Fingerprint 2 Click

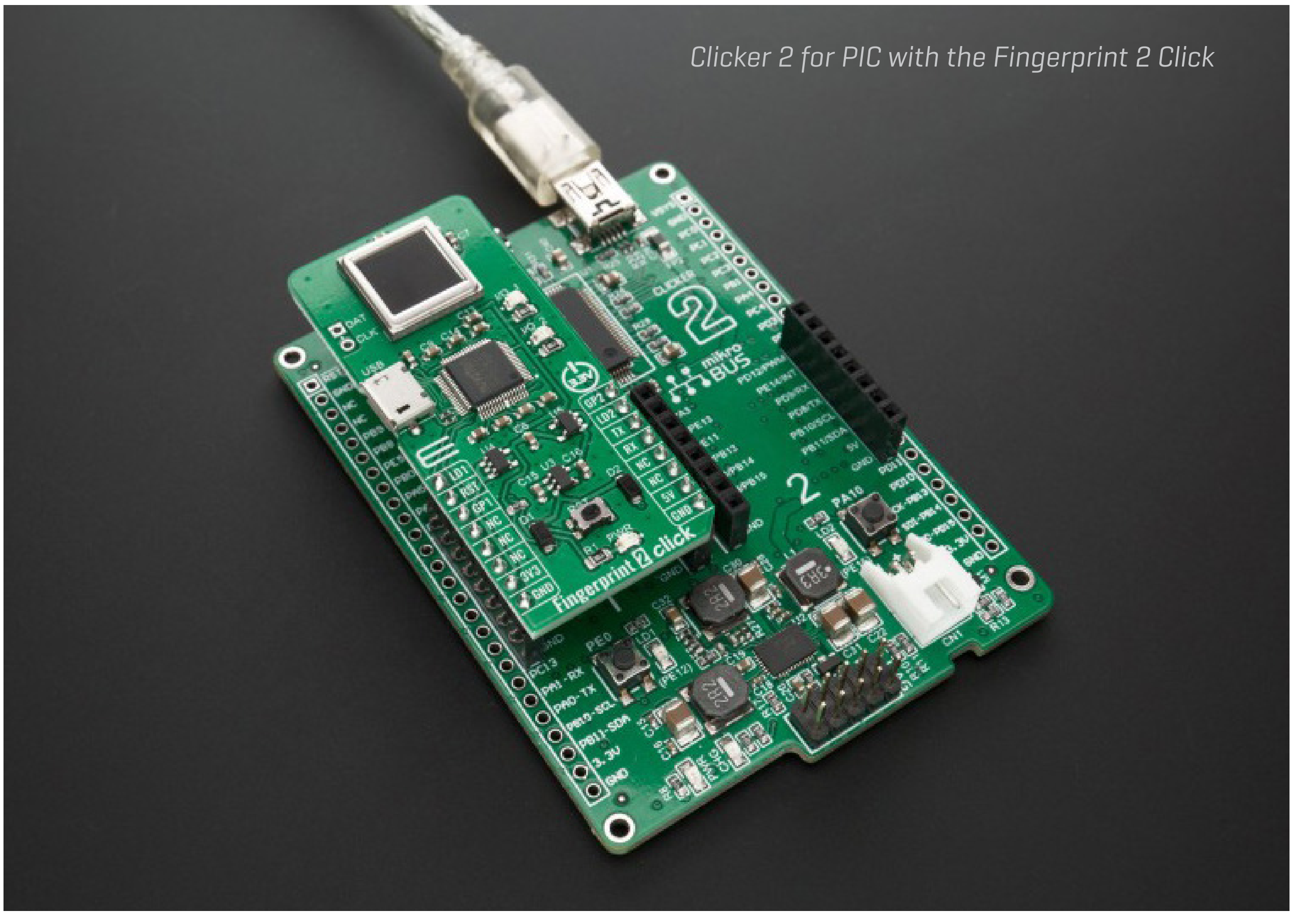
COMMUNICATION PROTOCOL

VERSION 1.0

MIKROE

1. Document scope

This document describes the communication protocol between Fingerprint 2 Click board (MIKROE-4119) and any development board with the mikroBUS connector (over UART interface) or PC. In this case the processor board or PC is Master (**Host**), and Fingerprint 2 Click is a Slave (**Device**).



