Erae API

The ERAE Touch API is a custom sysex library with messages enabling you to take full control of the ERAE Touch point detection and LEDs states. Each message is formatted as described below:

API Messages

API Mode enable

Full sysex message:

```
  0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x01 RECEIVER PREFIX BYTES 0xF7
```

Message break down:

```
  0xF0                                                                     SysEx message begin

       0x00 0x21 0x50 0x00 0x01 0x00 0x01                                  Erae Touch identifier prefix

                                          0x01                             Erae Touch current MIDI network ID
                                                                      (for now always 0x01)

                                               0x01                        Erae Touch Service

                                                    0x04                   Erae Touch [Service] API

                                                         0x01              Erae Touch [Service API] enable
command

                                                              RECEIVER PREFIX BYTES        Messages sent by Erae Touch API to
receiver will begin with these bytes (at least 1 byte and maximum 16 bytes)

                                                                                    0xF7   SysEx message end
```

Please send an API Mode disable message before sending a new API Mode enable message

API Mode disable

Full sysex message:

```
  0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x02 0xF7
```

Message break down:

```
  0xF0                                                                     SysEx message begin

       0x00 0x21 0x50 0x00 0x01 0x00 0x01                                  Erae Touch identifier prefix

                                          0x01                             Erae Touch current MIDI network ID
(for now always 0x01)

                                               0x01                        Erae Touch Service

                                                    0x04                   Erae Touch [Service] API

                                                         0x02              Erae Touch [Service API] disable
command

                                                              0xF7         SysEx message end
```

API Fingerstream message

This message is sent by the Erae Touch to your receiving device when the API mode is enabled. Messages are sent only when touching an API Zone.

Full sysex message:

```
0xF0 RECEIVER PREFIX BYTES DAT1 DAT2 XYZ1 ... XYZ14 CHKS 0xF7
```

Message break down:

0xF0	SysEx message begin
RECEIVER PREFIX BYTES	Bytes chosen in API Mode enable message
DAT1	Action type bits aaa (click 0b000 /slide 0b001 /release 0b010) & finger index bits ffff (finger index between 0 and 9) : 0b0aaaffff
DAT2	Zone identifier 0b0zzzzzzz
XYZ1 ... XYZ14	14 7-bitized bytes encoding the 3 floats (X,Y,Z)
CHKS	CHECKSUM of the 14 7-bitized XYZ bytes
0xF7	SysEx message end

Zone Boundary Request message

The API mode must be enabled for this message to be sent by the Erae Touch to your receiving device.

Full sysex message:

```
0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x10 ZONE 0xF7
```

Message break down:

0xF0	SysEx message begin
0x00 0x21 0x50 0x00 0x01 0x00 0x01	Erae Touch identifier prefix
0x01	Erae Touch current MIDI network ID (for now always 0x01)
0x01	Erae Touch Service
0x04	Erae Touch [Service] API
0x10	Erae Touch [Service API] Zone Boundary Request command
ZONE	API zone index
0xF7	SysEx message end

Zone Boundary Reply message

Full sysex message:

```
0xF0 RECEIVER PREFIX BYTES 0x7F 0x01 ZONE Width Height 0xF7
```

The returned size of the zone is 0x7F 0x7F if the zone is not used in the Erae Touch

Message break down:

0xF0	SysEx message begin
RECEIVER PREFIX BYTES	Bytes chosen in API Mode enable message
0x7F	Non finger data byte
0x01	Zone boundary reply byte
ZONE	Zone index
Width Height	Width & Height of the zone

0xF7	SysEx message end

Clear Zone Display

Full sysex message:

```
0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x20 ZONE 0xF7
```

Message break down:

0xF0	SysEx message begin
0x00 0x21 0x50 0x00 0x01 0x00 0x01	Erae Touch identifier prefix
0x01	Erae Touch current MIDI network ID (for now always 0x01)
0x01	Erae Touch Service
0x04	Erae Touch [Service] API
0x20	Erae Touch [Service API] Clear Zone Display
ZONE	API Zone to be cleared
0xF7	SysEx message end

Draw pixel

Full sysex message:

```
0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x21 ZONE XPOS YPOS RED GREEN BLUE 0xF7
```

Message break down:

0xF0	SysEx message begin
0x00 0x21 0x50 0x00 0x01 0x00 0x01	Erae Touch identifier prefix
0x01	Erae Touch current MIDI network ID (for now always 0x01)
0x01	Erae Touch Service
0x04	Erae Touch [Service] API
0x21	Erae Touch [Service API] Draw pixel
ZONE	Target API Zone Index
XPOS YPOS	(x,y) coordinates of the led
RED GREEN BLUE	RGB value to set the led to (7 bits values for each color i.e. 0 to 127 range)
0xF7	SysEx message end

Draw rectangle

Full sysex message:

```
0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x22 ZONE XPOS YPOS WIDTH HEIGHT RED GREEN BLUE 0xF7
```

Message break down:

0xF0	SysEx message begin
0x00 0x21 0x50 0x00 0x01 0x00 0x01	Erae Touch identifier prefix
0x01	Erae Touch current MIDI network ID (for now always 0x01)
0x01	Erae Touch Service
0x04	Erae Touch [Service] API
0x22	Erae Touch [Service API] Draw rectangle
ZONE	Target API Zone index

