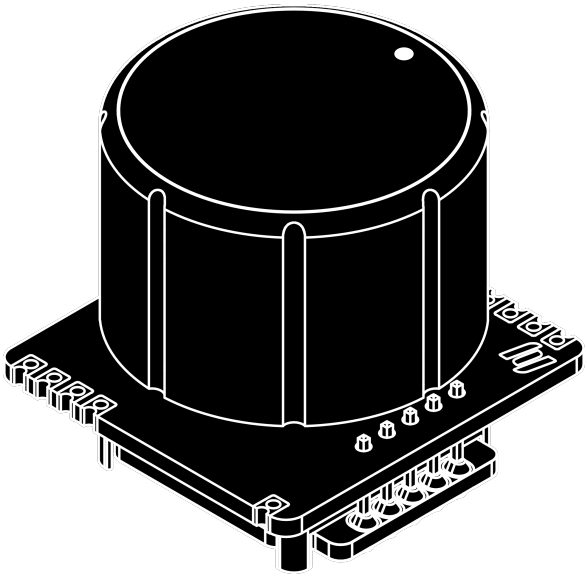
Introduction

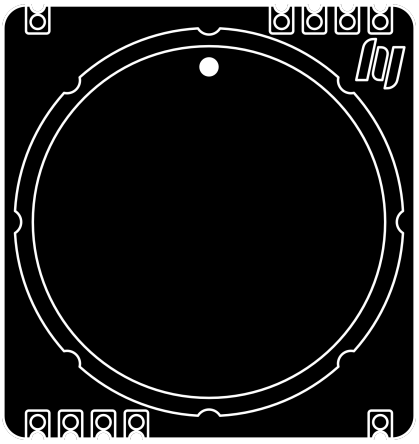
Magic Dot Knob Module MDKM24 is an integrated digital knob system for electronic musical instruments and pro audio applications. The Magic Dot Knob Module family of products is designed to enhance the UX of these applications by enabling adaptive visual feedback and adaptive haptic feedback.

MDKM24 provides adaptive visual feedback with the Magic Dot Display which can be used for parameter recall on the OEM user interface. The Magic Dot Display indicates one position at a time representing the current value of the knob, where this value is programmable over I²C.

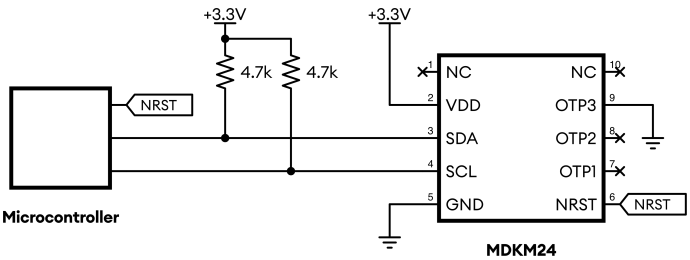
The adaptive haptic feedback system of MDKM24 can be programmed over I²C to trigger any of the 123 built-in haptic waveforms to accomplish a wide variety of feedback effects. Examples of haptic feedback applications include virtual detents, virtual endpoints, and tempo/rhythm preview. By manipulating haptic feedback and the Magic Dot Display simultaneously, MDKM24 can behave as an application-specific control such as an N-position rotary switch.

MDKM24 is powered by a single 3.3 V supply. The module connects electrically and mechanically by soldering the castellated edge connectors to the OEM system PCB. The proprietary footprint design of MDKM24 allows the module to mount from the top or the bottom of the PCB for a versatile range of vertical position adjustment.





Typical Application



Features

Knob Body — 30 mm diameter with anodized aluminum construction.

Magic Dot Display — For parameter recall and continuous motion control.

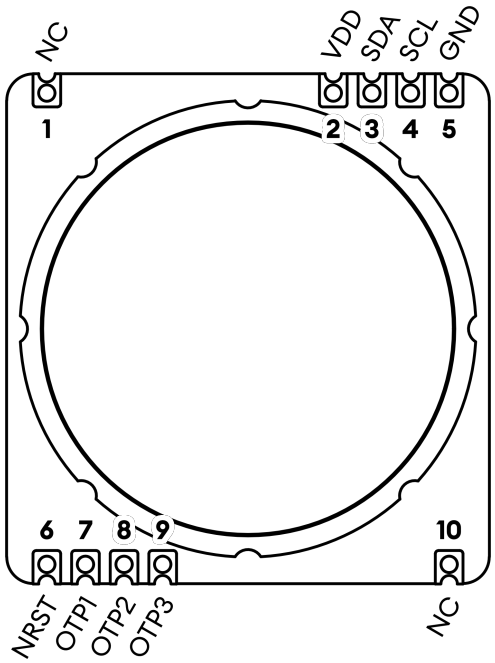
Haptic Feedback — For virtual mechanics and augmented tactility.

