/\*!

\* @file Adafruit\_Fingerprint.cpp

\*

\* @mainpage Adafruit Fingerprint Sensor Library

\*

\* @section intro\_sec Introduction

\*

\* This is a library for our optical Fingerprint sensor

\*

\* Designed specifically to work with the Adafruit Fingerprint sensor

\* ----> http://www.adafruit.com/products/751

\*

\* These displays use TTL Serial to communicate, 2 pins are required to

\* interface

\* Adafruit invests time and resources providing this open source code,

\* please support Adafruit and open-source hardware by purchasing

\* products from Adafruit!

\*

\* @section author Author

\*

\* Written by Limor Fried/Ladyada for Adafruit Industries.

\*

\* @section license License

\*

\* BSD license, all text above must be included in any redistribution

\*

\*/

#include "Adafruit\_Fingerprint.h"

//#define FINGERPRINT\_DEBUG

/\*!

\* @brief Gets the command packet

\*/

#define GET\_CMD\_PACKET(...) \

uint8\_t data[] = {\_\_VA\_ARGS\_\_}; \

Adafruit\_Fingerprint\_Packet packet(FINGERPRINT\_COMMANDPACKET, sizeof(data), \

data); \

writeStructuredPacket(packet); \

if (getStructuredPacket(&packet) != FINGERPRINT\_OK) \

return FINGERPRINT\_PACKETRECIEVEERR; \

if (packet.type != FINGERPRINT\_ACKPACKET) \

return FINGERPRINT\_PACKETRECIEVEERR;

/\*!

\* @brief Sends the command packet

\*/

#define SEND\_CMD\_PACKET(...) \

GET\_CMD\_PACKET(\_\_VA\_ARGS\_\_); \

return packet.data[0];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PUBLIC FUNCTIONS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#if defined(\_\_AVR\_\_) || defined(ESP8266) || defined(FREEDOM\_E300\_HIFIVE1)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Instantiates sensor with Software Serial

@param ss Pointer to SoftwareSerial object

@param password 32-bit integer password (default is 0)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Adafruit\_Fingerprint::Adafruit\_Fingerprint(SoftwareSerial \*ss,

uint32\_t password) {

thePassword = password;

theAddress = 0xFFFFFFFF;

hwSerial = NULL;

swSerial = ss;

mySerial = swSerial;

}

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Instantiates sensor with Hardware Serial

@param hs Pointer to HardwareSerial object

@param password 32-bit integer password (default is 0)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Adafruit\_Fingerprint::Adafruit\_Fingerprint(HardwareSerial \*hs,

uint32\_t password) {

thePassword = password;

theAddress = 0xFFFFFFFF;

#if defined(\_\_AVR\_\_) || defined(ESP8266) || defined(FREEDOM\_E300\_HIFIVE1)

swSerial = NULL;

#endif

hwSerial = hs;

mySerial = hwSerial;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Instantiates sensor with a stream for Serial

@param serial Pointer to a Stream object

@param password 32-bit integer password (default is 0)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Adafruit\_Fingerprint::Adafruit\_Fingerprint(Stream \*serial, uint32\_t password) {

thePassword = password;

theAddress = 0xFFFFFFFF;

hwSerial = NULL;

#if defined(\_\_AVR\_\_) || defined(ESP8266) || defined(FREEDOM\_E300\_HIFIVE1)

swSerial = NULL;

#endif

mySerial = serial;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Initializes serial interface and baud rate

@param baudrate Sensor's UART baud rate (usually 57600, 9600 or 115200)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Adafruit\_Fingerprint::begin(uint32\_t baudrate) {

delay(1000); // one second delay to let the sensor 'boot up'

if (hwSerial)

hwSerial->begin(baudrate);

#if defined(\_\_AVR\_\_) || defined(ESP8266) || defined(FREEDOM\_E300\_HIFIVE1)

if (swSerial)

swSerial->begin(baudrate);

#endif

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Verifies the sensors' access password (default password is

0x0000000). A good way to also check if the sensors is active and responding

@returns True if password is correct

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

boolean Adafruit\_Fingerprint::verifyPassword(void) {

return checkPassword() == FINGERPRINT\_OK;

