Interface Guide for MAX32664 Sensor Hub-Based Reference Design Platforms

UG7148; Rev 3; 10/20

Abstract

A number of reference design platforms, including the MAXREFDES101, MAXREFDES103, and MAXREFDES220, are available for Maxim customers to evaluate and develop a wide range of Maxim products for health-sensing applications. These platforms include biosensors, power-management ICs (PMIC), and microcontrollers that allow the capture of bio signals important to healthcare. Different embedded algorithm firmwares are available for this platform to calculate heart rate, SpO_2, or blood pressure. This document describes the commands and settings needed for communication with these platforms.

Table of Contents

List of Figures

List of Tables

Introduction

The sensor hub-based reference designs provide unique evaluation and development platforms in a wearable form factor. Each reference design include a biosensor (e.g., the MAX86141 or MAX30101), a variant of the MAX32664 that runs the algorithm firmware (e.g., the MAX32664A, B, C, or D, depending on the algorithm), a 6-axis accelerometer/gyroscope, a PMIC, and a host microcontroller (the MAX32630) that communicates with the MAX32664 sensor hub through an I2C interface. The host microcontroller firmware has a set of commands accessible by the user through the serial port or Bluetooth® LE[1] to configure the MAX32664 and sensors, start/stop the algorithm, and stream data. The same interface commands are used by the Maxim DeviceStudio PC GUI to communicate with the platform.

This document describes the protocol and commands to interface the host microcontroller firmware on the Maxim reference design platform, including:

- Generic commands common to all MAX32664 variants
- MAX32664 variant-specific commands and settings

[1]The Bluetooth LE connection does not support firmware updates. Also, the effective streaming speed is limited to 100Hz.

1 Architecture

Figure 1 depicts the high-level architecture of the MAX32664-based reference design platforms. Depending on the variant, it may include different biosensors and algorithms, as shown in **Table 1**. A host MCU bridges the user interface to the sensors and algorithms as seen in **Figure 1** with a set of commands through UART or Bluetooth LE. The user can launch the Maxim DeviceStudio Windows® GUI or an Android® application[2] to communicate with the platform using a graphical interface. Alternatively, a terminal program like Tera Term can provide more flexibility to directly send/receive commands/responses to/from the host.

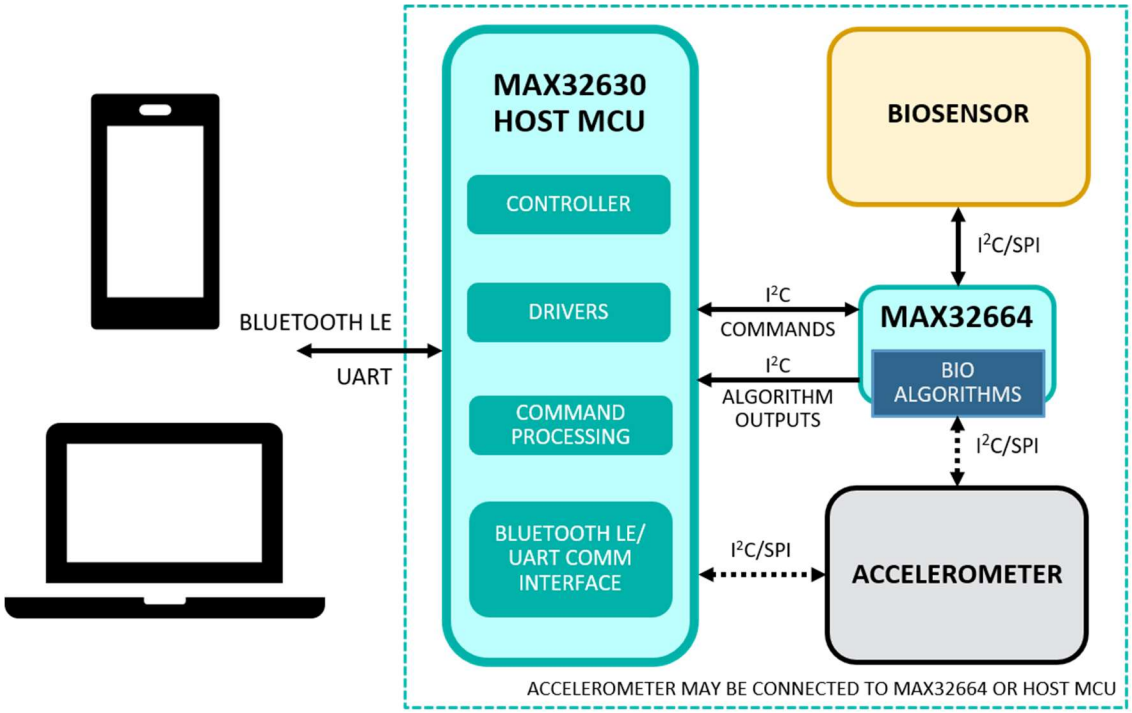

Figure 1. Architecture diagram.

