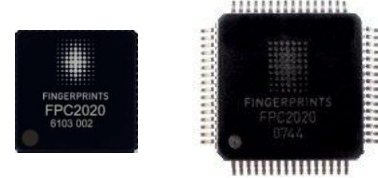


## Features

- Extremely easy to integrate minimizing time-to-market
- Supports area sensor FPC1011C
- One-To-One verification mode
- One-To-Many identification mode
- Flash interface for program and template storage
- Straightforward serial command interface
- Download/upload template functionality
- Single supply 2.5 – 3.3 V

## Application examples

- Access control systems
- Time & Attendance
- Locks, safes
- POS terminals



## General description

FPC2020 is an extremely small, fast and power efficient ASIC that acts as a biometric sub-system with a direct interface to the FPC1011C sensor as well as to an external flash memory for storing templates. Thanks to its small size and low power consumption it fits as well in door locks, card readers and safes as in smaller portable and battery powered devices without losing identification speed or performance.

FPC2020 can easily be integrated into virtually any application and be controlled by a host sending basic commands for enrolment and verification via the serial interface. Fingerprint templates are created automatically and stored in flash memory connected to FPC2020. Templates used for verification can also be uploaded/downloaded to an external storage, e.g. central database, smart card or portable flash memory. FPC2020 has no internal limitation in number of templates it can handle. Size of external flash memory will set the limitation.

## Quick reference data

PARAMETER	VALUE
Package	80 pin TQFP 12*12, 64 pin QFN 9 * 9 mm*
Number of templates	(2 Mbit Flash) 223 (4 Mbit Flash) 479 (8 Mbit Flash) 991**
Verification time	(1:1) 0.2 s (typical)
Identification time	(1:150) 1.0 s (typical) (1:500) 2.0 s (typical)
Enrolment time	7 s (typical)
Template Size	938 bytes
False-Rejection-Rate (FRR)	Adjustable, Data dependent
False-Acceptance-Rate (FAR)	Adjustable from 1/1,000 to 1/100,000
Host interface	Serial UART or SPI
Flash interface	SPI
Sensor interface	Direct interface to FPC1011C
Supply voltage	2,5 to 3.3 VDC
Average supply current active 3,3 V	70mA @ full speed, 25 mA @ half speed
Supply current idle mode 3,3 V	10 mA
Supply current sleep mode 3,3V	30 $\mu$ A
Crystal	7,3728 MHz
Frequency	7 – 96 MHz Depending on workload and power mode

\* The QFN package is only available in large volumes

\*\* Recommended number of templates during Identification: <500

This page is intentionally left blank.

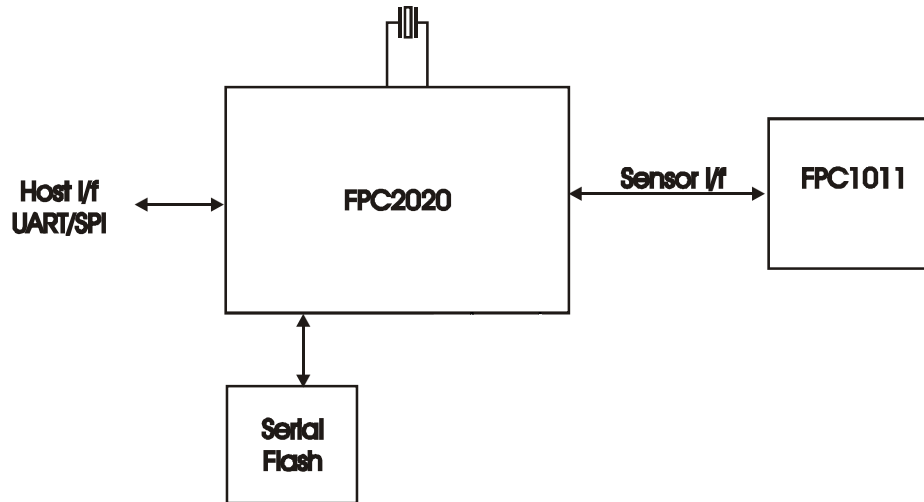
## Table of contents

Features.....	1
Application examples .....	1
General description .....	1
Quick reference data.....	1
Table of contents.....	3
Functional description .....	5
System overview .....	5
General description .....	5
Algorithm performance .....	5
Performance characteristics.....	6
Electrical characteristics.....	6
Normal operation.....	6
Absolute maximum ratings .....	7
Pin configuration .....	7
Software command interface.....	9
Serial interface settings .....	9
Command send structure .....	9
Response structure .....	10
SPI Timing Requirements.....	10
Table of commands.....	11
Table of response bytes .....	13
Command description .....	14
Capture image API_CAPTURE_IMAGE.....	14
Capture and enrol (RAM) API_CAPTURE_AND_ENROL_RAM.....	14
Capture and verify (RAM) API_CAPTURE_AND_VERIFY_RAM.....	14
Capture and verify (FLASH) API_CAPTURE_AND_VERIFY_FLASH.....	14
Capture and identify (FLASH) API_CAPTURE_AND_IDENTIFY_FLASH.....	15
Enrol (RAM) API_ENROL_RAM .....	15
Verify (RAM) API_VERIFY_RAM .....	15
Verify (FLASH) API_VERIFY_FLASH .....	15
Identify (FLASH) API_IDENTIFY_FLASH.....	16
Enrol (FLASH) API_ENROL_FLASH.....	16
Capture Enrol (FLASH) API_CAPTURE_AND_ENROL_FLASH .....	16
Upload image API_UPLOAD_IMAGE.....	16
Download image API_DOWNLOAD_IMAGE.....	16
Upload template API_UPLOAD_TEMPLATE .....	17
Download template API_DOWNLOAD_TEMPLATE .....	17
Copy template from RAM to FLASH API_COPY_TEMPLATE_FROM_RAM_TO_FLASH .....	17
Upload template from FLASH API_UPLOAD_TEMPLATE_FROM_FLASH.....	17
Delete template in RAM API_DELETE_TEMPLATE_RAM .....	17
Delete single template in FLASH API_DELETE_SLOT_IN_FLASH.....	17
Delete all templates in FLASH API_DELETE_ALL_IN_FLASH.....	18
Download template to FLASH API_DOWNLOAD_TEMPLATE_TO_FLASH .....	18
Security level (RAM) API_SECURITY_LEVEL_RAM.....	18
Security level (STATIC) API_SECURITY_LEVEL_STATIC.....	18
Get current security level API_GET_SECURITY_LEVEL.....	19
Firmware version API_FIRMWARE_VERSION.....	19
Firmware update API_FIRMWARE_UPDATE .....	19
Set baud rate (RAM) API_SET_BAUD_RATE_RAM.....	19
Set baud rate (STATIC) API_SET_BAUD_RATE_STATIC .....	20
Test hardware API_TEST_HARDWARE .....	20
Cancel current command API_CANCEL .....	20
Enter sleep mode API_ENTER_SLEEP_MODE .....	21
Power save mode (RAM) API_POWER_SAVE_MODE_RAM .....	21
Power save mode (STATIC) API_POWER_SAVE_MODE_STATIC.....	21
Get current power save mode API_GET_POWER_SAVE_MODE.....	21
Manage Advance Settings API_ADVANCED_SETTINGS .....	22
CRC calculation .....	23
Power Management.....	23
Boot Mode.....	25
FPC2020 QFN64 mechanical outline.....	27
FPC2020 TQFP80 mechanical outline.....	28
Pin assignment.....	29
Area sensor interface .....	30

Reset .....	31
Brown Out detection .....	31
Flash interface .....	32
Host interface .....	32
Clock .....	32
Schematic .....	34
Bom.....	35
Document revision history .....	36
Firmware information .....	36
Contact information .....	36

## Functional description

### System overview



**Figure 1**  
FPC2020 system configuration

### General description

Everything needed to form a biometric system based on FPC2020 is outlined in figure 1. The program code and the templates will be stored in a non-volatile external serial flash memory. At power on (boot) FPC2020 will download the program from flash to an internal program RAM and start the program execution. Except for the flash memory an external inexpressive crystal is needed in the system.

This, together with a sensor, is basically what is needed for a complete fingerprint identification system.

The sensor FPC1011C can be directly connected to FPC2020.

In Area mode, FPC2020 can also be used without the attached FLASH memory – in this case, all the FLASH related software commands will not work, and the application itself must be transmitted from the host upon system start.

The host CPU, selected and provided by the customer, is executing the main application, interfacing the FPC2020. Requirements on the host processor associated with the module communication are extremely low. Hence the host processor can be selected entirely to suit the main application at hand.

All CPU capabilities in FPC2020 processor are used to perform biometric related functions. Software (firmware) is supplied by Fingerprint

Cards and there is no space available for customer applications.

The FPC2020 processor acquires the fingerprint image from the fingerprint sensor. Thus there is no direct interaction between the host processor and the fingerprint sensor.

The interface between the host processor and the FPC2020 processor board is based on a simple-to-use serial UART or SPI command interface.

FPC2020 should be supplied with a single supply in the range of 2.5 to 3.3 V. FPC2020 will internally generate the needed core voltage of 1.8 V.

### Algorithm performance

The pre-loaded firmware is based on the high performing DAD algorithm developed by Fingerprint Cards.

## Performance characteristics

SYMBOL	PARAMETER	CONDITION	Verification	Identification	Identification	UNIT
			1 USER	10 USERS	100 USERS	
FRR	False-rejection-rate	High convenience	< 1	< 1	< 4	%
		Default	< 1	< 1	< 5	%
		High security	< 2	< 2	< 7	%
FAR	False-acceptance-rate	High convenience	< 0.1	< 1	1	%
		Default	< 0.01	< 0.1	0.1	%
		High security	< 0.001	< 0.01	0.01	%

**Table 1**

Algorithm performance characteristics, from a dataset of habituated users.

PROCESS	POWER SAVE MODE OFF			POWER SAVE MODE ON			UNIT
	MIN	TYP	MAX	MIN	TYP	MAX	
Enrolment	5	7	10	14	20	29	s
Verification 1:1	0.15	0.20	0.42	0.50	0.70	1.7	s
Identification 1:36	0.50	0.70	1.2	1.2	1.5	3.1	s
Identification 1:500	1.7	2.0	2.9	5.5	7.5	10	s

**Table 2**

Algorithm processing speed.