}

uint8\_t Adafruit\_Fingerprint::checkPassword(void) {

GET\_CMD\_PACKET(FINGERPRINT\_VERIFYPASSWORD, (uint8\_t)(thePassword >> 24),

(uint8\_t)(thePassword >> 16), (uint8\_t)(thePassword >> 8),

(uint8\_t)(thePassword & 0xFF));

if (packet.data[0] == FINGERPRINT\_OK)

return FINGERPRINT\_OK;

else

return FINGERPRINT\_PACKETRECIEVEERR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Get the sensors parameters, fills in the member variables

status\_reg, system\_id, capacity, security\_level, device\_addr, packet\_len

and baud\_rate

@returns True if password is correct

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::getParameters(void) {

GET\_CMD\_PACKET(FINGERPRINT\_READSYSPARAM);

status\_reg = ((uint16\_t)packet.data[1] << 8) | packet.data[2];

system\_id = ((uint16\_t)packet.data[3] << 8) | packet.data[4];

capacity = ((uint16\_t)packet.data[5] << 8) | packet.data[6];

security\_level = ((uint16\_t)packet.data[7] << 8) | packet.data[8];

device\_addr = ((uint32\_t)packet.data[9] << 24) |

((uint32\_t)packet.data[10] << 16) |

((uint32\_t)packet.data[11] << 8) | (uint32\_t)packet.data[12];

packet\_len = ((uint16\_t)packet.data[13] << 8) | packet.data[14];

if (packet\_len == 0) {

packet\_len = 32;

} else if (packet\_len == 1) {

packet\_len = 64;

} else if (packet\_len == 2) {

packet\_len = 128;

} else if (packet\_len == 3) {

packet\_len = 256;

}

baud\_rate = (((uint16\_t)packet.data[15] << 8) | packet.data[16]) \* 9600;

return packet.data[0];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to take an image of the finger pressed on surface

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_NOFINGER</code> if no finger detected

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

@returns <code>FINGERPRINT\_IMAGEFAIL</code> on imaging error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::getImage(void) {

SEND\_CMD\_PACKET(FINGERPRINT\_GETIMAGE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to convert image to feature template

@param slot Location to place feature template (put one in 1 and another in

2 for verification to create model)

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_IMAGEMESS</code> if image is too messy

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

@returns <code>FINGERPRINT\_FEATUREFAIL</code> on failure to identify

fingerprint features

@returns <code>FINGERPRINT\_INVALIDIMAGE</code> on failure to identify

fingerprint features

\*/

uint8\_t Adafruit\_Fingerprint::image2Tz(uint8\_t slot) {

SEND\_CMD\_PACKET(FINGERPRINT\_IMAGE2TZ, slot);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to take two print feature template and create a

model

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

@returns <code>FINGERPRINT\_ENROLLMISMATCH</code> on mismatch of fingerprints

\*/

uint8\_t Adafruit\_Fingerprint::createModel(void) {

SEND\_CMD\_PACKET(FINGERPRINT\_REGMODEL);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to store the calculated model for later matching

@param location The model location #

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_BADLOCATION</code> if the location is invalid

@returns <code>FINGERPRINT\_FLASHERR</code> if the model couldn't be written

to flash memory

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

uint8\_t Adafruit\_Fingerprint::storeModel(uint16\_t location) {

SEND\_CMD\_PACKET(FINGERPRINT\_STORE, 0x01, (uint8\_t)(location >> 8),

(uint8\_t)(location & 0xFF));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to load a fingerprint model from flash into buffer 1

@param location The model location #

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_BADLOCATION</code> if the location is invalid

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

uint8\_t Adafruit\_Fingerprint::loadModel(uint16\_t location) {

SEND\_CMD\_PACKET(FINGERPRINT\_LOAD, 0x01, (uint8\_t)(location >> 8),

(uint8\_t)(location & 0xFF));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to transfer 256-byte fingerprint template from the

buffer to the UART

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

uint8\_t Adafruit\_Fingerprint::getModel(void) {

SEND\_CMD\_PACKET(FINGERPRINT\_UPLOAD, 0x01);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to delete a model in memory

@param location The model location #

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_BADLOCATION</code> if the location is invalid

@returns <code>FINGERPRINT\_FLASHERR</code> if the model couldn't be written

to flash memory

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

uint8\_t Adafruit\_Fingerprint::deleteModel(uint16\_t location) {

SEND\_CMD\_PACKET(FINGERPRINT\_DELETE, (uint8\_t)(location >> 8),

(uint8\_t)(location & 0xFF), 0x00, 0x01);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to delete ALL models in memory

