

CS40L5x Configuration User Guide

Introduction

This user guide describes the functionality, features and configurations that are supported in the CS40L5x Advanced Haptic Driver with Waveform Memory DSP and Closed-Loop Algorithms. This user guide is applicable to the CS40L51, CS40L52, and the CS40L53 devices. For information on the configuration of the CS40L50 device, please see the CS40L50 configuration user guide.

Table of Contents

Introduction	1
Table of Contents	1
1 Power Configuration	2
1.1 Power Supply Configurations	2
1.1.1 Low Power Rails: Device Supply	2
1.1.2 High Power Rail: Amplifier Supply	4
1.2 Operational States	6
1.2.1 Reset State	6
1.2.2 Hibernate State	7
1.2.2.1 Wakeup Sources	8
1.2.3 Standby State	8
1.2.4 Active State	8
1.2.5 Operating State Transition examples	9
2 Firmware	10
2.1 Initialization Sequence	10
2.1.1 ROM Firmware	10
2.1.2 RAM Firmware	10
2.2 Interrupt	10
2.2.1 Hardware Interrupts	11
2.2.2 Firmware Interrupts	13
3 Haptic Playback	15
3.1 Haptic Wavetables	16
3.1.1 ROM Waveform	17
3.1.2 RAM Wavetable	17
3.1.3 Runtime Haptics Waveforms	19
3.2 Triggering Options	19
3.2.1 I ² C/SPI Mailbox Trigger	20
3.2.2 GPI Trigger	20
4 Device Features (Algorithms)	21
4.1 Click Compensation	22
4.2 Calibration	26
4.3 Dynamic F0	27
4.4 Runtime Haptics (RTH)	28
4.4.1 RTH Registers	29
4.4.2 RTH PCM Register Format	30
4.4.3 RTH PWLE Register Format	32
4.5 Load Diagnostics	37
4.6 Generic Haptic System Controls	38
5 Revision History	39

Important Notice:

No license to any intellectual property right is included with this component, and certain uses or product designs, including certain haptics-related uses or haptics-system designs, may require an intellectual property license from one or more third parties.

1 Power Configuration

1.1 Power Supply Configurations

The CS40L5x device supports a variety of power-supply configurations. The power-supply configuration may require register settings during device initialization; the required settings are downloaded by the host processor after power-up. Low power rails include system supply (VDD_P), digital input/output supply (VDD_IO), analog supply (VDD_A). The high-power rail provides the amplifier supply. For details about component rating and layout, see the schematic and layout guidelines.

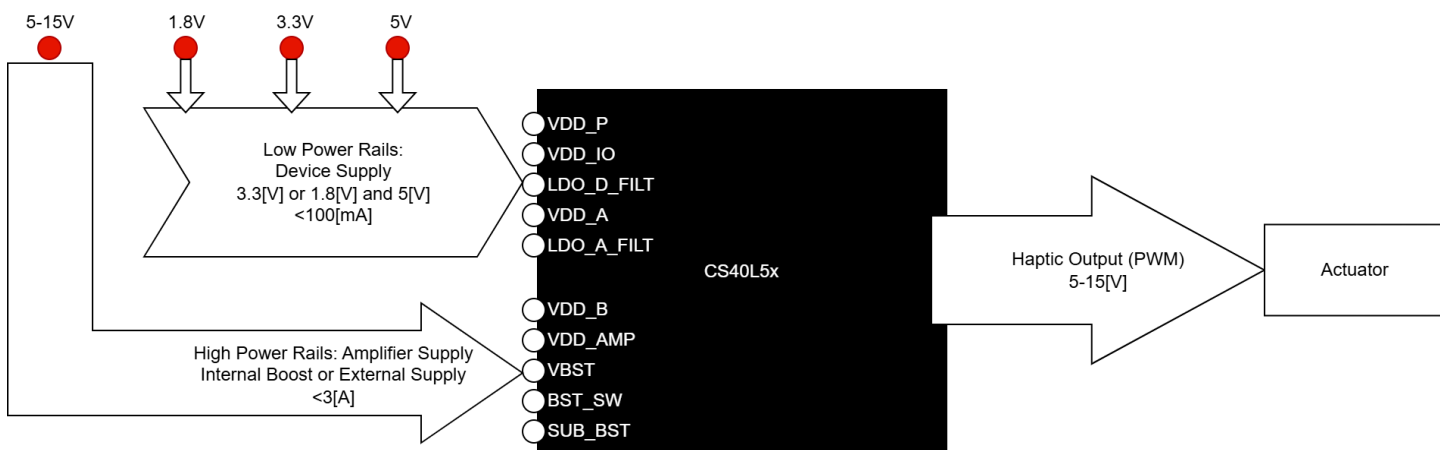
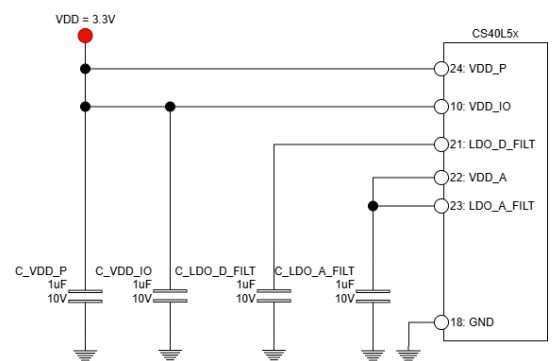
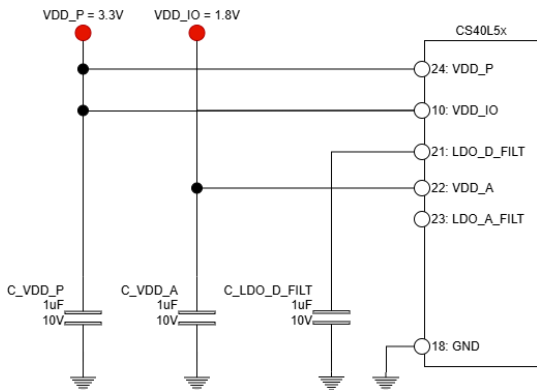
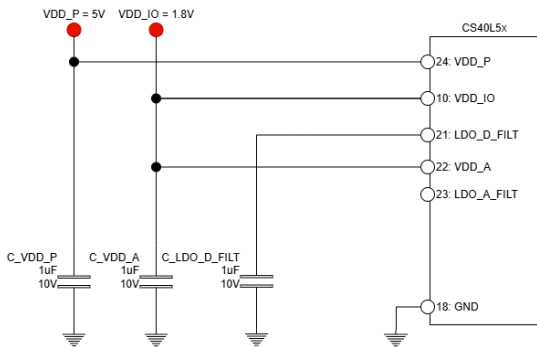


Figure 1 Power Supply Configurations

1.1.1 Low Power Rails: Device Supply

VDD_P and VDD_IO must be supplied externally. VDD_A can be supplied internally by an LDO; alternatively, VDD_A can be supplied externally. Below are shown the typical configurations.

Option 1: Single Supply	Description	VDD_P = VDD_IO = 3.3 V VDD_A = LDO_A_FILT
	Schematic	
	Software Configuration	No software configuration required

Option 2: Dual Supply	Description	VDD_P = VDD_IO = 3.3 V VDD_A = 1.8 V (external)										
	Schematic											
	Software Configuration	<table><tr><th>Operation</th><th>Address</th><th>Value</th></tr><tr><td>WRITE</td><td>0x0002A28</td><td>0x000000C0</td></tr><tr><td>WRITE</td><td>0x0002A30</td><td>0x00000004</td></tr></table> <p>These register writes must be applied during device initialization. See Section 2.1 for further details.</p>	Operation	Address	Value	WRITE	0x0002A28	0x000000C0	WRITE	0x0002A30	0x00000004	
Operation	Address	Value										
WRITE	0x0002A28	0x000000C0										
WRITE	0x0002A30	0x00000004										
Option 3: Dual Supply	Description	VDD_P = 5 V VDD_IO = VDD_A = 1.8 V (external)										
	Schematic											
	Software Configuration	<table><tr><th>Operation</th><th>Address</th><th>Value</th></tr><tr><td>WRITE</td><td>0x0002A28</td><td>0x000000C0</td></tr><tr><td>WRITE</td><td>0x0002A30</td><td>0x00000004</td></tr></table> <p>These register writes must be applied during device initialization. See Section 2.1 for further details.</p>	Operation	Address	Value	WRITE	0x0002A28	0x000000C0	WRITE	0x0002A30	0x00000004	
Operation	Address	Value										
WRITE	0x0002A28	0x000000C0										
WRITE	0x0002A30	0x00000004										

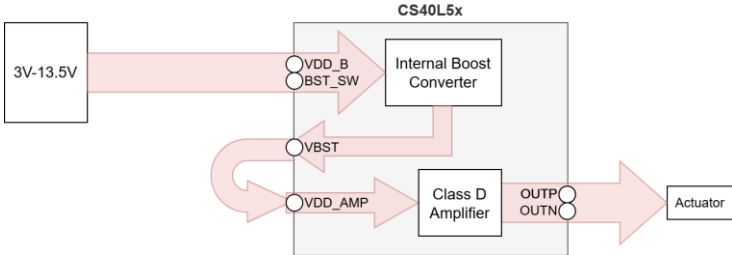
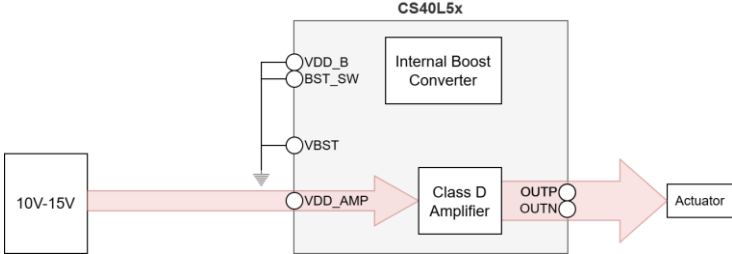
Note - All power supplies must be stable prior to release of the RESET pin. See Section 1.2.1 for more details.

1.1.2 High Power Rail: Amplifier Supply

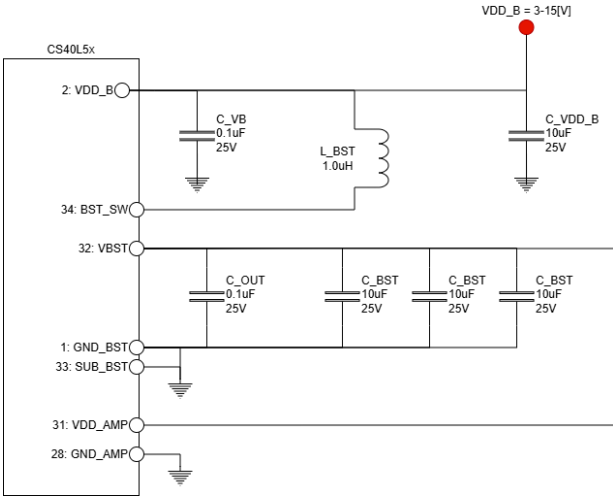
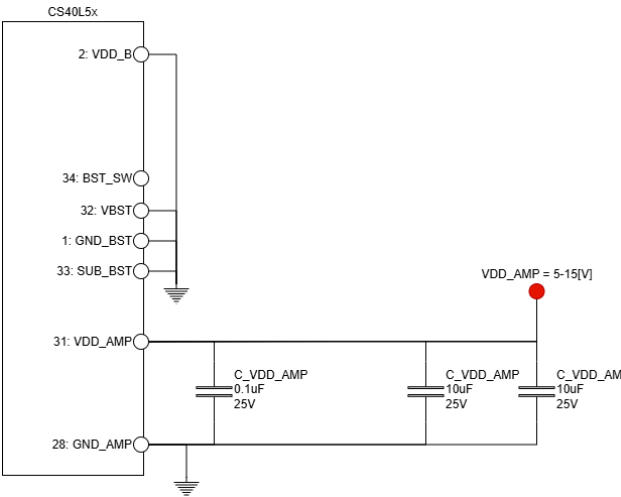
The Class D amplifier on CS40L5x can be powered internally through the boost converter, or else can be powered using an external supply. When powered internally, an external inductor (connected across VDD_B and BST_SW) forms part of the boost-converter circuit which supplies VDD_AMP. The internal boost supply supports Class-H tracking, which improves the amplifier efficiency; if an external VDD_AMP supply is used, the boost converter is bypassed.

The selection of the amplifier supply is based on the available power supply on the system. The internal boost converter is recommended when the system supply voltage VDD_B is lower. Bypassing the boost converter by using an external supply is recommended when the system supply voltage VDD_AMP is higher, or when the output haptic effect voltage is lower than the supplied voltage.

The table below describes the recommended amplifier supply with the internal boost and external amplifier supply configurations.

System Configuration	CS40L5x High Power Rail Configuration	Notes	Diagram
3 V – 15 V	Internal Boost supply - Direct power rail connection	When VBST ≤ VDD_B, the boost converter is bypassed automatically. See the notes in the next table for details of the boost-converter operating states.	
10 V – 15 V	External supply - Direct battery connection	Boost converter is disabled and bypassed	

The following table shows the recommended external components for internal/external amplifier supply configurations. The component ratings shown are typical. For further details and considerations of the external components, see the schematic and layout guidelines.

	Internal Boost Supply (VDD_B/BST_SW) [Default Configuration]	External VDD_AMP Supply									
Voltage Range	The recommended range is 3-15V.	While the supply voltage of the boost converter can be as low as 5 V, the recommendation is to use this supply configuration at higher voltages (e.g. 4S battery cell)									
Max current	4.5 A	3 A									
Schematic											
Software Configuration	The default configuration enables the internal boost setting.	To enable the external VDD_AMP supply, the CS40L5x_HAP2_Init_ExtVDDAmp.txt and the CS40L5x_HAP2_Errata_ExtVDDAmp.txt scripts from the relevant device packs should be run.									
Notes	<p>The boost converter supports transitions between active and bypass states automatically, which are dependent on the input supply VDD_B conditions and the boost output VBST configuration. Note that VBST is determined by Class H tracking, which is enabled by default.</p> <p>Active state: If the output boost voltage VBST is set to a value greater than VDD_B, the boost converter becomes active.</p> <p>Bypass state: If the output boost voltage VBST is less than or equal to VDD_B, the boost converter is bypassed automatically.</p> <table border="1" data-bbox="284 1459 893 1606"> <thead> <tr> <th>Voltage Condition</th><th colspan="2">Boost Converter State</th></tr> </thead> <tbody> <tr> <td>Input Supply Range (VDD_B and BST_SW)</td><td>VBST > VDD_B</td><td>VBST <= VDD_B</td></tr> <tr> <td>3V to 15V (1S to 3S cell)</td><td>Active state</td><td>Bypass State</td></tr> </tbody> </table>	Voltage Condition	Boost Converter State		Input Supply Range (VDD_B and BST_SW)	VBST > VDD_B	VBST <= VDD_B	3V to 15V (1S to 3S cell)	Active state	Bypass State	If the amplifier is powered using an external supply (VDD_AMP), the boost converter is disabled.
Voltage Condition	Boost Converter State										
Input Supply Range (VDD_B and BST_SW)	VBST > VDD_B	VBST <= VDD_B									
3V to 15V (1S to 3S cell)	Active state	Bypass State									

Note - Refer to the CS40L5x errata for additional configuration that may be revision-dependent.

