performance PLL used to support 48 as well as 44.1 KHz based audio sample rates

In combination with a STM32F7 Discovery the following use case examples are possible:

- AES3 audio in – processing – AES3 audio out (plus 2Vrms out)
- AES3 audio in – processing – Line Out (plus 2Vrms out)
- Line In or MIC in – processing – AES3 out (plus 2Vrms out)
- Line In or MIC in – processing – Line Out (plus AES3 out, plus 2Vrms out)
- Word Clock used as AES3/SPDIF/TOSLINK embedded audio clock (received clock)
- Word Clock from external input (AES11 balanced, BNC unbalanced), including option to use also GPS Disciplined Clock, e.g. 10 MHz input from a GPS receiver (Atomic Clock)
- Word Clock output (always active) in order to synchronize with other devices, AES11 (balanced) or BNC unbalanced

UART Shell

The DualMCU provides an USB-based UART interface. If a Silicon Labs CP21xx driver is installed, a UART terminal program, e.g. TeraTerm, Hyperterminal, can be used in order to control the DualMCU.

A simple UART Shell is provided for commands and parameters.

This UART Shell can be used also in combination with a connected STM32F7 Discovery. Updates on DualMCU will also be reflected on the Discovery UI (LCD).

The entire system can be powered via this USB device port, even without using the UART Shell. The external DC PSU input is optional.
Dual Power Mode (both power inputs connected) is possible so that it provides additional safety in terms of power rail failures.

UART Parameters

Baud rate:	**115200**
Data bits:	8
Stop bits:	1
Parity:	None
Flow control:	None
Needed driver:	Silicon Labs CP21xx

UART Commands

Commands are single character commands, e.g. h for help, followed by parameters provided as decimal or hexa-decimal values. Command and parameters are separated by spaces (not tabs). A command line is entered with a Newline or Carriage Return.
The command line can be corrected, modified with the Backspace key.

List of Commands

help, display, monitor commands:

h	- help, display list of commands
L	- display CPU load (in percentage)

system commands:

R	- reconfigure all with current settings
X	- system (soft) reset and restart with defaults (similar to hardware reset)

audio processing pipeline related commands:

f p1	- enable (1) or disable (0) all filters (bypass)
c p1	- configure audio pipeline, 0..3 (see following)
s p1	- set sample rate: 0 = 48KHz, 1 = 44.1 Khz
r p1	- select reference clock: 1 = intern (received on AES3, SPDIF), 0 = extern (from system clock, WC input etc.)
x p1	- set SRC rate (upsampling ratio): 0 = x1 1 = x2 2 = x4
i p1	- select audio clock input: 0 = TOSLINK 1 = SPDIF 2 = AES3 4 = not used

audio filter related commands:

e p1 p2 p3 p4 p5	- set gain (-9..9) for the 5 channels of 5-Band-Equalizer (EQ) Remark: channel 1 is limited to +8 as max.

Remarks:

- Trailing parameters can be omitted (for the rest of command line) – they are taken as 0 then.
- The command letter should start on first position in command line. Between command, parameters and between parameters any number of spaces is possible for separation.
  Tab is not supported or used.
- The maximal command line length is limited to 80 characters. If more characters are entered then the UART Shell will take the first 80 characters and process it like an entered line.
- Newline or Carriage Return can be used to enter a command line (to complete the user input).
- Backspace to delete characters from the end of line is supported.
  Other commands such as arrow keys, Delete, Bell are not supported (no command line edit).
- Escape sequences (often due to arrow keys, e.g. Up and Down) are ignored.
- The DualMCU will echo all characters entered – as UART Remote Echo (no local echo on terminal needed).

Audio Processing Pipeline

Besides the Audio Processing Pipeline which is built by pipelining the DualMCU with another board, e.g. STM32F7 Discovery or Raspberry Pi – the following information is just related to the filter and Audio Processing Pipeline inside the DualMCU itself.