The default communication channel is UART, with also USB (CDC class) supported. The UART baud rate of Fingerprint 2 Click is set to default configuration parameters:

- **Baud rate:** 9600
- **Parity:** NO
- **Data bits**: 8
- **Stop bit**: 1

and it can be changed to:

- **Baud rate**: 9600 / 19200 / 38400 / 57600 / 115200
- **Parity:** NO
- **Data bits:** 8
- **Stop bit**: 1

2. Protocol Format

The communication protocol format is in simply string form and it is case-sensitive.
Host sends command string to **Device**, all the ***command*** string is enclosed by ***<C>*** and ***</C>***, and **Device** responds with string enclosed by ***<R>*** and ***</R>***.

Basic response string:

OK_String: <R>OK</R>
NG_String: <R>NG</R>
FINISHED_String: <R>FINISHED</R>
PASS_String: <R>PASS</R>, <R>PASS_0</R> or <R>PASS_1</R>
FAIL_String: <R>FAIL</R>
OK_String: <R>OK</R>
INFO_String: <R>W=[width],H=[height]</R>

> *The [width] and [height] from "INFO String" are the width and height of fingerprint image dimension.*

3. Basic Command String

3.1 Register Fingerprint

Command string:

`<C>RegisterFingerprint</C>`

Host requests Device to register fingerprint, it will overwrite the stored registered fingerprint. It will register more fingerprints one by one up to 24 fingerprints at most, each one needs to enroll 3 times to do it.

Basic response string:

OK_String: `<R>OK</R>`
NG_String: `<R>NG</R>`

Device 's response:

1. Return OK_String
2. Device outputs fingerprint enroll and remove instructions to user
3. Return FINISHED_String when registration is complete

User can reset Device at any time when the registered fingerprint ID is stored.

3.2 Register One Fingerprint

Command string:

`<C>RegisterOneFp=xxx</C>`

"xxx" is the index to register fingerprint, and index 0 means the first registered fingerprint.

Host requests Device to register one fingerprint, it will overwrite the stored registered fingerprint with the same index. Each registration needs to enroll 3 times to do it.
Device 's response:

1. Device outputs fingerprint enroll and remove instructions to user
2. Return FINISHED_String when registration is complete, return FAIL_String if enroll is not well enough

3.3 Compare Fingerprint

Command string:

```
<C>CompareFingerprint</C>
```

Host requests Device to compare fingerprint. If there is no registered fingerprint, Device returns NG_String, otherwise it will act as below:

1. Return OK_String
2. Device outputs fingerprint enroll and remove instructions to user
3. If compare fails, it return FAIL_String

If compare succeeds, it returns:

`<R>PASS_0</R>` <= It matches with the first registered fingerprint.

`<R>PASS_1</R>` <= It matches with the second registered fingerprint.

And so on.

3.4 Query Fingerprint Information

Command string:

```
<C>FpImageInformation</C>
```

Host queries Device for the fingerprint image dimension, Device returns INFO_String, for example:

```
<R>W=300,H=600</R>
```

It means the fingerprint image is 300 pixels in width and 600 pixels in height. The image is a 8-bit gray image.

3.5 Scan and Upload Fingerprint Image

Command string:

```
<C>ScanFpImage</C>
```

Host requests Device to scan fingerprint image and send to Host, Device returns NG_String if it does not support this command, otherwise it will respond as below:

1. Return OK_String
2. Device outputs fingerprint enroll and remove instructions to user
3. Device enclose image binary data by <I> and </I> and send to Host

For the current Bynew EVB SDK1702, it is 176 pixels in width and 176 pixels in height for the fingerprint image, so there are 176x176=30976 bytes of image data will be enclosed by **<I>** and **</I>**.