MCU Driver Reference

File: ../cs40l5x/cs40l5x.c

Function: cs40l5x_initialize()

- Internal boost supply haptic_config.is_ext_bst = false
- External supply haptic_config.is_ext_bst = true

1.2 Operational States

The CS40L5x supports four operational states—Reset, Standby, Active, and Hibernate. The device transitions between these states to reduce power consumption, depending on the required functionality. The diagram below shows the state transitions, based on user inputs and firmware commands.

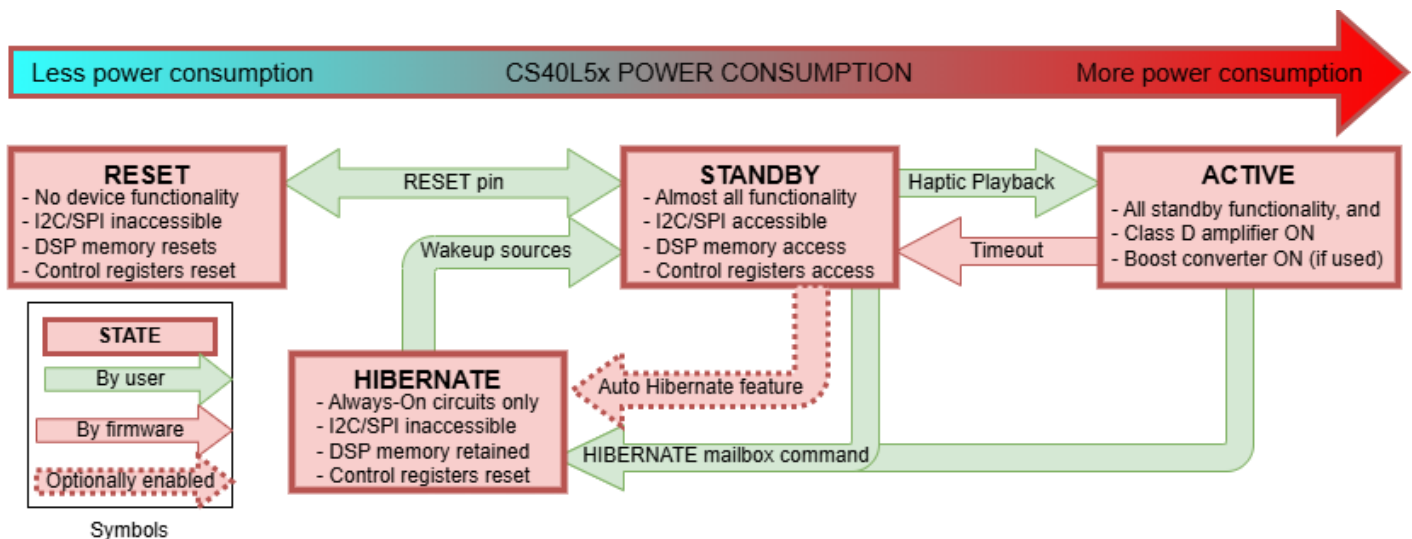


Figure 2 Operational States

MCU Driver Reference

...cs40l5x/cs40l5x.c

Function cs40l5x_power() manages the power management transitions

1.2.1 Reset State

The Reset State is the lowest power state and does not support any device functionality.

CS40L5x enters the RESET state when the RESET pin is driven low, or when power is not present. When the RESET pin is set high, the device transitions to Standby State.

Note - The CS40L5x does not have a power-supply sequence requirements, but all power supplies must be stable prior to release of the RESET pin.

The power-supply and reset timing is illustrated in the following figure.

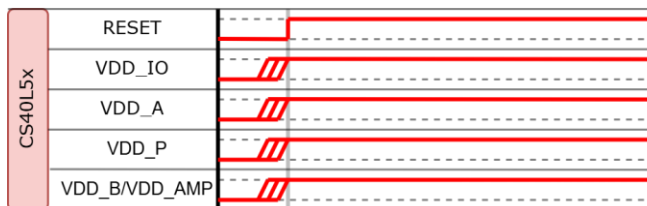


Figure 3 Power-On Reset Timing

1.2.2 Hibernate State

The Hibernate State is a low-power state that supports a minimal always-on capability while being ready to resume operation following a valid wake-up event. Almost all device functionality is disabled in the Hibernate State; only the always-on digital functions associated with wake-up are maintained. The Hibernate State provides power consumption savings when most of the device functionality is not required, but also introduces additional latency on haptic-effect triggers due to the wake-up time.

The device enters the Hibernate state when commanded via the mailbox. The device supports an optional Automatic Hibernate feature, which is enabled through the ALLOW_HIBERNATE command. The Automatic Hibernate allows the device to select the Hibernate State automatically using a timeout following any haptic output. See Section 1.2.5 below.

Mailbox command	Description	Register write sequence		
HIBERNATE	The device enters the hibernate state immediately, even if a haptic effect is playing.	HIBERNATE command:		
		Register Name	Register Address	Data
		DSP_VIRTUAL1_MBOX_1	0x0011020	0x02000001
ALLOW_HIBERNATE	The device enters the hibernate state after a user-set timeout. The default timeout is 4.88 seconds. This command enables the timer "Standby to Hibernate Timeout", which is started every time the device enters the Standby State. The PREVENT_HIBERNATE command (see Section 1.2.2.1) disables this timer.	ALLOW HIBERNATE command		
		Register Name	Register Address	Data
		DSP_VIRTUAL1_MBOX_1	0x0011020	0x02000004
		(Optional) The timer delay PM_PM_TIMER_TIMEOUT_TICKS3 is stored in the two appended 24-bit words. The appended value is the delay in units of 1/32768 seconds. The table below shows the default values for a 4.88 seconds time delay.		
		Register Name	Register Address	Default Data
		PM_TIMER_TIMEOUT_TICKS3_LSW	0x2803FF0	0x00027100
		PM_TIMER_TIMEOUT_TICKS3_MSW	0x2803FF4	0x00000000
		Value = append MSW[23:0]:LSW[23:0] Encoding: Value = Seconds * 32768 Example: 1.22s → 0x9C29 Decoding: Seconds = Value/32768 Example: 0x9C29 → 1.22 s		

Note - If Automatic Hibernate mode is enabled, the current consumption in the Standby State is increased from its typical value (~195 μ A) to approximately 1 mA while the Standby-to-Hibernate timer is active. When the timer expires, the device transitions to the Hibernate State and the current decreases to ~11 μ A. While the device is in the Hibernate State, any I2C/SPI transaction will wake up the device and restart the timer. See diagrams in Section 1.2.5.

1.2.2.1 Wakeup Sources

The CS40L5x device remains in the Hibernate State until a wakeup source is triggered as described in the table below.

Wakeup source	Description	User interaction:		
I2C/SPI transaction with address detection	Any I2C/SPI transaction targeted to the device's address wakes up the device.	Any I2C/SPI transaction (READ or WRITE) that targets the device address. Transactions on the same I2C/SPI bus that target a different address do not wake up the device.		
PREVENT_HIBERNATE mailbox command	The device enters the STANDBY state immediately and disables the ALLOW_HIBERNATE timer function.	PREVENT_HIBERNATE command		
		Register Name	Register Address	Register Data
		DSP_VIRTUAL1_MBOX_1	0x0011020	0x02000003
GPIO trigger: GPIO11 GPIO12 GPIO13 Refer to Section 3.2.2 for specific waveform assignments	If GPIO11, GPIO12, or GPIO13 is assigned to trigger a haptic effect (see Section 3.2.2), the device wakes when a valid edge is detected on the respective GPIO pin. Note that once triggered, the device transitions to active state to play the assigned waveform, and returns to standby. The device remains in the Standby State, unless the ALLOW_HIBERNATE command has been sent previously.			

Note - The first I2C/SPI transaction received while the device is in the Hibernate State wakes the device but is not acknowledged (NACK). Therefore, the host must retransmit the I2C/SPI command once the device has completed the wake-up transition (Hibernate to Standby).

1.2.3 Standby State

The Standby State is a low-power state that supports control-register access using the I2C or SPI interfaces. Read/write access to the DSP memory is also supported. The device enters the Standby State in the following conditions:

- The device is in the Reset State and the RESET pin is driven high
- The device is in the Hibernate State and a wake-up source is triggered
- The device is in the Active State, has completed playing a haptic effect, and the Active-to-Standby timer has expired

1.2.4 Active State

When the device is in Active State, all functional blocks are supported. Note that the boost converter, Class D amplifier, ASP interface, and other functions required for haptic output are only supported in the Active State. The device transitions to Active state when playing from the ASP I2S interface, or when a haptic effect is triggered (from GPIO or I2C/SPI, see Section 3). After the device has completed playing an effect, the device enters the Standby State after the Active-to-Standby user-configurable timer expires.

The Active to Standby timer delay PM_PM_TIMER_TIMEOUT_TICKS4 is stored in the two appended 24-bit words. This appended value is the delay in units of 1/32768 seconds. The table below shows the default values for a 244 ms delay.

Register Name	Register Address	Default Data	Register Data
PM_PM_TIMER_TIMEOUT_TICKS4_LSW	0x2803FF8	0x00001F40	Value = append MSW[23:0]:LSW[23:0] Encoding: Value = Seconds * 32768 Example: 1.22s → 0x9C29 Decoding: Seconds = Value/32768 Example: 0x9C29 → 1.22 s
PM_PM_TIMER_TIMEOUT_TICKS4_MSW	0x2803FFC	0x00000000	

1.2.5 Operating State Transition examples

Example state transitions - without the hibernation timer enabled - are illustrated in the following figure.

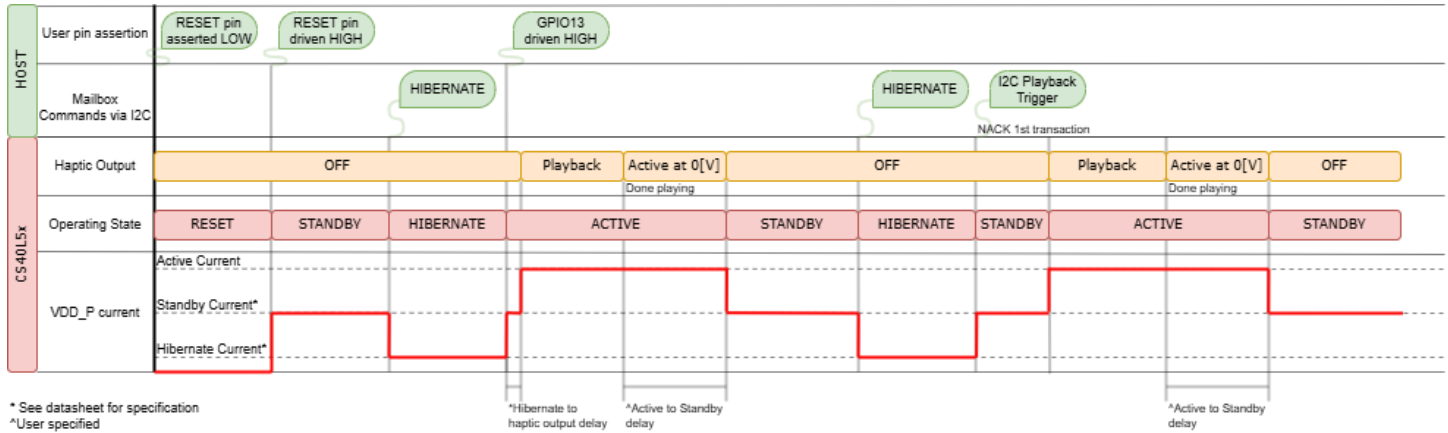


Figure 4 Operating State Transitions with Automatic Hibernation Disabled

Example state transitions - without the hibernation timer enabled - are illustrated in the following figure.

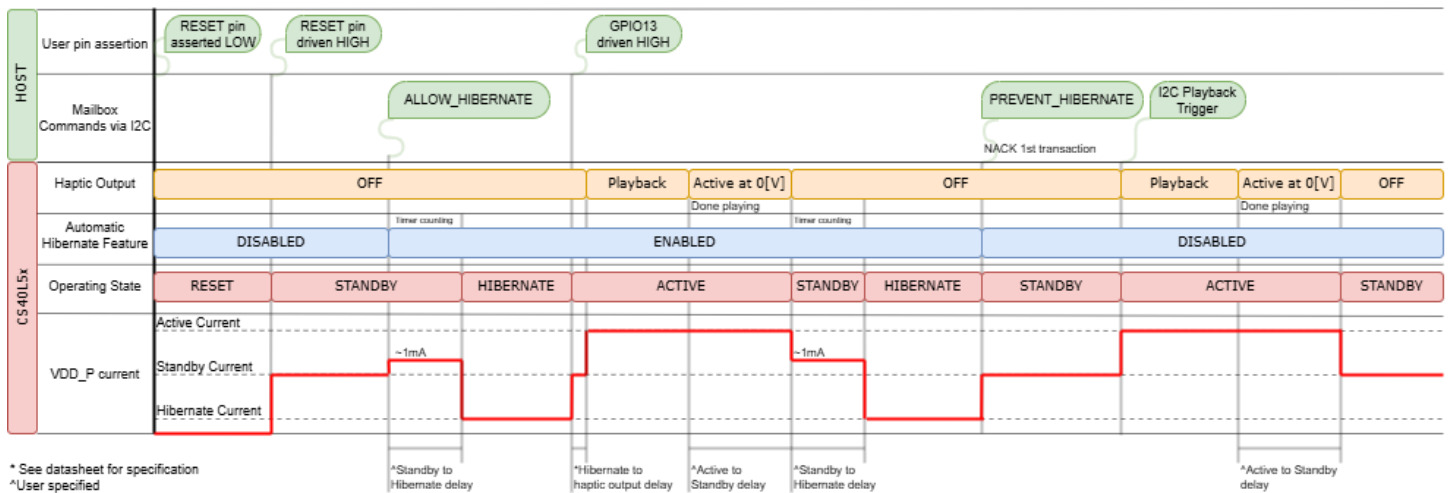


Figure 5 Operating State Transitions with Automatic Hibernation Enabled

2 Firmware

2.1 Initialization Sequence

When the CS40L5x transitions from the Reset State to the Standby State, the CS40L5x is initialized with default hardware configurations, and the ROM firmware is initialized with default firmware configurations. The initialization sequence duration is typically about 3 ms.