## Electrical characteristics

### Normal operation

Operating temperature: -20°C to +85°C

SYMBOL	PARAMETER	CONDITION	MIN	TYP	MAX	UNIT
V <sub>DD</sub>	Supply voltage		2.35	2.5-3.3	3.45	V
I <sub>DD</sub>	Supply current 3,3 V	Idle		10		mA
		Active full speed		70*		mA
		Active half speed		25*		mA
		Sleep		30		µA
<i>Digital inputs</i>						
V <sub>IL</sub>	Logic '0' voltage		0		0.8	V
V <sub>IH</sub>	Logic '1' voltage		0.7V <sub>DD</sub>		V <sub>DD</sub> +0.3	V
I <sub>IL</sub>	Logic '0' current (V <sub>I</sub> = GND)				±10	µA
I <sub>IH</sub>	Logic '1' current (V <sub>I</sub> = V <sub>DD</sub> )				±10	µA
<i>Digital outputs</i>						
V <sub>OL</sub>	Logic '0' output voltage				0.45	V
V <sub>OH</sub>	Logic '1' output voltage		0.85V <sub>DD</sub>			V

**Table 3**

Electrical characteristics

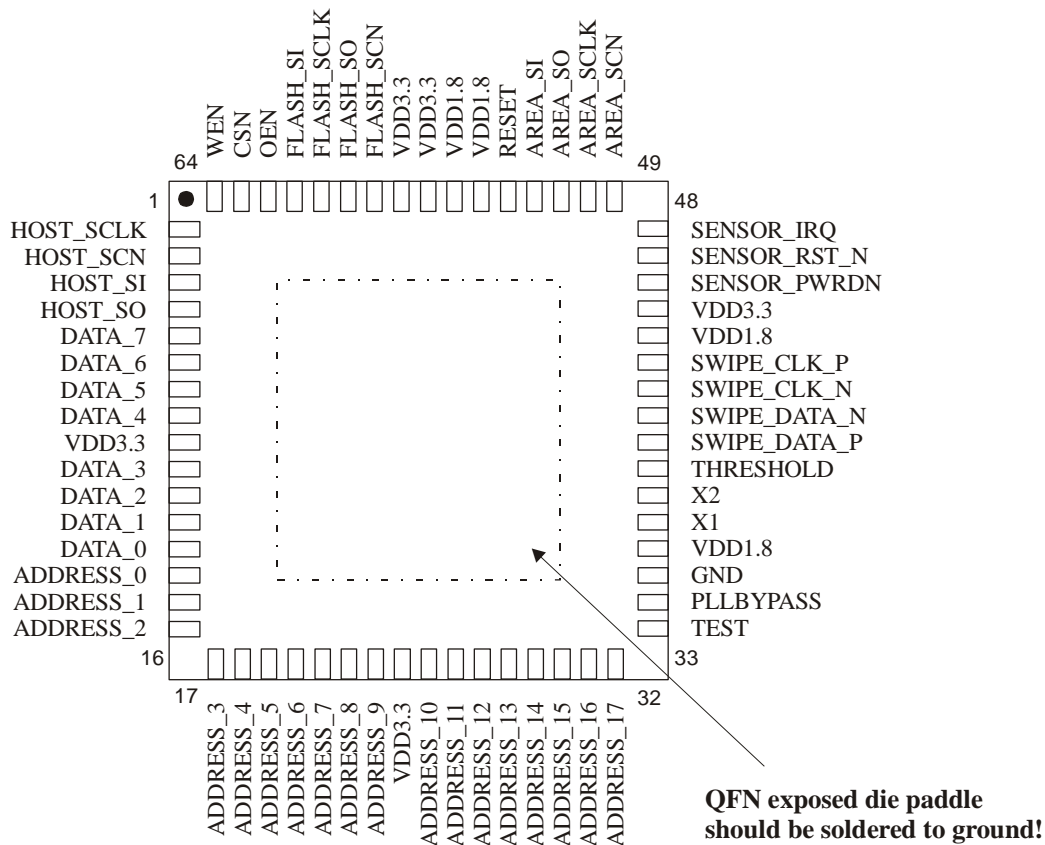
\* The average supply current for active mode during fingerprint verification.

### Absolute maximum ratings

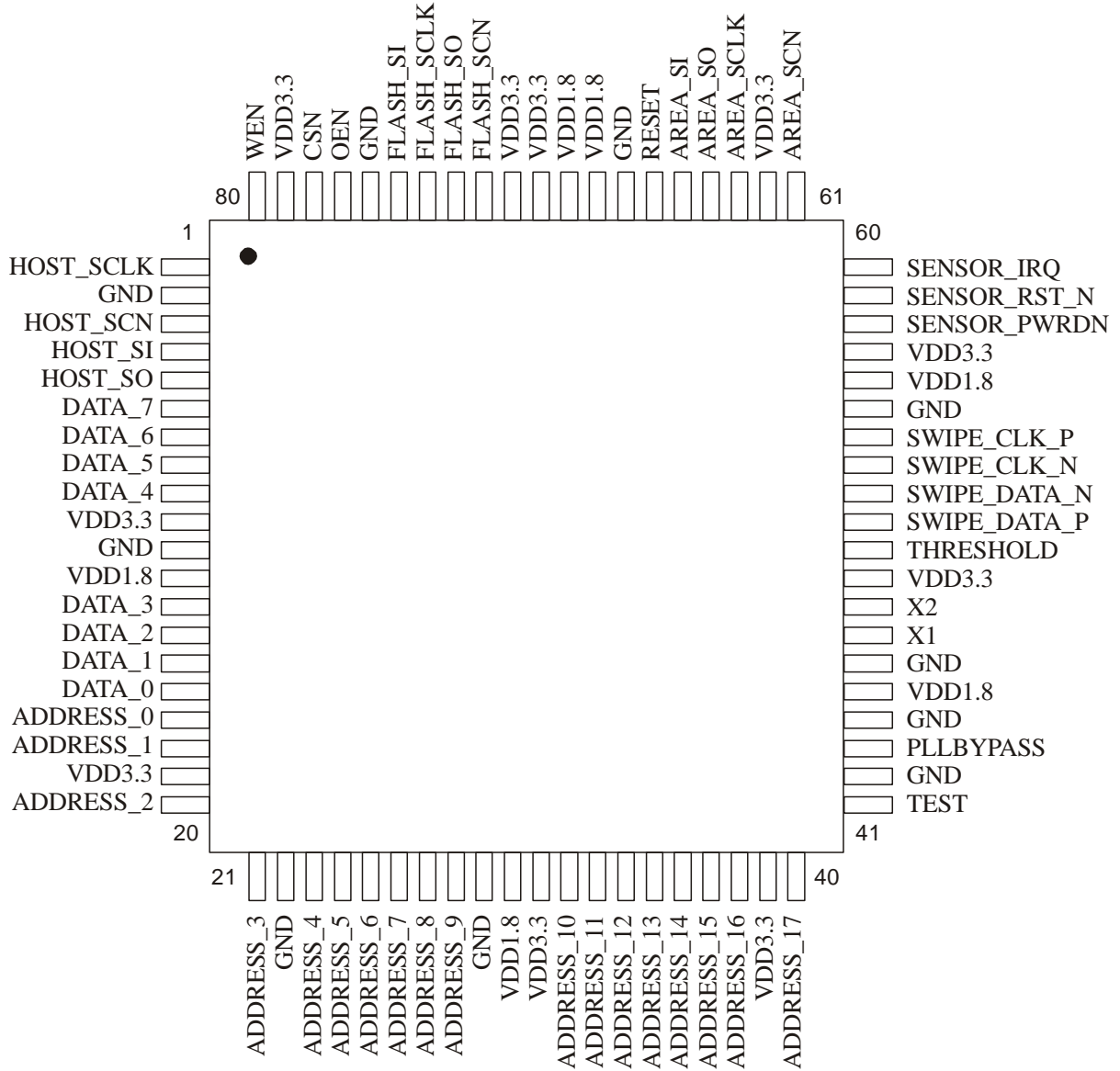
Operating temperature	-20°C to +85°C	<i>Note:</i> Stress beyond values listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated as normal operation this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
Storage temperature	-65°C to +150°C	
Supply voltage	-0.5 V to +4.6 V	
Input voltage	-0.5 V to $V_{DD} + 0.5 V$	

**Table 4**  
Absolute maximum ratings

### Pin configuration



**Figure 2**  
FPC2020 pinout for 64 pins QFN with exposed die paddle



**Figure 3**  
FPC2020 pinout for 80 pins TQFP

## Software command interface

To communicate with the area sensor module a serial command interface is used between the host processor and processor board. This interface is designed to be easy to use and performs the basic biometric functions needed in a fingerprint authentication system.

### Serial interface settings

The software settings for the serial protocol is (using UART):

- Communication speed: factory default baud rate set to 9600 baud (range from 9600 to 115200 baud)
- Format: 8 data bits, odd parity, one stop bit.
- Bit order: least significant bit first

, or (using SPI):

- Communication speed: Guaranteed maximum speed: 1MBit/s. Certain commands can requires a slower speed. For details, see the separate section on SPI timing requirements.
- SPI Mode: Mode 3, Clock Polarity High and Clock Phase Rising
- Chip Select: Active Low
- Bit order: most significant bit first

### Command send structure

Command, sent from host:

0	1	2	3	4	5
STX	IDX-LSB	IDX-MSB	COMMAND	PAYLOAD-LSB	PAYLOAD-MSB

- STX: Start byte, should be 0x02  
 IDX-LSB: Index value, least significant byte  
 IDX-MSB: Index value, most significant byte  
 COMMAND: Command byte  
 PAYLOAD-LSB: If any additional data is sent, the payload is a counter of how many bytes that will be sent (not including the CRC-code), otherwise zero.  
 PAYLOAD-MSB: Payload most significant byte, if no data, set to zero

If PAYLOAD != 0, then additional data should follow in the stream according to the following:

6	...	n	n+1	n+2	n+3	n+4
DATA-1	DATA-...	DATA-n	CRC-LSB	CRC-BYTE2	CRC-BYTE3	CRC-MSB

Note:

- 1) The CRC size (4 bytes) is not included in the payload counter. Its value is calculated from all the data bytes, and is used for checking if an error occurred during the transmission.
- 2) Do not send a new command before a response has been received from the previous command. One exception is when using the CANCEL-command, see "Command description / API\_CANCEL".
- 3) The default choice for IDX-LSB and IDX-MSB is 0x00, if nothing else is stated.

## Response structure

Response from device:

0	1	2	3
STX	RESULT	PAYLOAD-LSB	PAYLOAD-MSB

STX: Start byte, should be 0x02

RESULT: Result byte

PAYLOAD-LSB: If any additional data is sent, the payload is a counter of how many bytes that will be sent (not including the CRC-code), otherwise zero.

PAYLOAD-MSB: Payload most significant byte, if no data, set to zero

If PAYLOAD != 0, then additional data should follow in the stream according to the following:

4	...	n	n+1	n+2	n+3	n+4
DATA-1	DATA-...	DATA-n	CRC-LSB	CRC-BYTE2	CRC-BYTE3	CRC-MSB

*Note:*

The CRC size (4 bytes) is not included in the payload counter. Its value is calculated from all the data bytes, and is used for checking if an error occurred during the transmission.

## SPI Timing Requirements

The SPI interface of the FPC2020 is a slave interface, implying that the host (which is the master) determines when each data is sent to and from the FPC2020. Since the host cannot know when the FPC2020 has completed processing a given command, a polling process must be implemented by the host when trying to read the response for a given command request. (This is different from the UART interface, where the host knows that the received response is the correct response.)

The required implementation of the Request/Response process is the following:

1. Let the host send the 6 command bytes (+ a possible payload and crc).
2. Wait 5 ms.
3. Let the host send a random byte to the slave. It is what is returned to the host that is the first byte that (possibly) is the first byte of the response.
4. Check if the received byte is 0x02. If not, the slave is not ready, and need more time to complete processing the command. Typically, the FPC2020 returns a 0x52, as the 'BUSY' signal. Repeat steps 3-4 until a 0x02 byte is received as response.
5. This 0x02 value is the first byte in the regular response consisting of 4 bytes (+ a possible payload and crc). Read it as usual.