Castellated Connectors — For easy hardware integration with multiple mounting options.

Compact Form Factor — Versatile design fits in a variety of applications.

I²C Interface — For easy firmware integration with no setup required.

Pin Configuration



Pin Functions

Pin	Pin Name	Description
1	NC	Mechanical connection only
2	VDD	Power supply
3	SDA	Serial data
4	SCL	Serial clock
5	GND	Ground
6	NRST	Active-low reset
7	OTP1	Reserved; leave unconnected
8	OTP2	Reserved; leave unconnected
9	OTP3	Reserved; connect to ground
10	NC	Mechanical connection only

Absolute Maximum Ratings

Parameter	Min	Max	Unit
Supply voltage, V_{DD}	–0.3	4.0	V
Voltage at any input pin	–0.3	VDD + 0.3	V
Current into V_{DD} pin	—	215	mA
Storage temperature	–30	75	°C
Operating temperature	–20	60	°C

Electrical Characteristics

Symbol	Parameter	Min	Max	Unit
V_{DD}	Supply voltage	2.7	3.6	V
V_{IL}	Logic low "0" input voltage (SDA, SCL, NRST)	GND	0.4	V
V_{IH}	Logic high "1" input voltage (SDA, SCL, NRST)	1.4	VDD	V
f_{SCL}	Serial clock frequency (I^2C Fast Mode)	—	400	kHz

I²C Interface

MDKM24 has a 2-bit configurable I²C target address set by the 0Ω jumper resistors located on the bottom of the main PCB. The jumper resistors are called ADDRSEL0 and ADDRSEL1 and both are present by default. The presence of a jumper resistor sets the configuration bit low, whereas the absence of the jumper sets the configuration bit high. To set the bits, selectively keep or desolder the jumper resistors to achieve the desired I²C target address according to the following table.

ADDRSEL1	ADDRSEL0	I²C Target Address
0	0	0x20
0	1	0x21
1	0	0x22
1	1	0x23

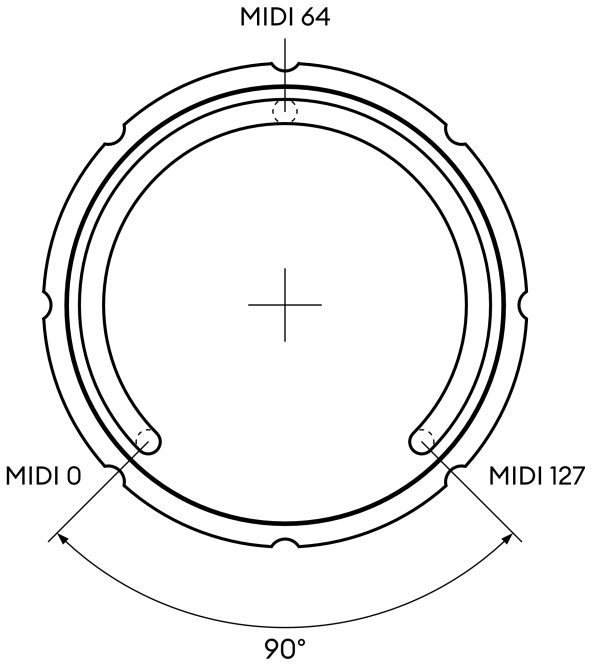
The I²C interface gives the OEM system control of the Magic Dot Display and the haptic feedback system.

Controlling the Magic Dot Display consists of reading and writing the *active position*, which is defined as the position of the dot indicator at any given time. The value of the active position is represented by the 7-bit MIDI CC range from 0 to 127 inclusive. To use the Knob Module as a hardware control input, read the active position by polling. To manipulate the active position for patch/program recall or other user interface effects, write a new value to the address 0x01.

