// Prototype of method used below to reduce errors when setting

// all register sub-values:

//----------------------------------------------------------------------------------

class NumberBits **{**

public**:**

// From - Creates a number where the bits of that number may be a subsection of a

// larger number. An example is a 3-bit number 5 that is a subsection of a full

// byte such as this: 0001 0100 Here we may have two 3-bit numbers followed by

// one 2-bit number.

template**<**class TNumType**>**

static TNumType

From**(**uint8\_t starting\_bit\_index\_zero\_based\_**,** uint8\_t number\_of\_bits**,** TNumType value**);**

**};**

// The code that sends the I2C config data looks like this:

//----------------------------------------------------------------------------------

// Supporting macro

#define BASIC\_ARRAY\_LENGTH(arr) (sizeof(arr) / sizeof(arr[0]))

// Send the I2C Config 1A

**for(**size\_t i **=** 0**;** i **<** BASIC\_ARRAY\_LENGTH**(**I2C\_power\_ALT\_1a**);** i **+=** 2 **)**

**{**

hal\_err\_code **=** HAL\_I2C\_Master\_Transmit**(&**hi2c4**,** TAS2780\_ADDR**,** **&**I2C\_power\_ALT\_1a**[**i**],** 2**,** HAL\_MAX\_DELAY**);**

ASSERT\_HP**(**hal\_err\_code **==** HAL\_OK**,** "Failed I2C transmit"**);**

**}**

osDelay**(** 10 **);**

// Send the I2C Config 1B after slight delay

**for(**size\_t i **=** 0**;** i **<** BASIC\_ARRAY\_LENGTH**(**I2C\_power\_ALT\_1b**);** i **+=** 2 **)**

**{**

hal\_err\_code **=** HAL\_I2C\_Master\_Transmit**(&**hi2c4**,** TAS2780\_ADDR**,** **&**I2C\_power\_ALT\_1b**[**i**],** 2**,** HAL\_MAX\_DELAY**);**

ASSERT\_HP**(**hal\_err\_code **==** HAL\_OK**,** "Failed I2C transmit 2"**);**

**}**

// Data is sent in 2 chunks with a small delay between them. Here is the data

// that is sent:

//----------------------------------------------------------------------------------

uint8\_t I2C\_power\_ALT\_1a**[]** **=** **{**

////// Pre-Reset Configuration

0x00**,** 0x01**,** //Page 0x01

0x37**,** 0x3A**,** //Bypass

0x00**,** 0xFD**,** //Page 0xFD

0x0D**,** 0x0D**,** //Access page

0x06**,** 0xC1**,** //Set Dmin

0x00**,** 0x01**,** //Page 0x01

0x19**,** 0xC0**,** //Force modulation

0x00**,** 0xFD**,** //Page 0xFD

0x0D**,** 0x0D**,** //Access page

0x06**,** 0xD5**,** //Set Dmin

////// Software Reset

0x00**,** 0x00**,** //Page 0x00

0x7F**,** 0x00**,** //Book 0x00

0x01**,** 0x01 //Software Reset

**};**

uint8\_t I2C\_power\_ALT\_1b**[]** **=** **{**

////// Post-Reset Configuration

0x00**,** 0x01**,** //Page 0x01

0x37**,** 0x3A**,** //Bypass

0x00**,** 0xFD**,** //Page 0xFD

0x0D**,** 0x0D**,** //Access page

0x06**,** 0xC1**,** //Set Dmin

0x06**,** 0xD5**,** //Set Dmin

// 10.2 Initial Device Configuration - PWR\_MODE0

0x00**,** 0x00**,** // Page 0x00

0x0E**,** 0x02**,** // TDM tx vsns transmit disable with slot 2

0x0F**,** 0x00**,** // TDM tx isns transmit disable with slot 0

0x00**,** 0x01**,** //Page 0x01

0x21**,** 0x00**,** //Disable Comparator Hysterisis

0x17**,** 0xC8**,** //SARBurstMask=0

0x19**,** 0x00**,** //LSR Mode

0x35**,** 0x74**,** //Noise minimized

0x00**,** 0xFD**,** //Page 0xFD

0x0D**,** 0x0D**,** //Access Page 0xFD

0x3E**,** 0x4A**,** //Optimal Dmin

0x0D**,** 0x00**,** //Remove access Page 0xFD

0x00**,** 0x00**,** //Page 0x00

// Class-D switching mode - PVDD Only Supply of Class D

0x03**,** NumberBits**::**From**<**uint8\_t**>(**6**,** 2**,** 0b10**)** **|**

// AMP\_LEVEL (range is 0x00 - 0x14)