Table 1. MAX32664 Sensor Hub-Based Reference Design Platforms

PLATFORM	SENSOR HUB VARIANT	SENSOR	FEATURE
MAXREFDES220	MAX32664A	MAX30101/2	Heart rate, SpO_2 using fingertip
MAXREFDES101	MAX32664B	MAX86140/1	Heart rate using wrist
MAXREFDES103	MAX32664C	MAX86140/1	Heart rate, SpO_2 using wrist
MAXREFDES220-BPT	MAX32664D	MAX30101/2	Heart rate, SpO_2, blood pressure using fingertip

Android is a registered trademark of Google Inc.
Windows is a registered trademark of Microsoft Corporation.

[2]Android application functionality is limited monitoring, displaying, and recording data with a subset of preselected configurations.

2 Communication Setup

The PC connection is established using a serial-to-USB cable, or Bluetooth LE interface. The COM port connection is set to:

- Baud = 115200 or 230400
- Parity = none
- Data bits = 8
- Stop bits = one

After a connection has been established, the PC client application (GUI or terminal) can send commands to the platform and can also listen for asynchronous messages from the platform. These messages may include responses to commands and streaming data sent to the PC automatically when a sensor is active.

Each command and each message to/from the platform must be terminated by a new-line character '`\n`'.

The PC-to-platform communication uses a command + response format with the PC acting as the master. The PC can request data, send configuration parameters, and so on. The platform then responds with the requested data and confirmation.

2.1 PC-to-Platform Command Format

A typical command includes a command string followed by a number of parameters (if required), separated by spaces:

```
<command> <optional parameter1> <optional parameter2> …
```

2.2 Platform-to-PC Response Format

The platform will respond to a command by echoing the command itself, followed by any necessary token/value pair (optional), an err/value pair (mandatory), and finally a newline ('\n') character (mandatory).

```
<echoed command> <token1>=<value1> … <tokenN>=<valueN> err=<status>\n
```

If the value of a token is a list of item or values (e.g., dumping registers of a sensor), the item/values are pairs that are comma separated within `{ }` brackets.

```
<echoed command> <token>= {<item1>,<value1>},{<item2>,<value2>},…,
{<itemN>,<valueN>}, err=<status>\n
```

The error code `=0` represents a successful command. Other values represent an error type as shown in **Table 2.**

Example:

```
reset
reset err=0

get_device_info
```

```
get_device_info platform=SmartSensor_MAX32660 firmware_ver=HSP2SPO2_3_2.0
sensors=ppg algo_ver_ppg=unknown part_name_ppg=max8614x hub_firm_ver=30.6.0
fw_algos=agc,scd,aec,wspo2,whrm  err=0


dump_reg ppg
dump_reg ppg reg_val={0,0},{1,0},{2,0},{3,0},{4,0},{5,0},{6,0},
{8,F},{9,0},{A,0},{B,0},{C,0},{D,0},{E,0},{F,0},{10,0},{11,0},{12,0},{13,F},{
14,FF},{15,FF},{16,FF},{17,FF},{18,0},{19,0},{1A,0},{1B,0},{1C,7C},{1D,7E},{1
E,1},{1F,20},{20,0},{21,0},{30,0},{FE,6},{FF,15} err=0
```

Table 2. Error Status Code Description

ERROR CODE	DESCRIPTION
0	Command is executed successfully
-1	An error with unspecified source occurred
-2	File system access error
-3	I²C access error
-4	Error in finding the appropriate driver
-5	Error in finding the sensor
-6	Error in finding library/algorithm
-7	Not enough memory
-8	General driver error
-254	Invalid or missing parameter
-255	Unknown command

3 Commands and Settings

3.1 Host Commands to Control Sensor Hub (All Variants)

The commands described in **Table 3** are generally used in different variants of the platform. The response to a command includes the echoed command and list of token/values (shown in **Table 3**) followed by the error status, as described in the previous section.

Table 3. Host Commands—General for All Variants

FORMAT	RESPONSE TOKEN/VALUE LIST (HEX)
`reset`	Reset the MAX32664 sensor hub. (Host will not be restarted.)
`get_device_info`	Get information about the connected platform, connected sensors, and algorithms. Response includes: platform, platform version, sensors list, algorithm version, and algorithm list
`read <sensor> <num>`	Start sensor(s) and algorithm (depending on command). Streaming begins after this command. `<sensor>`: • `ppg`: Sensor platform for heart rate or blood oxygen (SpO_2) • `bpt`: Sensor platform for blood pressure trending (BPT) `<num>`: • Streaming type. See the sensor hub variant-specific sections for more information.