The haptic feedback system is based on the DRV2605L haptic driver IC which contains a set of 123 pre-designed haptic waveforms in ROM. The Knob Module produces haptic effects by triggering these built-in waveforms one at a time. To trigger a haptic effect, write the corresponding effect ID to the address 0x02. Refer to Appendix A to see the effects listed by name and ID. Refer to Appendix B for examples of programming haptic effects over I²C.

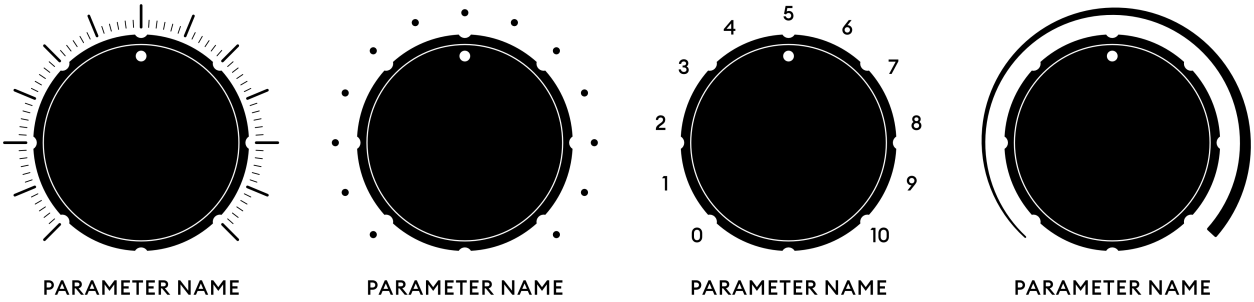
Display Properties

The left and right bounds of the Magic Dot Display are defined by a 90° gap oriented downward. The MIDI CC output values increase linearly from the left to the right bound.