NumberBits**::**From**<**uint8\_t**>(**1**,** 5**,** 0x00**),** //0x80

0x1A**,** 0x00**,** // Digital Volume Control - 00h = 0 dB, 01h = -0.5 dB, 02h = -1 dB, ..., C8h = -100 dB, C9h = mute

// over current event 0b = Disabled ; over temperature event 0b = Disabled (uses OUT\_N and OUT\_P pins)

0x06**,** NumberBits**::**From**<**uint8\_t**>(**5**,** 1**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**4**,** 1**,** 0b0**)** **|**

// Post-Filter Feedback 0b = Disabled ; SMODE\_EN 1b = Enabled

NumberBits**::**From**<**uint8\_t**>(**3**,** 1**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**2**,** 1**,** 0b1**)** **|**

// OC Threshold Control 0h = Nominal Value

NumberBits**::**From**<**uint8\_t**>(**0**,** 2**,** 0x0**),** //0x04

0x71**,** 0x03**,** //PVDD UVLO set to 2.76V

// PVDD input and PVDD UVLO enabled RESERVED

0x02**,** NumberBits**::**From**<**uint8\_t**>(**7**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**5**,** 2**,** 0b0**)** **|**

// Current sense is 1b = Powered down ; Voltage sense is 1b = Powered down

NumberBits**::**From**<**uint8\_t**>(**4**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**3**,** 1**,** 0b1**)** **|**

// Device operational mode 000b = Active without Mute

NumberBits**::**From**<**uint8\_t**>(**0**,** 3**,** 0b000**),** //0x98

// VBAT1S supply 0b = Supplied externally; IRQZ internal pull up 0b = Disabled

0x04**,** NumberBits**::**From**<**uint8\_t**>(**7**,** 1**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**6**,** 1**,** 0b0**)** **|**

// Low EMI spread spectrum is 1b = Enabled; Forward Path DC blocker 1h = 2 Hz

NumberBits**::**From**<**uint8\_t**>(**5**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 3**,** 0x1**),** //0x21

// RESERVED ; Thermal Foldback is 0b = Disabled

0x05**,** NumberBits**::**From**<**uint8\_t**>(**4**,** 4**,** 0x4**)** **|** NumberBits**::**From**<**uint8\_t**>(**3**,** 1**,** 0b0**)** **|**

// Record Path DC blocker 1h = 2 Hz

NumberBits**::**From**<**uint8\_t**>(**0**,** 3**,** 0x1**),** //0x41

// RESERVED ; Limiter dynamic headroom 0b = Disabled

0x1B**,** NumberBits**::**From**<**uint8\_t**>(**6**,** 2**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**5**,** 1**,** 0b0**)** **|**

// Limiter attack rate 01h = 40 μs/dB ; Limiter is 0b = Disabled / 1b = Enabled

NumberBits**::**From**<**uint8\_t**>(**1**,** 4**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 1**,** 0x0**),** //0x42

// RESERVED ; SBCLK to FSYNC ratio when AUTO\_RATE = 1 (disabled) 04h = 64

0x60**,** NumberBits**::**From**<**uint8\_t**>(**6**,** 2**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**2**,** 4**,** 0x04**)** **|**

// RESERVED

NumberBits**::**From**<**uint8\_t**>(**0**,** 2**,** 0x1**),** //0x11

// Invert audio amplifier ouput 0b = Normal ; CLASSD\_SYNC 0b = Synchronized to audio clocks

0x08**,** NumberBits**::**From**<**uint8\_t**>(**7**,** 1**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**6**,** 1**,** 0b1**)** **|**

// Sample rate based on 1b = 44.1 kHz ; Auto detection of TDM sample rate 0b = Enabled

NumberBits**::**From**<**uint8\_t**>(**5**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**4**,** 1**,** 0b0**)** **|**