Once CS40L5x indicates the default initialization sequence is done through the [Firmware Interrupt "AWAKE"](#), the user can proceed loading its configuration sequence. The following configuration sequence is recommended: 1) [power supply configurations](#), 2) errata fixes, 3) other configurations such as [interrupt](#) and [device features](#) and 4) [RAM wavetable](#) loading.

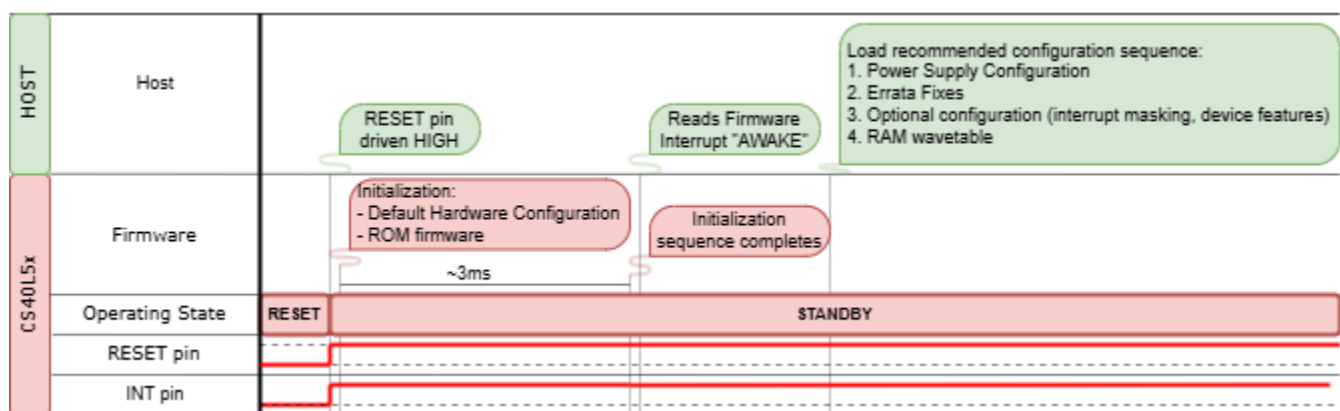


Figure 6 Device Initialization Sequence

2.1.1 ROM Firmware

The CS40L5x ROM firmware contains featured algorithms including [Calibration](#), [Click Compensation](#), [Dynamic F0](#), and [Run Time Haptics](#). It also contains predefined ROM waveforms, which are described in detail in Section 3.1.1.

2.1.2 RAM Firmware

CS40L5x RAM firmware may be developed to support new features, bug fixes, and patches. The CS40L5x ROM firmware is loaded automatically after a hardware reset; RAM firmware needs to be downloaded by the host processor after a hardware reset. The CS40L5x provides up to 24 kB memory available for a RAM firmware; the host system must provide sufficient memory storage for RAM firmware if needed. The CS40L51/52/53 devices support different algorithm features as described below:

2.2 Interrupt

The CS40L5x device provides an interrupt pin, INT, to alert the host to significant events. The INT pin is normally high (Logic 1) and is driven low (Logic 0) when a defined interrupt event occurs. The interrupt handling by the CS40L5x and the host is illustrated in the figure below.

On CS40L5x, each bit of the IRQ1_INT_X registers represents an interrupt event. When an interrupt event is detected, the corresponding bit IRQ1_INT_X[b] is set to 1. If the interrupt is unmasked (i.e. IRQ1_MASK_X[b]=0), the INT pin is asserted low. Interrupt masking is handled by the host and performed at initialization. If the host detects a falling edge on the INT pin, it should read the unmasked interrupt registers bits (IRQ1_INT_X[b]) to determine the source of the interrupt. To clear the interrupt signal, the host must write 1 to the respective interrupt bit.

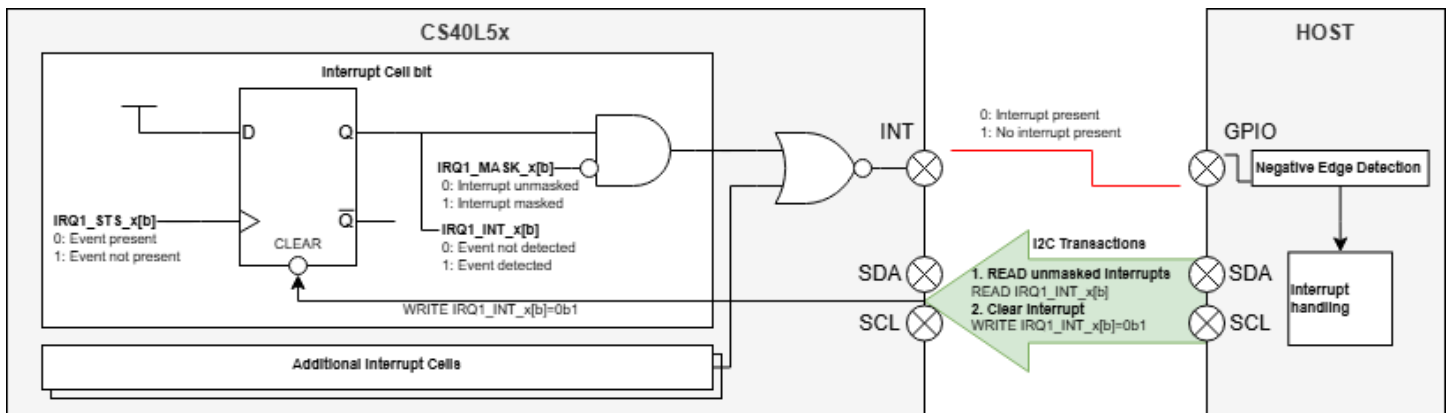


Figure 7 Interrupt Handling

The CS40L5x supports two types of interrupts: Hardware interrupts, and Firmware interrupts. Both types may trigger the INT pin when unmasked. While Hardware Interrupts are contained within interrupt cells only, Firmware Interrupts are contained on a single interrupt cell and accessed through the DSP mailbox.

Note - All hardware interrupts are masked by default. A subset of the firmware interrupts are unmasked (see Section 2.2.2 for details). The user can unmask specific interrupts by writing a 0 to the corresponding mask bit; this should be done during initialization.

2.2.1 Hardware Interrupts

Hardware interrupts are handled as shown in the following figure.

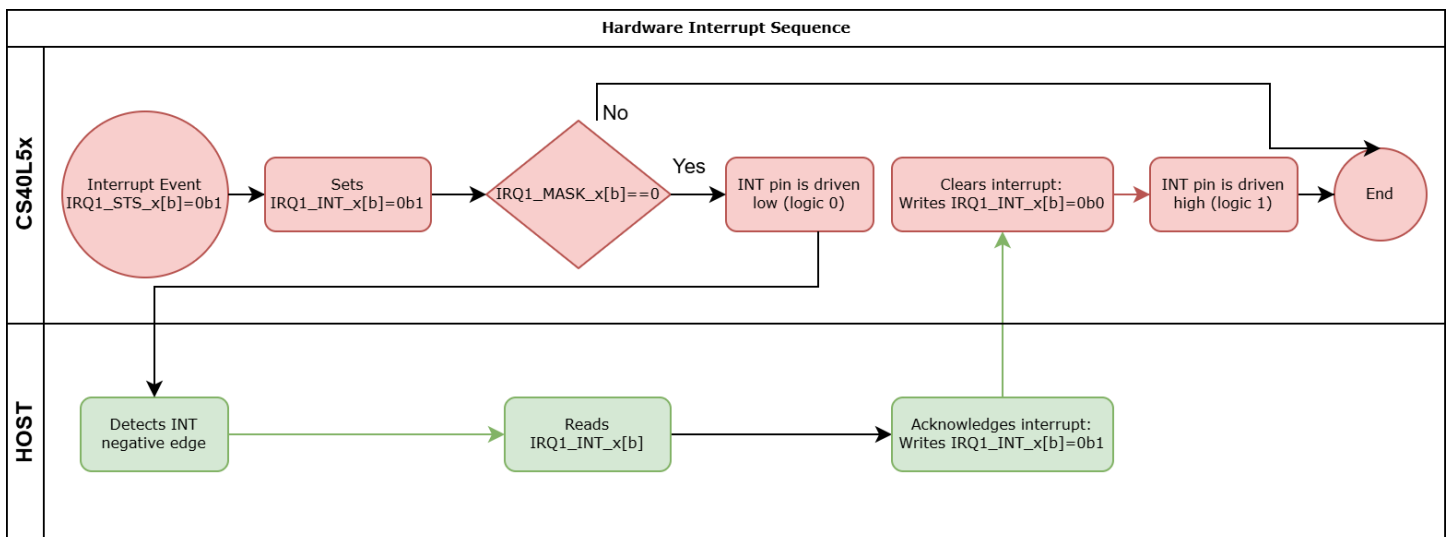


Figure 8 Hardware Interrupt Sequence

The hardware interrupt registers are identified in the following table.

Table 1 Hardware Interrupt Registers

Name	Address Range	Description	Type
IRQ1_INT_X	0x0000E010 - 0x0000E058	Each register bit indicates if an interrupt event has been detected	Read/Write
IRQ1_MASK_x	0x0000E090 - 0x0000E0D8	Each register bit indicates if an interrupt event is masked. Note that only the unmasked interrupts trigger the INT pin output.	Read/Write

The table below shows the most common use cases of hardware interrupts. For a complete list of interrupt event registers and the mask registers, refer to Section 7.12 in the CS40L5x datasheet.

Table 2 Hardware Interrupt Bits

Register name [bit]	Register Address (bit)	Interrupt Name	Description
IRQ1_INT_1 [31]	0x0000E010 (bit 31)	AMP_SHORT_ERR_INT1	Amplifier short error - triggered if an overcurrent (short circuit) is detected at the amplifier output
IRQ1_INT_8 [31]	0x0000E02C (bit 31)	TEMP_ERR_INT1	Overtemperature error - triggered if the die temperature exceeds the error threshold (~125°C)
IRQ1_INT_9 [8]	0x0000E030 (bit 8)	BST_ILIMIT_ERR_INT1	Boost current limit engagement - triggered if the average inductor current exceeds the limit threshold (~4 A) and the boost current limit is applied
IRQ1_INT_9 [7]	0x0000E030 (bit 7)	BST_SHORT_ERR_INT1	Boost Inductor short error - triggered if the boost overcurrent (inductor short-circuit) error is detected
IRQ1_INT_9 [6]	0x0000E030 (bit 6)	BST_UVP_ERR_INT1	Boost undervoltage error - triggered if the boost undervoltage error is detected, i.e., the output voltage (VBST) falls below the undervoltage threshold (~VDD_B x 1.1)
IRQ1_INT_10 [16]	0x0000E034 (bit 16)	UVLO_VddbATT_ERR_INT1	VDD_B UVLO error - triggered if VDD_B falls below the undervoltage lockout (UVLO) threshold
IRQ1_INT_18 [15]	0x0000E054 (bit 15)	GLOBAL_ERR_INT1	Global error - triggered when there is a critical error that places the device in the Global Error State to protect the device from permanent damage - Refer to Section 4.8.5 in the CS40L5x datasheet.

An example control sequence for a hardware interrupt is shown in the Figure below. The example assumes a haptic waveform which, if triggered, would require more current than the boost current limit. The host unmask the boost current limit interrupt and plays the waveform. The CS40L5x attempts to generate the waveform but, when the boost current reaches its limit, the waveform playback is stopped and the INT pin is driven low (Logic 0). When the host acknowledges the interrupt, the CS40L5x clears the interrupt and drives the INT pin high (Logic 1).

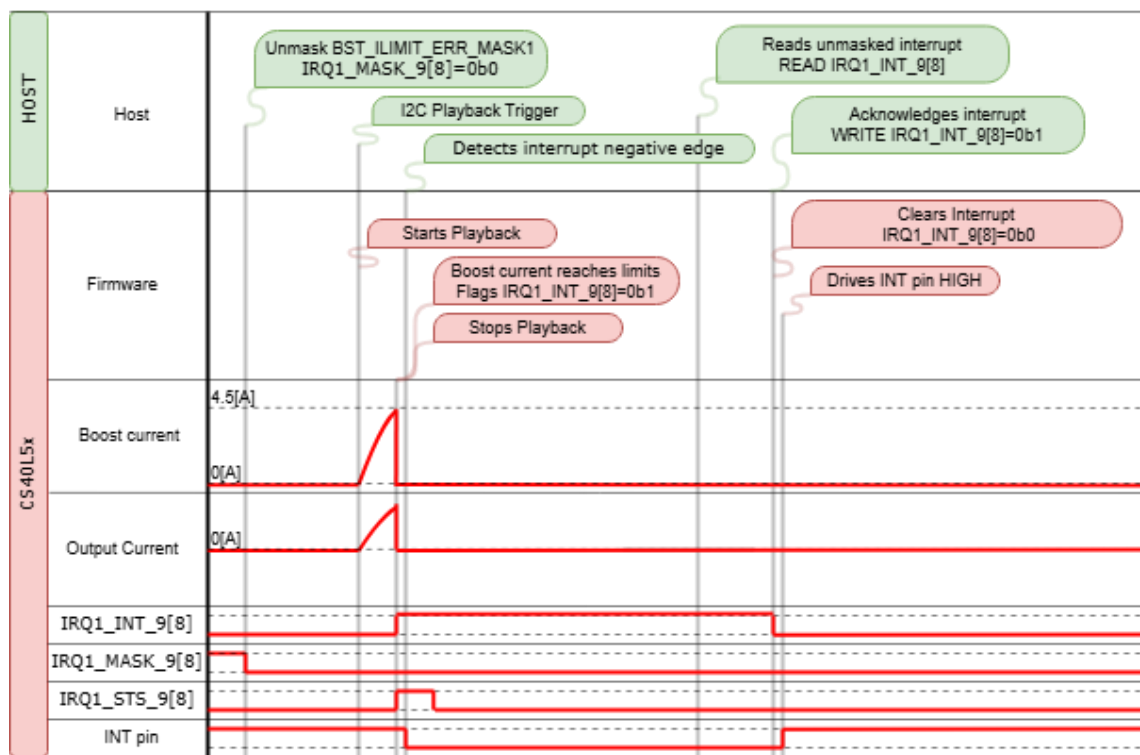


Figure 9 Boost Current Limit Interrupt Example

2.2.2 Firmware Interrupts

When a firmware interrupt event occurs, the event code is written into the DSP mailbox registers. The host reads the mailbox registers, updates the pointer on the device, and acknowledges the interrupt event via register write.