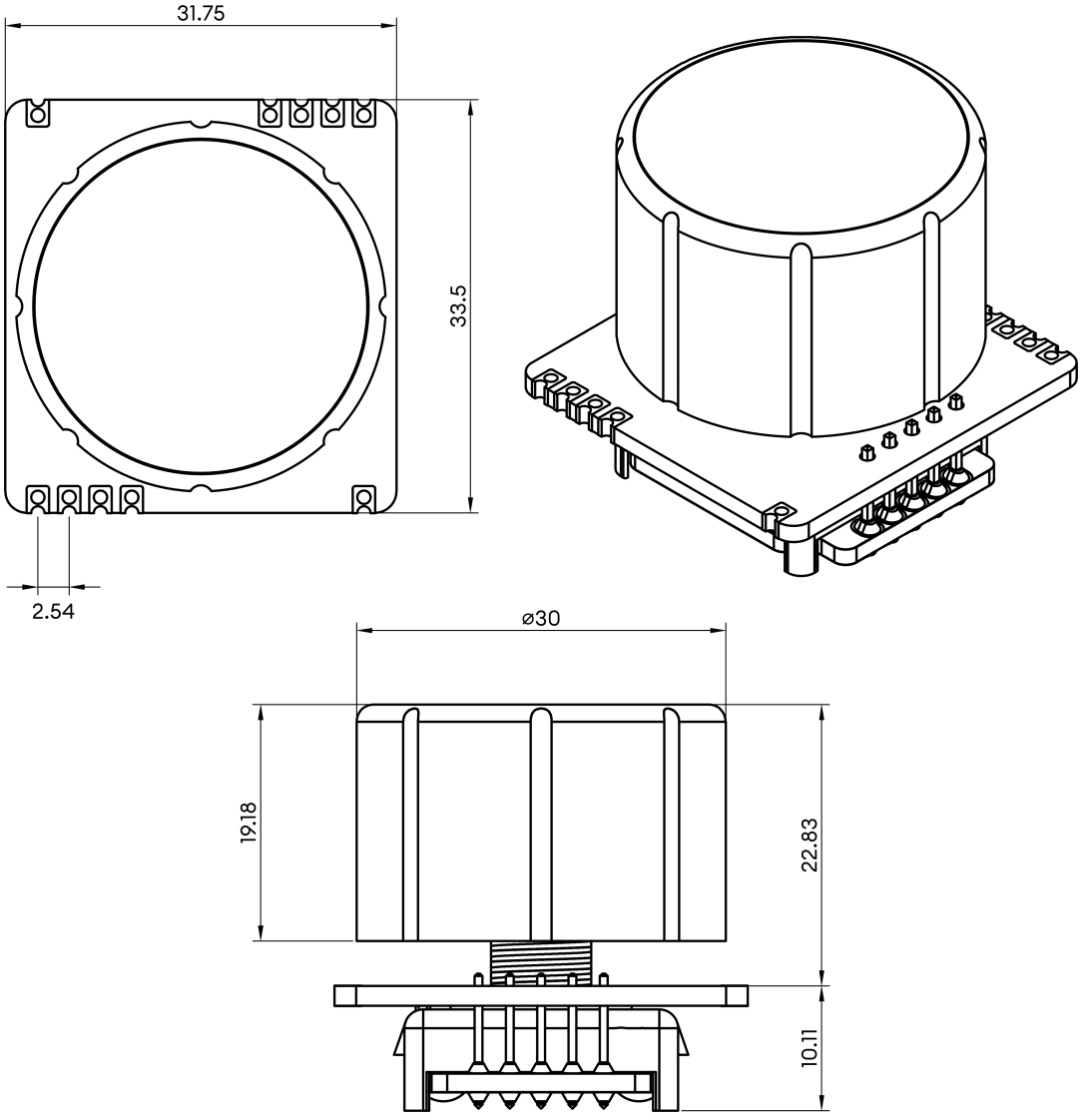


Function	I²C Transaction Type	Data Address	Data Size	Description
Get active position	Read	—	1 byte	The OEM system polls the Knob Module to get the current active position [0–127]
Set active position	Write	0x01	1 byte	The OEM system sets the active position of the Knob Module by overwriting the current active position [0–127]; for example, when recalling a patch/program
Play haptic effect	Write	0x02	1 byte	The OEM system triggers a one-shot haptic effect from the Knob Module with a certain waveform specified by an ID [1–123]; re-writing the waveform ID re-triggers the haptic effect

Display Properties — Continued
The following illustrations suggest some meter marking designs for an OEM application that would accurately correspond to the Magic Dot Display.