The SPI data transfer speed can be at least 2.5MBits/s during a single byte transmission, but certain delays are required between bytes, for all commands to work properly. This delay varies with the different commands. For example, uploading an image with CRC generation turned on requires a larger byte delay that if CRC generation is skipped.

The suggested default byte delay is 18ms, low speed byte delay is 30ms, and high speed byte delay is 4ms. The Low speed delay must be used during image upload with CRC, template upload with CRC, and the Firmware Upgrade command. The high speed byte delay can be used with image up/download without CRC. Effectively, this means that an image can be transferred with over 1.2MBit/s.

## Table of commands

BIOMETRIC COMMANDS	HEX	DESCRIPTION
API_CAPTURE_IMAGE	0x80	Capture image from sensor (before enrol).
API_CAPTURE_AND_ENROL_RAM	0x81	Enrol into RAM (includes Capture Image)
API_CAPTURE_AND_VERIFY_RAM	0x82	Verify against RAM (includes Capture Image)
API_CAPTURE_AND_VERIFY_FLASH	0x83	Verify against single FLASH slot (includes Capture Image) Set slot number (0 to 187) in IDX
API_CAPTURE_AND_IDENTIFY_FLASH	0x84	Identify against all FLASH slots (includes Capture Image)
API_ENROL_RAM	0x85	Enrol into RAM
API_VERIFY_RAM	0x86	Verify against RAM
API_VERIFY_FLASH	0x87	Verify against single FLASH slot Set slot number (0 to 187) in IDX
API_IDENTIFY_FLASH	0x88	Identify against all FLASH slots
API_ENROL_FLASH	0x92	Enrol into FLASH memory
API_CAPTURE_AND_ENROL_FLASH	0x93	Enrol into FLASH memory (includes Capture Image)

IMAGE TRANSFER	HEX	DESCRIPTION
API_UPLOAD_IMAGE	0x90	Upload image from RAM
API_DOWNLOAD_IMAGE	0x91	Download image to RAM

TEMPLATE HANDLING	HEX	DESCRIPTION
API_UPLOAD_TEMPLATE	0xA0	Upload template from RAM
API_DOWNLOAD_TEMPLATE	0xA1	Download template to RAM
API_COPY_TEMPLATE_RAM_TO_FLASH	0xA2	Copy template from RAM to permanent FLASH storage Set slot number (0 to 187) in IDX
API_UPLOAD_TEMPLATE_FROM_FLASH	0xA3	Upload template from single FLASH slot Set slot number (0 to 187) in IDX
API_DELETE_TEMPLATE_RAM	0xA4	Erase template from RAM
API_DELETE_SLOT_IN_FLASH	0xA5	Delete single slot in FLASH Set slot number (0 to 187) in IDX
API_DELETE_ALL_IN_FLASH	0xA6	Delete all FLASH slots
API_DOWNLOAD_TEMPLATE_TO_FLASH	0xA7	Download a template to FLASH

**Table 5**  
*Table of commands.*

ALGORITHM SETTINGS	HEX	DESCRIPTION
API_SECURITY_LEVEL_RAM	0xB0	Set security level, setting saved in RAM IDX-LSB: 0x04 = high convenience 0x05 = standard 0x06 = high security
API_SECURITY_LEVEL_STATIC	0xB1	Set security level, setting saved in non-volatile (static) memory
API_GET_SECURITY_LEVEL	0xB2	Get current security level, value sent as payload data

FIRMWARE COMMANDS	HEX	DESCRIPTION
API_FIRMWARE_VERSION	0xC0	Upload the version string for this device
API_FIRMWARE_UPDATE	0xC1	Start download of new firmware

COMMUNICATION COMMANDS	HEX	DESCRIPTION
API_SET_BAUD_RATE_RAM	0xD0	Set baud rate, setting saved in RAM IDX-LSB: 0x10 = 9600 baud 0x20 = 14400 baud 0x30 = 19200 baud 0x40 = 28800 baud 0x50 = 38400 baud 0x60 = 57600 baud 0x70 = 76800 baud 0x80 = 115200 baud
API_SET_BAUD_RATE_STATIC	0xD1	Set baud rate, setting saved in non-volatile (static) memory IDX-LSB: 0x00 = current speed (recommended usage) 0x10 = 9600 baud 0x20 = 14400 baud 0x30 = 19200 baud 0x40 = 28800 baud 0x50 = 38400 baud 0x60 = 57600 baud 0x70 = 76800 baud 0x80 = 115200 baud
API_TEST_HARDWARE	0xD2	Test hardware components

OTHER COMMANDS	HEX	DESCRIPTION
API_CANCEL	0xE0	Cancel ongoing command, only valid for: API_CAPTURE_AND_ENROL_RAM, API_CAPTURE_AND_VERIFY_RAM, API_CAPTURE_AND_VERIFY_AGAINST_FLASH, API_CAPTURE_AND_IDENTIFY_AGAINST_FLASH
API_ENTER_SLEEP_MODE	0xE1	Enter sleep mode (wake up by activating proper pin)
API_POWER_SAVE_MODE_RAM	0xE2	Set power save mode, setting saved in RAM IDX-LSB: 0x00 = enable 0x01 = disable
API_POWER_SAVE_MODE_STATIC	0xE3	Set power save mode, setting saved in non-volatile (static) memory IDX-LSB: 0x00 = enable 0x01 = disable
API_GET_POWER_SAVE_MODE	0xE5	Get current power save mode, value sent as payload data: 0x00 = enabled 0x01 = disabled
API_ADVANCED_SETTINGS	0xE8	Managing advanced settings, e.g. supply voltage control.

**Table 6**  
Table of commands (continued)

**Table of response bytes**

COMMAND	HEX
API_FAILURE	0x00
API_SUCCESS	0x01
API_NO_FINGER_PRESENT	0x02
API_FINGER_PRESENT	0x03
API_VERIFICATION_OK	0x04
API_VERIFICATION_FAIL	0x05
API_ENROL_OK	0x06
API_ENROL_FAIL	0x07
API_HW_TEST_OK	0x08
API_HW_TEST_FAIL	0x09
API_CRC_FAIL	0x0A
API_PAYLOAD_TOO_LONG	0x0B
API_PAYLOAD_TOO_SHORT	0x0C
API_UNKNOWN_COMMAND	0x0D
API_NO_TEMPLATE_PRESENT	0x0E
API_IDENTIFY_OK	0x0F
API_IDENTIFY_FAIL	0x10
API_INVALID_SLOT_NR	0x11
API_CANCEL_SUCCESS	0x12
API_APPL_CRC_FAIL	0x14
API_SYS_CRC_FAILED	0x16
API_LOW_VOLTAGE	0x17

**Table 7**  
Table of response bytes

## Command description

This section describes the individual commands of the main application, along with their parameters, and responses.

### Capture image

#### API\_CAPTURE\_IMAGE

An image is captured from the fingerprint sensor. The fingerprint image is placed in RAM and can be uploaded by the command API\_UPLOAD\_IMAGE. Calculation is done on the image to determine if a finger is present or not present on the sensor. No payload is sent with this command.

#### Response command:

- API\_NO\_FINGER\_PRESENT = No finger present on sensor
- API\_FINGER\_PRESENT = Finger present on sensor

No payload is received with the response from this command.

### Capture and enrol (RAM)

#### API\_CAPTURE\_AND\_ENROL\_RAM

An image is captured from the fingerprint sensor and enrolment of this image is done. The command waits for “finger present” before it starts the enrolment. This means that images are captured in a loop from the sensor until a finger is present. The command returns with response when the enrolment is complete or if the enrolment fails for any reason. After enrolment the template is stored in RAM and can be uploaded or moved to FLASH storage. No payload is sent with this command.

#### Response command:

- API\_ENROL\_OK = Enrolment successful
- API\_ENROL\_FAIL = Enrolment failed

No payload is received with the response from this command.

#### Note:

It is possible to cancel the current enrol operation by sending the command API\_CANCEL. This cancels the enrolment and the device returns to its normal command loop.

### Capture and verify (RAM)

#### API\_CAPTURE\_AND\_VERIFY\_RAM

A template must be present in RAM before starting the verification, either by using the Download Template command (API\_DOWNLOAD\_TEMPLATE) OR the command API\_CAPTURE\_ENROL\_RAM. Thereafter the verification can be started. This command also captures an image from the fingerprint sensor. The command waits for “finger present” before it starts the verification. This means that images are captured in a loop from the sensor until a finger is present. The command returns with response when the verification is complete or if the verification fails for any reason. No payload is sent with this command.

#### Response command:

- API\_VERIFICATION\_OK = Verification successful
- API\_VERIFICATION\_FAIL = Verification failed
- API\_NO\_TEMPLATE\_PRESENT = No template present

No payload is received with the response from this command.

#### Note:

It is possible to cancel the current verification operation by sending the command API\_CANCEL. This cancels the verification and the device returns to its normal command loop.

### Capture and verify (FLASH)

#### API\_CAPTURE\_AND\_VERIFY\_FLASH

The FLASH slot number must be given in the IDX bytes. This command first captures an image from the fingerprint sensor. The command waits for “finger present” before it starts the verification. This means that images are captured in a loop from the sensor until a finger is present. The command returns with response when the verification is complete or if the verification fails for any reason. No payload is sent with this command.

#### Response command:

- API\_VERIFICATION\_OK = Verification successful
- API\_VERIFICATION\_FAIL = Verification failed
- API\_NO\_TEMPLATE\_PRESENT = No template in given FLASH slot
- API\_INVALID\_SLOT\_NR = Wrong slot number

No payload is received with the response from this command.

*Note:*

It is possible to cancel the current verification operation by sending the command `API_CANCEL`. This cancels the verification and the device returns to its normal command loop.

### **Capture and identify (FLASH)      `API_CAPTURE_AND_IDENTIFY_FLASH`**

Identification is made against all FLASH slots. This command first captures an image from the fingerprint sensor. The command waits for “finger present” before it starts the identification. This means that images are captured in a loop from the sensor until a finger is present. The command returns with response when the identification is complete or if the identification fails for any reason. No payload is sent with this command.

*Response command:*

- `API_IDENTIFY_OK` = Identification successful
- `API_IDENTIFY_FAIL` = Identification fails

In a successful identification, the slot index is received as payload in two bytes (LSB first) plus the 4 CRC bytes.

*Note:*

It is possible to cancel the current identification operation by sending the command `API_CANCEL`. This cancels the identification and the device returns to its normal command loop.

### **Enrol (RAM)      `API_ENROL_RAM`**

A fingerprint image must be present in RAM before starting the enrolment, either by capturing an image from the fingerprint sensor using the command `API_CAPTURE_IMAGE` OR by using the Download Image Command (`API_DOWNLOAD_IMAGE`). The command returns with response when the enrolment is complete or if the enrolment fails for any reason. After enrolment the template is stored in RAM and can be uploaded or moved to FLASH storage. No payload is sent with this command.

*Response command:*

- `API_ENROL_OK` = Enrolment successful
- `API_ENROL_FAIL` = Enrolment failed

No payload is received with the response from this command.