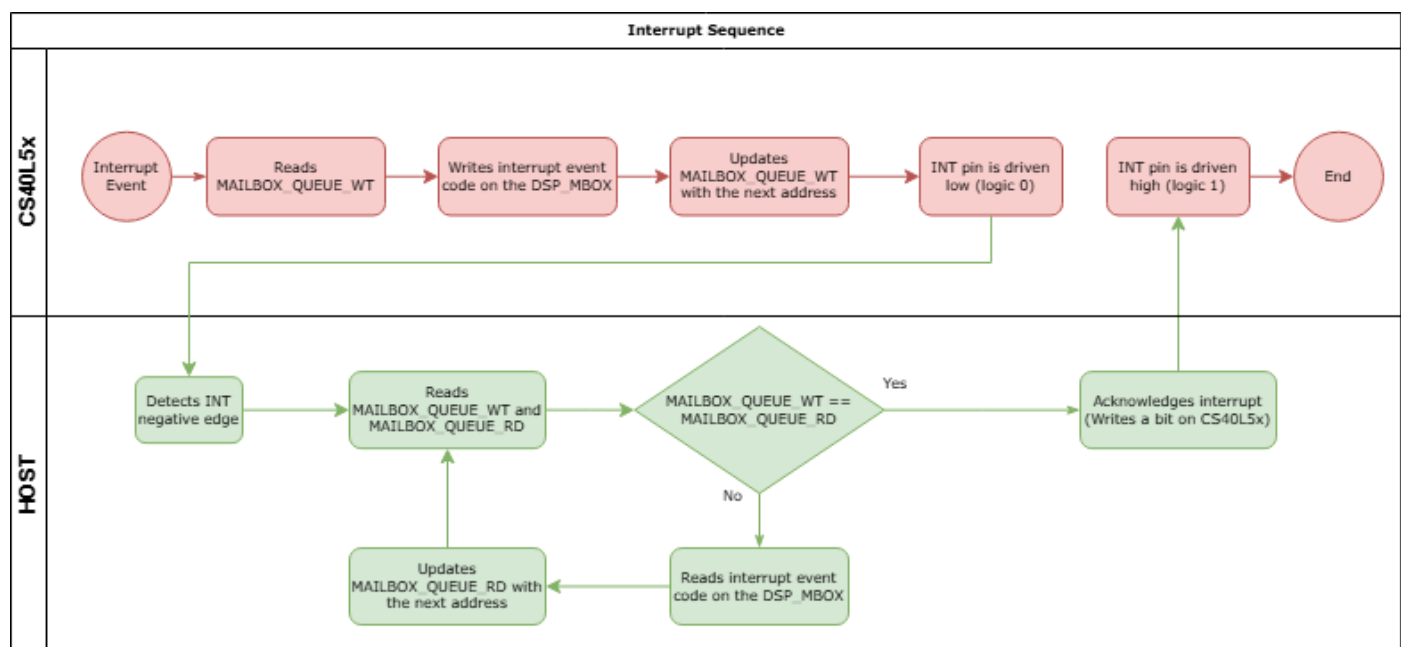


Figure 10 Firmware Interrupt Sequence

The following table describes the host interactions with the mailbox queue.

Mailbox Circular Buffer	To find the register mailbox containing the interrupt source, the host reads the MAILBOX_QUEUE_RD register and updates it with the next unread address from the buffer.																														
	<table><tr><th>Register Name</th><th>Address</th><th>Range</th><th>Description</th><th>Owner</th></tr><tr><td>MAILBOX_QUEUE_RD</td><td>0x28042CC</td><td>0x00011004 - 0x0001101C</td><td>Address of oldest unread element in queue.</td><td>Host</td></tr><tr><td>MAILBOX_QUEUE_WT</td><td>0x28042CC</td><td>0x00011004 - 0x0001101C</td><td>Address of next empty element in queue.</td><td>CS40L5x</td></tr></table>	Register Name	Address	Range	Description	Owner	MAILBOX_QUEUE_RD	0x28042CC	0x00011004 - 0x0001101C	Address of oldest unread element in queue.	Host	MAILBOX_QUEUE_WT	0x28042CC	0x00011004 - 0x0001101C	Address of next empty element in queue.	CS40L5x															
	Register Name	Address	Range	Description	Owner																										
	MAILBOX_QUEUE_RD	0x28042CC	0x00011004 - 0x0001101C	Address of oldest unread element in queue.	Host																										
	MAILBOX_QUEUE_WT	0x28042CC	0x00011004 - 0x0001101C	Address of next empty element in queue.	CS40L5x																										
Interrupt events are stored in DSP mailbox registers in a circular buffer approach.																															
<div><table><tr><th>Name</th><th>Address</th><th>Value</th></tr><tr><td>MAILBOX_QUEUE_RD</td><td>0x2830FC8h</td><td>Unread DSP_MBOX Address</td></tr><tr><td>MAILBOX_QUEUE_WT</td><td>0x2830FC4h</td><td>Next DSP_MBOX Address</td></tr></table><div><div>Points to</div><div>Points to</div><div><div>DSP_MBOX address Circular buffer</div><div>0x0011004</div><div>0x0011008</div><div>0x001100C</div><div>0x0011010</div><div>0x0011014</div><div>0x0011018</div><div>0x001101C</div></div></div></div>				Name	Address	Value	MAILBOX_QUEUE_RD	0x2830FC8h	Unread DSP_MBOX Address	MAILBOX_QUEUE_WT	0x2830FC4h	Next DSP_MBOX Address																			
Name	Address	Value																													
MAILBOX_QUEUE_RD	0x2830FC8h	Unread DSP_MBOX Address																													
MAILBOX_QUEUE_WT	0x2830FC4h	Next DSP_MBOX Address																													
Note that the host should read the mailbox and update the queue pointers as fast as possible to avoid mailbox overflow (MAILBOX_STATUS 0x2803FCC = 0x00000006).																															
Interrupt codes	The following interrupt codes are written to the DSP_MBOX mailbox addresses																														
	<table><tr><th>Interrupt Name</th><th>Code</th><th>Description</th></tr><tr><td>HAPTIC_COMPLETE_MBOX</td><td>0x01000000</td><td>End of haptic waveform playback from a mailbox trigger source</td></tr><tr><td>HAPTIC_COMPLETE_GPIO</td><td>0x01000001</td><td>End of haptic waveform playback from a GPI trigger source</td></tr><tr><td>HAPTIC_TRIGGER_MBOX</td><td>0x01000010</td><td>Start of a mailbox playback</td></tr><tr><td>HAPTIC_TRIGGER_GPIO</td><td>0x01000011</td><td>Start of a GPI playback</td></tr><tr><td>INIT</td><td>0x02000000</td><td>Device transitions from reset to standby</td></tr><tr><td>AWAKE</td><td>0x02000002</td><td>Device transitions from hibernate to standby</td></tr><tr><td>PERMANENT_SHORT_DETECTED</td><td>0x0C000C1C</td><td>A short circuit was detected at the amplifier output before playback</td></tr><tr><td>RUNTIME_SHORT_DETECTED</td><td>0x0C000C1D</td><td>A short circuit was detected at the amplifier output during playback</td></tr></table>			Interrupt Name	Code	Description	HAPTIC_COMPLETE_MBOX	0x01000000	End of haptic waveform playback from a mailbox trigger source	HAPTIC_COMPLETE_GPIO	0x01000001	End of haptic waveform playback from a GPI trigger source	HAPTIC_TRIGGER_MBOX	0x01000010	Start of a mailbox playback	HAPTIC_TRIGGER_GPIO	0x01000011	Start of a GPI playback	INIT	0x02000000	Device transitions from reset to standby	AWAKE	0x02000002	Device transitions from hibernate to standby	PERMANENT_SHORT_DETECTED	0x0C000C1C	A short circuit was detected at the amplifier output before playback	RUNTIME_SHORT_DETECTED	0x0C000C1D	A short circuit was detected at the amplifier output during playback	
	Interrupt Name	Code	Description																												
	HAPTIC_COMPLETE_MBOX	0x01000000	End of haptic waveform playback from a mailbox trigger source																												
	HAPTIC_COMPLETE_GPIO	0x01000001	End of haptic waveform playback from a GPI trigger source																												
	HAPTIC_TRIGGER_MBOX	0x01000010	Start of a mailbox playback																												
	HAPTIC_TRIGGER_GPIO	0x01000011	Start of a GPI playback																												
	INIT	0x02000000	Device transitions from reset to standby																												
	AWAKE	0x02000002	Device transitions from hibernate to standby																												
	PERMANENT_SHORT_DETECTED	0x0C000C1C	A short circuit was detected at the amplifier output before playback																												
RUNTIME_SHORT_DETECTED	0x0C000C1D	A short circuit was detected at the amplifier output during playback																													
Note that interrupt codes not listed in the table above are reserved. Contact your Cirrus Logic representative for more information.																															
Acknowledge (by Host)	To acknowledge the interrupt, the host writes a 0b1 to DSP_VIRTUAL2_MBOX_WR_INT1. Then, CS40L5x drives the interrupt pin high (logic 1)																														
	<table><tr><th>Name</th><th>Address</th><th>Value</th><th>Notes</th></tr><tr><td>IRQ1_INT_2</td><td>0x000E014</td><td>0x00200000</td><td>Write 0b1 to DSP_VIRTUAL2_MBOX_WR_INT1 (IRQ_INT_2[21])</td></tr></table>			Name	Address	Value	Notes	IRQ1_INT_2	0x000E014	0x00200000	Write 0b1 to DSP_VIRTUAL2_MBOX_WR_INT1 (IRQ_INT_2[21])																				
Name	Address	Value	Notes																												
IRQ1_INT_2	0x000E014	0x00200000	Write 0b1 to DSP_VIRTUAL2_MBOX_WR_INT1 (IRQ_INT_2[21])																												

	Host	Firmware	Haptic Output	MAILBOX_QUEUE_RD	MAILBOX_QUEUE_WT	DSP_MBOX_2	DSP_MBOX_3	INT pin
I2C Playback Trigger								
Detects INT pin negative edge								
Reads MAILBOX_QUEUE								
Reads DSP_MBOX_2								
Updates MAILBOX_QUEUE_RD								
Acknowledges interrupt WRITE IRQ1_INT_2[21]=0b1								
Detects INT pin negative edge								
Reads MAILBOX_QUEUE								
Reads DSP_MBOX_3								
Updates MAILBOX_QUEUE_RD								
Acknowledges interrupt WRITE IRQ1_INT_2[21]=0b1								
Starts Playback								
Updates MAILBOX_QUEUE_WT								
Drives INT pin LOW								
Clears Interrupt WRITE IRQ1_INT_2[21]=0b0								
Drives INT pin HIGH								
End of Playback								
Updates MAILBOX_QUEUE_WT								
Drives INT pin LOW								
Clears Interrupt WRITE IRQ1_INT_2[21]=0b0								
Drives INT pin HIGH								
MAILBOX_QUEUE_RD								
MAILBOX_QUEUE_WT								
DSP_MBOX_2								
DSP_MBOX_3								
INT pin								

3 Haptic Playback

Host

I2C triggering
Through DSP_MAILBOX

GPIO Triggering:

- GPIO3 - GPIO10
- GPIO4 - GPIO11*
- GPIO5 - GPIO12*
- GPIO6 - GPIO13*

* also available in hibernate state

CS40L5x

ROM Wavetable

0	ROM0
1	ROM1
...	ROM#
26	ROM26

RAM Wavetable

0	RAM0
1	RAM1
...	RAM#
127	RAM127

RTH




Haptic Effect from Wavetable

Actuator

AN0680R2

3.1 Haptic Wavetables

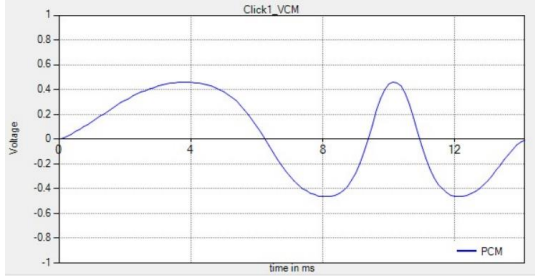
Haptic effects stored in wavetables are indexed in the formats shown in the table below. These effects are developed using the SCS software haptic-control utility.

Type	Description	Example
PCM	<p>A PCM waveform specifies the instantaneous voltage over time as a pulse-code modulated (PCM) signal. These waveforms are useful for clicks, and for other short, finely tuned effects.</p> <p>User defined parameters (optional, not required):</p> <p>F0: Resonant frequency</p> <p>ReDC: Actuator resistance</p>	<p>Example 160 Hz PCM:</p> 
PWLE	<p>A piecewise linear envelope (PWLE) specifies the amplitude envelope of a variable-frequency sine wave over time. These waveforms are useful for long-duration, low-resolution effects, such as buzzes. They are composed of sections in which the following parameters are specified:</p> <p>Level (Full Scale)</p> <p>Duration (ms)</p> <p>Frequency (Hz)</p> <p>Notes:</p> <p>For section 1, only the level is user defined. The duration of section 1 is always 0.</p> <p>A section of duration 0 is required when changing the level instantaneously.</p>	<p>PWLE defined in SCS:</p> 
Ringtone	<p>Ringtones are specified as a series of PCM / PWLE effects to play in series, with inserted delays, and changes of amplitude as needed. These waveforms are useful for creating long effects which combine or repeat other effects.</p>	<p>Ringtone using PilotTone1_160Hz as index 1 and PWLE above as index 2</p> 

CS4045x supports playback from predefined waveforms stored in ROM. Additionally, the application-specific waveforms can be created and loaded into the RAM wavetable. Runtime Haptics (RTH) also allows the playback of effects with parameters specified by the user.

3.1.1 ROM Waveform

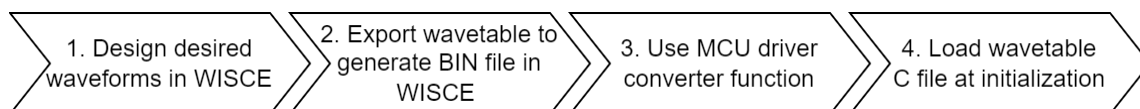
The ROM waveform below can be used to verify software and hardware functionality during development. To trigger the ROM waveform, see Section 3.2.