`get_format <sensor> <num>`	Show the streaming report format in a comma-delimited list, tagging each field.
	`<sensor>`:
	• `ppg`: Sensor platform for heart rate or blood oxygen (SpO_2)
	• `bpt`: Sensor platform for BPT
	`<num>`:
	• Streaming type. See the sensor hub variant command sections for more information.
	Example:
	`get format ppg 4`
	`format=smpleCnt,grnCnt,led2,led3,grn2Cnt,irCnt,redCnt,accelX,accelY,accelZ,opMode,hr,hrconf,rr,rrconf,activity,r,wspo2conf,spo2,wspo2percentcomplete,wspo2lowSNR,wspo2motion,wspo2lowpi,wspo2unreliableR,wspo2state,scdstate err=0`
	Refer to the Quick Start Guide for the appropriate sensor hub variant for the description of each field.

`stop`	Stop algorithm and sensors streaming data.
`pause <value>`	Disable display of streaming data in terminal. `<value>`: • `0`: Turn off pause (resume displaying data). • `1`: Turn on pause (pause displaying data).
`get_reg <sensor> <addr>`	Get the value of a sensor register (in hex). `<sensor>`: • `ppg`: Sensor platform for heart rate or blood oxygen (SpO_2) • `bpt`: Sensor platform for BPT `<addr>`: • Address[2] of register in hex

`set reg <sensor> <addr> <value>`	Set the value of a sensor register (in hex). `<sensor>`: • `ppg`: Sensor platform for heart rate or blood oxygen (SpO_2) • `bpt`: Sensor platform for BPT `<addr>`: • Address[2] of register in hex `<value>`: • Value[2] of register in hex
`dump_reg <sensor>`	Dump all registers and their values for a sensor register (in hex). `<sensor>`: • `ppg`: Sensor platform for heart rate or blood oxygen (SpO_2) • `bpt`: Sensor platform for BPT Response includes a list of comma-separated `{reg,val}` pairs, e.g., `{0,0},{1,FF},{2,F0},etc`
`set_cfg blepower <hex val>`	Set power level as a 1-byte value. 0x00 = 3dbm 0x01 = 0dbm 0x02 = -3dbm 0x03 = -6dbm 0x04 = -9dbm 0x05 = -12dbm 0x06 = -15dbm
`get_battery_level`	Get the battery charge % (0 to 100 in decimal), subject to firmware support.

[2]For the list of registers and their usage, refer to the sensor data sheet.

3.1.1 Sensor Hub Authentication for Maxim Wellness Library

In addition to the algorithms embedded in MAX32664 sensor hub (e.g., heart rate, SpO_2), a group of higher-level algorithms are also available as part of the Maxim Wellness library. This set of algorithms utilizes sensor hub output data. These algorithms include heart rate variation, stress evaluation, sleep quality assessment, respiration rate detection, and sports coaching.

Maxim Wellness library algorithms require an authentication mechanism to validate the presence of the sensor hub. The authentication process includes a series of message exchanges between the library and the MAX32664 sensor hub, facilitated by the host as the message forwarder. In an authentication session, the host sends commands to the sensor hub and provides the responses to the Maxim Wellness library. Similarly, it also receives responses from the library and provides them to the sensor hub.

Table 4 shows the host commands used during an authentication session and Error! Reference source not found. shows their sequence.

Table 4. Host Commands—Authentication for Maxim Wellness Library

FORMAT	RESPONSE TOKEN/VALUE LIST (HEX)
`get_cfg sh_dhparams`	Get a random authentication initials from the sensor hub. The value is a 6-byte long initial string. Example: `get_cfg sh_dhparams value=372A1D880772 err=0`
`set cfg sh dhlpublic <value1> <value2> … <value12>`	Send a 12-byte public key to the sensor hub. Values are in hex and space separated. Example: `set cfg sh dhlpublic 00 01 02 03 04 05 06 07 08 09 0A 0B`
`get_cfg sh_dhrpublic`	Get a 12-byte remote public key from the sensor hub. The value is a 12-byte long string. Example: `get cfg sh dhrpublic value=375374713666774477645652 err=0`
`get_cfg sh_auth`	Get a 32-byte authentication sequence from the sensor hub. The value is a 32-byte long string. Example: `get cfg sh auth value=4D46040CB2E362ACEE145352C4C6BD28EB03D776802EC017B473C07842258F47 err=0`