### **Verify (RAM)      `API_VERIFY_RAM`**

A template AND a fingerprint image must be present in RAM before starting the verification. To handle the template use the command Download Template (`API_DOWNLOAD_TEMPLATE`) OR the command `API_CAPTURE_ENROL_RAM`. To handle the image use the command Download Image (`API_DOWNLOAD_IMAGE`) OR the command `API_CAPTURE_IMAGE`. Thereafter the verification can be started. The command returns with response when the verification is complete or if the verification fails for any reason. No payload is sent with this command.

*Response command:*

- `API_VERIFICATION_OK` = Verification successful
- `API_VERIFICATION_FAIL` = Verification failed
- `API_NO_TEMPLATE_PRESENT` = No template present

No payload is received with the response from this command.

### **Verify (FLASH)      `API_VERIFY_FLASH`**

A fingerprint image must be present in RAM before starting the verification, use the command Download Image (`API_DOWNLOAD_IMAGE`) OR the command `API_CAPTURE_IMAGE`. The FLASH slot number must be given in the `IDX` bytes. The command returns with response when the verification is complete or if the verification fails for any reason. No payload is sent with this command.

*Response command:*

- `API_VERIFICATION_OK` = Verification successful
- `API_VERIFICATION_FAIL` = Verification failed
- `API_NO_TEMPLATE_PRESENT` = No template in given FLASH slot
- `API_INVALID_SLOT_NR` = Wrong slot number

No payload is received with the response from this command.

## Identify (FLASH)

### API\_IDENTIFY\_FLASH

A fingerprint image must be present in RAM before starting the verification, use the command Download Image (API\_DOWNLOAD\_IMAGE) OR the command API\_CAPTURE\_IMAGE. Identification is made against all FLASH slots. The command returns with response when the identification is complete or if the identification fails for any reason. No payload is sent with this command.

*Response command:*

- API\_IDENTIFY\_OK = Identification successful
- API\_IDENTIFY\_FAIL = Identification failed
- API\_NO\_TEMPLATE\_PRESENT = All FLASH slots are empty

In a successful identification, the slot index is received as payload in two bytes (LSB first), plus the 4 CRC bytes.

## Enrol (FLASH)

### API\_ENROL\_FLASH

A fingerprint image must be present in RAM before starting the enrolment, either by capturing an image from the fingerprint sensor using the command API\_CAPTURE\_IMAGE OR by using the Download Image Command (API\_DOWNLOAD\_IMAGE). The command returns with response when the enrolment is complete or if the enrolment fails for any reason. After enrolment the template is stored in FLASH and can be uploaded or moved to FLASH storage. The desired FLASH slot number must be given in the IDX bytes. No payload is sent with this command.

*Response command:*

- API\_ENROL\_OK = Enrolment successful
- API\_ENROL\_FAIL = Enrolment failed
- API\_INVALID\_SLOT\_NR = Incorrect FLASH slot number

No payload is received with the response from this command.

## Capture Enrol (FLASH)

### API\_CAPTURE\_AND\_ENROL\_FLASH

This command first captures an image from the fingerprint sensor. The command waits for “finger present” before it starts the verification. This means that images are captured in a loop from the sensor until a finger is present. The command then returns with a response when the enrolment is complete or if the enrolment fails for any reason. After enrolment the template is stored in FLASH and can be uploaded or moved to FLASH storage. The desired FLASH slot number must be given in the IDX bytes. No payload is sent with this command.

*Response command:*

- API\_ENROL\_OK = Enrolment successful
- API\_ENROL\_FAIL = Enrolment failed
- API\_INVALID\_SLOT\_NR = Incorrect FLASH slot number

No payload is received with the response from this command.

## Upload image

### API\_UPLOAD\_IMAGE

By using this command it is possible to upload the fingerprint image present in RAM. The response is the API\_SUCCESS command followed by the image data. The size of image data is 30400 bytes. The first byte is the upper left pixel and then data follows row-wise (X-direction). Each pixel has one byte value (256 gray scales). There is no image header. No payload is sent with this command. Note that if IDX\_LSB = 0x01, the CRC generation is skipped. This will increase the maximum SPI transfer speed for this command.

*Response command:*

- API\_SUCCESS = Upload successful
- API\_FAILURE = Upload failed

The received payload in a successful upload consists of 30400 bytes plus the 4 CRC bytes.

## Download image

### API\_DOWNLOAD\_IMAGE

By using this command it is possible to download a fingerprint image to RAM. The size of image data must be 30400 bytes. The first byte is the upper left pixel and then data follows row-wise (X-direction). Each pixel has one byte value (256 grey scales). There is no image header. Payload size of this command must be 30400 bytes, plus the 4 CRC-bytes sent after the payload. Note that if IDX\_LSB = 0x01, the CRC check is skipped (but it must still be included in the request). This will increase the maximum SPI transfer speed for this command.

*Response command:*

- API\_SUCCESS = Download successful

- API\_FAILURE = Download failed  
No payload is received with the response from this command.

### Upload template

### API\_UPLOAD\_TEMPLATE

After a successful enrolment the template is uploaded from RAM using the Upload Template command (API\_UPLOAD\_TEMPLATE). The response is the API\_SUCCESS command followed by the template data. The template consists of 938 bytes binary data with no public information. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Upload successful
- API\_FAILURE = Upload failed

The received payload in a successful upload consists of 938 bytes plus the 4 CRC bytes.

### Download template

### API\_DOWNLOAD\_TEMPLATE

Before verification the template is downloaded to RAM using the Download Template command (API\_DOWNLOAD\_TEMPLATE). The size of the template is 938 bytes. Payload size of this command must be 938 bytes, plus the 4 CRC-bytes sent after the payload.

*Response command:*

- API\_SUCCESS = Download successful
- API\_FAILURE = Download failed

No payload is received with the response from this command.

### Copy template from RAM to FLASH

### API\_COPY\_TEMPLATE\_FROM\_RAM\_TO\_FLASH

This command copies the template currently in RAM to FLASH. The FLASH slot number must be given in the IDX bytes. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Template storage successful
- API\_FAILURE = Template storage failed
- API\_INVALID\_SLOT\_NR = Wrong slot number

No payload is received with the response from this command.

### Upload template from FLASH

### API\_UPLOAD\_TEMPLATE\_FROM\_FLASH

This command uploads the template from FLASH. The FLASH slot number must be given in the IDX bytes. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Upload successful
- API\_FAILURE = Upload failed
- API\_INVALID\_SLOT\_NR = Wrong slot number

The received payload in a successful upload consists of 938 bytes plus the 4 CRC bytes.

### Delete template in RAM

### API\_DELETE\_TEMPLATE\_RAM

This command deletes the template currently stored in RAM. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Template removal successful
- API\_FAILURE = Template removal failed

No payload is received with the response from this command.

### Delete single template in FLASH

### API\_DELETE\_SLOT\_IN\_FLASH

By using the command Delete slot in FLASH one can choose which slot to delete (include slot number in index value of command). No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Template removal successful
- API\_FAILURE = Template removal failed
- API\_INVALID\_SLOT\_NR = Wrong slot number

No payload is received with the response from this command.

**Delete all templates in FLASH**                      **API\_DELETE\_ALL\_IN\_FLASH**

It is possible to delete all templates in FLASH by issuing the command Delete all in FLASH. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Template removal successful
- API\_FAILURE = Template removal failed

No payload is received with the response from this command.

**Download template to FLASH**                      **API\_DOWNLOAD\_TEMPLATE\_TO\_FLASH**

Downloads a template from the host directly into the FLASH memory (and into RAM). The FLASH slot number must be given in the IDX bytes. The size of the template is 938 bytes. Payload size of this command must be 938 bytes, plus the 4 CRC-bytes sent after the payload.

*Response command:*

- API\_SUCCESS = Download successful
- API\_FAILURE = Download failed
- API\_INVALID\_SLOT\_NR = Wrong slot number

No payload is received with the response from this command.

**Security level (RAM)**                                      **API\_SECURITY\_LEVEL\_RAM**

The security level to be used during verification and identification can be set by the command Set security level. The value of the security level should be set in the index value (IDX-LSB) of the command. The factory default security level is set to value 0x04. The value is stored in RAM and the setting is lost after reset. The security level is not stored together with the template. During enrolment there is no effect when changing the security threshold. The created template will support all security settings. No payload is sent with this command.

VALUE (IDX-LSB)	SECURITY LEVEL
0x04	High convenience (factory default)
0x05	Standard
0x06	High security

*Response command:*

- API\_SUCCESS = New security level set
- API\_FAILURE = Security level out of range

No payload is received with the response from this command.

**Security level (STATIC)**                                      **API\_SECURITY\_LEVEL\_STATIC**

The security level to be used during verification and identification can be set by the command Set security level. The value of the security level should be set in the index value (IDX-LSB) of the command. The factory default security level is set to value 0x04. The value is stored in non-volatile memory and the setting is saved even after reset. This means that the factory default value will be changed. The security level is not stored together with the template. During enrolment there is no effect when changing the security threshold. The created template will support all security settings. No payload is sent with this command.

VALUE (IDX-LSB)	SECURITY LEVEL
0x04	High convenience (factory default)
0x05	Standard
0x06	High security

*Response command:*

- API\_SUCCESS = New security level set
- API\_FAILURE = Security level out of range

### Get current security level

### API\_GET\_SECURITY\_LEVEL

This command returns the value of the current security setting that the module uses. The value is sent as payload data. No payload is sent with this command.

VALUE	SECURITY LEVEL
0x04	High convenience
0x05	Default
0x06	High security

*Response command:*

- API\_SUCCESS = Command OK
- API\_FAILURE = Command fail

No payload is received with the response from this command.

### Firmware version

### API\_FIRMWARE\_VERSION

This command returns the firmware version of the main application. The response is the API\_SUCCESS command followed by the firmware version string. No payload is sent with this command.

*Response command:*

- API\_SUCCESS = Request successful, version string follows as payload
- API\_FAILURE = Request failed

A payload + 4 CRC bytes will be received in a successful request. The size of this payload could vary with the version of the firmware.

### Firmware update

### API\_FIRMWARE\_UPDATE

It is possible to update the pre-loaded firmware, by downloading the new program to the FLASH memory. This is done in several steps – filling one FLASH memory slot at a time.

The new firmware is divided in N pieces of 256 bytes each, and each such 256-byte code payload is sent to FPC2020 using the API\_FIRMWARE\_UPDATE command, with the IDX bytes starting at '1' for the first command, and ending at 'N' for the last command.

Thus, a complete upgrade requires N successful applications of this command. If the upgrade is not completed, or fails in any way, the application will start in 'Boot Mode' after next reset. For more info on this mode, see the section on Boot Mode. Payload size of this command must be 256 bytes, plus the 4 CRC-bytes sent after the payload.

*Response command:*

- API\_SUCCESS = Upgrade successful
- API\_CRC\_FAIL = Firmware is corrupt

No payload is received with the response from this command.

If the firmware is corrupt it is still possible to set baud rate and perform firmware updates.

### Set baud rate (RAM)

### API\_SET\_BAUD\_RATE\_RAM

It is possible to change baud rate for the serial communication between host and FPC2020. The table below shows the available baud rates. Factory default baud rate is 9600. The selected value should be set in the IDX-LSB byte of the command. The value is stored in RAM and the setting is lost after reset. No payload is sent with this command.