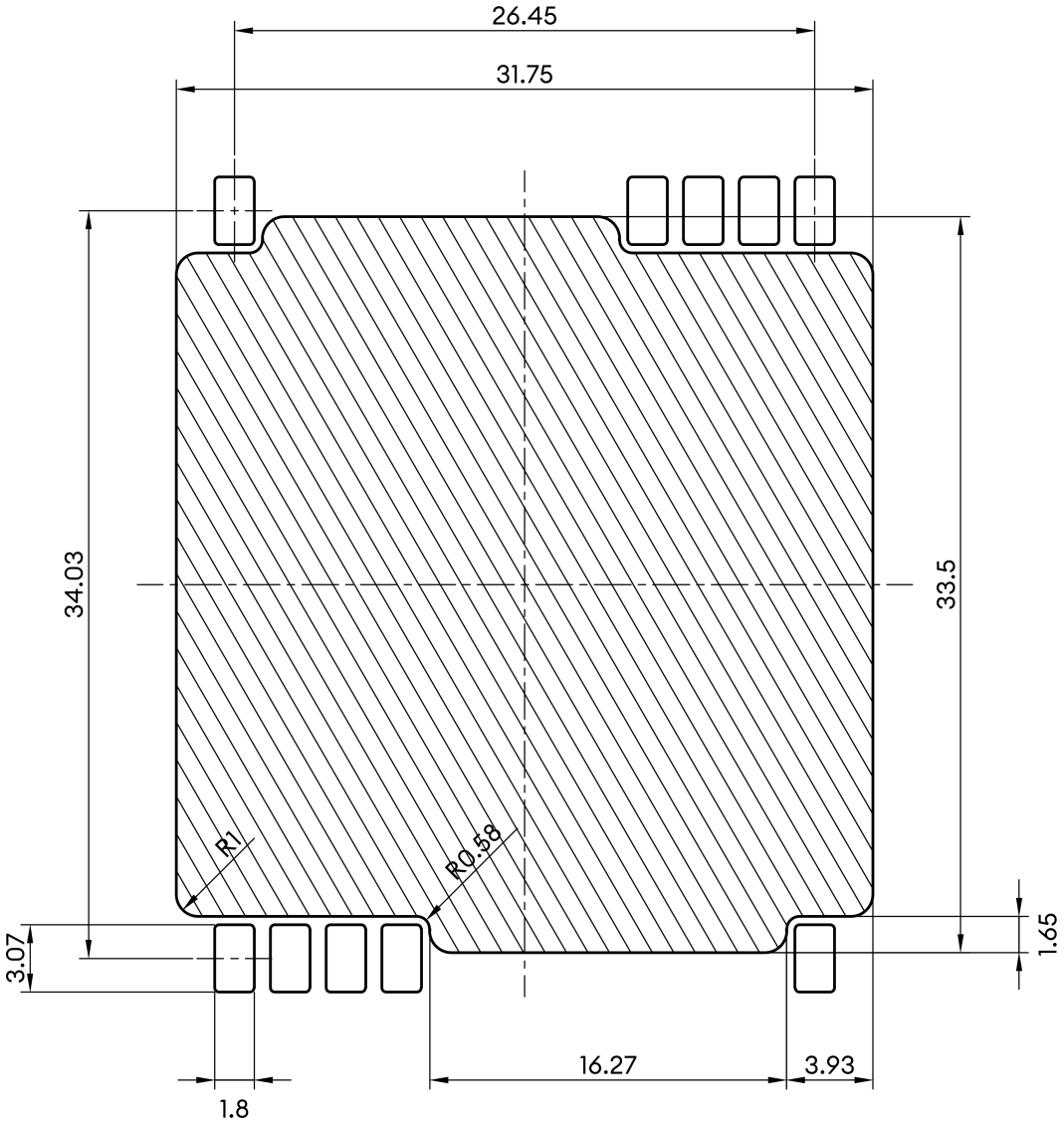
Mechanical Dimensions



Units: millimeters

Mechanical Dimensions — Continued
Note that the main PCBA of the Knob Module contains integrated circuits and other components; their representation has been omitted in this datasheet for simplicity. The height of the tallest exposed component on the top layer of the PCBA is 2.5 mm.

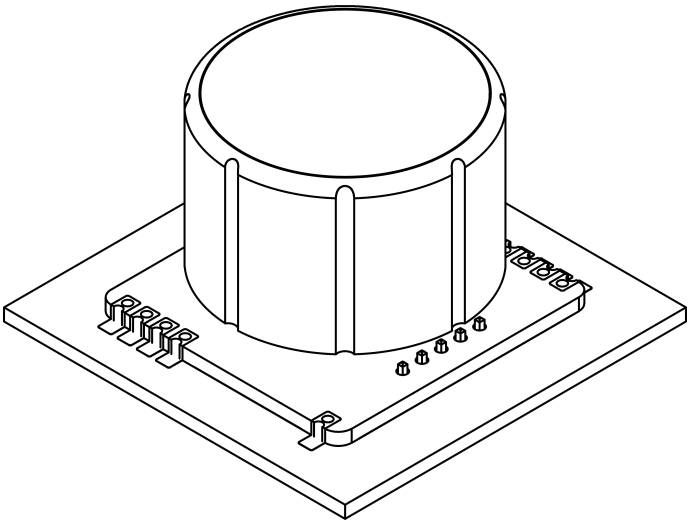
Recommended Footprint



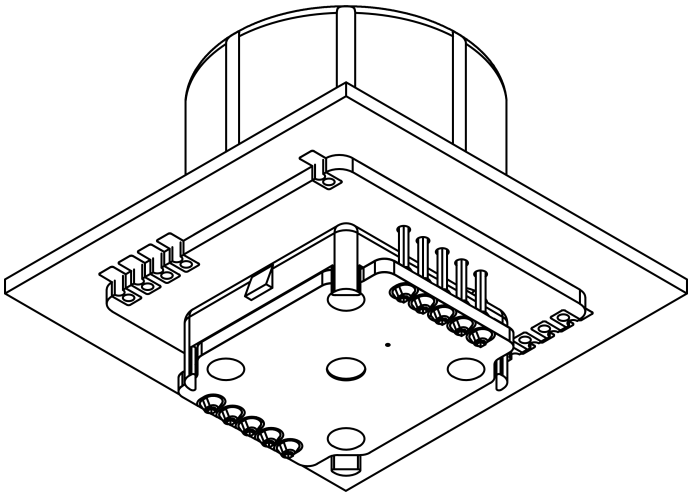
Units: millimeters

Recommended Footprint — Continued
The castellated connectors can mount to the top or the bottom of the OEM system PCB while the knob faces upward. When mounting to the top versus the bottom, there is a vertical position adjustment range of 3.2 mm assuming the OEM system PCB has a standard 1.6 mm thickness.