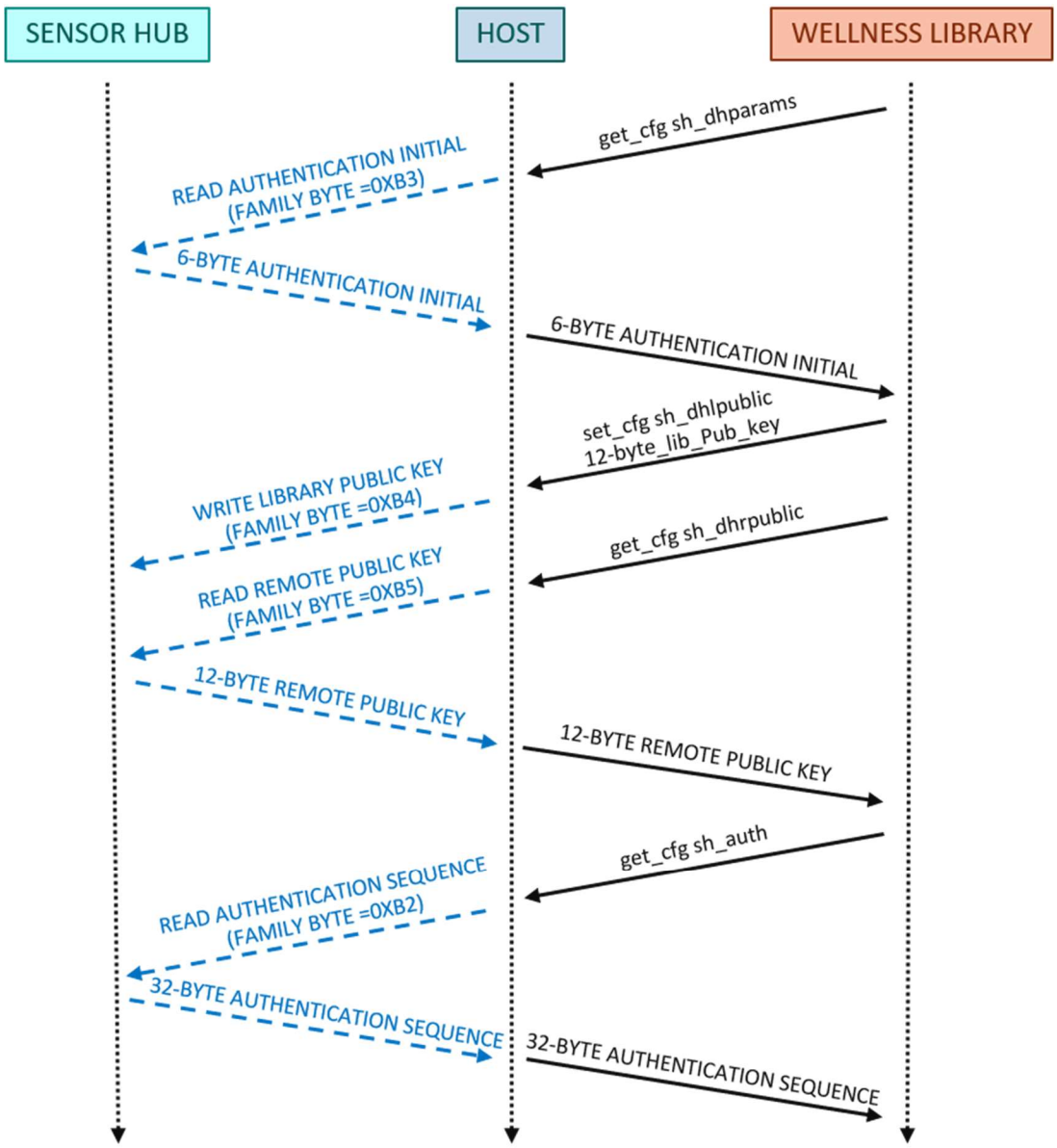

Figure 2. Authentication sequence.

3.2 Host Commands for the MAX32664A

The commands in **Table 5** are specific to the MAX32664A sensor hub. Unless otherwise specified, `<values>` are entered in hex.

Note: Each host command may invoke one or more MAX32664 sensor hub command to complete the desired action. For the list of sensor hub commands and the description of the fields in the streaming data, refer to the **AN7087 MAX32664A Quick Start Guide**.