VALUE (IDX-LSB)	BAUD RATE
0x10	9600 (factory default)
0x20	14400
0x30	19200
0x40	28800
0x50	38400
0x60	57600
0x70	76800
0x80	115200

*Response command:*

- API\_SUCCESS = Baud rate change accepted
  - API\_FAILURE = Baud rate out of range
- No payload is received with the response from this command.

*Note:*

Once a baud rate change has been accepted, next command must be sent with new baud rate. However the response command above is sent with the old baud rate.

**Set baud rate (STATIC) API\_SET\_BAUD\_RATE\_STATIC**

It is possible to change baud rate for the serial communication between host and FPC2020. The table below shows the available baud rates. Factory default baud rate is 9600. The selected value should be set in the IDX-LSB byte of the command. The value is stored in non-volatile memory and the setting is saved even after reset. This means that the factory default value will be changed.

Note that if the index 0 is used, the currently active baud rate will be stored. This is the recommended option, since it will ensure that a non-compatible UART speed not is permanently selected. No payload is sent with this command.

VALUE (IDX-LSB)	BAUD RATE
0x00	Currently active baudrate
0x10	9600 (factory default)
0x20	14400
0x30	19200
0x40	28800
0x50	38400
0x60	57600
0x70	76800
0x80	115200

*Response command:*

- API\_SUCCESS = Baud rate change accepted
  - API\_FAILURE = Baud rate out of range
- No payload is received with the response from this command.

*Note:*

Once a baud rate change has been accepted, next command must be sent with new baud rate. However the response command above is sent with the old baud rate.

**Test hardware API\_TEST\_HARDWARE**

This command tests the FPC2020 hardware. It performs a check of the different components on the module. No payload is sent with this command.

*Response command:*

- API\_HW\_TEST\_OK = Hardware check successful
  - API\_HW\_TEST\_FAIL = Hardware check failed, contact technical support
- No payload is received with the response from this command.

**Cancel current command API\_CANCEL**

It is possible to cancel the following ongoing commands:

- API\_CAPTURE\_AND\_ENROL\_RAM
- API\_CAPTURE\_AND\_ENROL\_FLASH
- API\_CAPTURE\_AND\_VERIFY\_RAM
- API\_CAPTURE\_AND\_VERIFY\_FLASH
- API\_CAPTURE\_AND\_IDENTIFY\_FLASH

The module will respond with API\_CANCEL\_SUCCESS and the return to normal command loop. No payload is sent with this command.

*Response command:*

- API\_CANCEL\_SUCCESS = Cancel successful
- API\_FAILURE = Cancel failed

No payload is received with the response from this command.

### Enter sleep mode

### API\_ENTER\_SLEEP\_MODE

SLEEP MODE is entered by issuing the command API\_ENTER\_SLEEP\_MODE. In SLEEP MODE the device runs on low power. To wake up the device, a wakeup interrupt must occur. This is triggered by one of three possible signals: HOST\_SI (active low), HOST\_SCN (active low) or SENSOR\_IRQ (active high). No payload is sent with this command. Before the device enter SLEEP MODE it responds with one of the following:

*Response command:*

- API\_SUCCESS = Request accepted, entering SLEEP MODE
- API\_FAILURE = Request failed

No payload is received with the response from this command.

### Power save mode (RAM)

### API\_POWER\_SAVE\_MODE\_RAM

In POWER SAVE MODE the module reduces the clock frequency of the processor by half to lower power consumption. To enter POWER SAVE MODE, issue the command Power Save Mode with the value 0 in the IDX-LSB byte. To exit POWER SAVE MODE, issue the command Power Save Mode with the value 1 in the IDX-LSB byte. The setting is stored in RAM and the setting is lost after reset. No payload is sent with this command. The factory default setting is that the module is NOT in power down mode (value=1).

VALUE (IDX-LSB)	DESCRIPTION
0	Half speed
1	Low speed

*Response command:*

- API\_SUCCESS = Request accepted, entering POWER SAVE MODE
- API\_FAILURE = Request failed

No payload is received with the response from this command.

### Power save mode (STATIC)

### API\_POWER\_SAVE\_MODE\_STATIC

In POWER SAVE MODE the module reduces the clock frequency of the processor by half to lower power consumption. To enter POWER SAVE MODE issue the command Power Save Mode with the value 0 in the IDX-LSB byte. To exit POWER SAVE MODE issue the command Power Save Mode with the value 1 in the IDX-LSB byte. The setting is stored in non-volatile memory and the setting is saved even after reset. This means that the factory default setting (value=1) will be changed. No payload is sent with this command.

VALUE (IDX-LSB)	DESCRIPTION
0	Half speed
1	Full speed

*Response command:*

- API\_SUCCESS = Request accepted, entering POWER SAVE MODE
- API\_FAILURE = Request failed

No payload is received with the response from this command.

### Get current power save mode

### API\_GET\_POWER\_SAVE\_MODE

This command returns the value of the current setting of power save mode. The value is received as payload data. No payload is sent with this command.

VALUE	DESCRIPTION
0	Half speed
1	Full speed

*Response command:*

- API\_SUCCESS = Command OK
- API\_FAILURE = Command fail

The received payload in a successful upload consists of 1 byte plus the 4 CRC bytes.

## Manage Advance Settings

## API\_ADVANCED\_SETTINGS

This command is used both to get current, and to set current advanced settings. The only advanced setting presently supported is the Supply Voltage Control.

The Supply Voltage Control uses the 'Brown Out Detection' functionality in FPC2020 to check that the Supply Voltage is at a proper level. If enabled, the system will respond with the response code API\_LOW\_VOLTAGE to any command, whenever the supply voltage falls below the specified level. (See the section on Brown Out Detection on details of how to determine the specified level.)

The Supply Voltage Control can be enabled temporarily (in RAM) or statically (in non-volatile memory). In the latter case, the setting will remain after a system reset. The factory setting is a disabled Supply Voltage Control.

If this command is sent with a one-byte payload, it can be used to put the system in a desired mode, by turning on/off individual bits:

COMMAND PAYLOAD VALUE	DESCRIPTION
Bit 0 (LSB) = 0	Disable Supply Voltage Control
Bit 0 (LSB) = 1	Enable Supply Voltage Control
Bit 1 = 0	Do not store setting statically
Bit 1 = 1	Store setting statically

If this command is sent with no payload, it will return a response including a one byte payload, which represents the state that the system is currently in:

RESPONSE PAYLOAD VALUE	DESCRIPTION
0	Supply Voltage Control disabled
1	Supply Voltage Control enabled

### Response command:

- API\_SUCCESS = Command OK
- API\_FAILURE = Command fail

Note that the CRC of 4 bytes are always added to the 1 byte payload. Note also that the Index bytes of the command structure are NOT used with this command.

## CRC calculation

The CRC calculation can be implemented as a table of pre-computed effects to ensure efficiency. The CRC value is 32 bits long. The table is indexed by the byte to be encoded and thus the table contains 256 double words (256 \* 32 bits).

The CRC algorithm implementation was initially developed by the University of California, Berkeley and its contributors, but has been changed and somewhat simplified to fit the embedded nature of FPC2020. The algorithm uses the CCITT-32 CRC Polynomial.

The source code for the CRC implementation is available on the CD, or as a download from the Fingerprint Cards homepage. It should compile with no or small changes on most environments.

## Power Management

The FPC2020 uses an external crystal with a frequency of 7.3728 Mhz, but this clock is internally multiplied to generate a different frequencies, depending on the current state of the system. The 'Power Save Mode' commands (API\_POWER\_SAVE\_MODE\_RAM, and API\_POWER\_SAVE\_MODE\_STATIC) allows the user to set the system in two different modes: Full Speed (Power Save Mode=disabled), or Half Speed (Power Save Mode=enabled).

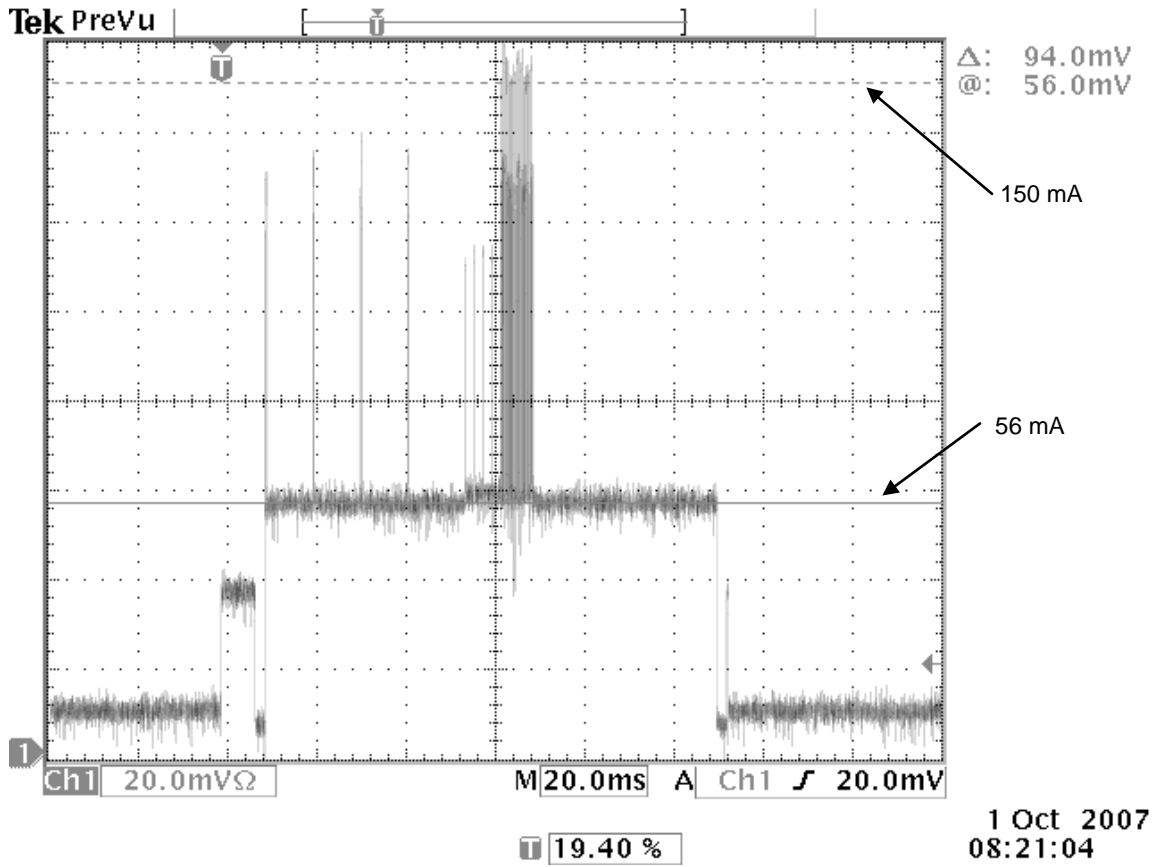
In Full Speed Mode, the following frequencies are used in the processor:

- 7 Mhz during Idle mode (waiting for commands)
- 59 MHz during processing of most commands
- 96 MHz during the most computationally complex commands (enrol, verify, and identify).

