FPR-220 API Manual

1.mxComFingerDriver.class

1.1 Get Device Info

```
/**
 * This function returns version of Fingerprint Collection Module.
 *
 * @return version
 */
 int mxGetComDevVersion(String szDevNodeName, int nBaudRate, byte[] szVersion);
```

1.2 Detect Finger

```
/**
 * The device obtains the fingerprint image and returns the image area.
 *
 * @param oArea Fingerprint area percentage
 */
int mxGetImageAndArea(String szDevNodeName, int ibaud, int[] oArea);
```

1.3 Capture Image

```
/**
 * This function returns image captured from Fingerprint Collection Module.
 *
 * @param dwWaitTime Timeout time (unit: milliseconds)
 * @param lpImgData Image data (output parameter, at least 256*360 bytes)
 */
int mxGetFingerImageWithCompression(String szDevNodeName, int iBaudRate, int
dwWaitTime, byte[] lpImgData);
```

1.4 Enroll

```
//first step
/**
 * The device obtains the fingerprint image and returns the image area.
 *
 * @param oArea Fingerprint area percentage
 */
int mxGetImageAndArea(String szDevNodeName, int ibaud, int[] oArea);

//second step
/**
 * Extract fingerprint features from the in-device image buffer to the specified
storage area.
 *
 * @param iFlag   0 indicates the memory cache (BufferID is 0 to 2 feature area, 3
template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param PageID  the serial number of the feature storage buffer, which is the memory
area or flash area specified by Flag.
 */
int mxGenTz(String szDevNodeName, int ibaud, short iFlag, short PageID);
```

1.5 Verify

```
//first step
/**
 * The device obtains the fingerprint image and returns the image area.
 *
 * @param oArea Fingerprint area percentage
 */
int mxGetImageAndArea(String szDevNodeName, int ibaud, int[] oArea);

//second step
/**
 * Extract fingerprint features from the in-device image buffer to the specified
storage area.
 *
```

```
 * @param iFlag   0 indicates the memory cache (BufferID is 0 to 2 feature area, 3
template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param PageID  the serial number of the feature storage buffer, which is the memory
area or flash area specified by Flag.
 */
int mxGenTz(String szDevNodeName, int ibaud, short iFlag, short PageID);

//third step
/**
 * Specify the storage area data in the device for comparison.
 *
 * @param iFlag   0 indicates the memory cache (BufferID is 0 to 2 feature area, 3
template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param PageID  the serial number of the feature storage buffer, which is the memory
area or flash area specified by Flag.
 * @param oMatchResult 1 succeeds, other fails
 */
int mxMatch(String szDevNodeName, int ibaud, short iFlagA, short PageIDA, short iFlagB,
short PageIDB, int[] oMatchResult);
```

1.6 Search

```
//first step
/**
 * The device obtains the fingerprint image and returns the image area.
 *
 * @param oArea Fingerprint area percentage
 */
int mxGetImageAndArea(String szDevNodeName, int ibaud, int[] oArea);

//second step
/**
 * Extract fingerprint features from the in-device image buffer to the specified
storage area.
 *
 * @param iFlag   0 indicates the memory cache (BufferID is 0 to 2 feature area, 3
template area). 1 indicates that flash bufferids range from 0 to 999.
 * @param PageID  the serial number of the feature storage buffer, which is the memory
area or flash area specified by Flag.
 */
int mxGenTz(String szDevNodeName, int ibaud, short iFlag, short PageID);

//third step
/**
 * In-device storage area data for searching.
 *
 * @param iBufferID   Memory buffer ID 0,1,2.
```

```
 * @param iStartPage  The starting number of the searched FLASH is 0 or the upper limit of the storage capacity of the device. In the case of FLASH, it is 0~the upper limit of the storage capacity of the device
 * @param iPageNum Search span For example iStartPage = 0, iPageNum = 256, the search start address is 256 memory addresses starting from 0
 * @param oNum Output the address number of the search hit
 */
int mxSearch(String szDevNodeName, int ibaud, short iBufferID, short iStartPage, short iPageNum, short[] oNum);
```

1.7 Remove

```
/**
 * This function returns the result of delete the feature or template specified in memory buffer or flash.
 *
 * @param flag     0: for memory buffer, 1: for flash
 * @param bufferId if flag is 0, bufferId is 0, 1, 2, if flag is 1, flash page Id is 0~999
 */
int mxDeleteTz(String szDevNodeName, int ibaud, short iFlag, short PageID);
```

1.8 Clear

```
/**
 * This function returns the result of emptying the fingerprint repository.
 *
 */
int mxClearTz(String szDevNodeName, int ibaud);
```

1.9 Upload

```
/**
 * This function is to return fingerprint features to upload data.
 *
 * @param flag      0: for memory buffer, 1: for flash
 * @param PageID    if flag is 0, bufferId is 0, 1, 2, if flag is 1, flash page Id is 0~999
 * @param TzBuffer  Feature buffer (feature size is 1024B)
 * @param ioTzLen   Feature length
 */
int mxUpTz(String szDevNodeName, int ibaud, short iFlag, short PageID, byte[] TzBuffer, short[] ioTzLen);
```

1.10 Modify Fingerprint Device BaudRate

```
/**
 * Modify the baud rate of the fingerprint device
 *
 * @param NewBaudRate [1: 2400] [2: 9600] [3: 19200] [4: 57600] [5: 115200] [6: 230400]
[7: 460800] [8: 921600]
 */
int mxSetFingerBaudRate(String szDevNodeName, int iBaudRate, int dwWaitTime, int
NewBaudRate);
```

2. MxImage.class

```
public class MxImage {

    /**
     * Image width
     */
    public final int width;
    /**
     * Image width
     */
    public final int height;
    /**
     * Image data
     */
    public final byte[] data;
    /**
     * 1: Grayscale image, 3: rgb, 4:argb
     */
    public final int channels;
    /**
     * Tag
     */
    public Object tag;
}
```