Example:

```
set_cfg ppg agc 1
read ppg 0
…
stop
```

Table 5. Host Commands—MAX32664A

FORMAT	RESPONSE TOKEN/VALUE LIST (HEX)
CONFIGURATION: ALGORITHM	
`set cfg ppg agc <value>`	Enable/disable AGC
STREAMING-RELATED COMMANDS	
`read ppg 0`	Algorithm (heart rate + SpO_2) + sensor hub accelerometer + sensor streaming at 100Hz. Use `get format ppg 0` for streaming format.
`read ppg 1`	Algorithm (heart rate + SpO_2) streaming at 100Hz (sensor hub accelerometer is enabled). Use `get format ppg 1` for streaming format.
`read ppg 2`	Algorithm (heart rate + SpO_2) + host-side accelerometer + sensor streaming at 100Hz. Use `get format ppg 2` for streaming format.
`read ppg 3`	Algorithm (heart rate + SpO_2) + sensor streaming at 100Hz, no accelerometer is used. Use `get format ppg 3` for streaming format.
`read ppg 4`	Algorithm (heart rate + SpO_2) + sensor hub accelerometer + sensor streaming at 100Hz, extended report. Use `get format ppg 4` for streaming format.
`read ppg 5`	Algorithm (heart rate + SpO_2) + host-side accelerometer + sensor streaming at 100Hz, extended report. Use `get format ppg 5` for streaming format.
`read ppg 6`	Algorithm (heart rate + SpO_2) + sensor streaming at 100Hz, extended report; no accelerometer is used. Use `get format ppg 6` for streaming format.

3.3 Host Commands for the MAX32664B and MAX32664C

The commands in **Table 6** are specific to both the MAX32664B and MAX32664C sensor hubs, except SpO_2-related commands that are only supported by the MAX32664C. Unless otherwise specified, <values> are entered in hex.

Note: Each host command may invoke one or more MAX32664 sensor hub commands to complete the desired action. For the list of sensor hub commands and the description of the fields in the streaming data, refer to the **AN6922 MAX32664B Quick Start Guide** or **AN6924 MAX32664C Quick Start Guide**.

Example:

```
set_cfg accel_sh 1
set_cfg stream ascii
read ppg 4
…
stop
```

Table 6. Host Commands—MAX32664B and MAX32664C

FORMAT	RESPONSE TOKEN/VALUE LIST (HEX)
CONFIGURATION: SENSOR HUB	
set_cfg stream <type>	Define the format of streaming data <type> as: • ascii: Suitable when using terminal • bin: When using GUI (automatically set by GUI) [default]
set_cfg report <type>	Set type of streaming report <type> as: • 1: Brief report [default] • 2: Extended report with more info (subject to firmware support)
set_cfg accel_sh <value>	Set type of accelerometer used as in **Figure 1**. <value>: • 0: Host-side accelerometer • 1: Sensor hub-side accelerometer [default]
set_cfg flash log <value>	Enable/disable logging streaming data in flash.
set_cfg i2c_addr <value>	Change the MAX32664 I^2C slave address from default (0xAA) to <value>. The new address will be effective only AFTER sending the response of this command to the host. This should only be used if multiple I^2C slaves are connected to the host.
set_cfg sh_mode <value>	Use this command with <value> = 1 to shut down the MAX32664. **Restart is only possible by power cycle or toggling RSTN.**
get_cfg sh_counter	Reads the internal 32-bit counter of the sensor hub for debugging purposes. The LS byte represents chip revision (A1 or A2).

`set cfg hubaccel op cfg <value1> <value2> <value3>`	Enable/disable wake up on motion detection for the sensor hub KX122 accelerometer (three 8-bit values):
	`<value1>`: Enable wake up on motion:
	• `0`: Disabled
	• `1`: Enabled
	
	`<value2>`: WUFC*: the time in seconds in which motion should be present before a wakeup interrupt.
	WUFC = desired time (s) x 25
	
	`<value3>`: ATH*: the motion level threshold
	ATH = desired threshold (g) x 16
	
	Example: For a 0.2s time and 0.5g, set WUFC to 5 and ATH to 8:
	`set cfg hubaccel op cfg 01 05 08`
	
	To disable wake up on motion:
	`set cfg hubaccel op cfg 00 FF FF.`

`set_cfg accel_model <value>*`	Selects and initializes accel connectd to system 0 = KX122 (Kionics) 1 = LIS2DS12 (ST microelectronics) *System detects accel on boot-up. This function is a forced assign and initialize for accel device.
`set cfg ppg hr conf level <value>` `get_cfg ppg hr_conf_level`	Set/get the minimum heart-rate confidence level (unsigned 8-bit) to update the heart rate (for platform display only). **Values are in decimal.**
`set cfg ppg hr expiration <value>` `get_cfg ppg hr_expiration`	Set/get the heart-rate value expiration duration (unsigned 8-bit) in seconds (for platform display only). **Values are in decimal.**
`set_cfg event_mode`	Sets Host's behavior for data acquisition from MAX32664C **0**: poll mode where MAX32664C is polled periodically for measurement data **1**: interrupt mode where MAX32664C informs measurement data availability via mfio pin interrupt. **Values are in decimal**
`get_cfg event_mode`	Returns event reporting mode of sensor hub • 0 = poll mode. MFIO pin is input to ME11 • 1= mfio mode,MFIO pin is output from ME11 which is used as interrupt line. **Values are in decimal**