For a greater range of vertical position adjustment, this footprint could be adapted to use pin headers that provide a vertical offset. For this mounting scheme, connect the pin headers to the through-holes included with the castellated connectors.



Top Mount



Bottom Mount

Ordering Information

Product family	Display path diameter
MDKM = Magic Dot Knob Module	24 = 24 mm display path diameter

Revision History

Revision	Description	Date
v1.0.0	Initial release for MDKM24 v2.1.0	2025-09-05
v1.1.0	Add haptics programming interface (firmware v2.2.0) and supporting appendices	2025-11-03

Appendix A: Haptic Effects List

This list of pre-designed haptic waveforms represents the full range of haptic effects that can be produced by MDKM24. To trigger a haptic effect, identify the desired effect from the name column and write the corresponding ID to address 0x02.

This list has been reproduced from the datasheet of the DRV2605L haptic driver which is designed by Texas Instruments. These pre-designed haptic waveforms are designed by Immersion Corporation and licensed to Texas Instruments as part of the TouchSense 2200 software which is embedded in the DRV2605L haptic driver.

Effect ID	Waveform Name	Effect ID	Waveform Name	Effect ID	Waveform Name
1	Strong Click – 100%	42	Long Double Sharp Click Medium 2 – 80%	83	Transition Ramp Up Long Smooth 2 – 0 to 100%
2	Strong Click – 60%	43	Long Double Sharp Click Medium 3 – 60%	84	Transition Ramp Up Medium Smooth 1 – 0 to 100%
3	Strong Click – 30%	44	Long Double Sharp Tick 1 – 100%	85	Transition Ramp Up Medium Smooth 2 – 0 to 100%
4	Sharp Click – 100%	45	Long Double Sharp Tick 2 – 80%	86	Transition Ramp Up Short Smooth 1 – 0 to 100%
5	Sharp Click – 60%	46	Long Double Sharp Tick 3 – 60%	87	Transition Ramp Up Short Smooth 2 – 0 to 100%
6	Sharp Click – 30%	47	Buzz 1 – 100%	88	Transition Ramp Up Long Sharp 1 – 0 to 100%
7	Soft Bump – 100%	48	Buzz 2 – 80%	89	Transition Ramp Up Long Sharp 2 – 0 to 100%
8	Soft Bump – 60%	49	Buzz 3 – 60%	90	Transition Ramp Up Medium Sharp 1 – 0 to 100%
9	Soft Bump – 30%	50	Buzz 4 – 40%	91	Transition Ramp Up Medium Sharp 2 – 0 to 100%
10	Double Click – 100%	51	Buzz 5 – 20%	92	Transition Ramp Up Short Sharp 1 – 0 to 100%
11	Double Click – 60%	52	Pulsing Strong 1 – 100%	93	Transition Ramp Up Short Sharp 2 – 0 to 100%
12	Triple Click – 100%	53	Pulsing Strong 2 – 60%	94	Transition Ramp Down Long Smooth 1 – 50 to 0%
13	Soft Fuzz – 60%	54	Pulsing Medium 1 – 100%	95	Transition Ramp Down Long Smooth 2 – 50 to 0%
14	Strong Buzz – 100%	55	Pulsing Medium 2 – 60%	96	Transition Ramp Down Medium Smooth 1 – 50 to 0%
15	750 ms Alert 100%	56	Pulsing Sharp 1 – 100%	97	Transition Ramp Down Medium Smooth 2 – 50 to 0%
16	1000 ms Alert 100%	57	Pulsing Sharp 2 – 60%	98	Transition Ramp Down Short Smooth 1 – 50 to 0%
17	Strong Click 1 – 100%	58	Transition Click 1 – 100%	99	Transition Ramp Down Short Smooth 2 – 50 to 0%