XPOS YPOS	(x,y) coordinates of the bottom left edge of the rectangle
WIDTH HEIGHT	Width & height of the rectangle
RED GREEN BLUE	RGB value to set the led to (7 bits values for each color i.e. 0 to 127 range)
0xF7	SysEx message end

Draw image

Full sysex message:

0xF0 0x00 0x21 0x50 0x00 0x01 0x00 0x01 0x01 0x01 0x04 0x23 ZONE XPOS YPOS WIDTH HEIGHT BIN BIN … BIN CHKS 0xF7

Message break down:

0xF0	SysEx message begin
0x00 0x21 0x50 0x00 0x01 0x00 0x01	Erae Touch identifier prefix
0x01	Erae Touch current MIDI network ID (for now always 0x01)
0x01	Erae Touch Service
0x04	Erae Touch [Service] API
0x23	Erae Touch [Service API] Clear Zone Display
ZONE	Target API Zone index
XPOS YPOS	(x,y) coordinates of the bottom left edge of the image
WIDTH HEIGHT	Width & height of the image
BIN … BIN	7-bitized 24 bits RGB data of the pixels, going from left to right and bottom to top
CHKS	Checksum (XOR) of all the ‘BIN’ bytes
0xF7	SysEx message end

When displaying a large image, it will be best to break down the image into subimages with no more than 32 pixels each. This keeps the messages short enough for it to be managed by your operating system properly.

Example:
Draw an image on API Zone 1 at location (bottom left) x = 5, y = 3 of size width = 2 height = 2
We want to send 24 bit rgb data “white, red, green, blue” (0xFFFFFF, 0xFF0000, 0x00FF00, 0x0000FF) from left to right and bottom to top

The RGB data to be bitized is:
FF FF FF FF 00 00 00 FF 00 00 00 FF

Message:

F0 00 21 50 00 01 00 01 01 01 04 23 01 05 03 02 02 78 7F 7F 7F 7F 00 00 00 44 7F 00 00 00 7F 3C F7

Message break down:

Draw an image:	F0 00 21 50 00 01 00 01 01 01 04 23
API Zone 1:	01
Position x = 5, y = 3:	05 03
Width = 2, height = 2:	02 02
Bitized 24 bits RGB data:	78 7F 7F 7F 7F 00 00 00 44 7F 00 00 00 7F
Checksum of the bitized data:	3C
End of message:	F7

7-bit-izing in Python

```python
from functools import reduce

def bitize7chksum(byteArray):
  bitized7Arr = sum(([sum((el & 0x80) >> (j+1) for j,el in enumerate(data[i:min(i+7,len(data))]))] + [el & 0x7F for el in data[i:min(i+7,len(data))]] for i in range(0, len(data), 7)),[])
  return bitized7Arr + [reduce(lambda x,y: x^y, bitized7Arr)]
```

7-bit-izing & 7-un-bit-izing in C++

```cpp
#include <cstdint>
#include <cstddef>

/**
* @brief Get size of the resulting 7 bits bytes array obtained when using the bitize7 function
*/
constexpr size_t bitized7size(size_t len)
{
 return len / 7 * 8 + (len % 7 ? 1 + len % 7 : 0);
}

/**
* @brief Get size of the resulting 8 bits bytes array obtained when using the unbitize7 function
*/
constexpr size_t unbitized7size(size_t len)
{
 return len / 8 * 7 + (len % 8 ? len % 8 - 1 : 0);
}

/**
* @brief 7-bitize an array of bytes and get the resulting checksum
*
* @param in Input array of 8 bits bytes
* @param inlen Length in bytes of the input array of 8 bits bytes
* @param out An output array of bytes that will receive the 7-bitized bytes
* @return the output 7-bitized bytes XOR checksum
*/
constexpr uint8_t bitize7chksum(const uint8_t* in, size_t inlen, uint8_t* out)
{
 uint8_t chksum = 0;
 for (size_t i{0}, outsize{0}; i < inlen; i += 7, outsize += 8)
 {
   out[outsize] = 0;
   for (size_t j = 0; (j < 7) && (i + j < inlen); ++j)
   {
     out[outsize] |= (in[i + j] & 0x80) >> (j + 1);
     out[outsize + j + 1] = in[i + j] & 0x7F;
     chksum ^= out[outsize + j + 1];
   }
   chksum ^= out[outsize];
 }
 return chksum;
}

/**
* @brief 7-unbitize an array of bytes and get the incomming checksum
*
* @param in Input array of 7 bits bytes
* @param inlen Length in bytes of the input array of 7 bits bytes
* @param out An output array of bytes that will receive the 7-unbitized bytes
* @return the input 7-bitized bytes XOR checksum
*/
constexpr uint8_t unbitize7chksum(const uint8_t* in, size_t inlen, uint8_t* out)
```

```
{
 uint8_t chksum = 0;
 for (size_t i{0}, outsize{0}; i < inlen; i += 8, outsize += 7)
 {
   chksum ^= in[i];
   for (size_t j = 0; (j < 7) && (j + 1 + i < inlen); ++j)
   {
     out[outsize + j] = ((in[i] << (j + 1)) & 0x80) | in[i + j + 1];
     chksum ^= in[i + j + 1];
   }
 }
 return chksum;
}
```