Waveform Name	Waveform Type	Duration (ms)	F0 (Hz)	ReDC (Ohms)	Image
Click1_VCM	Click	14.25	80	10	

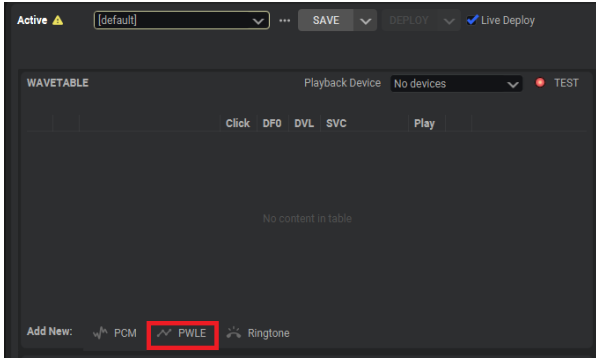
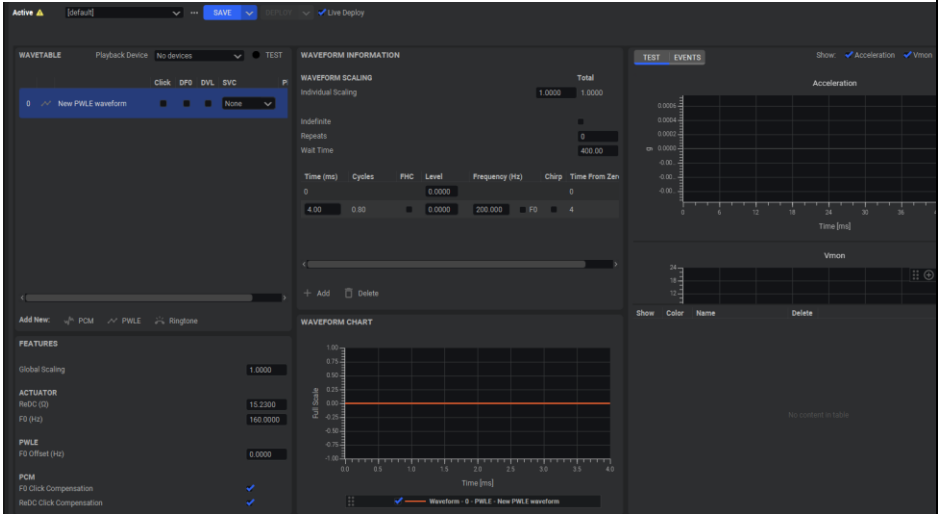
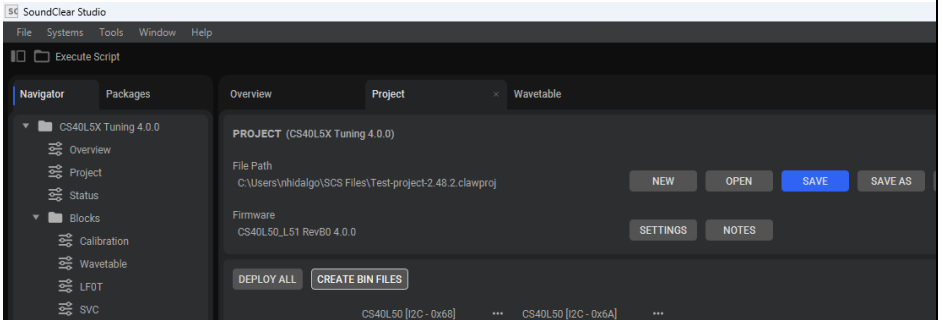
3.1.2 RAM Wavetable

The RAM wavetable allows users to load applications-specific waveforms during CS40L5x initialization, to be used at a later time. The CS40L5x device has 11,994 bytes available to store these waveforms in RAM. Note that each RAM waveform contains a waveform header (15 bytes) and waveform data.

The user can create waveforms (PCM, PWLE and Ringtone) and store them as a wavetable *.bin* file using the Cirrus Logic SCS Tool. This wavetable bin is converted to *.c/h* files using the *firmware_converter.py* tool in the Cirrus Logic MCU Driver code to be included in the user's MCU code.



The procedure to create application-specific waveforms using the SCS software tool is described in the following table.

Step	Description	Example
1	Design desired waveforms in SCS.	<p>Refer to the CS40L5x board user guide for the platforms and software used in this example. The figure below shows the steps and parameters used to create a PWLE.</p>  
2	Export wavetable to generate a BIN file in SCS.	<p>To export the preset file, navigate to the project tab. Then click on the "Create bin files" button. Name the file "CS40L5x-WT" and choose Binary Wavetable File (*.bin) file type.</p> 

Step	Description	Example										
3	<p>Refer to the MCU Driver Quick Start Guide to obtain the Firmware Converter Tool. This tool creates .c and .h files.</p> <p>Save the Wavetable BIN file in the same directory as the firmware converter and run the firmware converter tool</p> <div><p>MCU Driver Reference File</p><p>path: ..\tools\firmware_converter</p><p>File: firmware_converter.py</p></div>	<p>Run the firmware_converter.py tool from the Windows PowerShell</p> <pre>PS C:\MCU-Drivers\tools\firmware_converter> python.exe .\firmware_converter.py fw_img_v1 cs40l52 ROM --wmdr .\CS40L52-WT.bin --wmdr-only firmware_converter Convert from WMFW/WMDR ("BIN") Files to C Header/Source SDK version 4.27.2 - d201cda89044f25be78a00605a197f376226234 Command: fw_img_v1 Part Number: cs40l52 WMFW Path: ROM WMDR Path: .\CS40L52-WT.bin No suffix Input Symbol ID Header: None Generated Symbol ID Header cs40l52_sym.h Exported to files: cs40l52_fw_img.h cs40l52_fw_img.c Exit. PS C:\MCU-Drivers\tools\firmware_converter></pre>										
4	<p>Include the newly created "cs40l5x_fw_img.c", "cs40l5x_fw_img.h" and "cs40l5x_sym.h" to the project.</p>	<p>The "makefile" script on the MCU driver sample application runs the firmware_converter.py and includes the .c and .h files on the initialization sequence. Refer to the MCU Driver Quick Start Guide for more details.</p> <div><p>MCU Driver Reference</p><p>File path: ..\cs40l5x</p><p>File: bsp_cs40l5x.c</p><p>Function: bsp_dut_boot(void)</p></div>										
5	<p>Play the RAM waveform via MAILBOX. Refer to Section 3.2.1 for details.</p>	<table><tr><th>Operation</th><th>Register Name</th><th>Address</th><th>Value</th><th>Notes</th></tr><tr><td>WRITE</td><td>DSP_VIRTUAL1_MBOX_1</td><td>0x0011020</td><td>0x01000001</td><td>Play RAM waveform index 1</td></tr></table> 	Operation	Register Name	Address	Value	Notes	WRITE	DSP_VIRTUAL1_MBOX_1	0x0011020	0x01000001	Play RAM waveform index 1
Operation	Register Name	Address	Value	Notes								
WRITE	DSP_VIRTUAL1_MBOX_1	0x0011020	0x01000001	Play RAM waveform index 1								

3.1.3 Runtime Haptics Waveforms

The RAM and ROM waveforms must be defined and loaded at initialization. Runtime Haptics (RTH) waveforms can be created, loaded and played at runtime. Refer to Section 4.4 for details.

3.2 Triggering Options

There are two ways to trigger a haptic effect on CS40L5x - via I2C/SPI mailbox trigger and GPI triggering.

3.2.1 I²C/SPI Mailbox Trigger

To trigger a haptic effect from an I²C/SPI transaction, the corresponding mailbox command and wavetable index are written to the mailbox inbound address 0x0011020.

Mailbox Command	Value	Function	Example	
ROM WAVEFORM	0x0180000A	ROM waveform playback.	Example 1: Play ROM waveform	
			Address	Value
			0x0011020	0x0180000A
RAM WAVEFORM	0x01000000 - 0x0100007F	RAM wavetable playback. The last 2 digits correspond to the RAM wavetable index.	Example 1: Play RAM15 waveform (0x0F)	
			Address	Value
			0x0011020	0x0100000F
			Example 2: Play RAM35 waveform (0x23)	
			Address	Value
0x0011020	0x01000023			
RTH WAVEFORM	0x01400000 - 0x01400001	RTH wavetable playback. The last digit corresponds to the RTH wavetable index. There are only 2 indexes supported in RTH wavetable (index 0 and 1).	Example 1: Play RTH0 waveform (0x00)	
			Address	Value
			0x0011020	0x01400000
			Example 2: Play RTH1 waveform (0x01)	
Address	Value			
0x0011020	0x01400001			

MCU Driver Reference

File: .../cs40l5x/cs40l5x.c

Function: cs40l5x_trigger()

3.2.2 GPI Trigger


A GPIO rising or falling edge can be configured to trigger haptic effects from ROM wavetable, RAM wavetable or Runtime Haptics (RTH) waveforms. This configuration is stored in GPIO event registers and can be changed after reset.

The GPIO Event register format is defined as follows:

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	Reserved (0x00)							RTH Waveform 0=ROM/RAM 1=RTH	Reserved (0x00)				Attenuation 0 = 0dB 1 = -1dB 2 = -2dB 3 = -3dB 4 = -4dB 5 = -5dB 6 = -6dB 7 = -7dB	ROM/RAM TABLE 0=ROM 1=RAM (Ignored for RTH effects)	PLAYBACK INDEX 0x00–0x7F = Waveform Index 0x0-0x01 = RTH Waveform									

Note - Setting a GPIO Event register to 0x000001FF disables the corresponding GPIO edge as a source of haptic effect triggering. Specifically for GPIO11-GPIO13, disabling a haptic effect also disables the GPI triggers as wakeup sources after the next hibernate-wake cycle.

An example configuration for GPIO13 is described below.

Intended haptic effect	Register write		Output
GPIO13 rising edge plays ROM waveform 10 at -7dB. GPIO13 falling edge has no effect	Address	Value	
	0x02804178	0x00000E0A	
	0x0280417C	0x000001FF	

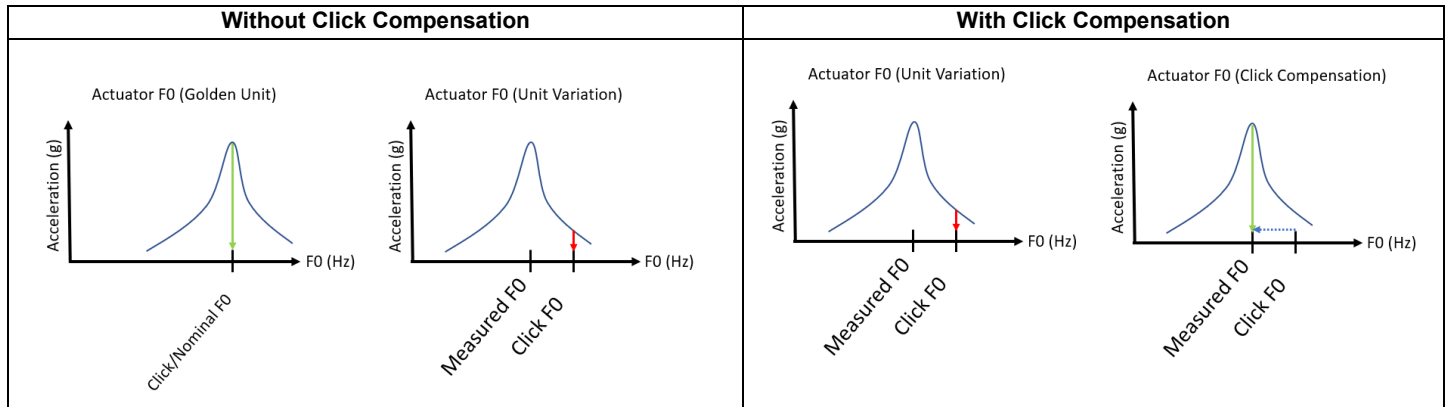
4 Device Features (Algorithms)

CS40L5x is haptic amplifier that supports waveform playback and also integrates advanced algorithms and features to improve the user experience. The [Click Compensation](#) feature enables stronger haptic effects, modifying the nominal parameters of a designed waveform (such as the resonance frequency, F0, and actuator resistance, ReDC,) to compensate for actuator variations arising from changes in temperature, voltage, and manufacturing tolerances. Part-to-part parameters are estimated through algorithms such as [Calibration](#), which happens at the production line. The [Dynamic F0](#) algorithm allows for temperature and voltage variations by estimating F0 every time a waveform is played.

Feature	Description	CS40L51	CS40L52	CS40L53
Calibration (Re and F0)	Algorithm that allows measurement of the haptic actuator electrical parameters like Fo and Re.	Yes	Yes	Yes
Click compensation (Re and F0)	Algorithm that automatically adjusts the click waveform performance to compensate for changes in the Re and F0.	Yes	Yes	Yes
Haptics synthesizer	Haptics playback engine that enables generation of haptics effects.	Yes	Yes	Yes
Runtime haptics (RTH)	Dynamic haptic waveform generation with minimized delays.	Yes	Yes	Yes
Load diagnostics	Algorithm that allows to check the status of the load connected to the output of the haptics amplifier.	Yes	Yes	—
Closed-loop algorithms (SVC)	Algorithm that enables direct control of the actuator back EMF/acceleration in real-time.	Yes	Yes	—
Audio to haptics (A2H)	Algorithm that automatically generate haptics effects from input audio content.	Yes	—	—
Live F0 tracking (LF0T)	Realtime resonance frequency tracking and playback.	Yes	—	—
Active vibration compensation	Algorithm that actively compensates for external vibration conditions by adjusting the haptics effect playback.	Yes	—	—
Wavetable sample rate	Sample rate supported for PCM waveforms in the wavetable.	8kHz, 24 kHz, 48kHz	8kHz, 24kHz	8kHz

4.1 Click Compensation

When a waveform is designed for an actuator, it is designed with the nominal resonant frequency (F0) of the actuator in mind. This results in haptic effects that work well when the F0 of the actuator is close to the nominal F0. However, either due to manufacturing tolerance, or temperature, the actuator F0 might be far enough away from the nominal F0 to cause ringing or loss in acceleration strength. As the actuator F0 moves away from the designed click F0, acceleration strength and braking performance is lost. The CS40L5x Click Compensation feature accounts for this shift by modifying the waveform by a factor that depends on the calibrated F0 and ReDC, and the nominal F0 and ReDC of the waveform. The user can expect a stronger haptic effect when F0 compensation is enabled. The ReDC compensation produces a less noticeable user effect than the F0 compensation, but is recommended for improved power consumption.



Click compensation is calculated by taking the following ratios:

$$\text{Frequency}_{\text{factor}} = \frac{\text{VIBEGEN_F0_OTP_STORED}}{\text{F0_WAVEFORM}} \quad \text{Amplitude}_{\text{factor}} = \frac{\text{VIBEGEN_REDC_OTP_STORED}}{\text{REDC_WAVEFORM}}$$

Where:

- VIBEGEN_F0_OTP_STORED is the nominal F0 of the actuator stored in CS40L5x. This value can be obtained from F0 estimation in [Calibration](#), or manually set by the user
- F0_WAVEFORM is the nominal F0 of the particular waveform
- VIBEGEN_REDC_OTP_STORED is the nominal ReDC of the actuator stored in CS40L5x. This value can be obtained from ReDC estimation in [Calibration](#), or manually set by the user
- REDC_WAVEFORM is the nominal ReDC of the particular waveform

The table below shows the format for the VIBEGEN_x_OTP_STORED fields.

Name	Address (may change if a RAM FW is used)	Default Value	Description
VIBEGEN_F0_OTP_STORED	0x2805C00	0x003C0000	Format: Q10.14 Default: 240Hz <div style="background-color: #ffffcc; padding: 10px; margin-top: 10px;"> When Dynamic F0 is enabled, the estimated Dynamic F0 value will be used for Click Compensation and the VIBEGEN_F0_OTP_STORED is ignored. See Section 4.3 for details. </div>
VIBEGEN_REDC_OTP_STORED	0x2805C04	0x0001D000	Format: Q7.17 $\frac{(Ohms)(2.9)}{24} = Value$ Default: 7.5Ohms Example: 4Ohms → 0x0000F776

The waveform F0 and ReDC parameters are set depending on which wavetable the waveform belongs:

- ROM wavetable: F0 and ReDC are fixed. See Section 3.1.1.
- RAM wavetable: F0 and ReDC are defined by the user when designing the waveforms through SCS. See Section 3.1.2 for details.
- RTH waveforms: F0 and ReDC are defined by the user. See Section 4.4 for more details.