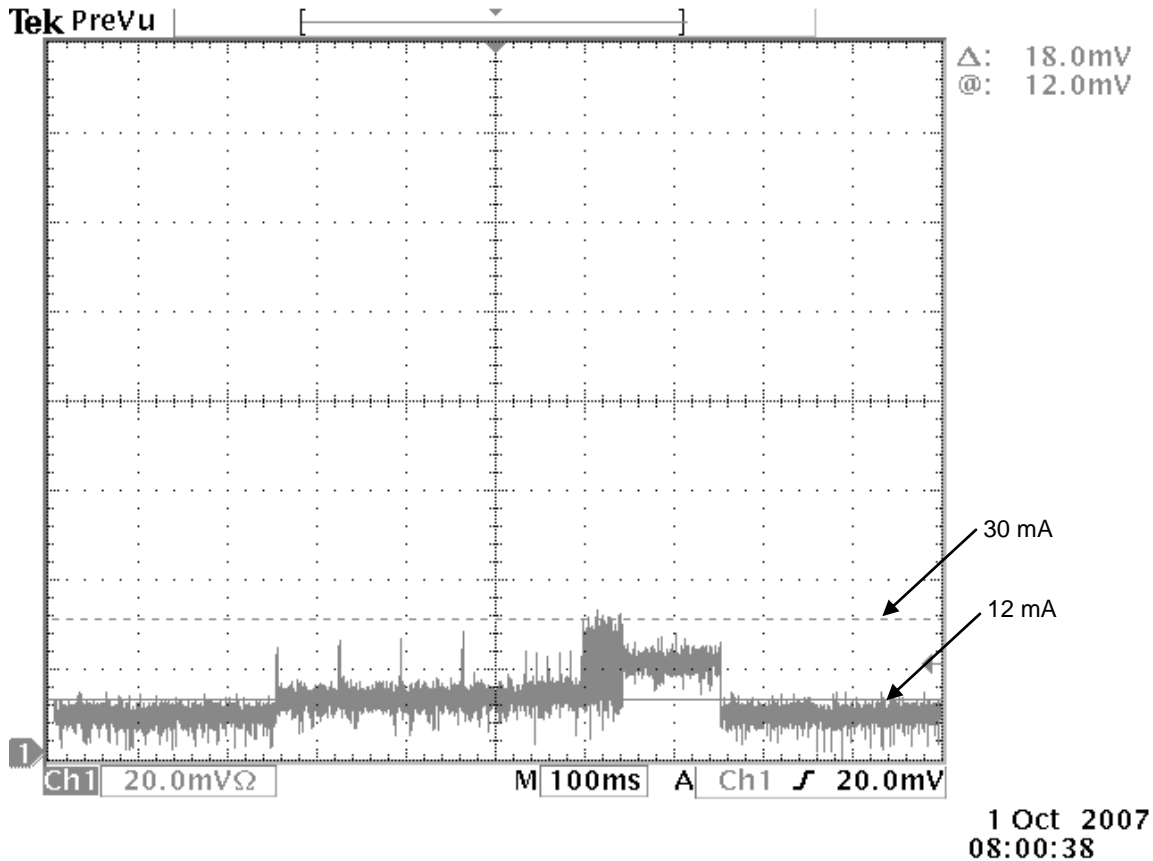
In Half Speed Mode, measures have been taken to reduce the overall power, but also to limit the current peaks. The frequencies used in this mode are:

- 7 Mhz during Idle mode (waiting for commands)
- 29 MHz during processing of all commands
- 14 MHz during the most power consuming hardware supported computations (during enrol, verify, and identify).

The following images illustrate the current usage for the command API\_VERIFY\_RAM in Full Speed vs. Half Speed Mode.



**Figure 4**  
Supply current during a verification in active full speed



**Figure 5**  
Supply current during a verification in active half speed

## Boot Mode

At start-up of FPC2020, a boot sequence (located in ROM) is executed, which downloads the main application code located in the attached FLASH memory. If no errors are encountered during this download process, the boot sequence terminates and leaves control to the main application. This is the default behaviour, which typically always should occur in the standard set-up. The boot sequence takes 180 ms.

If the main application code is corrupt in some way (i.e. FLASH memory is not present or damaged, or a previous upgrade command was terminated in an uncontrolled way), the boot sequence instead waits for input over the UART/SPI host interface. (Default host UART speed is 9600 baud, and maximum SPI speed is 1 MBit/s.)

In this mode, FPC2020 only handles the following commands (the communication protocol is identical to what is described in the earlier sections):

- API\_RUN\_SUPPLIED\_PRG (Command code: 0xE6)
- API\_INIT\_SFR\_REGS (Command code: 0xE7)

All other commands receive the response API\_APPL\_CRC\_FAILED or API\_SYS\_CRC\_FAILED, depending on which part of the normal system startup that failed (Application or System).

With the API\_RUN\_SUPPLIED\_PRG-command, the user can send the main application code or the FLASH Installer application as a payload. Both applications are provided by Fingerprint Cards. The Flash Installer application is used to program a system using Flash memory with the main program before its first usage.

Return values are:

- API\_SUCCESS = Download of program successful – New app is launched.
- API\_CRC\_FAILED = Download of program failed – Hardware reset required.

With the API\_INIT\_SFR\_REGS-command, the user can modify some internal registers of FPC2020, to change the UART speed. Note that this command is optional, and it has no affect on the SPI interface. Sending a payload of size two bytes, according to the following table, changes the bootmode baudrate. Note that 4 CRC bytes are necessary, but their actual values are ignored.

PAYLOAD BYTES	BAUD RATE
0xAA 0x33	9600 (factory default)
0xAA 0x75	14400
0xAA 0x98	19200
0xAA 0xBB	28800
0xAA 0xCC	38400
0xAA 0xDD	57600
0xAA 0xE6	76800
0xAA 0xEE	115200

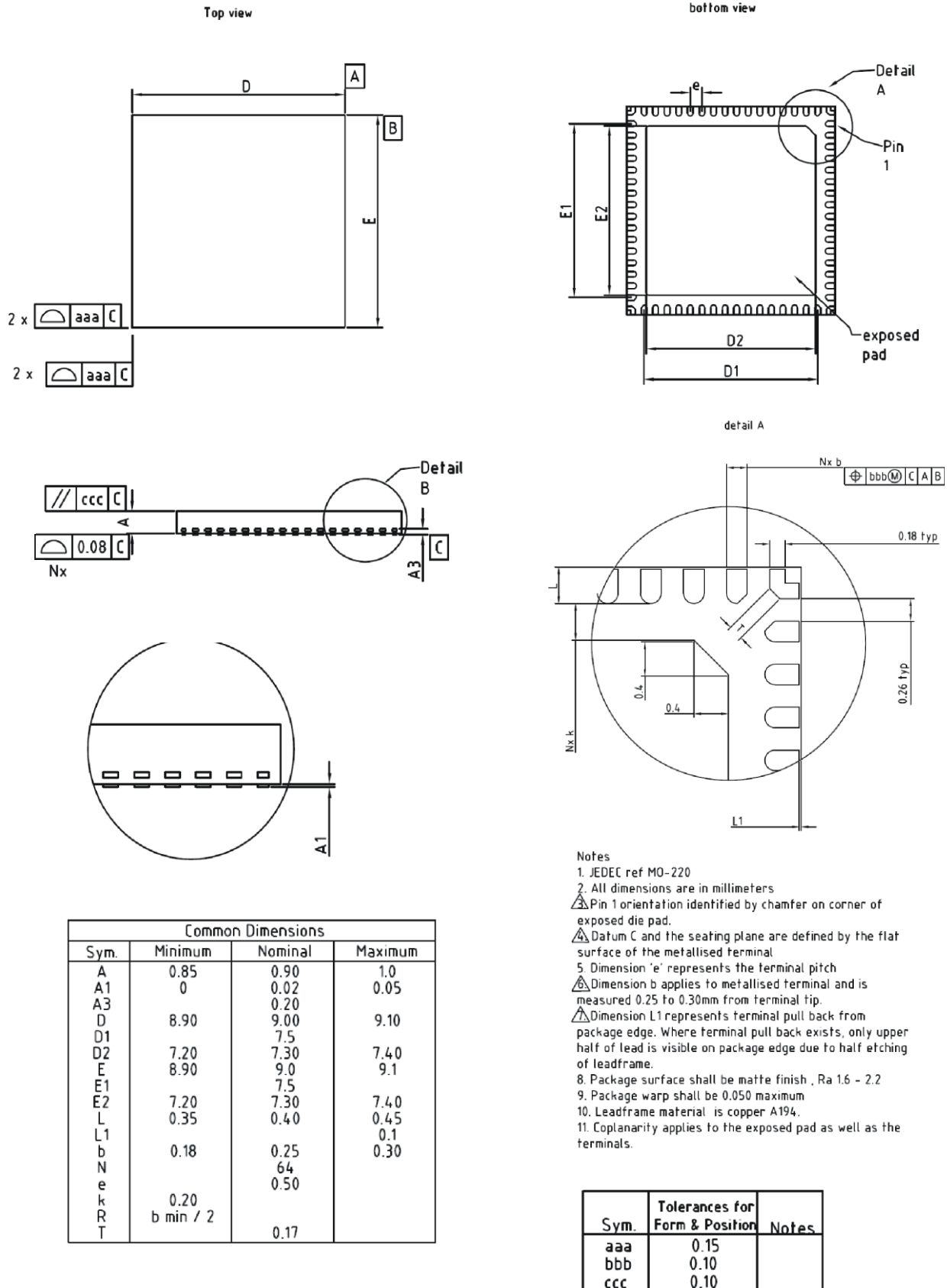
Return values are:

- API\_SUCCESS
- API\_FAILURE

= Internal registers adjusted.

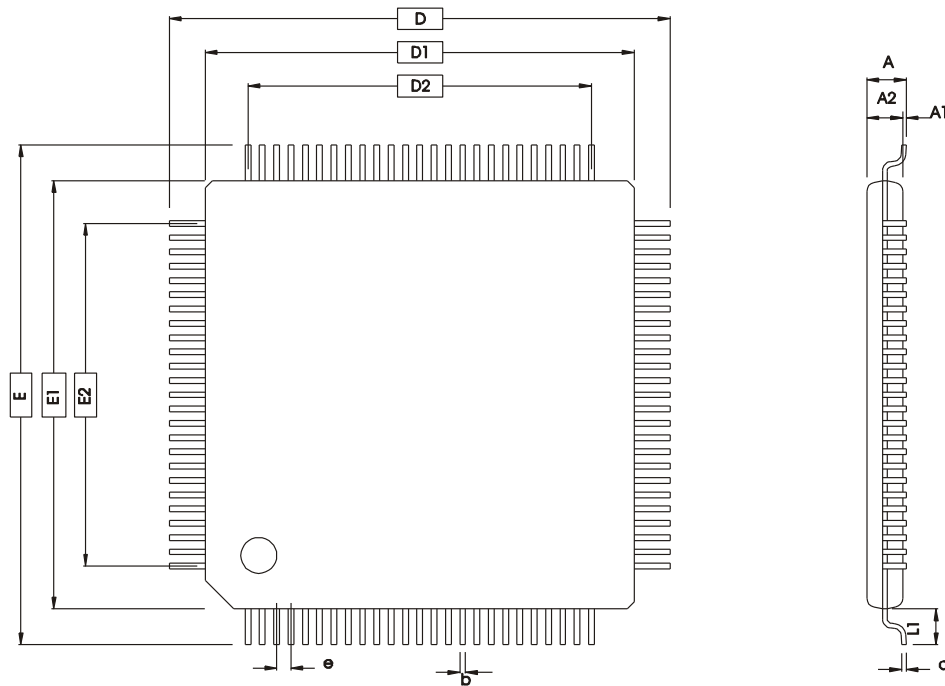
= Process failed.

