Low Power Microphone Acquisition and Processing for Always-on Applications Based on Microcontrollers
Architecture I: standalone $\mu$C

- Microcontroller used to implement the complete application, targeting low power (example: wearable, remote controller)
Architecture II: $\mu$C connect to application processor

- Microcontroller used for low-power voice detection and microphone acquisition (example: $\mu$C used as audio sensor hub)
**Architecture III:** standalone µC connected to the cloud

- High-End Microcontroller used to implement the complete application: low-power voice detection, microphone acquisition, cloud connection and voice-answer decoding
  
  *(example: Amazon Alexa)*
# Different types of MEMS microphones

<table>
<thead>
<tr>
<th>Microphone type</th>
<th>Pro</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog</td>
<td>Power consumption</td>
<td>ADC performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC power consumption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>External amplifier</td>
</tr>
<tr>
<td>Digital I2S</td>
<td>Integration</td>
<td>Power consumption</td>
</tr>
<tr>
<td>Digital PDM</td>
<td>Standard</td>
<td>Digital interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power consumption</td>
</tr>
<tr>
<td></td>
<td>Multimode</td>
<td>Digital interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power consumption</td>
</tr>
</tbody>
</table>
PDM: Pulse Density Modulation

• Relative density (local average) of the pulses corresponds to the analog signal's amplitude

• Quantization noise is very high, but is pushed to very high frequency
PDM to PCM conversion

- Increase sample resolution from 1 to 16 bits
- Decrease sampling frequency from 2 MHz to e.g. 48 KHz
  - Low-pass filtering and downsampling
How to acquire a PDM microphone with a microcontroller

- **Software**: PDMtoPCM library provided with the STM32Cube

- **Hardware**: DFSDM = Digital Filter for Sigma Delta Modulators
Standard Architecture

PDM Input

DFSDM HW IP

PDM LP Filter and Decimation

SW processing

Signal Conditioning

PCM Output

PCM Output

Voice Trigger Detection

Indicator (trigger ID)

Led Blink

Caption:

STMicroelectronics

Sensory

Audio flow

IT

WWW.SENSOREXPO.COM

#SENSORS17
TrulyHandsfree™ Voice Control

- World’s leading (by far!) “always-on always listening” phrase spotting for wakeup words and hands-free control
  - Fast, reliable, noise robust and far field
  - Fixed, User-Enrolled and User-Defined voice triggers
  - Speaker Verification and Identification
  - Phrase-spotted command sets up to 50 words in limited listening window
  - “Trigger to Search” – no pause needed between trigger and following command/query
  - Numerous awards and implementations in over 2B products

- Deeply Embedded on STM32
  - small footprint/low power
Platform used for the tests

- Flexible board power supply
  Through USB or external source
- Integrated ST-Link/V2.1
  Mass-storage device flash programming
  Virtual COM port for communications
- 2 push buttons, 2 color LEDs
- Arduino™ extension connectors
  Easy access for add-ons
- One STM32 MCU flavor with 64 pins
- Morpho extension headers
  Direct access to all MCU I/Os
Microcontroller Block Diagram

STM32L452

- **Connectivity**
  - USB Device
  - 1x SD/SDIO/MMC
  - 3x SPI
  - 4x I²C
  - 1x CAN
  - 1x Quad SPI (Dual Flash)
  - 4x USART + 1x ULP UART

- **Digital**
  - TRNG, 1x SAI, DFSDM (4 channels)

- **I/Os**
  - Up to 83 I/Os
  - Touch-sensing controller

- **ARM® Cortex®-M4 CPU**
  - 80 MHz
  - FPU
  - MPU
  - ETM

- **DMA**

- **ART Accelerator™**
  - Up to 512-Kbyte Flash with ECC
  - Single Bank

- **Timers**
  - 8 timers including:
    - 1x 16-bit advanced motor control timers
    - 2x ULP timers
    - 4x 16-bit-timers
    - 1x 32-bit timers

- **Analog**
  - 1x 16-bit ADC, 1x DAC
  - 2x comparators, 1x op amp
  - 1x temperature sensor

WWW.SENSORSEXPO.COM
#SENSORS17
Using DFSMD in low-power voice acquisition

- DFSMD acquired with DMA
  - Microphone acquisition is performed in sleep mode
  - Every 16ms the µC wakes up to process the audio
Power Consumption example

using microphone clocked at 1 MHz

<table>
<thead>
<tr>
<th>μC @ 1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>1.2 mA</td>
</tr>
</tbody>
</table>

run

run

sleep

run

sleep

Min | Max | Average
---|-----|--------
235.76 μA | 2.22 mA | 1.18 mA
How to optimize from here: Sound Detector

DFSDM HW IP
- PDM Input
- PDM LP Filter and Decimation
- Signal Conditioning
- PCM Output

SW processing
- Voice-Trigger Detection
- Indicator

Low-Power Sound Detector (LPSD)

Caption:
- STMicroelectronics
- Sensory

Audio flow
IT
Power Consumption example

*using microphone clocked at 1 MHz*

### LPSD state

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>~360 µA</td>
</tr>
</tbody>
</table>