18	Strong Click 2 – 80%		59	Transition Click 2 – 80%		100	Transition Ramp Down Long Sharp 1 – 50 to 0%
19	Strong Click 3 – 60%		60	Transition Click 3 – 60%		101	Transition Ramp Down Long Sharp 2 – 50 to 0%
20	Strong Click 4 – 30%		61	Transition Click 4 – 40%		102	Transition Ramp Down Medium Sharp 1 – 50 to 0%
21	Medium Click 1 – 100%		62	Transition Click 5 – 20%		103	Transition Ramp Down Medium Sharp 2 – 50 to 0%
22	Medium Click 2 – 80%		63	Transition Click 6 – 10%		104	Transition Ramp Down Short Sharp 1 – 50 to 0%
23	Medium Click 3 – 60%		64	Transition Hum 1 – 100%		105	Transition Ramp Down Short Sharp 2 – 50 to 0%
24	Sharp Tick 1 – 100%		65	Transition Hum 2 – 80%		106	Transition Ramp Up Long Smooth 1 – 0 to 50%
25	Sharp Tick 2 – 80%		66	Transition Hum 3 – 60%		107	Transition Ramp Up Long Smooth 2 – 0 to 50%
26	Sharp Tick 3 – 60%		67	Transition Hum 4 – 40%		108	Transition Ramp Up Medium Smooth 1 – 0 to 50%
27	Short Double Click Strong 1 – 100%		68	Transition Hum 5 – 20%		109	Transition Ramp Up Medium Smooth 2 – 0 to 50%
28	Short Double Click Strong 2 – 80%		69	Transition Hum 6 – 10%		110	Transition Ramp Up Short Smooth 1 – 0 to 50%
29	Short Double Click Strong 3 – 60%		70	Transition Ramp Down Long Smooth 1 – 100 to 0%		111	Transition Ramp Up Short Smooth 2 – 0 to 50%
30	Short Double Click Strong 4 – 30%		71	Transition Ramp Down Long Smooth 2 – 100 to 0%		112	Transition Ramp Up Long Sharp 1 – 0 to 50%
31	Short Double Click Medium 1 – 100%		72	Transition Ramp Down Medium Smooth 1 – 100 to 0%		113	Transition Ramp Up Long Sharp 2 – 0 to 50%
32	Short Double Click Medium 2 – 80%		73	Transition Ramp Down Medium Smooth 2 – 100 to 0%		114	Transition Ramp Up Medium Sharp 1 – 0 to 50%
33	Short Double Click Medium 3 – 60%		74	Transition Ramp Down Short Smooth 1 – 100 to 0%		115	Transition Ramp Up Medium Sharp 2 – 0 to 50%
34	Short Double Sharp Tick 1 – 100%		75	Transition Ramp Down Short Smooth 2 – 100 to 0%		116	Transition Ramp Up Short Sharp 1 – 0 to 50%
35	Short Double Sharp Tick 2 – 80%		76	Transition Ramp Down Long Sharp 1 – 100 to 0%		117	Transition Ramp Up Short Sharp 2 – 0 to 50%
36	Short Double Sharp Tick 3 – 60%		77	Transition Ramp Down Long Sharp 2 – 100 to 0%		118	Long buzz for programmatic stopping – 100%
37	Long Double Sharp Click Strong 1 – 100%		78	Transition Ramp Down Medium Sharp 1 – 100 to 0%		119	Smooth Hum 1 (No kick or brake pulse) – 50%
38	Long Double Sharp Click Strong 2 – 80%		79	Transition Ramp Down Medium Sharp 2 – 100 to 0%		120	Smooth Hum 2 (No kick or brake pulse) – 40%
39	Long Double Sharp Click Strong 3 – 60%		80	Transition Ramp Down Short Sharp 1 – 100 to 0%		121	Smooth Hum 3 (No kick or brake pulse) – 30%
40	Long Double Sharp Click Strong 4 – 30%		81	Transition Ramp Down Short Sharp 2 – 100 to 0%		122	Smooth Hum 4 (No kick or brake pulse) – 20%
41	Long Double Sharp Click Medium 1 – 100%		82	Transition Ramp Up Long Smooth 1 – 0 to 100%		123	Smooth Hum 5 (No kick or brake pulse) – 10%

Appendix B: Programming Examples

The following examples demonstrate how to program MDKM24 for adaptive visual and haptic effects. The examples represent the OEM system firmware and are written in C-like pseudocode.

