Philips Sensing Platform

SY1901 PSP API Specification

Approved

Rev. 1.2 — 30-Jan-2023

Philips Confidential

Document information

Info	Content
Keywords	Sensing Platform, API, Specification
Abstract	This document describes the application programming interface for PSP library.

Realtek

PHILIPS

Revision history

Rev	Date	Author	Description
1.0	10-Aug-2021	JaWo	First version
1.1	25-Nov-2021	JaWo	Update Ambient channel requirements for different metrics Update SpO2 input metrics requirement.
	24-May-2022 07-Nov-2022	JaWo JaWo	Update Legal Information Update legal disclaimer, update date format
1.2	30-Jan-2023	JaWo	Add motion streams, multi-stream handling

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Table of Contents

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

1 Introduction

1.1 Purpose & scope

This specification describes the Philips Sensing Platform software library (referred to hereafter as 'PSPL'). The main functionality of PSP is to extract health related metrics of a person. PSP comes in different deployment formats, suited for specific computational platforms.

The purpose of this specification is to give a high-level overview of the PSP functionality, mechanisms and metrics. More precisely, to describe the requirements and protocol that an application (typically, software running on a microcontroller in a wearable device) must follow to use PSP.

The detailed software API aspects (such as function signatures and datatypes) are beyond the scope of this specification; they are provided in documentation accompanying the various deployment formats (refer to section 5).

1.1.1 PSP variants

The contents of this specification apply to all products, being variants of the PSP library, except where explicitly stated. In particular, the set of supported extracted metrics differs per variant.

1.1.2 PSP versions

This specification is generally applicable to all PSP versions 6.1.* or above, except where explicitly stated.

Realtek

1.2 Important safety notice

The accuracy of the metrics, output by PSP, may vary due to a variety of causes. The PSP application shall at all time and by all necessary means warrant the user's safety also in occurrence of metric inaccuracies.

In particular, using the sensing technology that provides the input data to PSP outside of its specified operational conditions increases the probability of inaccuracies. While this sensing technology is out of scope for PSP, it is still strongly advised to limit the application's intended use within the sensing technology's operational conditions.

1.3 Definitions, acronyms & abbreviations

Abbreviation	Description
ACC	3-axis acceleration sensor
AEE	Active Energy Expenditure
API	Application Programming Interface
BMR	Basal Metabolic Rate
BPM	Beats Per Minute
HR	Heart Rate

© Koninklijke Philips N.V. 2022. All rights reserved.

HW	Hardware
LPAEE	Low Power Active Energy Expenditure
LPHR	Low Power Heart Rate
NTP	Network Time Protocol
PPG	Photo plethysmography measures the volumetric changes of the arteries caused by the pumping of the heart. From this measurement average heart rate and heart rate variability can be extracted.
PSP	Philips Sensing Platform
PSPL	Philips Sensing Platform Library
PSP-S	Philips Sensing Platform – Sleep Sensing by PPG, a metric extraction co-processing library running on mobile, used together with PSP.
TEE	Total Energy Expenditure means the AEE + Basal Metabolic Rate

1.4 References

Abbrev.	Title	Author(s)	Version
SY1902	PSP Metrics Specification	Philips	1.2
SY1903	PSP-S API specification	Philips	1.0

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

2 Application context

2.1 Applying PSP stand-alone

The main function of PSP is to extract health related output metrics of a person from a combination of input metrics, in particular sensor metrics data (such as acceleration) and external personal metrics (e.g. the person's height). PSP can extract a variety of output metrics (see Figure 1). These metrics are thus available on the wearable device running the PSP library, and can e.g. be used for the user interface on the wearable device.

Figure 1 – PSP in a stand-alone application

Section 4 specifies which input metrics are required by PSP for each of the extracted metrics.

2.2 Applying PSP in combination with PSP-S

For certain metrics whose extraction is resource (computing or memory) intensive, PSP requires additional extraction by the companion library, e.g. PSP-S library for Sleep Stages(Figure 2). While PSP normally runs on a processor of a wearable device, PSP-S library may run in another, resource richer environment, such as on a mobile device.

Realtek

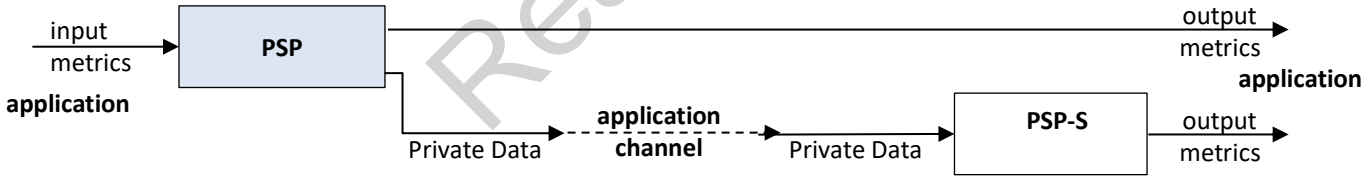

Figure 2 – PSP in an application including PSP-S library

PSP provides the Private Data metric at a rate of 1 packet per minute. It is the responsibility of the application (typically comprising the wearable host processor, connectivity such as Bluetooth, and a mobile app incorporating PSP-S) to provide this data to PSP-S integrally and in the **exact same order** as it was provided by PSP. PSP-S will then extract the metrics.

