Table 7.1 Bootloader Status

Register value	Status
0xA5	Bootloader is available
0xEE	No bootloader

*Note the bootloader is available on the standard IQS5xx-B000 firmware; this could possibly be unavailable on custom firmware versions.

7.9 Version Information

7.9.1 Product Number

The different IQS5xx devices can be identified by their relevant product numbers.

Table 7.2 Product Number

Product Number (decimal)	Device
40	IQS550
58	IQS572
52	IQS525

7.9.2 Project Number

The project number for the generic B000 project is 15 (decimal) for all devices.

7.9.3 Major and Minor Versions

These will vary as the B000 is updated, this datasheet relates to the version as indicated at the bottom of the *Overview* Section *1*.

7.10 Unique ID

A 12-byte unique ID can be read from memory map address 0xF000 – 0xF00B. This number gives each individual IC a unique identifier.

7.11 Switch Input

The SW_IN (switch input) pin, when enabled (*SW_INPUT*), will display the state of the input pin to the master controller (*SWITCH_STATE*). This state is updated before each I²C session.

The input can be configured as active LOW or active HIGH (*SW_INPUT_SELECT*). For active LOW, an internal pull-up resistor (typical value of 40kΩ) is connected to the SW_IN pin.

A change in the state of the SW_IN can also trigger an event, see Section 8.8.1. This input can be used as an additional switch or proximity sensor, and has the ability to wake the IQS5xx from the extreme (<1uA) low power suspend state.

8 I²C

The IQS5xx communicates via the standard I²C communication protocol.

Clock stretching can occur, thus monitoring the availability of the SCL is required, as per standard I²C protocol.

8.1 Data Ready (RDY)

An additional RDY I/O indicates (active HIGH) when the communication window is available with new data for optimal response. Polling can however be used, but is not recommended. RDY should be connected to an interrupt-on-change input for easier implementation and optimal response time.

8.2 Slave Address

The default 7-bit device address is ‘1110100’. The device address can be modified during programming. The full address byte will thus be 0xE9 (read) or 0xE8 (write).

8.3 16-bit Addressing

The I²C employs a 16-bit address to access all individual registers in the memory map.

8.4 I²C Read

The master can read from the device at the *current address* if the address is already set up, or when reading from the default address.

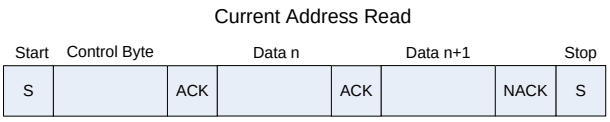

Figure 8.1 Current Address Read

The master can perform a *random read* by specifying the address. A WRITE is performed to set up the address, and a repeated start is used to initiate the READ section.

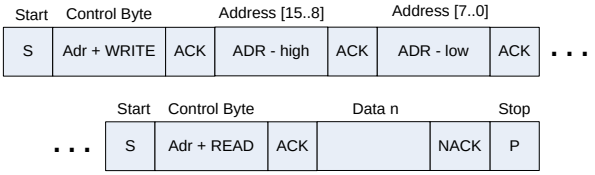

Figure 8.2 Random Read

8.4.1 Default Read Address

When a new communication window begins, the configurable *default read address* is used if a current address read is performed (no address is specified). If an application will always read from a specific register, the IQS5xx can be configured to point to the required register, negating the need to specify the address at each new communication window, allowing for faster data reading.

8.5 I²C Write

The master uses a *Data Write* to write settings to the device. A 16-bit data address is always required, followed by the relevant data bytes to write to the device.

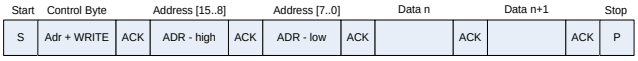

Figure 8.3 Data Write

8.6 I²C Timeout

If the communication window is not serviced within the *I²C timeout* period (in milliseconds), the session is ended (RDY goes LOW), and processing continues as normal. This allows the system to continue and keep reference values up to date even if the master is not responsive.

8.7 End of Communication Session / Window

Unlike the previous A000 implementation, an I²C STOP will **not** terminate the communication window. When all required I²C transactions have been completed, the communication session must be terminated manually. This is achieved by sending the *End Communication Window* command, by writing a single byte (any data) to the address 0xEEEE, followed by a STOP. This will end the communication window, RDY will go low and the IQS5xx will continue with a new sensing and processing cycle.

8.8 Event Mode Communication

The device can be set up to bypass the communication window when no activity is sensed (*EVENT_MODE*). This is usually enabled since the master does not want to be interrupted unnecessarily during every cycle if no activity occurred. The communication will resume (RDY will indicate available data) if an enabled event occurs. It is recommended that the RDY be placed on an interrupt-on-pin-change input on the master.

8.8.1 Events

Numerous events can be individually enabled to trigger communication, they are:

- Trackpad events (*TP_EVENT*): event triggered if there is a change in X/Y value, or if a finger is added or removed from the trackpad
- Proximity events (*PROX_EVENT*): event only triggers if a channel has a change in a proximity state
- Touch events (*TOUCH_EVENT*): event only triggers if a channel has a change in a touch state
- Snap (*SNAP_EVENT*): event only triggers if a channel has a change in a snap state
- Re-ATI (*REATI_EVENT*): one cycle is given to indicate the Re-ATI occurred (*REATI_OCCURRED*).
- Proximity on ALP (*ALP_PROX_EVENT*): event given on state change
- Switch input (*SW_INPUT_EVENT*): event triggers if there is a change in the input pin state.