Contents:

Example 1: Haptics at centerpoint and endpoints

Note how the example uses different haptic effects for the endpoints versus the centerpoint.

```
if (I2C_read_flag) {
  I2C_Master_Receive(KNOB_MODULE_TARGET_ADDRESS, i2c_rx_buffer, 1);
  I2C_read_flag = 0;
  read_position = i2c_rx_buffer[0];

  // Play haptic effect when Knob Module position reaches the lower bound
  if ((read_position == 0) && (previous_read_position != 0)) {
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS; // 0x02
    i2c_tx_buffer[1] = 0x0E; // Effect 0x0E is "Strong Buzz - 100%"
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  // Play haptic effect when Knob Module position reaches the upper bound
  else if ((read_position == 127) && (previous_read_position != 127)) {
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x0E; // Effect 0x0E is "Strong Buzz - 100%"
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  // Play haptic effect when Knob Module position reaches the centerpoint
  else if ((read_position == 64) && (previous_read_position != 64)) {
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x01; // Effect 0x01 is "Strong Click - 100%"
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  previous_read_position = read_position;
}
```

Example 2: Haptic virtual detents

This example creates 32 virtual detents using a haptic effect at increments of 4 position steps. Note how the example uses the haptic effect called “Sharp Tick” to emulate the sensation of mechanical detents.

Caution: The Knob Module position output sequence is interpolated from 96 to 128 steps such that every third out of four steps in [0–127] are skipped (2, 6, 10, . . .). This example or a similar program would not work for the condition `read_position % 4 == 2`.

```
if (I2C_read_flag) {
  I2C_Master_Receive(KNOB_MODULE_TARGET_ADDRESS, i2c_rx_buffer, 1);
  I2C_read_flag = 0;
  read_position = i2c_rx_buffer[0];

  // Play haptic effect for virtual detent
  if ((read_position != previous_read_position) && (read_position % 4 == 1)) {
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS; // 0x02
    i2c_tx_buffer[1] = 0x18; // Effect 0x18 is "Sharp Tick 1 – 100%"
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  previous_read_position = read_position;
}
```

Example 3: Rotary switch mode

This example makes the Knob Module behave as a rotary switch with five positions, where the positions are evenly spaced from the lower bound to the upper bound of the Magic Dot Display. Note how the example simultaneously involves setting the active position and playing haptic effects.

```
if (I2C_read_flag) {
  I2C_Master_Receive(KNOB_MODULE_TARGET_ADDRESS, i2c_rx_buffer, 1);
  I2C_read_flag = 0;
  read_position = i2c_rx_buffer[0];

  if ((read_position < previous_read_position) && (read_position < 32)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS; // 0x01
    i2c_tx_buffer[1] = 0; // Set active position
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS; // 0x02
    i2c_tx_buffer[1] = 0x25; // Effect 0x25 is "Long Double Sharp Click Strong 1 – 100%"
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  else if ((read_position > previous_read_position) && (read_position < 32)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS;
    i2c_tx_buffer[1] = 32;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x25;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  else if ((read_position < previous_read_position) && (read_position < 64)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS;
    i2c_tx_buffer[1] = 32;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x25;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  else if ((read_position > previous_read_position) && (read_position < 64) && (read_position > 32)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS;
    i2c_tx_buffer[1] = 64;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x25;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  else if ((read_position < previous_read_position) && (read_position < 96)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS;
    i2c_tx_buffer[1] = 64;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x25;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  // ... Repeat for (read_position < 96) && (read_position > 64) and setting active position = 96
  else if ((read_position > previous_read_position) && (read_position < 127) && (read_position > 96)) {
    i2c_tx_buffer[0] = KNOB_MODULE_POSITION_ADDRESS;
    i2c_tx_buffer[1] = 127;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
    i2c_tx_buffer[0] = KNOB_MODULE_HAPTIC_ADDRESS;
    i2c_tx_buffer[1] = 0x25;
    I2C_Master_Transmit(KNOB_MODULE_TARGET_ADDRESS, i2c_tx_buffer, 2);
  }
  previous_read_position = read_position;
}
```