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_BADLOCATION</code> if the location is invalid

@returns <code>FINGERPRINT\_FLASHERR</code> if the model couldn't be written

to flash memory

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

uint8\_t Adafruit\_Fingerprint::emptyDatabase(void) {

SEND\_CMD\_PACKET(FINGERPRINT\_EMPTY);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to search the current slot 1 fingerprint features to

match saved templates. The matching location is stored in <b>fingerID</b> and

the matching confidence in <b>confidence</b>

@returns <code>FINGERPRINT\_OK</code> on fingerprint match success

@returns <code>FINGERPRINT\_NOTFOUND</code> no match made

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::fingerFastSearch(void) {

// high speed search of slot #1 starting at page 0x0000 and page #0x00A3

GET\_CMD\_PACKET(FINGERPRINT\_HISPEEDSEARCH, 0x01, 0x00, 0x00, 0x00, 0xA3);

fingerID = 0xFFFF;

confidence = 0xFFFF;

fingerID = packet.data[1];

fingerID <<= 8;

fingerID |= packet.data[2];

confidence = packet.data[3];

confidence <<= 8;

confidence |= packet.data[4];

return packet.data[0];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Control the built in LED

@param on True if you want LED on, False to turn LED off

@returns <code>FINGERPRINT\_OK</code> on success

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::LEDcontrol(bool on) {

if (on) {

SEND\_CMD\_PACKET(FINGERPRINT\_LEDON);

} else {

SEND\_CMD\_PACKET(FINGERPRINT\_LEDOFF);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Control the built in Aura LED (if exists). Check datasheet/manual

for different colors and control codes available

@param control The control code (e.g. breathing, full on)

@param speed How fast to go through the breathing/blinking cycles

@param coloridx What color to light the indicator

@param count How many repeats of blinks/breathing cycles

@returns <code>FINGERPRINT\_OK</code> on fingerprint match success

@returns <code>FINGERPRINT\_NOTFOUND</code> no match made

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::LEDcontrol(uint8\_t control, uint8\_t speed,

uint8\_t coloridx, uint8\_t count) {

SEND\_CMD\_PACKET(FINGERPRINT\_AURALEDCONFIG, control, speed, coloridx, count);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor to search the current slot fingerprint features to

match saved templates. The matching location is stored in <b>fingerID</b> and

the matching confidence in <b>confidence</b>

@param slot The slot to use for the print search, defaults to 1

@returns <code>FINGERPRINT\_OK</code> on fingerprint match success

@returns <code>FINGERPRINT\_NOTFOUND</code> no match made

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::fingerSearch(uint8\_t slot) {

// search of slot starting thru the capacity

GET\_CMD\_PACKET(FINGERPRINT\_SEARCH, slot, 0x00, 0x00, (uint8\_t)(capacity >> 8),

(uint8\_t)(capacity & 0xFF));

fingerID = 0xFFFF;

confidence = 0xFFFF;

fingerID = packet.data[1];

fingerID <<= 8;

fingerID |= packet.data[2];

confidence = packet.data[3];

confidence <<= 8;

confidence |= packet.data[4];

return packet.data[0];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Ask the sensor for the number of templates stored in memory. The

number is stored in <b>templateCount</b> on success.

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::getTemplateCount(void) {

GET\_CMD\_PACKET(FINGERPRINT\_TEMPLATECOUNT);

templateCount = packet.data[1];

templateCount <<= 8;

templateCount |= packet.data[2];

return packet.data[0];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Set the password on the sensor (future communication will require

password verification so don't forget it!!!)