3.6 Check Number of Registered Fingerprint

Command string:

<C>CheckRegisteredNo</C>

Device returns this value by ***<R>*** and ***</R>***, for example:

<R>0<R> It means there is no registered fingerprint.
<R>2<R> It means there are 2 registered fingerprints.
<R>10<R> It means there are 10 registered fingerprints.

And so on.

3.7 Set Baudrate

Command string:

<C>Baudrate=[baudrate]</C>
<C>Baudrate=9600</C> <= Set baudrate as 9600,N,8,1
<C>Baudrate=19200</C> <= Set baudrate as 19200,N,8,1
<C>Baudrate=38400</C> <= Set baudrate as 38400,N,8,1
<C>Baudrate=57600</C> <= Set baudrate as 57600,N,8,1
<C>Baudrate=115200</C> <= Set baudrate as 115200,N,8,1

If the setting value is not supported, **Device** will have no response, for example, **Device** will not respond to ***<C>Baudrate=50000</C>***. If the setting value is supported, **Device** returns OK_String to change baudrate accordingly, and then **Device** will reset itself to run from beginning. **Device** will keep this new baudrate until being changed again. Hence before new baudrate setting, **Host** must make sure itself can support the new baudrate, otherwise the communication between **Host** and **Device** will break after execution of baudrate setting command.

4. Communication Example:

```
<UART>
// Check the number of registered fingerprint
Host -> <C>CheckRegisteredNo</C> -> Device

// Device returns the number
Host <- <R>0<R> <- Device

// Register fingerprint
Host -> <C>RegisterFingerprint</C> -> Device

// Device return OK_String
Host <- <R>OK<R> <- Device
// Device output enroll instructions to user

// After registration complete, Device returns FINISHED_String
Host <- <R>FINISHED<R> <- Device

// Compare fingerprint
Host -> <C>CompareFingerprint</C> -> Device

// Device returns OK_String
Host <- <R>OK</R> <- Device
// Device output enroll instructions to user

// After compare, Device returns PASS_String or FAIL_String
Host <- <R>FAIL</R> <- Device or
Host <- <R>PASS_0</R> <- Device or
Host <- <R>PASS_1</R> <- Device

// Query fingerprint information
Host -> <C>FpImageInformation</C> -> Device

// Device return INFO_String
Host <- <R>W=176, H=176</R> <- Device
```

It means it is 176 pixels in width and 176 pixels in height for fingerprint image dimension. The size of fingerprint image is 176x176 = 30976 bytes.

```
// Scan and upload fingerprint image
Host -> <C>ScanFpImage</C> -> Device

//Device returns OK_String
Host <- <R>OK</R> <- Device
// Device output enroll instructions to user

// Device enclose image data and send to Host
Host <- <I>[30976 bytes of fingerprint image binary data]</I> <- Device

// Set baudrate 115200
Host -> <C>Baudrate=115200</C> -> Device

//Device returns OK_String
Host <- <R>OK</R> <- Device
```

Then Device will reset to 115200,N,8,1 to operate, Host must make sure it can also support this baudrate.

5. Extended Command String

5.1 Get Device FW Version

Command string:

<C>GetFWVer</C>

Device returns its version string enclosed by ***<R>*** and ***</R>***, "<R>0123</R>"means version 1.23.

5.2 Clear Registered Fingerprint

Command string:

<C>ClearRegisteredFp</C>

Device clear all registered fingerprints and the returns OK_String.

5.3 Clear One Fingerprint

Command string:

<C>ClearOneFp=xxx</C>

Device clear the registered fingerprint with index "xxx" and the returns OK_String or FAIL_String. Index 0 means the first registered fingerprint.