## FPC2020 QFN64 mechanical outline



**Figure 6**  
FPC2020 QFN64 mechanical outline

## FPC2020 TQFP80 mechanical outline



SYMBOL	MILLIMETER		
	MIN.	NOM.	MAX.
A	—	—	1.20
A1	0.05	—	0.15
A2	0.95	1.00	1.05
D	14.00 BSC.		
D1	12.00 BSC.		
D2	9.50		
E	14.00 BSC.		
E1	12.00 BSC.		
E2	9.50		
b	0.17	0.20	0.27
c	0.09	—	0.20
e	0.50 BSC.		
L1	1.00 REF		

NOTES :

- DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION.  
ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE  
MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
- DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.  
ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE  
LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY  
MORE THAN 0.08mm.  
DAMBAR CAN NOT BE LOCATED ON THE LOWER RADIUS  
OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION  
AND AN ADJACENT LEAD IS 0.07mm FOR 0.4mm and  
0.5mm PITCH PACKAGES.

**Figure 7**  
FPC2020 TQFP80 mechanical outline

## Pin assignment

PIN TQFP	PIN QFN	SIGNAL NAME	DESCRIPTION
1	1	HOST_SCLK	SPI Mode: SPI clk UART Mode: Not used connect to "0" or "1"
2	-	GND	Ground
3	2	HOST_SCN	Selects between UART and SPI HOST_SCN = "0" => SPI Mode HOST_SCN = "1" => UART Mode
4	3	HOST_SI	SPI Mode: SPI data in UART Mode: Uart data in
5	4	HOST_SO	SPI Mode: SPI data out UART Mode: Uart data out
6	5	DATA_7	Leave unconnected reserved for GPIO function in a future
7	6	DATA_6	Leave unconnected reserved for GPIO function in a future
8	7	DATA_5	Leave unconnected reserved for GPIO function in a future
9	8	DATA_4	Leave unconnected reserved for GPIO function in a future
10	9	VDD3.3	2.5 – 3.3 Voltage supply
11	-	GND	Ground
12	-	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
13	10	DATA_3	Leave unconnected reserved for GPIO function in a future
14	11	DATA_2	Leave unconnected reserved for GPIO function in a future
15	12	DATA_1	Leave unconnected reserved for GPIO function in a future
16	13	DATA_0	Leave unconnected reserved for GPIO function in a future
17	14	ADDRESS_0	Leave unconnected
18	15	ADDRESS_1	Leave unconnected
19	-	VDD3.3	2.5 – 3.3 Voltage supply
20	16	ADDRESS_2	Leave unconnected
21	17	ADDRESS_3	Leave unconnected
22	-	GND	Ground
23	18	ADDRESS_4	Leave unconnected
24	19	ADDRESS_5	Leave unconnected
25	20	ADDRESS_6	Leave unconnected
26	21	ADDRESS_7	Leave unconnected
27	22	ADDRESS_8	Leave unconnected
28	23	ADDRESS_9	Leave unconnected
29	-	GND	Ground
30	-	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
31	24	VDD3.3	2.5 – 3.3 Voltage supply
32	25	ADDRESS_10	Leave unconnected
33	26	ADDRESS_11	Leave unconnected
34	27	ADDRESS_12	Leave unconnected
35	28	ADDRESS_13	Pull down required
36	29	ADDRESS_14	Pull down required
37	30	ADDRESS_15	Pull down required
38	31	ADDRESS_16	Pull down required
39	-	VDD3.3	2.5 – 3.3 Voltage supply
40	32	ADDRESS_17	Pull down required
41	33	TEST	Production test should be connected to "0"
42	-	GND	Ground
43	34	PLLBYPASS	Bypasses the internal Pll should be connected to "0"
44	35	GND	Ground
45	36	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
46	-	GND	Ground
47	37	X1	Crystal oscillator input
48	38	X2	Crystal oscillator output
49	-	VDD3.3	2.5 – 3.3 Voltage supply

50	39	THRESHOLD	Voltage level used for Brown out detection. If the Voltage goes below this level the processor will be stop all accesses towards the flash.
51	40	SWIPE_DATA_P	Leave unconnected
52	41	SWIPE_DATA_N	Leave unconnected
53	42	SWIPE_CLK_N	Leave unconnected
54	43	SWIPE_CLK_P	Leave unconnected
55	-	GND	Ground
56	44	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
57	45	VDD3.3	2.5 – 3.3 Voltage supply
58	46	SENSOR_PWRDN	Can be used to turn of the power to the sensor SENSOR_PWRDN = "0" => Sensor is active SENSOR_PWRDN = "1" => Sensor could be tuned off
59	47	SENSOR_RST_N	Reset signal to the sensor
60	48	SENSOR_IRQ	Can be used as wake up the system if there is a finger detect function in the system. FPC2020 will get an interuped when SENSOR_IRQ is "1". Connect this signal to "0" if the function is not used.
61	49	AREA_SCN	Sensor chip select SPI signal
62	-	VDD3.3	2.5 – 3.3 Voltage supply
63	50	AREA_SCLK	Sensor clock SPI signal
64	51	AREA_SO	Sensor data out SPI signal pull down required
65	52	AREA_SI	Sensor data in SPI signal pull down required
66	53	RESET	Reset signal should be kept low for 10 ms at power up to ensure a correct behavior at start up. This could be achieved with a RC net.
67	-	GND	Ground
68	54	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
69	55	VDD1.8	Internal generated 1.8 V core voltage should be externally decoupled to ground
70	56	VDD3.3	2.5 – 3.3 Voltage supply
71	57	VDD3.3	2.5 – 3.3 Voltage supply
72	58	FLASH_SCN	Flash chip select SPI signal pull up required
73	59	FLASH_SO	Flash data out SPI signal pull down required
74	60	FLASH_SCLK	Flash clock SPI signal
75	61	FLASH_SI	Flash data in SPI signal pull down required
76	-	GND	Ground
77	62	OEN	Pull down required
78	63	CSN	Pull down required
79	-	VDD3.3	2.5 – 3.3 Voltage supply
80	64	WEN	Pull down required

**Table 8**

FPC2020 Area Pin configuration for the 64 pins QFN

### Area sensor interface

FPC2020 can be directly connected to FPC1011C with only a few passive components see table below. It is also possible to add extra functions for a more power efficient solution. By letting the SENSOR\_PWRDN signal to control the power supply to the sensor it is possible to limit the power consumption for the sensor to only be active then needed. It is important to ensure that there is no power drops on the power due to the switching of the sensor supply. If there is some kind of hard ware finger present function available in the system it could be connected to the SENSOR\_IRQ signal to only wake up the system then a finger is present on the sensor.

FPC2020	FPC1011C
AREA_SCN	CS_N
AREA_SCLK	SCK
AREA_SO	SI
AREA_SI	SO
SENSOR_RST_N	RSTN
-	VDD 2.5 – 3.3 V
-	GND

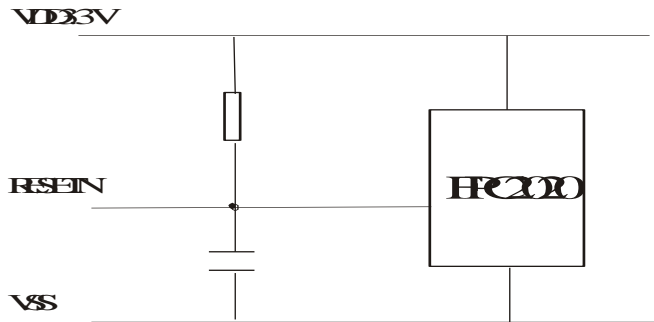
**Table 9**

Connection between FPC2020 & FPC1011C

## Reset

The external reset signal should be active for 10 ms after power up of FPC2020 to get a proper startup sequence. This could be achieved by having a RC net on the reset signal or by letting the host processor manage this. The reset signal is asynchronous activated while releasing will be synchronized internal in FPC2020 and requires a clock. The processor will not start until the reset signal is released.

After the system is powered up the reset block monitors the voltage supply and if the voltage is below 2.1 V a reset signal will be generated by FPC2020 it self. This is done to protect the processor for running with the power supply out of range.



**Figure 8**  
Implementation of a delayed reset signal

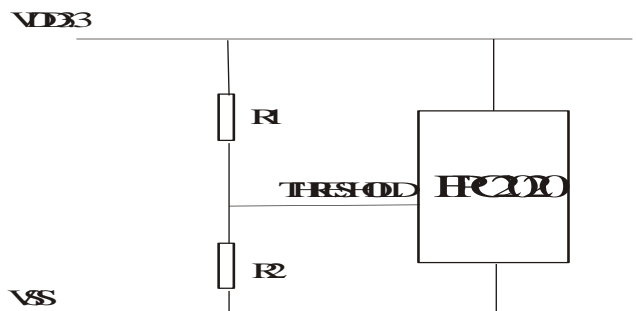
## Brown Out detection

The function of the Brown Out Detector is to generate an interrupt signal to the processor whenever the voltage supply falls below a predetermined threshold. This interrupt signal will temporary stop all commands to FPC2020 until the power is ok again. The brown-out detector compares a divided replica of the input voltage with the internal reference of 1.2V. A voltage divider of two serial connected resistors (denoted R1 and R2) should be placed between pins GND and the voltage supply and midpoint should be connected to input pin THRESHOLD. Different flash circuits have different supply voltages and the threshold level for the brown out should be adjusted to this.

Threshold (V)	R1(Mohm)	R2 (Mohm)
2.3	1.1	1.2
2.5	1.3	1.2
2.7	1.5	1.2
2.9	1.7	1.2

**Table 10**  
Recommended values for R1 and R2

Depending on the tolerance for the selected resistors and the build in uncertainty for brown out detection of 5 % there will be a total uncertainty for the system. To get a stable system it's requires a margin between operating voltage and the selected threshold to avoid unnecessary stops in the system. If this function is not needed a low threshold (1.8V) should be selected.



**Figure 9**  
Configuration resistor for threshold voltage level

## Flash interface

The Flash interface is a serial SPI interface and supports number of different flash circuits from different manufacturers. Since the flash is used for template storage it requires that the flash is a small sector type. FPC2020 have been tested with number flash circuits and it is strongly recommended to use one of the tested devices.