Audio Filter Chain

The DualMCU provides two bi-directional stereo I2S (actually TDM, up to 16 channels) interfaces:

- *Right-Hand-Side* SAI1 Port: used to connect DIGI-FP as Input and Output module
- *Left-Hand-Side* SAI2 Port: used to connect other nodes, processing engines, e.g. Discovery board, I2S DACs, USB-to-I2S interfaces, Raspberry Pi etc.

The audio is routed through the DualMCU via firmware. In any of the audio pathes (from right-to-left for input, from left-to-right for output) filters can be inserted (mounted and connected). Such a filter chain can consistent of any number of pipelined audio filters, e.g. DC-Blocker + 5-Band-EQ + FFT + etc. It is considered following as one filter block ("filter container").

The entire audio filter chain is based on 32bit processing: audio is processed internally as 32bit samples, even the DIGI-FP with SRC chip support just 24bit formats.

The maximal audio sample rate supported by the firmware is 192 KHz.

Floating Point processing via hardware FPU is used. In terms of using the latesst STM32F767 MCU – Double Precision is used.

Word Clock

The entire DualMCU plus DIGI-FP audio pipeline is based on the same Word Clock (LRCK). All interfaces are running synchronized with the same clock reference. Therefore, no concerns in terms of audio distortions due to crossing asynchronous clock boundaries.
Such an asynchronous clock boundary exists just in case the STM32F7 Discovery on-board ADC/DAC is used, relevant for Line In/MIC and Line Out. But the firmware takes care about this issue in order to provide distortion, jitter-free audio processing through the entire system.

The Word Clock can be selected as:

- Use the AES3, SPDIF, TOSLINK embedded audio clock (received audio with clock) – "internal"
- Use the on-board high-accurate oscillator (XTAL) and PLL:
  the native sample rate is 48KHz (24.576 MHz XTAL) but via the PLL also sample rate as 44.1 KHz can be configured and used (and all multiples of it, up to 192 KHz).
- Use an external Word Clock input (AES11 XLR balanced, BNC unbalanced): the Word Clock is feed to the PLL which will provide an accurate internal audio clock (LRCK). A wide range of Word Clock frequency is supported, e.g. 42 .. 52 KHz for 48 KHz sample rate audio. All audio interfaces are using this Word Clock (is selected) and will follow automatically a changed Word Clock frequency.

Electrical Parameters:

BNC:	TTL signal, unbalanced (ground based), nominal 5V, range: 2 … 7V Impedance: 75 Ohm
AES11:	XLR balanced (symetrical), nominal +/-5, range: +/-2 .. +/-7V Impedance: 110 Ohm
Frequency:	42..52 KHz for 48 KHz based audio 38..50 KHz for 44.1 KHz based audio

- Use an external GPS Disciplined Clock: a GPS receiver can provide a clock signal, e.g. 10 MHz. Such one can be received on a dedicated interface (SMA connector).

  Electrical Parameters:

SMA:	TTL signal, unbalanced (ground based), nominal 3.3V, range: 2..7V Impedance: 75 Ohm
Frequency:	nominal 10 MHz, any other frequency, e.g. 49 KHz possible via PLL config

Remarks:

- If the Word Clock input fails or is not connected – the internal clock reference is used.
- Not used, especially not connected Word Clock inputs, should not be selected via configuration command. It can result in received noise and randomly toggeling PLL Lock. Otherwise it is recommended to terminate the inputs, e.g. to have a 75R resistor on BNC input if not used but enabled.
- The inputs do not have a termination resistor. Make sure to have one in the clock distribution network.
  (There is not an option to enable/disable termination on inputs, except to solder a fixed termination resistor which would be active all the time, not equipped as default).
- The Word Clock outputs (both) are active all the time. They provide the Word Clock selected and used by the system. In case of external Word Clock input - the Word Clock outputs will provide the same, just regenerated Word Clock frequency.
  In case of using embedded audio clock (“internal”, received on AES3, SPDIF etc.) or “extern” as running with on-board clock generation – this frequency is provided.
  The Word Clock outputs provide always the same reference audio clock.