`set_cfg scdsm`	Enables/disables Skin Contact State Machine (SCDSM) on MAX32664C. **0:** disable **1:** enable SCDSM must be enabled only when MFIO mode is used and after algorithm measurement started. **Values are in decimal**
`get cfg scdsm`	Gets the status of SCDSM. 0: disabled 1:enabled
CONFIGURATION: ALGORITHM	
`set cfg wearablesuite spo2cal <valueA> <valueB> <valueC>` `get_cfg wearablesuite spo2cal`	Set/get SpO_2 calibration coefficients A, B, and C as described in the **AN6924 MAX32664C Quick Start Guide.** `<valueA>`, `<valueB>`, and `<valueC>` are space-separated, 32-bit signed integer values equal to 10^5 x measured float calibration coefficient A, B, and C. Get response is [`valueA|valueB|valueC`]. **Only supported by the MAX32664C.**
`set cfg wearablesuite spo2motionperiod <value>` `get cfg wearablesuite spo2motionperiod`	Set/get SpO_2 motion-detection period (unsigned 16-bit integer) in seconds. **Only supported by the MAX32664C.**
`set cfgwearablesuite spo2motionthreshold <value>` `get cfg wearablesuite spo2motionthreshold`	Set/get SpO_2 motion-detection threshold (signed 32-bit integer, equal to 10^5 x actual float threshold value). `<value1>`: MSB in 4-byte signed integer `<value4>`: LSB in 4-byte signed integer **Only supported by the MAX32664C.**
`set cfg wearablesuite spo2afecontrltimeout <value>` `get cfg wearablesuite spo2afecontrltimeout`	Set/get SpO_2 AGC timeout (unsigned 8-bit) in seconds. **Only supported by the MAX32664C.**
`set cfg wearablesuite spo2timeout <value>` `get cfg wearablesuite spo2timeout`	Set/get SpO_2 algorithm timeout (unsigned 8-bit) in seconds. **Only supported by the MAX32664C.**
`set cfg wearablesuite initialhr <value>` `get cfg wearablesuite initialhr`	Set/get initial heart rate value (unsigned 8-bit).
`set cfg wearablesuite personheight <value>` `get cfg wearablesuite personheight`	Set/get height (unsigned 16-bit) in centimeters.
`set cfg wearablesuite personweight <value>` `get cfg wearablesuite personweight`	Set/get weight (unsigned 16-bit) in kilograms.
`set cfg wearablesuite personage <value>` `get cfg wearablesuite personage`	Set/get age (unsigned 8-bit) in years.

`set cfg wearablesuite persongender <value>` `get cfg wearablesuite persongender`	Set/get gender: • 00: Male • 01: Female
`set cfg wearablesuite algomode <value>` `get_cfg wearablesuite algomode`	Set/get algorithm operation mode (can be switched in runtime): • 00: Continuous HRM + Continuous SpO_2 [default] • 01: Continuous HRM + One-Shot SpO_2 • 02: Continuous HRM • 03: Sampled HRM • 04: Sampled HRM + One-Shot SpO_2 • 05: Activity Tracking ONLY • 06: SpO_2 Calibration **Note: Modes 0, 1, 4, and 6 are only supported by the MAX32664C.**
`set cfg wearablesuite aecenable <value>` `get cfg wearablesuite aecenable`	Set/get AEC enable: • 00: Disable • 01: Enable [default]
`set cfg wearablesuite scdenable <value>` `get cfg wearablesuite scdenable`	Set/get SCD enable: • 00: Disable • 01: Enable [default]
`set cfg wearablesuite targetpdperiod <value>` `get cfg wearablesuite targetpdperiod`	Set/get adjusted target photodetector (PD) current period (unsigned 16-bit) in seconds.
`set cfg wearablesuite motionthreshold <value>` `get cfg wearablesuite motionthreshold`	Set/get motion magnitude threshold (unsigned 16-bit) in 0.1g.
`set cfg wearablesuite minpdcurrent <value>` `get cfg wearablesuite minpdcurrent`	Set/get minimum PD (unsigned 16-bit) current in 0.1mA.
`set cfg wearablesuite initpdcurrent <value>` `get cfg wearablesuite initpdcurrent`	Set/get initial PD current (unsigned 16-bit) in 0.1mA.
`set cfg wearablesuite targetpdcurrent <value>` `get cfg wearablesuite targetpdcurrent`	Set/get target PD (unsigned 16-bit) current in 0.1mA. **Works only if Auto Target PD Current Calculation is enabled.**
`set cfg wearablesuite autopdcurrentenable <value>` `get cfg wearablesuite autopdcurrentenable`	Set/get automatic calculation of target PD current: • 00: Disable • 01: Enable [default]
`set cfg wearablesuite mintintoption <value>` `get cfg wearablesuite mintintoption`	Set/get minimum integration time: • 00: 14.8µs • 01: 29.4µs • 02: 58.7µs • 03: 117.3µs