Click compensation is disabled by default. Use the following register to enable Click Compensation.

Register Name	Register Address	Default Value	Notes
VIBEGEN_COMPENSATION_ENABLE	0x2805C30	0x00000000	0x00000000 = Disables Click Compensation 0x00000001 = Enables F0 Compensation only 0x00000002 = Enables ReDC Compensation only 0x00000003 = Enables both F0 and ReDC compensation

An example control sequence for ReDC compensation is shown in the Figure below. The example shows the user playing ROM13 via I2C/SPI mailbox triggering. The user then updates the ReDC value obtained earlier from calibration, enables ReDC Compensation, and plays the same click effect now with compensation. Note that the REDC_WAVEFORM parameter of ROM13 is stored in the ROM wavetable.

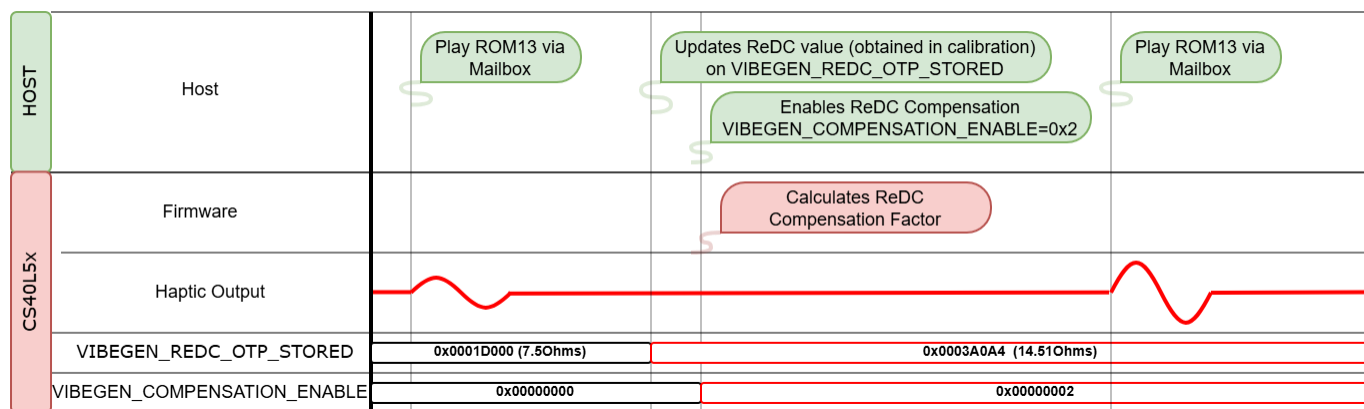


Figure 13 ReDC Compensation from Calibration



An example control sequence for F0 compensation is shown in the Figure below. The example shows the user playing ROM13 via I2C/SPI mailbox triggering. The user then updates the F0 value obtained earlier from calibration, enables F0 Compensation, and plays the same click effect now with compensation. Note that the F0_WAVEFORM parameter of ROM13 is stored in the ROM wavetable.

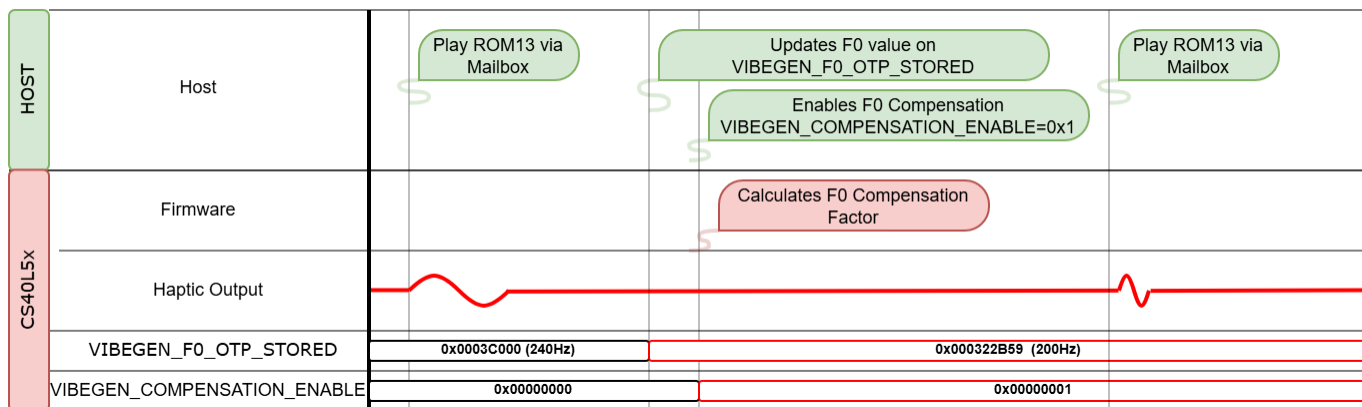


Figure 14 F0 Compensation from User-Defined F0



4.2 Calibration

CS40L51, CS40L52, and the CS40L53 all support a calibration feature that provides the ability to track resonant frequency (F0) and resistance (ReDC) of an actuator without any external components or sensors. These parameters are measured using the integrated output voltage (VMON) and current (IMON) monitoring capabilities of CS40L5x. It is recommended to be used as part of the factory-line calibration process. The result of the calibration could be used by other algorithms that compensate for any part-to-part variation in actuator parameters.

The table below describes the calibration procedure.

Step	Description	Notes			
0	[Optional] Set F0 estimation parameters	Name	Address (B0 ROM)	Default Value	Description
		F0_ESTIMATION_FREQ_SPAN	0x2802F70	0x00140000	The frequency (Hz) span of the chirp Format: Q9.14 Default: 80 Hz
		F0_ESTIMATION_FREQ_CENTRE	0x2802F74	0x003C0000	The center frequency (Hz) of the chirp Format: Q9.14 Default: 160 Hz
		F0_ESTIMATION_CHIRP_AMP	0x2802F78	0x000CCCCD	The chirp amplitude (FS) Format: Q0.23 Default: 0.1 FS
		F0_ESTIMATION_REDC*	0x2802F7C	0x0003C000	The actuator Re DC estimate Format: Q8.15 Default: 14.5 Ohm
		* Note - For more accurate results, running ReDC estimation before F0 estimation and copying that value for F0 estimation is recommended. Otherwise, the F0 estimation will use the default value.			
1	Send calibration request to DSP_VIRTUAL1_MBOX_1	Write the desired calibration command to the DSP_VIRTUAL1_MBOX_1 register 0x0011020			
		Name	Command	Description	
		F0_EST	0x07000001	Starts the F0 Estimation chirp	
		REDC_EST	0x07000002	Starts the ReDC Estimation	
		REDC_F0_EST	0x07000003	Starts ReDC estimation first, then F0 estimation	
		The CS40L5x acknowledges this command by writing 0x00000000 to the DSP_VIRTUAL1_MBOX_1 0x0011020 register.			
2	[Optional] Read calibration status (allow for a 30 millisecond delay)	See Section 2.2.2 for details on how to read the mailbox buffer			
		Name	Command	Description	
		F0_EST_START	0x07000011	F0 Estimation started	
		F0_EST_DONE	0x07000021	F0 Estimation complete	
		REDC_EST_START	0x07000012	ReDC Estimation started	
		REDC_EST_DONE	0x07000022	ReDC Estimation complete	
3	Read calibration results (allow for a 30 millisecond delay)	Calibration results are stored in the following registers			
		Name	Address (B0 ROM)	Default Value	Description
		F0_ESTIMATION_F0_EST	0x24017C0	0x00000000	Estimate of the F0 Format: Q9.14 Default: 0 Hz
		SVC_RE_EST_STATUS	0x3401110	0x00000000	Estimate of the Re DC Format: S8.15 Default: 0 Ohm

Note - Once calibration is done, the host should store the calibration values in its memory. If the user wants to use these values for click compensation, they need to be copied to the VIBEGEN_x_OTP_STORED registers. See Section 4.1 for details.

An example control sequence for ReDC compensation is shown in the Figure below.

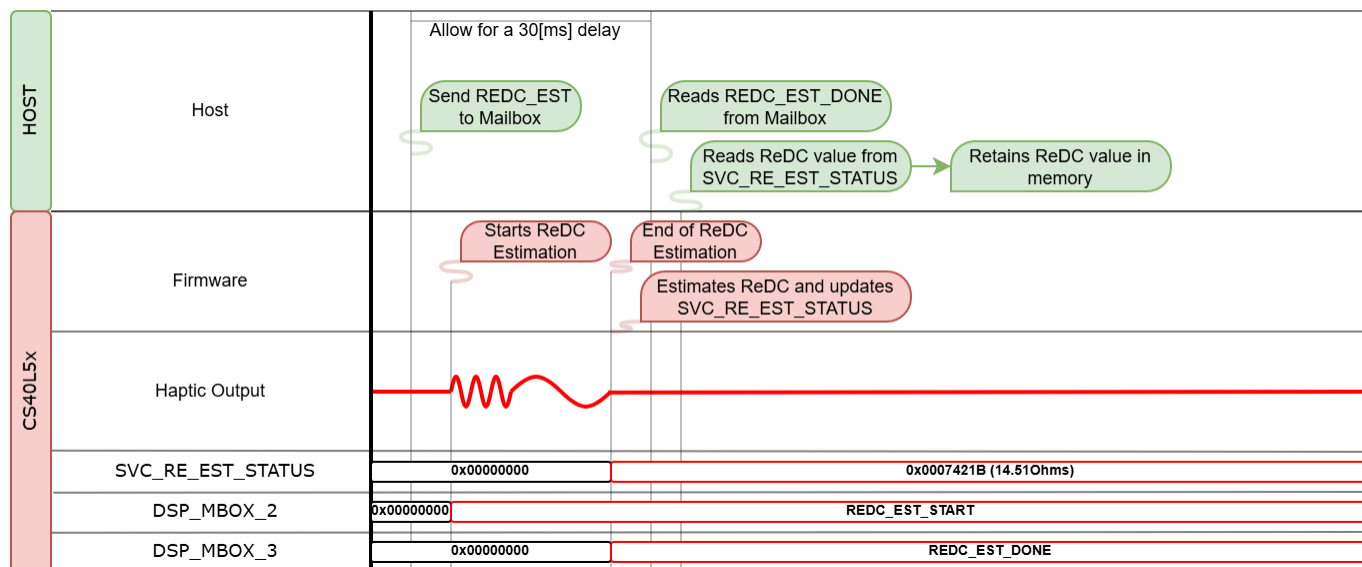


Figure 15 ReDC Calibration

4.3 Dynamic F0

Dynamic F0 is an algorithm that estimates F0 after a haptic waveform playback, which could vary due to actuator manufacturing or temperature/environmental changes. This is done by the "Ring-Down" method. When the device stops driving the actuator, the mass may still be in motion, and may "ring". While the actuator is still moving, it generates back-EMF (bEMF), which is measured through VMON/IMON. The diagram below shows an example of how F0 is determined from ringing after a waveform ends playback.

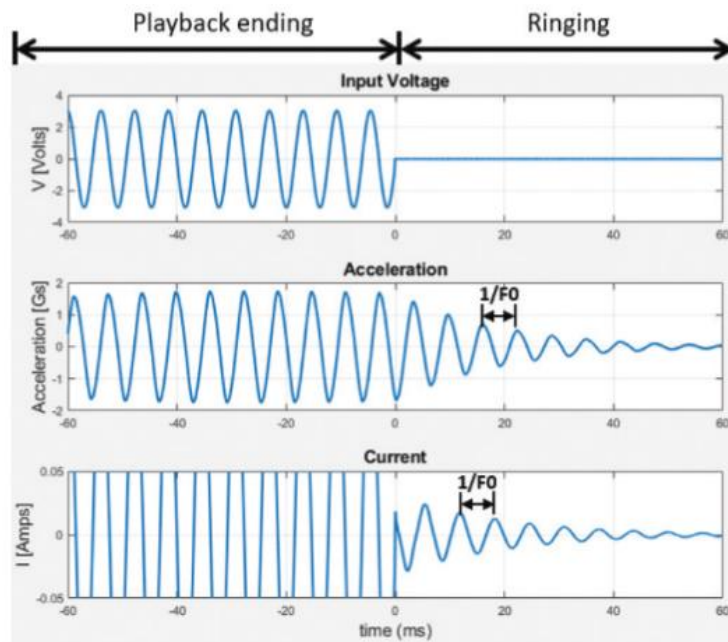


Figure 16 Dynamic F0

If Dynamic F0 is enabled, the CS40L5x measures the bEMF through VMON/IMON. If the IMON peaks are above the ringing threshold, Dynamic F0 estimates and updates the F0 for that particular waveform. The table below shows the registers to enable Dynamic F0 and set the ringing threshold.

Name	Address	Default Value	Description
DYNAMIC_F0_DYNAMIC_F0_ENABLED	0x0280285C	0x00000000	0x0 = Dynamic F0 is disabled (Default) 0x1 = Dynamic F0 is enabled
DYNAMIC_F0_IMONRINGPPTHRESHOLD	0x02802860	0x000020C5	Ringdown threshold factor. Dynamic F0 estimates a new F0 when the IMON measured is above this threshold. Format: Q1.23 Default: ROM A1: 0x00008312 → 0.004 ROM B0: 0x000020C5 → 0.001

After the dynamic F0 has been estimated for a particular waveform, the CS40L5x stores it on the Dynamic F0 table every time the waveform is played and the ringdown is above the threshold.

Note - ROM and RAM waveforms are stored and played using a wavetable index. The Dynamic F0 function supports F0 estimation for ROM and RAM waveforms, but uses the same index to store the estimated F0 for either wavetable. Therefore, Dynamic F0 should not be enabled for ROM and RAM waveforms with the same index.

Note - The Dynamic F0 algorithm only tracks F0 for the first 20 waveforms played with Dynamic F0 enabled, regardless of the number of waveforms stored in ROM or RAM with Dynamic F0 enabled.

If Click Compensation for PCM is enabled, CS40L5x uses the Dynamic F0 for the F0 compensation. Similarly for PWLEs - the CS40L5x uses the estimated Dynamic F0 when a section is set to play at F0. Otherwise, it will use the VIBEGEN_F0_OTP_STORED value.