Filter Pipeline

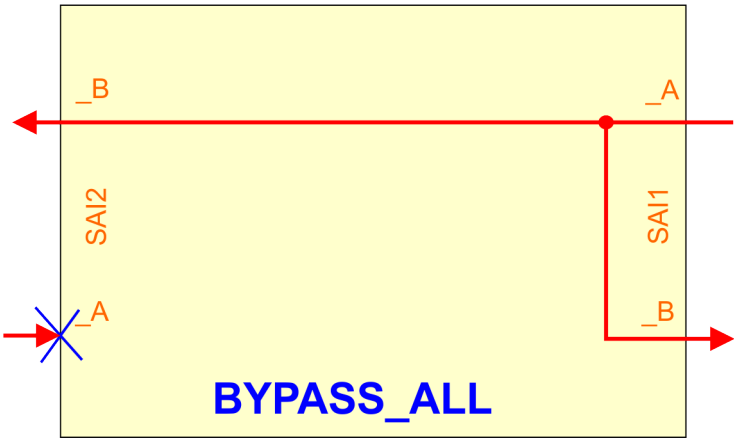
Via the UART Shell command '**c**' the Audio Processing Pipeline is configured. It means: the selection of inputs ans outputs is done, in combination with the mounting of the filter block. There are different types of audio routing schemas which depend also on existing of a STM32F7 Discovery board (or Raspberry Pi, on the left-hand-side SAI2 port).

The parameter 'p1' for this 'c' (configure chain) can be taken from following table.

Bit 1 | Bit 0	0 - *NO_DISCO*	1 - *WITH_DISCO*
0	p1 = **0** ***BYPASS_ALL***	p1 = **1** ***FILTER_A***
1	p1 = **2** ***LOOPBACK_INT***	p1 = **3** ***FILTER_B***

Table 1: The parameter for the 'c' command, the potential Audio Processing Pipeline modes

BYPASS_ALL

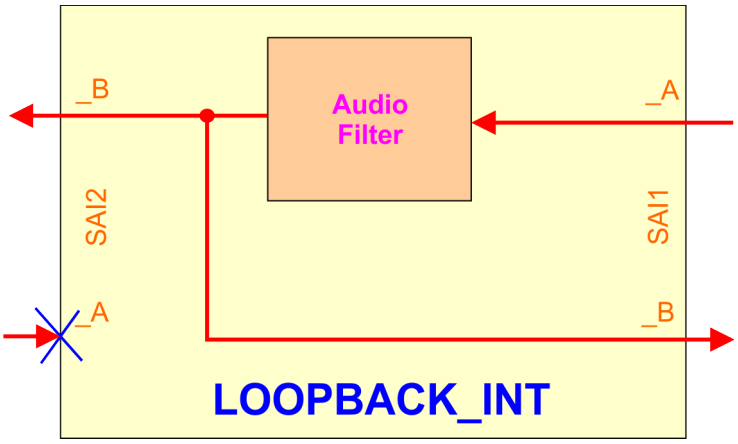


This Audio Processing Pipeline is used for testing or when DualMCU runs in standalone mode, without any filter activated.

It results in the highest performance and very low overhead for the firmware (lowest CPU load as reference when the performance of system is measured).

The DIGI-FP acts as input and output for the non-processed audio (1:1 fed-through and loop back).

LOOPBACK_INT



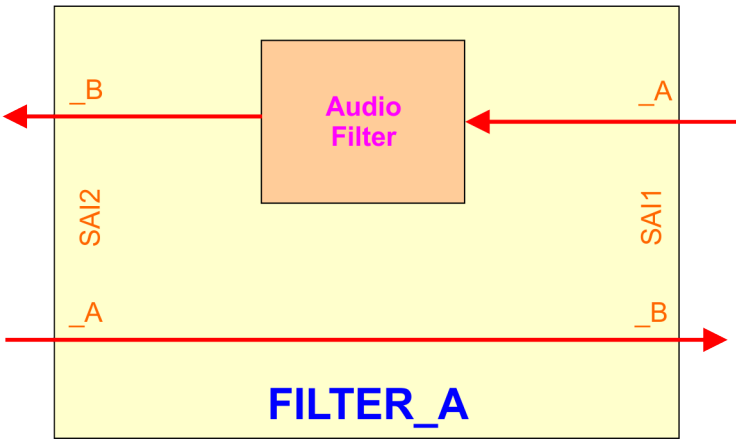
This Audio Processing Pipeline is the default if the DualMCU is running in standalone mode. All filters can be active. The DIGI-FP acts as input and output for the processed audio.