### Voice Trigger Detection

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>1.2 mA</td>
</tr>
</tbody>
</table>
Sound Detector considerations

- LPSD (Low-Power Sound Detector) is provided by Sensory and is integrated in the voice-recognition engine.
- The impact of a custom sound detector has to be evaluated with the third-party voice-recognition provider.
  - Audio is processed only after sound detection, therefore the voice recognition might miss the beginning of the trigger when it’s said in a quiet environment.
How to optimize from here: **ULP with watchdog**

DFSDM HW IP:
- PDM Input
- PDM LP Filter and Decimation
- Analog watch dog

SW processing:
- Signal Conditioning
- PCM Output
  - Used to enter/exit ULPSD state

Low-Power Sound Detector (LPSD):
- PCM Output
  - Low-Power Sound Detector (LPSD)

Voice-Trigger Detection:
- Indicator

Caption:
- STMicroelectronics
- Sensory

Audio flow

IT
Power Consumption example

*using microphone clocked at 1 MHz*

### ULPSD state

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>~130 µA</td>
</tr>
</tbody>
</table>

### LPSDL state

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>~360 µA</td>
</tr>
</tbody>
</table>

### Voice Trigger Detection

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>1.2 mA</td>
</tr>
</tbody>
</table>
ULP with watchdog considerations

- During ULP the µC is not buffering the audio in RAM, therefore the voice recognition might miss the beginning of the trigger when it’s said in a quiet environment.
  - If the watchdog is tuned correctly the system should wake up from ULP and stay in LPSD mode when there is minimum background noise (user in the room) and enter in ULP mode only for long periods without any noise (for example at night).
How to optimize from here: *multimode microphones*

<table>
<thead>
<tr>
<th>Mode</th>
<th>Clock</th>
<th>Power consumption</th>
<th>SNR (speech freq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>~1MHz - ~3MHz</td>
<td>600/700µA</td>
<td>~64 dB SNR</td>
</tr>
<tr>
<td>Low Power</td>
<td>~350KHz - ~800KHz</td>
<td>~250µA</td>
<td>~64 dB SNR</td>
</tr>
</tbody>
</table>

- datasheets specify only SNR at normal low-power frequency (768KHz)
- we made some tests at lower frequencies with a few parts:

<table>
<thead>
<tr>
<th>Fclk</th>
<th>SNR (dB)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>64.305</td>
</tr>
<tr>
<td>768</td>
<td>64.184</td>
</tr>
<tr>
<td>600</td>
<td>59.541</td>
</tr>
<tr>
<td>384</td>
<td>43.099</td>
</tr>
</tbody>
</table>

(*) the measurement BW is 20Hz – 8kHz.
By limiting the upper measurement BW, the user will see improved SNR at lower Fclk, at the expense of audio bandwidth.
Power Consumption example

*using multimode microphone clocked at 500 kHz*

**ULPSD state**

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>~70µA</td>
</tr>
</tbody>
</table>

**LPSD state**

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>~200µA</td>
</tr>
</tbody>
</table>

**Voice Trigger Detection**

<table>
<thead>
<tr>
<th>µC @1.8V</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L452</td>
<td>1.2 mA</td>
</tr>
</tbody>
</table>
Multimode microphones considerations

- Lower is the required microphone clock lower is the microcontroller internal clock needed to acquire the microphone
  - Standard clock in low power mode is 768KHz (16KHz audio obtained by decimation by 48)

- From datasheet the multimode microphones work with frequencies lower than 768KHz, but from measurements there is a trade off between SNR and clock

- The power consumption of a multi mode microphone in low-power mode (768KHz) is much lower than a standard microphone (250µA vs 650/700µA)
Possible improvements

**LPSD:** Change the sound detector in order to detect only speech and not sounds. Such a VAD will require more MIPS (more power consumption)
- What’s the statistic of each power mode in a real use case?

**Clock scaling:** use low microphone clock only for the ULP mode and switch to higher clock while executing LPSD and voice recognition

**Using SMPS:** Possibility to supply μC VCORE logic with an external DC/DC (bypass of internal LDO regulators)
Example with SMPS

- Possibility to supply µC VCORE logic with an external DC/DC (bypass of internal LDO regulators)
- Allow to get lower power consumption on same SW application.

<table>
<thead>
<tr>
<th>Freq</th>
<th>Algorithm</th>
<th>SMPS efficiency</th>
<th>SMPS ON</th>
<th>SMPS OFF</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>While(1) @ 3.3V</td>
<td>85%</td>
<td>37µA/MHz</td>
<td>93µA/MHz</td>
<td>60%</td>
</tr>
<tr>
<td>(26MHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td>85%</td>
<td>39µA/MHz</td>
<td>108µA/MHz</td>
<td>64%</td>
</tr>
<tr>
<td>(80MHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

**Tuning:** Tuning of the parameters depends on the target application/use case.

**Different levels of low-power modes:** depending on target power consumption you can decide to implement only certain low-power modes or certain microphone clocks/configuration

**Overall system:** Look at the overall system requirement not only at the power consumption in the lowest mode.