`set cfg wearablesuite minfsmpoption <value>` `get cfg wearablesuite minfsmpoption`	Set/get minimum sampling rate and averaging: • 00: 25sps, avg = 1 • 01: 50sps, avg = 2 • 02: 100sps, avg = 4 • 03: 200sps, avg = 8 • 03: 400sps, avg = 16
`set cfg wearablesuite maxtintoption <value>` `get cfg wearablesuite maxtintoption`	Set/get maximum integration time: • 0: 14.8µs • 1: 29.4µs • 2: 58.7µs • 3: 117.3µs
`set cfg wearablesuite maxfsmpoption <value>` `get cfg wearablesuite maxfsmpoption`	Set/get maximum sampling rate and averaging: • 0: 25sps, avg = 1 • 1: 50sps, avg = 2 • 2: 100sps, avg = 4 • 3: 200sps, avg = 8 • 4: 400sps, avg = 16
`set cfg wearablesuite inittintoption <value>` `get cfg wearablesuite inittintoption`	Set/get initial integration time: • 0: 14.8µs • 1: 29.4µs • 2: 58.7µs • 3: 117.3µs [default]
`set cfg wearablesuite initfsmpoption <value>` `get cfg wearablesuite initfsmpoption`	Set/get initial sampling rate and averaging: • 0: 25sps, avg = 1 • 1: 50sps, avg = 2 • 2: 100sps, avg = 4 [default] • 3: 200sps, avg = 8 • 4: 400sps, avg = 16
`set cfg wearablesuite whrmledpdconfig <value>` `get cfg wearablesuite whrmledpdconfig`	Set/get LED PD configuration for 2 channels of WHRM (MS byte channel 1, and LS byte channel 2). For each channel, 4-bit MSB is LED # and 4-bit LSB is PD #: • LED #: 0–2 for LED1–LED3; 7: LED not used • PD #: 0–1 for PD1–PD2; 3: PD not used **For single-channel case, use the appropriate settings for channel 1, and set LED and PD for channel 2 as unused (0x73).**
`set cfg wearablesuite spo2ledpdconfig <value>` `get cfg wearablesuite spo2ledpdconfig`	Set/get LED PD configuration for SpO_2 (MS byte: IR channel; LS byte: red channel). For each red or IR channel, 4-bit MSB is LED # and 4-bit LSB is PD #: • LED #: 0–2 for LED1–LED3; 7: LED not used • PD #: 0–1 for PD1–PD2; 3: PD not used
`Set_cfg scd_led <value>`	Wavelength selection for Skin contact detection only execution mode. • 0 = Green LED • 1 = IR LED • 2 = Red LED
STREAMING-RELATED COMMANDS	
`read ppg 3`	Enables Skin contact detection algorithm only. Enable accelerometer and algorithm in mode 3. Set report period to 1 and start data streaming at 25Hz rate. Use `get format ppg 3` for streaming format.

`read ppg 4`	Algorithm + sensor streaming at 25Hz: Enable accelerometer, sensor, and algorithm; set report period to 1 and start data streaming at 25Hz rate. Use `get format ppg 4` for streaming format.
`read ppg 5`	Algorithm streaming at 1Hz: Enable accelerometer, sensor, and algorithm; set report period to 25 and start data streaming at 1Hz rate. Only algorithm data is streamed. Use `get format ppg 5` for streaming format.
`read ppg 5*`	*For IBI mode where output samples are reported when IBI is calculated. Follow the same procedure as regular “read ppg 5” with setting report period to 0xFF.
`read ppg 6`	Raw sensor data streaming at 25Hz: Enable accelerometer and sensor; set report period to 1 and start data streaming raw data (algorithm not running) at 25Hz rate.
`read ppg 7`	Raw accelerometer data stream only if there is a sensor hub accelerometer motion event. This command requires motion detection for sensor hub KX122 accelerometer to be enabled (`set cfg hubaccel op cfg 01 <value1> <value2>`).
`Read ppp 9`	Algorithm + sensor streaming at 25Hz: Enable accelerometer, sensor, and algorithm; set report period to 1 and start data streaming at 25Hz rate. Use `get format ppg 9` for streaming format. Custom algorithm fields for Android GUI to be employed in Maxim Wellness Library