5.4 Query Device State

Command string:

<C>GetDS</C>

Device returns its state by <R>DS=HH</R>, HH is a 2-digit HEX number string, it has 8 bits defined as below:

Bit 0 – 1: password exists, 0: none
Bit 1 – 1: registered fingerprint exists, 0:none
Bit 2 – 1: unlocked, 0:locked
Bit 3 – 1: Demo mode, 0: Slave mode

Bit 4 – 1: Compare error 3 times will clear registered fingerprint in Demo mode, 0: it will not clear registered fingerprint
Bit 5 – 1: output system message, 0: don 't output system message
Bit 6 – (not used)
Bit 7 – 0

5.5 Query compare successful string

Command string:

<C>GetSuccStr</C>

Device returns <R>SUCC=sss</R>, "sss" is the output string when fingerprint compare successfully. For example, <R>SUCC=Pass! Welcome back. </R> It means Device will output string of "Pass! Welcome back." When compare successfully.

5.6 Query compare fail string

Command string:

<C>GetFailStr</C>

Device returns <R>FAIL=sss</R>, "sss" is the output string when fingerprint compare fail.

5.7 Set Output String When Compare Successfully

Command string:

<C>SetSuccStr=sss</C>

"sss" is the output string when fingerprint compare successfully. Device returns OK_String after store it.

5.8 Set Output String When Compare Fail

Command string:

<C>SetFailStr=sss</C>

"sss" is the output string when fingerprint compare fail. Device returns OK_String after store it.
For example, <C>SetFailStr=Fail! Try again.</C> It sets to output string of "Fail! Try again." When compares fail.

5.9 Unlock By Fingerprint

Command string:

<C>UnlockCompareFp</C>

Host requests Device to compare fingerprint in order to unlock device. If there is no registered fingerprint, Device returns NG_String, otherwise Device will respond as below:

1. Return OK_String
2. Device outputs fingerprint enroll and remove instructions to user
3. If compare fails, it return FAIL_String. If compare succeeds, it returns

<R>PASS_0</R> <= It matches with the first registered fingerprint.
<R>PASS_1</R> <= It matches with the second registered fingerprint

If compare succeeds, Device gets into unlocked state, otherwise it gets into locked state.

5.10 Unlock By Password

Command string:

<C>UnlockComparePWD=sss</C>

Host requests Device to check password in order to unlock device. "sss" is the password string to compare. If there is no stored password to compare, it returns NG_String, otherwise it returns OK_String when it compare successfully, or returns NG_String when compare fail.
If compare succeeds, Device gets into unlocked state, otherwise it gets into locked state.

5.11 Query Password

Command string:

<C>GetPWD</C>

If there is no stored password, Device returns NG_String, otherwise it returns <R>PWD=sss</R>, "sss" is the stored password string.

5.12 Set Password

Command string:

```
<C>SetPWD=sss</C>
```

"sss" is the password string to set, Device returns OK_String after store it.

5.13 Clear Password

Command string:

```
<C>ClearPWD</C>
```

Clear stored password. Device returns OK_String after delete it.

5.14 Lock Device

Command string:

```
<C>LockDevice</C>
```

This command will make Device get into locked state immediately, and returns OK_String. But if there is no stored password and no registered fingerprint, Device returns NG_String and will not get into locked state, because under such situation it cannot get into locked state.

5.15 Search KEY by ID

Command string:

```
<C>SearchKeyByID=sss</C>
```

To search stored KEY by ID string, "sss" is the ID string to search. Device returns `<R>KEY=sss</R>` if it exists, "sss" is the KEY string, otherwise Device returns NG_String.

5.16 Set KEY

Command string:

`<C>SetKey=sss</C>`

Set KEY as string of "sss", ID will be the ID string of <C>SearchKeyByID=sss</C>, so the previous command must be "Search KEY by ID ".
Device returns OK_String after store the KEY.

For example, send command <C>SearchKeyByID=bn@gmail.com</C> first,
and then <C>SetKey=bn888999</C>; It will set a data at secured area with ID is "bn@gmail.com" and KEY is "bn888999 ".

5.17 Delete Current KEY

Command string:

`<C>DeleteCurrentKey</C>`

Delete the KEY with the ID string of <C>SearchKeyByID=sss</C>, so the previous command must be "Search KEY by ID ". Device returns OK_String after clear the KEY.