Vendor	Size (Mbit)	Part number
Atmel	4	AT45DB041D
Atmel	8	AT45DB081D
ST	2	M45PE20
ST	4	M45PE40
ST	8	M45PE80

**Table 11**

*Recommended flash circuits to use together with FPC2020. Grey shadow devices are not fully verified yet.*

Before a system with flash can be used the main application needs to be loaded into the flash memory. There is three alternative summarized in the table below how this can be done:

Alternative	Program format	Programmer
Unmounted components	Intel hex	Intel hex programmer
PCB with test pads	Intel hex	Intel hex programmer
HOST interface UART/SPI	Bin	Ordinary FPC2020 commands UART or SPI

**Table 12**

*Flash program alternatives*

FPC delivers the main program in a standard Intel hex or bin format. The Intel hex format is to be used together with a standard flash programmer. The programming needs to be done before the flash is mounted on the PCB or as alternative the PCB could contain test pads for flash programming. The bin format could be used after the flash is mounted and the host (SPI or UART) interface to FPC2020 is used for programming the flash. The bin format is also used for program updates. For a system with a flash that hasn't been program yet it's required that a flash installer program is installed first. The program is downloaded into RAM in FPC2020 with the UART or SPI host interface and after that the program to be stored in flash is downloaded. For me information see Boot Mode chapter.

The only way to stop FPC2020 booting from the flash is to disconnect or reset the flash circuit during the boot sequence. Normally this should not be necessary. But if an unforeseen hardware or software error has occurred it could be vital. It also gives the possibility to start up a temporary program from the host interface while the original program is intact in the flash memory.

## Host interface

The host interface is a combined UART and SPI interface. HOST\_SI and HOST\_SO are used for both UART and the SPI interface. While HOST\_SCN is a select signal between UART and SPI interface. HOST\_SCN is also used as a chip select signal if the SPI interface is used. HOST\_SCLK is only used as clock for the SPI interface. The UART interface is using LVTTTL/LVCMOS levels and if RS232 levels are required a converting circuit is needed.

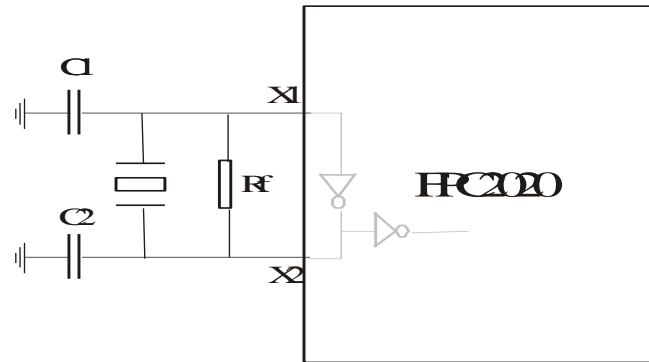
If FPC2020 is in power save mode or in sleep mode FPC2020 could be waken by toggle HOST\_SCN or HOST\_SI. Be aware of the start up time for FPC2020 that could cause that FPC2020 needs to be waked up before start sending commands. For more information pleas look in chapter Power modes.

## Clock

FPC2020 needs to have an external clock of 7,3728 MHz to work correct. It could be a clock with LVCMOS levels or a crystal or ceramic resonator. If a clock is used as external clock source it should be connected to X1 and X2 should be leaved unconnected.

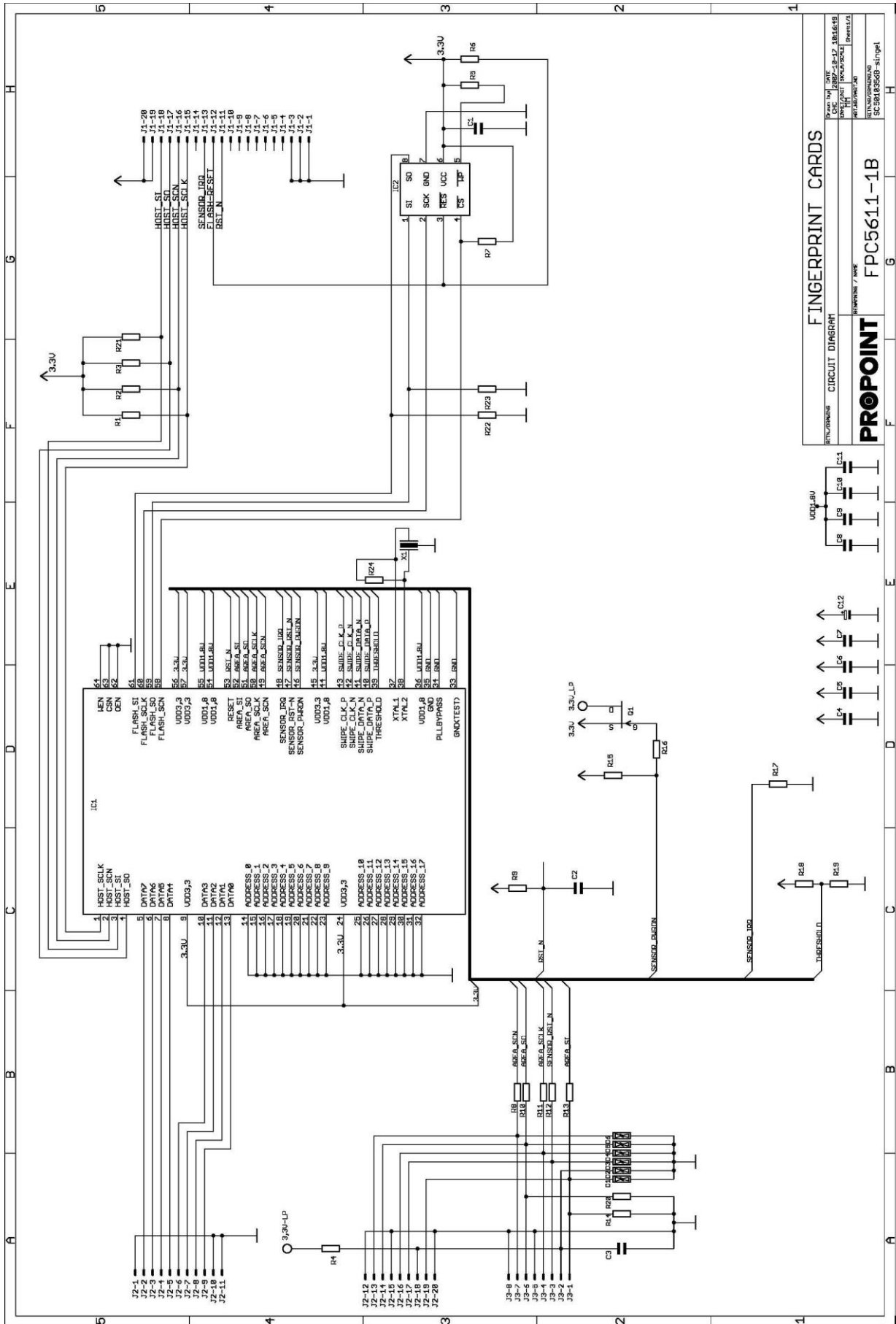
If a crystal or a ceramic resonator is used X1 and X2 are input and output, respectively, of an inverting amplifier used as crystal oscillator, as shown in figure below. Values for C1, C2 and Rf depend of the

crystal or resonator in use. Please follow the recommended values from the manufacture of the crystal or resonator.



**Figure 10**  
*Circuit example for crystal or resonator used as clock source*

## Schematic



## Bom

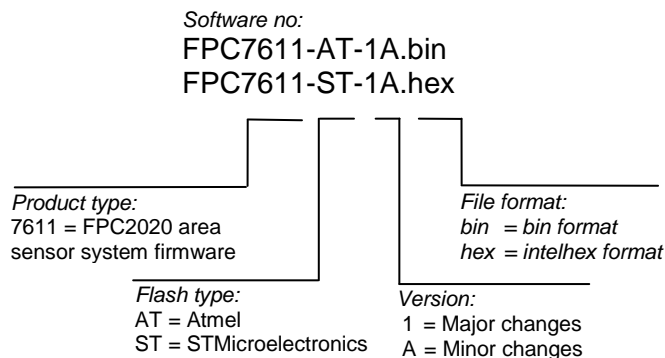
Part Type	Designator	Comments
100 nF	C1	Serial Flash
10 nF	C2	RESET
4.7 uF	C3	Area sensor
100 nF	C4	FPC2020 3,3 V
100 nF	C5	FPC2020 3,3 V
100 nF	C6	FPC2020 3,3 V
100 nF	C7	FPC2020 3,3 V
100 nF	C8	FPC2020 1,8 V
100 nF	C9	FPC2020 1,8 V
100 nF	C10	FPC2020 1,8 V
100 nF	C11	FPC2020 1,8 V
100 uF	C12	3,3 V
CDS2C16GTH	D1	Ceradiodes ESD protection
CDS2C15GTA	D2	Ceradiodes ESD protection
CDS2C16GTH	D3	Ceradiodes ESD protection
CDS2C16GTH	D4	Ceradiodes ESD protection
CDS2C16GTH	D5	Ceradiodes ESD protection
CDS2C16GTH	D6	Ceradiodes ESD protection
47 kohm	R1	Host int
47 kohm	R2	Host int
47 kohm	R3	Host int
10 ohm	R4	Area Sensor power filter
47 kohm	R5	Serial Flash
47 kohm	R6	Serial Flash
47 kohm	R7	Serial Flash
10 ohm	R8	Area sensor
47 kohm	R9	pull up reset
10 ohm	R10	Area sensor
10 ohm	R11	Area sensor
10 ohm	R12	Area sensor
10 ohm	R13	Area sensor
47 kohm	R14	Pull up area_si
47 kohm	R15	3V3_LP
4,7 kohm	R16	3V3_LP
47 kohm	R17	pull down sensor_irq
1 Mohm	R18	Threshold
1 Mohm	R19	Threshold
47 kohm	R20	Area sensor
47 kohm	R21	Host int
47 kohm	R22	Flash int
47kohm	R23	Flash int
1 Mohm	R24	Rf
FPC2020	IC1	FPC2020 fingerprint ASIC
AT45DB081D-SU-2,5	IC2	Serial Flash Package 8S2-EIAJ SOIC 0.209" Body
SI2315DS	Q1	Vishay MOSFET
CSTCR7M37G53-R0	X1	7,3728 MHz keramisk resonator

## Document revision history

REVISION	DATE	CHANGE	Firmware version*
A	2007-10-08	First version	Version 1A
B	2007-10-16	-Corrected the factory default security setting info. -Added a feedback resistor in the schematic for the oscillator. -Added a 80 pins TQFP package	Version 1A

\* This field states which firmware version that is valid for a certain revision of the specification. Minor changes in the firmware will only result in a new firmware release, but no new revision of the specification. All major software changes will result in an updated specification. A major change in the firmware will result in stepping the number in the version while a minor change will result in stepping the letter in the version.

## Firmware information



## Contact information

*Main office*  
Fingerprint Cards AB  
P.O. Box 2412  
SE-403 16 Göteborg  
Sweden

*Telephone*  
+46 (0)31 60 78 20

*Fax*  
+46 (0)31 13 73 85

*Web site*  
[www.fingerprints.com](http://www.fingerprints.com)

*E-mail*  
[sales@fingerprints.com](mailto:sales@fingerprints.com)  
[tech@fingerprints.com](mailto:tech@fingerprints.com)

*Visiting address*  
Västra Hamngatan 8