4.4 Runtime Haptics (RTH)

While RAM and ROM waveforms have to be predefined and loaded at initialization, Runtime Haptics (RTH) waveforms can be created, loaded and played at runtime. It currently supports Runtime Haptics PCM (RTH PCM) and Runtime Haptics Piecewise Linear Envelope (RTH PWLE) waveforms. The RTH waveforms do not need to be stored for future playback as they are dynamically defined by the user. The diagram below shows the procedure to use RTH.

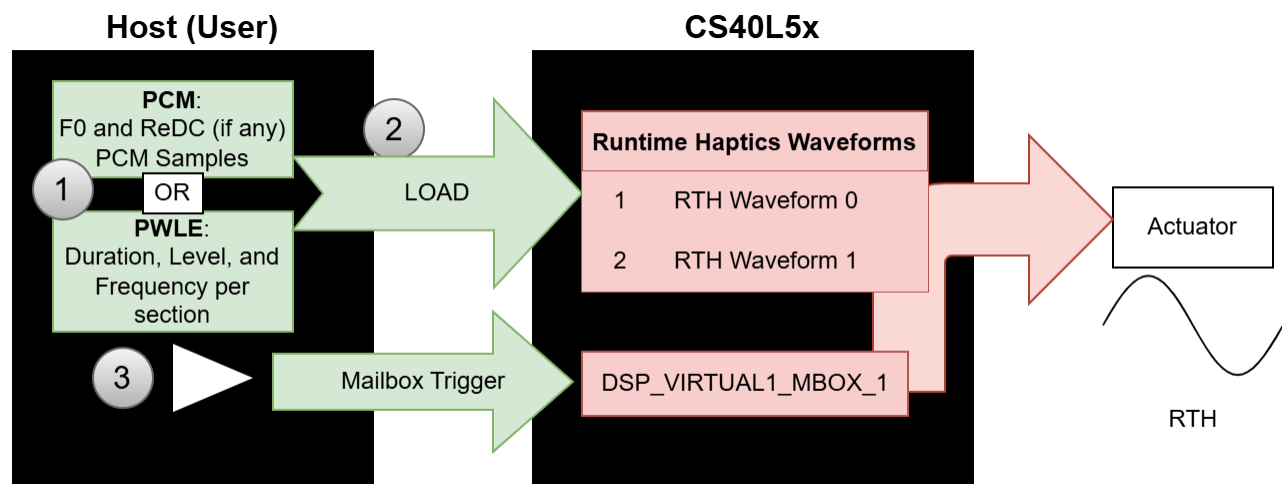


Figure 17 Run-Time Haptics

1. Define the desired waveform following the format below.
 - a. For RTH PCM, the user defines
 - i. F0 and ReDC (optional)
 - ii. PCM samples up to 378 (required)
 - b. For RTH PWLE, the user defines
 - i. Number of sections up to 63
 - ii. Duration, level, and frequency of each section
2. Load the waveform to a RTH index address
3. Play the RTH waveform via I2C/SPI mailbox trigger
4. (Optional) Modify any of the parameters above, and repeat steps 2 and 3.

The RTH wavetable supports up to 378 samples for PCM waveforms, and up to 63 sections for PWLE waveforms.

4.4.1 RTH Registers

The RTH waveform registers are summarized in the following table. These registers may change in a RAM FW variant.

Field Name	RTH Waveform - Index 0	RTH Waveform - Index 1	Register Data
	Register Address	Register Address	
RTH_WAVEFORM_TYPE	0x2807770	0x280797C	Value : RTH_PCM: 0x0000_0008 (Default) RTH_PWLE: 0x0000_000C Format : Q24.0
RTH_WAVEFORM_DATA_1	0x280777C	0x2807988	Value: See RTH_PCM_DATA See RTH_PWLE_DATA Format : See RTH_PCM_DATA See RTH_PWLE_DATA
RTH_WAVEFORM_DATA_2	0x2807780	0x280798C	
...	
RTH_WAVEFORM_DATA_128	0x2807978	0x2807B84	

4.4.2 RTH PCM Register Format

The RTH PCM register formats are defined as follows:

Field Name	RTH Waveform - Index 0	RTH Waveform - Index 1	RTH PCM Field Description and Register Data																															
	Register Address (ROM A1)	Register Address (ROM A1)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RTH_WAVEFORM_TYPE	0x2807770	0x280797C	Description: RTH Waveform Type Value: 0x0000_0008 (RTH_PCM) Format: Q24.0																															
RTH_WAVEFORM_DATA_1	0x280777C	0x2807988	Description: Reserved Value: 0x00 (Default)								Desc: Reserved Value: 0b00 (Default)		Description: RTH PCM Length - Number of PCM samples Value: [0x000001 - 0x00017A] (1-378 PCM samples) Default: 0x000 Format: Q22.0																					
RTH_WAVEFORM_DATA_2	0x2807780	0x280798C	Description: Reserved Value: 0x00 (Default)								Description: F0 value of RTH PCM waveform Value: $(freq - 50) \times 8$ Default: 0x000 (F0 compensation is not performed) Example: 100 Hz \rightarrow 0x190 Format: Q9.3											Description: ReDC value of RTH PCM waveform Value: $\frac{ReDC}{2.9} \times 24$ Default: 0x000 (ReDC compensation is not performed) Example: 8 Ω \rightarrow 0x190 Format: Q5.7												
RTH_WAVEFORM_DATA_3	0x2807784	0x2807990	Description: Reserved Value: 0x00 (Default)								Description: PCM Sample 1 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 2 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 3 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7													
RTH_WAVEFORM_DATA_4	0x2807788	0x2807994	Description: Reserved Value: 0x00 (Default)								Description: PCM Sample 4 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 5 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 6 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7													
...																															
RTH_WAVEFORM_DATA_128	0x2807978	0x2807B84	Description: Reserved Value: 0x00 (Default)								Description: PCM Sample 376 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 377 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7					Description: PCM Sample 378 (8 kHz) Value: -128 to 127 Default: 0x00 Format: S0.7													

An example control sequence to add a PCM waveform to RTH Index 0 is described as follows:

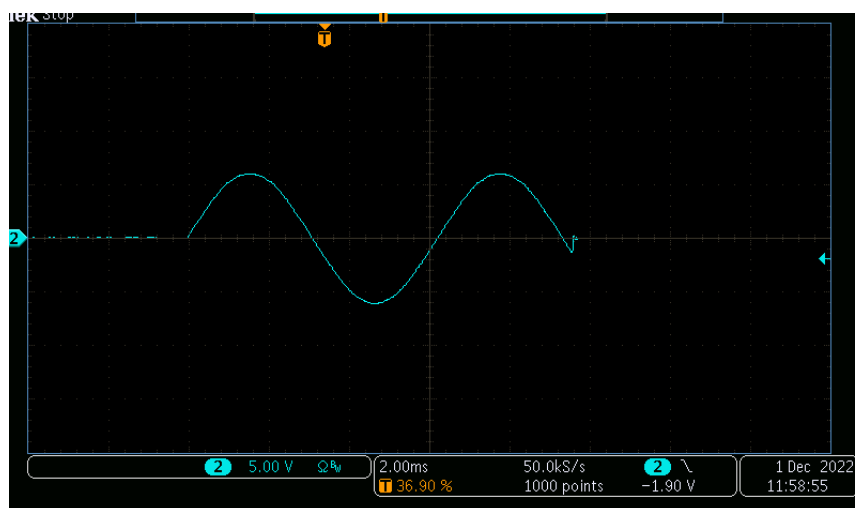
1. Load PCM waveform to RTH index 0

Register Name	Register Address	Data	Notes
RTH WAVEFORM TYPE	0x2807770	0x00000008	RTH Waveform type: PCM = 0x8
RTH WAVEFORM DATA 1	0x280777C	0x0000004E	PCM samples in bytes: 0x4E
RTH WAVEFORM DATA 2	0x2807780	0x005F0074	F0 = 240 Hz → 0x5F0, ReDC = 7.5 Ohm → 0x074
RTH WAVEFORM DATA 3	0x2807784	0x0000070D	PCM Samples (24-bit words = 3 samples per word)
RTH WAVEFORM DATA 4	0x2807788	0x0012181E	PCM Samples
RTH WAVEFORM DATA 5	0x280778C	0x0022272B	PCM Samples
RTH WAVEFORM DATA 6	0x2807790	0x002D3031	PCM Samples
RTH WAVEFORM DATA 7	0x2807794	0x00323231	PCM Samples
RTH WAVEFORM DATA 8	0x2807798	0x00302D2B	PCM Samples
RTH WAVEFORM DATA 9	0x280779C	0x0027221E	PCM Samples
RTH WAVEFORM DATA 10	0x28077A0	0x0018120D	PCM Samples
RTH WAVEFORM DATA 11	0x28077A4	0x000700F9	PCM Samples
RTH WAVEFORM DATA 12	0x28077A8	0x00F3EDE8	PCM Samples
RTH WAVEFORM DATA 13	0x28077AC	0x00E2DDD9	PCM Samples
RTH WAVEFORM DATA 14	0x28077B0	0x00D5D2D0	PCM Samples
RTH WAVEFORM DATA 15	0x28077B4	0x00CECDCD	PCM Samples
RTH WAVEFORM DATA 16	0x28077B8	0x00CED0D2	PCM Samples
RTH WAVEFORM DATA 17	0x28077BC	0x00D5D9DD	PCM Samples
RTH WAVEFORM DATA 18	0x28077C0	0x00E2E8ED	PCM Samples
RTH WAVEFORM DATA 19	0x28077C4	0x00F3F900	PCM Samples
RTH WAVEFORM DATA 20	0x28077C8	0x00070D12	PCM Samples
RTH WAVEFORM DATA 21	0x28077CC	0x00181E22	PCM Samples
RTH WAVEFORM DATA 22	0x28077D0	0x00272B2D	PCM Samples
RTH WAVEFORM DATA 23	0x28077D4	0x00303132	PCM Samples
RTH WAVEFORM DATA 24	0x28077D8	0x00323130	PCM Samples
RTH WAVEFORM DATA 25	0x28077DC	0x002D2B27	PCM Samples
RTH WAVEFORM DATA 26	0x28077E0	0x00221E18	PCM Samples
RTH WAVEFORM DATA 27	0x28077E4	0x00120D07	PCM Samples
RTH WAVEFORM DATA 28	0x28077E8	0x0000F9F3	PCM Samples

2. Play via I2C/SPI mailbox command

Operation	Register Name	Register Address	Data	Notes
WRITE	DSP_VIRTUAL1_MBOX_1	0x0011020	0x01400000	Play RTH index 0

Waveform



MCU Driver Reference

File: ../cs40l5x/cs40l5x_ext.c

Function: cs40l5x_trigger_pcm()

4.4.3 RTH PWLE Register Format

The RTH PWLE register formats are defined as follows:

AN0680R1

Section Flags			
Bit 3	Bit 2	Bit 1	Bit 0
Chirp Mode Flag Selects whether the frequency ramps up/down linearly from previous section. 0 = No ramp 1 = Ramp enabled	Reserved Value: 0	Half-Cycles Flag Determines whether the duration field is in Time (ms) or Half-cycle units. 0 = Time (ms) 1 = Half cycles	Reserved Value: 1

An example control sequence to add a PWLE waveform to RTH Index 1 is described as follows:

1. Define number of sections and parameters

Section	Duration (ms) / Half-Cycles	Level (FS)	Frequency (Hz)	Flags
1	0 (Reserved)	0.5	200 (Reserved)	None
2	10	0.5	200	None
3	0	0.25	400	None
4	5	0.25	400	None
5	0	0.5	100	None
6	40	0.5	500	Chirp mode Only
7	0	0.25	200	None
8	3	0.5	200	Half Cycles Only

2. Load PWLE waveform to RTH index 1

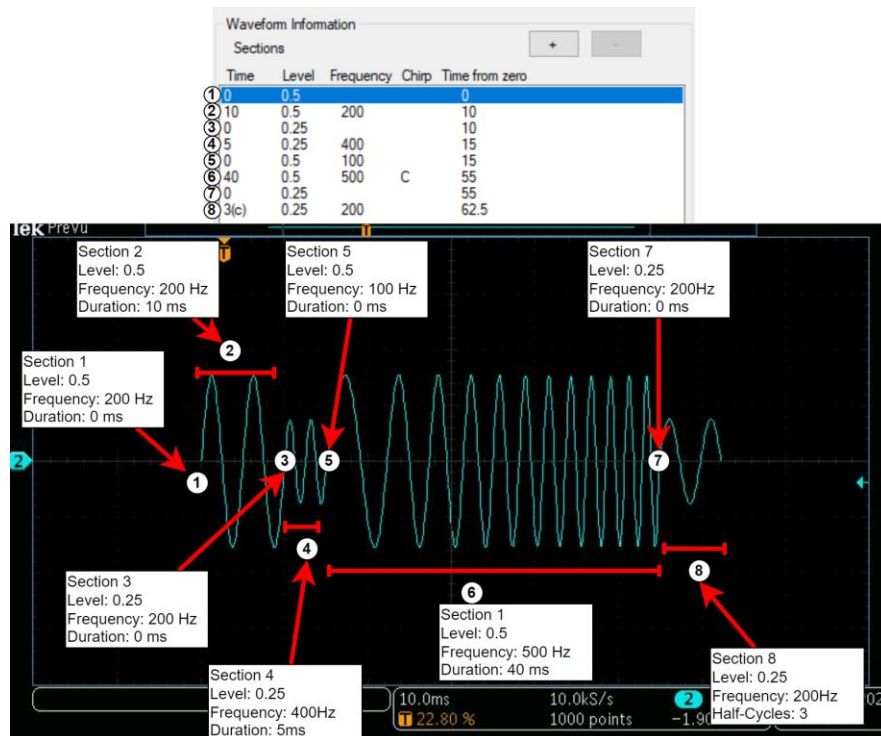
Register Name	Register Address	Data	Notes w/o Vb
RTH_WAVEFORM_TYPE	0x280797C	0x0000000C	RTH Waveform type: PWLE = 0xC
RTH_WAVEFORM_DATA_1	0x2807988	0x003FFFFFFF	Reserved: 0x003FFFFFFF
RTH_WAVEFORM_DATA_2	0x280798C	0x00000000	[23:4]: Reserved: 0x0 [03:00]: Number of sections MS bits: 0x08
RTH_WAVEFORM_DATA_3	0x2807990	0x00800004	[23:20]: Number of sections LS bits: 0x08 [19:04]: Section 1 Duration (Reserved): 0x0000 [3:0]: Section 1 Level MS bits: 0x400 → 0.5 FS
RTH_WAVEFORM_DATA_4	0x2807994	0x00003201	[23:16]: Section 1 Level LS bits: 0x400 → 0.5 FS [15:04]: Section 1 Frequency (Reserved): 0x320 → 200 Hz [03:00]: Section 1 Flags (Reserved): 0x1
RTH_WAVEFORM_DATA_5	0x2807998	0x00000284	[23:20]: Reserved: 0x0 [19:04]: Section 2 Duration: 0x0028 → 10 ms [3:0]: Section 2 Level MS bits: 0x400 → 0.5 FS
RTH_WAVEFORM_DATA_6	0x280799C	0x00003201	[23:16]: Section 2 Level LS bits: 0x400 → 0.5 FS [15:04]: Section 2 Frequency: 0x320 → 200 Hz [03:00]: Section 2 Flags: 0x1
RTH_WAVEFORM_DATA_7	0x28079A0	0x00000002	[23:20]: Reserved: 0x0 [19:04]: Section 3 Duration: 0x0000 → 0 ms [03:00]: Section 3 Level MS bits: 0x200 → 0.25 FS
RTH_WAVEFORM_DATA_8	0x28079A4	0x00003201	[23:16]: Section 3 Level LS bits: 0x200 → 0.25 FS [15:04]: Section 3 Frequency: 0x320 → 200 Hz [03:00]: Section 3 Flags: 0x1
RTH_WAVEFORM_DATA_9	0x28079A8	0x00000142	[23:20]: Reserved: 0x0 [19:04]: Section 4 Duration: 0x0014 → 5 ms [03:00]: Section 4 Level MS bits: 0x200 → 0.25 FS
RTH_WAVEFORM_DATA_10	0x28079AC	0x00006401	[23:16]: Section 4 Level LS bits: 0x200 → 0.25 FS [15:04]: Section 4 Frequency: 0x640 → 400 Hz [03:00]: Section 4 Flags: 0x1
RTH_WAVEFORM_DATA_11	0x28079B0	0x00000004	[23:20]: Reserved: 0x0 [19:04]: Section 5 Duration: 0x0000 [03:00]: Section 5 Level MS bits: 0x400 → 0.5 FS
RTH_WAVEFORM_DATA_12	0x28079B4	0x00001901	[23:16]: Section 5 Level LS bits: 0x400 → 0.5 FS [15:04]: Section 5 Frequency (Reserved): 0x190 → 100 Hz [03:00]: Section 5 Flags: 0x1
RTH_WAVEFORM_DATA_13	0x28079B8	0x00000A04	[23:20]: Reserved: 0x0 [19:04]: Section 6 Duration: 0x00A0 → 10 ms [03:00]: Section 6 Level MS bits: 0x400 → 0.5 FS