5.18 Delete KEY By ID

Command string:

`<C>DeleteKeyByID=sss</C>`

Use ID string of "sss" to search and delete its KEY. If the KEY exists, Device will delete it and returns OK_String, otherwise it returns NG_String. For example, <C>DeleteKeyByID=MyHomeNumber</C>, it will search and delete the KEY with ID is "MyHomeNumber ".

5.19 Delete All KEY

Command string:

`<C>DeleteAllKey</C>`

If any KEY exists, Device will delete all the KEYs and returns OK_String, otherwise it returns NG_String.

5.20 List ALL KEY

Command string:

`<C>ListAllKey</C>`

Device lists all ID and KEY and then returns OK_String.

5.21 Set Timeout of Unlock

Command string:

`<C>UnlockTimeout=d</C>`

To set the timeout of unlock period, it will go back to locked state when time is up. “d” is a decimal digit string, unit is second, 0 means no timeout. Device accepts the setting and returns OK_String. For example, <C>UnlockTimeout=10</C>, it means the timeout period of unlocked state is 10 seconds, after 10 seconds, Device will go back from unlocked to locked state.

5.22 Query Timeout of Unlocked State

Command string:

`<C>GetUnlockTimeout</C>`

Device returns <R>Timeout=d</R>, d is the decimal digit string of timeout, o means no timeout setting.

5.23 Set Unlocked GPIO State

Command string:

`<C>SetUnlockGPIO=p,h,HHHHHHHH</C>`

p: '0'~'7' for GPIO port from PA; PB to PH.
h: '0'~'1' for lower or higher 8 GPIO pins.
HHHHHHHH: 8 HEX digits to indicate each corresponding GPIO pin setting.

Each pin has 4 bits as below:

Bit 3: 1 – the pin is in use, 0 – the pin is not in use
Bit 2 ~ Bit 1: (not used)
Bit 0: 1 – output 1 when unlocked state, 0 – output 0 when unlocked state.

For example, <C>SetUnlockGPIO=2,1,88888810</C>, it is to set PC higher 8 pins, PC.8 to PC.15. When in unlocked state, PC.8 outputs 0, PC.9 outputs 1, and PC.10 to PC.15 are ignored. When in locked state, the outputs are reversed, so PC.8 outputs 1, PC.9 outputs 0, and PC.10 to PC.15 are still ignored.
Device returns OK_String after store the setting.

5.24 Query Unlocked GPIO State

Command string:

<C>GetUnlockGPIO[p,h]</C>

Device returns <R>UnlockGPIO[p,h]=HHHHHHHH</R> to indicate the GPIO setting to Host. The meanings of p, h and HHHHHHHH are the same as above command "Set Unlocked GPIO State".
For example, <R>UnlockGPIO[2,1]=88888810</R>, it indicates that when in unlocked state, PC.8 outputs 0, PC.9 outputs 1, and PC.10 to PC.15 are ignored.

5.25 Enable System Message

Command string:

<C>EnableSysMsg</C>

Request Device to return some system message automatically, it includes more operation description. Device returns OK_String after accepting it.

5.26 Disable System Message

Command string:

<C>DisableSysMsg</C>

Request Device not to return some system message automatically, it includes more operation description. Device returns OK_String after accepting it.

5.27 Enable Registration When Compare Fail In Demo Mode

Command string:

<C>EnableErrRegFpInAuto</C>

In demo mode, if fingerprint compare fail more than or equal to 3 times, it will clear all registered fingerprint and request user to register again. Device returns OK_String after accepting it. It is default enabled.

5.28 Disable Registration When Compare Fail In Demo Mode

Command string:

```
<C>DisableErrRegFpInAuto</C>
```

In demo mode, even fingerprint compare fail, it will not clear all registered fingerprint and request user to register again. Device returns OK_String after accepting it.

5.29 Set Communication Channel

Command string:

<C>SetCommCh=x</C> To set UART or USB as the communication channel with host.

<C>SetCommCh=UART</C> <= To set UART as the communication channel default

<C>SetCommCh=USB</C> <= To set USB as the communication channel.

When set to USB channel, it is a virtual COM to host.

mikroe.com

mikroe.com/mikrobus

libstock.com