`set cfg report period <value>`
`get_cfg report_period`
	Configures the report period per number of samples.
	For example, if the `<value>` is 1 (default), the report is generated every sample (40ms). If the `<value>` is 25, the report is generated once every 25 samples (1s).
	Note: To change the report period for `read ppg 4` or `read ppg 5`, this command should be sent AFTER streaming is started.
	Example to generate report once every 10s:
	`pause 1`
	`read ppg 4`
	`set cfg report period FA`
	`pause 0`

`set cfg scdpowersaving <value1> <value2> <value3>`	Enable and initialize the SCD-based, power-saving mode: `<value1>`: 0/1 for disable/enable `<value2>`: Initial LED off-time period in Probing state in 10xsec. (It will try 3 times before going to Off-Skin state and off-time will double each time.) `<value3>`: Off-skin state timeout in 10xsec Before enabling this feature, use `set cfg accel sh <value>` to choose the sensor hub or host-side accelerometer for motion detection. This command invokes streaming. Streaming will continue if there is a skin contact event. Otherwise, it will perform according to the SCD state machine and resume streaming if there is a motion or skin contact. Refer to the **AN6924 MAX32664C Quick Start Guide** for more information on SCD State and Motion Detection for Power Saving.
`stop sensors`	Stops sensor streaming data if `read ppg 6` is used.

*****As defined in the KX122 data sheet.

3.4 Host Commands for the MAX32664D

The commands in **Table 7** are specific to the MAX32664D sensor hub. Unless otherwise specified, <values> are entered in hex.

Note: Each host command may invoke one or more MAX32664 sensor hub commands to complete the desired action. For the complete list of sensor hub commands, and the description of the fields in the streaming data, refer to the **MAX32664D Quick Start Guide**.

Example:

Run calibration:

```
set_cfg bpt date_time 141019 163730
set_cfg bpt sys_bp 120 122 125
set_cfg bpt dia_bp 80 81 82
set_cfg bpt spo2_coefs 00026f60 ffcb1d12 00abf37b
read bpt 0
…
stop
```

Run Estimation:

```
read bpt 1
…
stop
```

Table 7. Host Commands—MAX32664D

FORMAT	RESPONSE TOKEN/VALUE LIST (HEX)
CONFIGURATION: SENSOR HUB	
`set cfg bpt date time <value1> <value2>`	Sets the date and time. <value1>: • 3-byte date as DDMMYY <value2>: • 3-byte time as HHMMSS **Values are in decimal.** Example: `set cfg bpt date time 181120 163730`
CONFIGURATION: ALGORITHM	
`set cfg bpt sys bp <value1> <value2> <value3>`	Provide three measurement results of systolic blood pressure performed on a subject using a medically approved device for calibration purposes. **Values are in decimal. (mmHg)**
`set cfg bpt dia bp <value1> <value2> <value3>`	Provide three measurement results of diastolic blood pressure performed on a subject using a medically approved device for calibration purposes. **Values are in decimal. (mmHg)**
`set cfg bpt spo2 coefs <valueA> <valueB> <valueC>`	Set SpO_2 calibration coefficients A, B, and C as described in **AN6921 MAX32664D Quick Start Guide.** <valueA>, <valueB>, and <valueC> are space-separated, 32-bit signed integer values equal to 10^5 x measured float calibration coefficient A, B, and C.
`get cfg bpt cal result` `set cfg bpt cal result <value>`	Read/write the result of calibration as an 824-byte calibration vector <value>.

`set_cfg bpt sys_dia <index> <systolic bp> <diastolic bp>` *	Provide measurement results of systolic blood pressure performed on a subject using a medically approved device for each calibration data stage <index>< 0 to 2>
set_cfg bpt cal_index <index> *	Set calibration data stage, <index>< 0 to 2> followingly need to send calibration stage data via `set cfg bpt cal result <value> with 512byte calibration stage data`
STREAMING-RELATED COMMANDS	
`read bpt 0`	Starts streaming for the calibration process on a subject after providing calibrated systolic and diastolic blood pressure of the subject as described above. Stop when the progress reaches 100%. Use `get format bpt 0` for format.
`read bpt 1`	Starts streaming for measuring blood pressure, SpO_2, and heart rate. Stop when the progress reaches 100%. Use `get format bpt 1` for format.

*** Valid for MAX32664D release version 40.5.0 with Host MAX32630 FW version 1.5.009**

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
3	10/20	MAX32663C command updates	17
2	8/20	MAX32663D command updates	20
1	6/20	MAX32663C SCDSM updates	14
0	1/20	Initial release	—