Register Name	Register Address	Data	Notes w/o Vb
RTH_WAVEFORM_DATA_14	0x28079BC	0x00007D09	[23:16]: Section 6 Level LS bits: 0x400 → 0.5 FS [15:04]: Section 6 Frequency: 0x7D0 → 500 Hz [03:00]: Section 6 Flags: 0x9 → Chirp mode Flag
RTH_WAVEFORM_DATA_15	0x28079C0	0x00000002	[23:20]: Reserved: 0x0 [19:04]: Section 7 Duration: 0x0000 → 0 ms [03:00]: Section 7 Level MS bits: 0x200 → 0.25 FS
RTH_WAVEFORM_DATA_16	0x28079C4	0x00003201	[23:16]: Section 7 Level LS bits: 0x200 → 0.25 FS [15:04]: Section 7 Frequency: 0x320 → 200 Hz [03:00]: Section 7 Flags: 0x1
RTH_WAVEFORM_DATA_17	0x28079C8	0x00000032	[23:20]: Reserved: 0x0 [19:04]: Section 7 Duration: 0x0003 → 3 half-cycles [03:00]: Section 7 Level MS bits: 0x200 → 0.25 FS
RTH_WAVEFORM_DATA_18	0x28079CC	0x00003203	[23:16]: Section 7 Level LS bits: 0x200 → 0.25 FS [15:04]: Section 7 Frequency: 0x320 → 200 Hz [03:00]: Section 7 Flags: 0x3 → Half Cycles Flag

3. Play via I2C/SPI mailbox command

Operation	Register Name	Register Address	Data	Notes
WRITE	DSP_VIRTUAL1_MBOX_1	0x0011020	0x01400001	Play RTH index 1

Waveform



MCU Driver Reference

File: ../cs40l5x/cs40l5x_ext.c

Function: cs40l5x_trigger_pwle_advanced()

4.5 Load Diagnostics

The CS40L51 and CS40L52 devices support a diagnostic function which can be commanded by the host. The diagnostics are configured using programmable ranges for key operating parameters of the haptics driver and the LRA. The host is notified when the diagnostics are complete. An indication is provided for any LRA parameter that is outside the configured acceptable range. VDD_B, VDD_AMP, boost inductor short circuit, amplifier open circuit, and amplifier short circuit conditions are also indicated.

To run the diagnostics, the host should perform the following steps:

1. If LRA parameter diagnostics (including input-voltage checks) are required, set the DIAGNOSTICS_RUN_LRA_CHECKS bit:

Register Name	Register Address	Description
DIAGNOSTICS_RUN_LRA_CHECKS	0x03400FA0	If set to 1, this register enables the LRA diagnostics (comprising F0, Q, bEMF, Zres, ReDC, Le, and input voltage checks). This bit should be set to 1 before sending RUN_DIAGNOSTICS mailbox command.

2. Configure the programmable thresholds for each diagnostic:

Register Name	Register Address	Default Value	Description	Threshold Range
DIAGNOSTICS_VDD_AMP_MIN	0x03400FAC	0 V	VDD_AMP minimum threshold.	0–22.57 V in 5 mV steps
DIAGNOSTICS_VDD_AMP_MAX	0x03499FA8	22.57 V	VDD_AMP maximum threshold.	0–22.57 V in 5 mV steps
DIAGNOSTICS_VDD_B_MIN	0x03400FB0	0	VDD_B minimum threshold.	0–22.57 V in 5 mV steps
DIAGNOSTICS_VDD_B_MAX	0x03400FB4	22.57 V	VDD_B maximum threshold.	0–22.57 V in 5 mV steps
DIAGNOSTICS_REDC_MIN	0x03400FB8	0 Ω	ReDC minimum threshold.	0–256 Ω in 1 Ω steps
DIAGNOSTICS_REDC_MAX	0x03400FBC	256 Ω	ReDC maximum threshold.	0–256 Ω in 1 Ω steps
DIAGNOSTICS_LE_MIN	0x03400FC0	0 H	Le minimum threshold.	0–1 H in 1 H steps
DIAGNOSTICS_LE_MAX	0x03400FC4	1 H	Le maximum threshold.	0–1 H in 1 H steps
DIAGNOSTICS_F0_MIN	0x03400FC8	0	F0 minimum threshold.	0–512 Hz in 1 Hz steps
DIAGNOSTICS_F0_MAX	0x03400FCC	512 Hz	F0 maximum threshold.	0–512 Hz in 1 Hz steps
DIAGNOSTICS_Q_MIN	0x03400FD0	0	Q minimum threshold.	0–128
DIAGNOSTICS_Q_MAX	0x03400FDC	128	Q maximum threshold.	0–128
DIAGNOSTICS_BEMF_MIN	0x03400FD8	0 V	bEMF minimum threshold.	0–24 V in 1 V steps
DIAGNOSTICS_BEMF_MAX	0x03400FDC	1 V	bEMF maximum threshold.	0–24 V in 1 V steps
DIAGNOSTICS_ZRES_MIN	0x03400FE0	0 Ω	Zres minimum threshold.	0–256 Ω in 1 Ω steps
DIAGNOSTICS_ZRES_MAX	0x03400FE4	256 Ω	Zres maximum threshold.	0–256 Ω in 1 Ω steps

3. Send the RUN_DIAGNOSTICS mailbox command:

Register Name	Mailbox Command	Description
RUN_DIAGNOSTICS	0x0300000C	This mailbox command starts the diagnostics.

4. Wait for the DIAGNOSTICS_DONE mailbox message:

Register Name	Mailbox Message	Description
DIAGNOSTICS_DONE	0x03000000	If received by the host, this mailbox message indicates that the diagnostics are complete.

5. Read the DIAGNOSTICS_RESULT:

Register Name	Register Address	Result Bitfield	Description
DIAGNOSTICS_RESULT	0x03400FA8	Bit 18: Zres too high Bit 17: Zres too low Bit 16: bEMF too high Bit 15: bEMF too low Bit 14: Q too high Bit 13: Q too low Bit 12: F0 too high Bit 11: F0 too low Bit 10: Le too high Bit 9: Le too low Bit 8: ReDC too high Bit 7: ReDC too low Bit 6: VDD_B too high Bit 5: VDD_B too low Bit 4: VDD_AMP too high Bit 3: VDD_AMP too low Bit 2: Boost inductor short circuit detected Bit 1: Amplifier open circuit detected Bit 0: Amplifier short circuit detected	Diagnostics result for amplifier short circuit, amplifier open circuit, boost inductor short circuit, and LRA checks (if configured to run). The host should read this register after the DIAGNOSTICS_DONE mailbox message has been received.

4.6 Generic Haptic System Controls

This section contains common features requested from the Haptics System FW API.

Name	Address	Default Value	Description
HAPTICS_SYSTEM_SOURCE_ATTENUATION	0x0280_30B8	0x0000_0000	Format: S21.2 Units: dB Range: 0 - 2097151.75 Default: 0 dB Example: 6 dB attenuation → 0x0000_0018
HAPTICS_SYSTEM_SOURCE_INVERT	0x0280_30C0	0x0000_0000	Format: Boolean Units: N/A Range: 0 → False, 1 → True Default: False Example: Invert waveform → 0x0000_0001
AMP_PROT_EN	0x0280_30E4	0x0000_0000	Format: Boolean Units: N/A Range: 0 → False, 1 → True Default: False Example: Enable amplifier over-voltage protection → 0x0000_0001
AMP_PROT_GPIO	0x0280_30E8	0x0000_0000	Format: N/A Units: N/A Range: GPIO pin (e.g. 0=GPIO1, 1=GPIO2, 2=GPIO3, etc.) Default: 0 (GPIO1) Example: Use GPIO2 for overvoltage protection → 0x0000_0001

Name	Address	Default Value	Description
AMP_PROT_VDDB_RISE_THRESH	0x0280_30EC	0x0000_0000	Format: N/A Units: N/A Range: GPIO pin (e.g. 0=GPIO1, 1=GPIO2, 2=GPIO3, etc.) Default: 0 (GPIO1) Example: Use GPIO2 for overvoltage protection → 0x0000_0001
AMP_PROT_VDDB_FALL_THRESH	0x0280_30F0	0x0000_0000	Format: N/A Units: N/A Range: GPIO pin (e.g. 0=GPIO1, 1=GPIO2, 2=GPIO3, etc.) Default: 0 (GPIO1) Example: Use GPIO2 for overvoltage protection → 0x0000_0001

5 Revision History

Revision	Changes
R1 NOV 2025	• Initial release
R2 DEC 2025	• Re-released without confidential restrictions

Contacting Cirrus Logic Support

For all product questions and inquiries, contact a Cirrus Logic Sales Representative.

To find the one nearest you, go to www.cirrus.com.

IMPORTANT NOTICE

The products and services of Cirrus Logic International (UK) Limited; Cirrus Logic, Inc.; and other companies in the Cirrus Logic group (collectively either “Cirrus Logic” or “Cirrus”) are sold subject to Cirrus Logic’s terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, indemnification, and limitation of liability. Software is provided pursuant to applicable license terms. Cirrus Logic reserves the right to make changes to its products and specifications or to discontinue any product or service without notice. Customers should therefore obtain the latest version of relevant information from Cirrus Logic to verify that the information is current and complete. Testing and other quality control techniques are utilized to the extent Cirrus Logic deems necessary. Specific testing of all parameters of each device is not necessarily performed. In order to minimize risks associated with customer applications, the customer must use adequate design and operating safeguards to minimize inherent or procedural hazards. Cirrus Logic is not liable for applications assistance or customer product design. The customer is solely responsible for its overall product design, end-use applications, and system security, including the specific manner in which it uses Cirrus Logic components. Certain uses or product designs may require an intellectual property license from a third party. Features and operations described herein are for illustrative purposes only and do not constitute a suggestion or instruction to adopt a particular product design or a particular mode of operation for a Cirrus Logic component.

CIRRUS LOGIC PRODUCTS ARE NOT DESIGNED, TESTED, INTENDED OR WARRANTED FOR USE (1) WITH OR IN IMPLANTABLE PRODUCTS OR FDA/MHRA CLASS III (OR EQUIVALENT CLASSIFICATION) MEDICAL DEVICES, OR (2) IN ANY PRODUCTS, APPLICATIONS OR SYSTEMS, INCLUDING WITHOUT LIMITATION LIFE-CRITICAL MEDICAL EQUIPMENT OR SAFETY OR SECURITY EQUIPMENT, WHERE MALFUNCTION OF THE PRODUCT COULD CAUSE PERSONAL INJURY, DEATH, SEVERE PROPERTY DAMAGE OR SEVERE ENVIRONMENTAL HARM. INCLUSION OF CIRRUS LOGIC PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER’S RISK AND CIRRUS LOGIC DISCLAIMS AND MAKES NO WARRANTY, EXPRESS, STATUTORY OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, WITH REGARD TO ANY CIRRUS LOGIC PRODUCT THAT IS USED IN SUCH A MANNER. IF THE CUSTOMER OR CUSTOMER’S CUSTOMER USES OR PERMITS THE USE OF CIRRUS LOGIC PRODUCTS IN SUCH A MANNER, CUSTOMER AGREES, BY SUCH USE, TO FULLY INDEMNIFY CIRRUS LOGIC, ITS OFFICERS, DIRECTORS, EMPLOYEES, DISTRIBUTORS AND OTHER AGENTS FROM ANY AND ALL LIABILITY, INCLUDING ATTORNEYS’ FEES AND COSTS, THAT MAY RESULT FROM OR ARISE IN CONNECTION WITH THESE USES.

This document is the property of Cirrus Logic, and you may not use this document in connection with any legal analysis concerning Cirrus Logic products described herein. No license to any technology or intellectual property right of Cirrus Logic or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property rights. Any provision or publication of any third party’s products or services does not constitute Cirrus Logic’s approval, license, warranty or endorsement thereof. Cirrus Logic gives consent for copies to be made of the information contained herein only for use within your organization with respect to Cirrus Logic integrated circuits or other products of Cirrus Logic, and only if the reproduction is without alteration and is accompanied by all associated copyright, proprietary and other notices and conditions (including this notice). This consent does not extend to other copying such as copying for general distribution, advertising or promotional purposes, or for creating any work for resale. This document and its information is provided “AS IS” without warranty of any kind (express or implied). All statutory warranties and conditions are excluded to the fullest extent possible. No responsibility is assumed by Cirrus Logic for the use of information herein, including use of this information as the basis for manufacture or sale of any items, or for infringement of patents or other rights of third parties. Cirrus Logic, Cirrus, the Cirrus Logic logo design, and SoundClear are among the trademarks of Cirrus Logic. Other brand and product names may be trademarks or service marks of their respective owners.

Copyright © 2025 Cirrus Logic, Inc. and Cirrus Logic International Semiconductor Ltd. All rights reserved.