@param password 32-bit password code

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_PACKETRECIEVEERR</code> on communication error

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t Adafruit\_Fingerprint::setPassword(uint32\_t password) {

SEND\_CMD\_PACKET(FINGERPRINT\_SETPASSWORD, (uint8\_t)(password >> 24),

(uint8\_t)(password >> 16), (uint8\_t)(password >> 8),

(uint8\_t)(password & 0xFF));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Helper function to process a packet and send it over UART to the

sensor

@param packet A structure containing the bytes to transmit

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Adafruit\_Fingerprint::writeStructuredPacket(

const Adafruit\_Fingerprint\_Packet &packet) {

mySerial->write((uint8\_t)(packet.start\_code >> 8));

mySerial->write((uint8\_t)(packet.start\_code & 0xFF));

mySerial->write(packet.address[0]);

mySerial->write(packet.address[1]);

mySerial->write(packet.address[2]);

mySerial->write(packet.address[3]);

mySerial->write(packet.type);

uint16\_t wire\_length = packet.length + 2;

mySerial->write((uint8\_t)(wire\_length >> 8));

mySerial->write((uint8\_t)(wire\_length & 0xFF));

#ifdef FINGERPRINT\_DEBUG

Serial.print("-> 0x");

Serial.print((uint8\_t)(packet.start\_code >> 8), HEX);

Serial.print(", 0x");

Serial.print((uint8\_t)(packet.start\_code & 0xFF), HEX);

Serial.print(", 0x");

Serial.print(packet.address[0], HEX);

Serial.print(", 0x");

Serial.print(packet.address[1], HEX);

Serial.print(", 0x");

Serial.print(packet.address[2], HEX);

Serial.print(", 0x");

Serial.print(packet.address[3], HEX);

Serial.print(", 0x");

Serial.print(packet.type, HEX);

Serial.print(", 0x");

Serial.print((uint8\_t)(wire\_length >> 8), HEX);

Serial.print(", 0x");

Serial.print((uint8\_t)(wire\_length & 0xFF), HEX);

#endif

uint16\_t sum = ((wire\_length) >> 8) + ((wire\_length)&0xFF) + packet.type;

for (uint8\_t i = 0; i < packet.length; i++) {

mySerial->write(packet.data[i]);

sum += packet.data[i];

#ifdef FINGERPRINT\_DEBUG

Serial.print(", 0x");

Serial.print(packet.data[i], HEX);

#endif

}

mySerial->write((uint8\_t)(sum >> 8));

mySerial->write((uint8\_t)(sum & 0xFF));

#ifdef FINGERPRINT\_DEBUG

Serial.print(", 0x");

Serial.print((uint8\_t)(sum >> 8), HEX);

Serial.print(", 0x");

Serial.println((uint8\_t)(sum & 0xFF), HEX);

#endif

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

@brief Helper function to receive data over UART from the sensor and

process it into a packet

@param packet A structure containing the bytes received

@param timeout how many milliseconds we're willing to wait

@returns <code>FINGERPRINT\_OK</code> on success

@returns <code>FINGERPRINT\_TIMEOUT</code> or

<code>FINGERPRINT\_BADPACKET</code> on failure

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t

Adafruit\_Fingerprint::getStructuredPacket(Adafruit\_Fingerprint\_Packet \*packet,

uint16\_t timeout) {

uint8\_t byte;

uint16\_t idx = 0, timer = 0;

#ifdef FINGERPRINT\_DEBUG

Serial.print("<- ");

#endif

while (true) {

while (!mySerial->available()) {

delay(1);

timer++;

if (timer >= timeout) {

#ifdef FINGERPRINT\_DEBUG

Serial.println("Timed out");

#endif

return FINGERPRINT\_TIMEOUT;

}

}

byte = mySerial->read();

#ifdef FINGERPRINT\_DEBUG

Serial.print("0x");

Serial.print(byte, HEX);

Serial.print(", ");

#endif

switch (idx) {

case 0:

if (byte != (FINGERPRINT\_STARTCODE >> 8))

continue;

packet->start\_code = (uint16\_t)byte << 8;

break;

case 1:

packet->start\_code |= byte;

if (packet->start\_code != FINGERPRINT\_STARTCODE)

return FINGERPRINT\_BADPACKET;

break;

case 2:

case 3:

case 4:

case 5:

packet->address[idx - 2] = byte;

break;

case 6:

packet->type = byte;

break;

case 7:

packet->length = (uint16\_t)byte << 8;

break;

case 8:

packet->length |= byte;

break;

default:

packet->data[idx - 9] = byte;

if ((idx - 8) == packet->length) {

#ifdef FINGERPRINT\_DEBUG

Serial.println(" OK ");

#endif

return FINGERPRINT\_OK;

}

break;

}

idx++;

}

// Shouldn't get here so...

return FINGERPRINT\_BADPACKET;

}