// Sample rate of the TDM bus 100b = 44.1/48 kHz; TDM frame start polarity 0b = Low to High on FSYNC

NumberBits**::**From**<**uint8\_t**>(**1**,** 3**,** 0b100**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 1**,** 0b0**),** //0x68

// TDM RX sample justification 0b = Left ; TDM RX start of frame to time slot 0 offset 0x1

0x09**,** NumberBits**::**From**<**uint8\_t**>(**6**,** 1**,** 0b0**)** **|** NumberBits**::**From**<**uint8\_t**>(**1**,** 5**,** 0x1**)** **|**

// TDM RX capture clock polarity 0b = Rising edge of SBCLK

NumberBits**::**From**<**uint8\_t**>(**0**,** 1**,** 0b0**),** //0x02

// current and voltage data 00b = 16 bits; TDM RX time slot select 11b = Stereo downmix;

0x0A**,** NumberBits**::**From**<**uint8\_t**>(**6**,** 2**,** 0b00**)** **|** NumberBits**::**From**<**uint8\_t**>(**4**,** 2**,** 0b11**)** **|**

// TDM RX word length 00b = 16 bits ; TDM RX time slot length 00b = 16 bits

NumberBits**::**From**<**uint8\_t**>(**2**,** 2**,** 0b00**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 2**,** 0b10**),** //0x32

// TDM RX Right Channel Time Slot ; TDM RX Left Channel Time Slot

0x0C**,** NumberBits**::**From**<**uint8\_t**>(**4**,** 4**,** 0x1**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 4**,** 0x00**),** // 0x10

0x0D**,** 0x03**,** // TDM\_CFG4 - Transmit settings (from PPC config)

**};**

uint8\_t I2C\_mute**[]** **=** **{**

0x00**,** 0x00**,** //Page 0x00

//-- send mute

// PVDD input and PVDD UVLO enabled ; RESERVED

0x02**,** NumberBits**::**From**<**uint8\_t**>(**7**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**5**,** 2**,** 0b0**)** **|**

// Current sense is 1b = Powered down

NumberBits**::**From**<**uint8\_t**>(**4**,** 1**,** 0b1**)** **|**

// Voltage sense is 1b = Powered down ; Device operational mode 000b = Active with Mute

NumberBits**::**From**<**uint8\_t**>(**3**,** 1**,** 0b1**)** **|** NumberBits**::**From**<**uint8\_t**>(**0**,** 3**,** 0b001**),**

// Digital Volume Control - 00h = 0 dB, 01h = -0.5 dB, 02h = -1 dB, ..., C8h = -100 dB, C9h = mute

0x1A**,** 0xC9

**};**

// After I2C sends config, this code sends the 2 sounds data buffers

// and then sends the mute:

//----------------------------------------------------------------------------------

hal\_err\_code **=** HAL\_I2S\_Transmit**(&**hi2s2**,** **&**cooking\_complete\_alert**[**0**],** BASIC\_ARRAY\_LENGTH**(**cooking\_complete\_alert**),** HAL\_MAX\_DELAY**);**

ASSERT\_HP**(**hal\_err\_code **==** HAL\_OK**,** "Failed I2S transmit 2"**);**

osDelay**(** 100 **);**

hal\_err\_code **=** HAL\_I2S\_Transmit**(&**hi2s2**,** **&**\_Test\_44\_16\_stereo\_HighAmplificationEndingSilence\_BE**[**0**],** BASIC\_ARRAY\_LENGTH**(**\_Test\_44\_16\_stereo\_HighAmplificationEndingSilence\_BE**),** HAL\_MAX\_DELAY**);**

ASSERT\_HP**(**hal\_err\_code **==** HAL\_OK**,** "Failed I2S transmit 2"**);**

//Send MUTE

**for(**size\_t i **=** 0**;** i **<** BASIC\_ARRAY\_LENGTH**(**I2C\_mute**);** i **+=** 2 **)**

**{**

hal\_err\_code **=** HAL\_I2C\_Master\_Transmit**(&**hi2c4**,** TAS2780\_ADDR**,** **&**I2C\_mute**[**i**],** 2**,** HAL\_MAX\_DELAY**);**

**}**