The proximity/touch/snap events are therefore mostly aimed at channels that are used for traditional buttons, where you want to know only when a status is changed.

8.8.2 Force Communication

The master can initiate communication with the IQS5xx, even while RDY is LOW. The

IQS5xx will clock stretch until an appropriate time to complete the I²C transaction. The master firmware will not be affected (as long as clock stretching is correctly handled).

For optimal program flow, it is suggested that RDY is used to sync on new data from the IQS5xx. The forced method is only recommended if the master must perform I²C and *Event Mode* is active.

NOTE: If the IQS5xx is in a low-power state when the master forces the communication, the first addressing will respond with a NACK. The master must repeat the addressing (wait a minimum of 150us after the I²C STOP before retrying), and the IQS5xx will be ready and ACK the transaction.

Figure 8.4 shows a forced communication transaction. Communication starts with RDY = LOW. The IQS5xx is in a low power state on the first request, and a NACK is sent. After the second request the IQS5xx responds with an ACK. The IQS5xx clock stretches until an appropriate time to communicate (to prevent interference with the capacitive measurements). When appropriate, the clock is released and the transaction completes as normal. RDY is not set during a forced communication transaction.

Figure 8.4 Forced communication

8.9 Memory Map Registers

The registers available in the memory map, via I²C, are provided in this section. The memory map starts with a READ-ONLY section, followed by a READ/WRITE section. The read/write permissions are indicated by the shading in the 'R' (read) and/or 'W' (write) columns.

Certain registers in the memory map have defaults loaded from non-volatile memory, which can be configured during programming; these are highlighted also in the 'E²' column.

Table 8.1 Direct-Addressable Memory Map

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Details	R	W	E²
0x0000 - 0x0001	Product number (2 bytes)								(See 7.9)			
0x0002 - 0x0003	Project number (2 bytes)											
0x0004	Major version											
0x0005	Minor version											
0x0006	Bootloader status								(See 7.8.1)			
0x0007 - 0x000A	Open (4 bytes)											
0x000B	Max touch column				Max touch row				(See 3.5.5)			
0x000C	Previous cycle time [ms]								(See 4.1.1)			
0x000D	-	-	SWIPE_Y-	SWIPE_Y+	SWIPE_X+	SWIPE_X-	PRESS_AND_HOLD	SINGLE_TAP	Gesture Events 0			
0x000E	-	-	-	-	-	ZOOM	SCROLL	2_FINGER_TAP	Gesture Events 1			
0x000F	SHOW_RESET	ALP_REATI_OCCURRED	ALP_ATI_ERROR	REATI_OCCURRED	ATI_ERROR	CHARGING_MODE			System Info 0			
0x0010	-	-	SWITCH_STATE	SNAP_TOGGLE	RR_MISSED	TOO_MANY_FINGERS	PALM_DETECT	TP_MOVE-MENT	System Info 1			
0x0011	Number of fingers								(See 5.2.1)			
0x0012 - 0x0013	Relative X [pixels] (2 bytes)								(See 5.2.2)			
0x0014 - 0x0015	Relative Y [pixels] (2 bytes)											
0x0016 - 0x0017	Absolute X position [pixels] (2 bytes)								(See 5.2.3)			
0x0018 - 0x0019	Absolute Y position [pixels] (2 bytes)											
0x001A - 0x001B	Touch strength (2 bytes)								(See 5.2.4)			

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Details	R	W	E²
0x001C	Touch area / size								(See 5.2.5)			
0x001D : 0x0038	Repeat: *Absolute X* *Absolute Y* *Touch strength* *Touch area / size* For fingers 2 - 5											
0x0039 - 0x0058	Prox status (32 bytes)								(See 8.10.5)			
0x0059 - 0x0076	Touch status (30 bytes)											
0x0077 - 0x0094	Snap status (30 bytes)											
0x0095 - 0x01C0	Count values (300 bytes)								(See 8.10.6)			
0x01C1 - 0x02EC	Delta values (300 bytes)											
0x02ED - 0x02EE	ALP count value (2 bytes)								(See 3.3.2)			
0x02EF - 0x0302	ALP individual count values (20 bytes)											
0x0303 - 0x042E	Reference values (300 bytes)								(See 8.10.6)			
0x042F - 0x0430	ALP LTA (2 bytes)								(See 3.4.2)			
0x0431	ACK_ RESET	-	AUTO_ ATI	ALP_ RESEED	RESEED	MODE_SELECT			System Control 0			
0x0432	-	-	-	-	-	-	RESET	SUSPEND	System Control 1			
0x0433 - 0x0434	Open (2 bytes)											
0x0435 - 0x043E	ALP ATI compensation (10 bytes)								(See 3.6.2)			
0x043F - 0x04D4	ATI compensation (150 bytes)											