This LOOPBACK_INT can be used also to forward the processed audio to a STM32F7 Discovery board or other external I2S DACs (when connected on the left-hand-side).

Remark:

> The on-board DAC for the 2Vrms RCA output will not provide any audio output signal.

FILTER_A



The *FILTER_A* Audio Processing Pipeline can be used either for DualMCU standalone or with STM32F7 Discovery, Raspberry Pi etc.

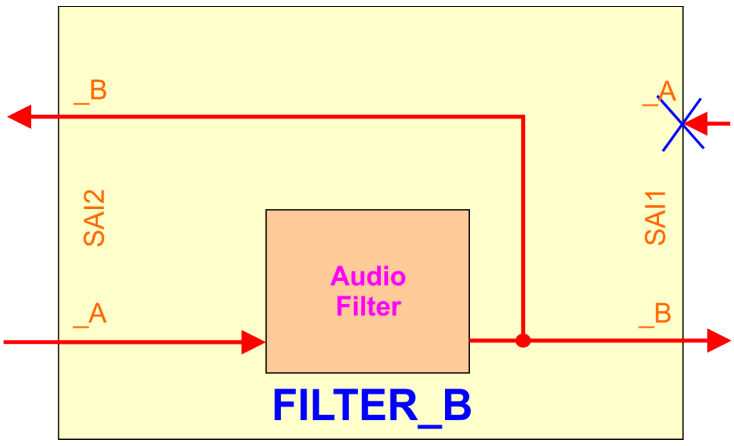
In standalone mode - a jumper on DualMCU board can be used in order to loop back the audio on left-hand-side externally.

If the STM32F7 Discovery is connected then this board could loop back the audio signal, if needed, e.g. for AES3 In and Out but with further audio processing inside STM32F7 Discovery.

Remark:

> If STM32F7 Discovery loops back the audio or if the jumper for external loop back is connected in standalone mode – the on-board DAC for RCA 2Vrms output will provide processed analog audio output.

FILTER_B



The *FILTER_B* Audio Processing Pipeline is used only in combination with a connected STM32F7 Discovery board. It is used when:

- STM32F7 Discovery or Raspberry Pi, other external I2S component will act as the audio inout, audio source
- Used if Line In/MIC is selected as audio source.

Remark:

> The analog audio output on the on-board DAC (2Vrms RCA) will provide the non-processed audio signal.

Attention:

> Do not select this *FILTER_B* mode if STM32F7 Discovery will loop back the same signal or if the loop back jumper is connected on DualMCU. It results in oscillation.

Use of LEDs

The DualMCU has two green LEDs on the lower edge of the PCB.

The LEDs are used as indicators controlled by the firmware:

- Left LED: flashes with 1 second period as indication that the firmware processes the task loop
- Right LED: flashes very fast (most of the time on) if audio is received, a new clock trigger has been received.
  The right LED will turn off if no audio signal is received or there is not any audio clock available for the firmware to process audio signals.

Inter-MCU Communication

In case another module such as the STM32F7 Discovery board or Raspberry Pi is used and connected – the DualMCU can be controlled via a SPI interface.

This SPI interface is capable to use half-duplex bi-directional communication up to 25 Mbps.

The DualMCU runs as a SPI slave (Discovery should be SPI master).

SPI mode 0 is used, with separate unidirectional MOSI and MISO signals.

More detailed information about the SPI Inter-MCU commands, packet format etc. is provided on request.