PSP-S provides time information along with its output metrics. For this, PSP internally keeps track of elapsing Time, but the absolute time has to be set (at least once) by the PSP application. See section 4.4 for more details.

Precise operation and application of PSP-S is out of scope here; for more information, refer to [SY1903].

© Koninklijke Philips N.V. 2022. All rights reserved.

3 Functions

PSP supports the following functions.

Function	Semantics
lifecycle	
<instantiate>	Create an PSP instance. How instances are created depends on the actual deployment format; refer to section 5 for details per deployment format
<terminate>	Terminate the specified PSP instance. How instances are terminated depends on the actual deployment format; refer to section 5 for details per deployment format
extraction control	
EnableMetrics	Enable specified output metrics to be provided by PSP
DisableMetrics	Disable specified output metrics to not be provided anymore by PSP
extraction	
ListRequiredMetrics	Get a list of all input metrics that are required for the enabled output metrics.
ListUpdatedMetrics	Get a list of all output metrics that were updated as a result of Process
SetMetric	Set the value of the specified input metric
GetMetric	Get the value of the specified output metric
Process	Perform the actual extraction, based on the enabled output metrics and on the input metrics having been set since the last Process. Must be called after setting exactly one value for all required input metrics
configuration	
GetVersion	Get the version of PSP, uniquely identified by the combination of returned elements • main version • major revision • minor revision • internal configuration (not interpretable by the application, intended only for support purposes)

Table 1 – PSP functions

Realtek

3.1 Life cycle; single-user versus multi-user

Before starting the metrics extraction functionality (by enabling any output metrics), an instance of PSP must be created. Likewise, the PSP instance should be terminated after definitively ending the extraction operation.

The PSP instance maintains the state of the PSP extraction, which actually is the result of all input metrics data that PSP has processed so far. Naturally, these input metric data are person-specific. Therefore, an instance of PSP is specific for (i.e. associated with) *a single person*.

In a typical single-user device such as an activity tracker, sportswatch or smartwatch, just one PSP instance needs to be used. This also means that the device is *person specific* and may not be used alternatingly (i.e. shared over time) by multiple persons.

In case a device does need to be alternatingly used by multiple persons, this can be realized by multiple PSP instances (one per device user) which can be maintained independently in parallel.

3.1.1 PSP instance: unique source ID

When creating an PSP instance, PSP accepts a source ID number. In applications where PSP is applied in combination with PSP-S (see section 2.2), this source ID is used to associate PSP and PSP-S instances, in which case the source ID number should be non-zero and unique for each PSP instance (e.g. per copy of the wearable device in which PSP is applied). Thus,

© Koninklijke Philips N.V. 2022. All rights reserved.

PSP-S can verify that the Private Data stream it receives as input always comes from the same PSP instance, giving additional robustness to data channel implementation errors e.g. in multi-user PSP-S applications.

In stand-alone PSP applications (see section 2.1), or when the application does not desire the additional PSP - PSP-S association robustness, no unique source ID needs to be provided; in these cases, the source ID number should be set to 0.

3.2 Extraction control

As described in section 2, PSP provides output metrics, which are either extracted from its input metrics, or are propagated input metrics. For more detailed information, refer to section 4.

The application must specify via EnableMetrics() which output metrics it wants PSP to provide. Likewise, the application can also disable output metrics again. The enabling and disabling is fully dynamic, i.e. the application has full freedom to enable or disable metrics together or one by one over time in line with application needs, e.g. in response to end user function activation/deactivation.

3.3 Extraction

While PSP is extracting, ListRequiredMetrics() indicates which input metrics are required in relation to the enabled output metrics. It is the responsibility of the application to constantly provide any required input metrics and to call Process() so that the extraction of output metrics can take place (see also the protocol in section 3.4).

After each Process(), PSP indicates via ListUpdatedMetrics() which metrics have been updated, so that the application may selectively get these via GetMetric(). It is not mandatory to get updated metrics; however, the metric value may be updated (i.e. overwritten) in a subsequent Process().

The application is not bound to getting only updated metrics; it may get any metric at any time, yielding the last updated value. As long as the value is not updated, multiple gets will result in the same metric value being repeatedly provided[1].

The list of required input metrics may actually vary dynamically over time: PSP may e.g. only need PPG data once in a while to extract a given metric. By this, PSP allows the application to minimize resource usage. In particular, it enables the application to switch off sensor hardware for the period that a metric sensed by that hardware is not required, so as to minimize power usage.

3.4 Calling protocol

PSP functions may be called according to the protocol state machine of Figure 3. Note the states indicated are conceptual only, PSP library does not report its state at the API.

[1] The application can also detect this from the index field of the metric being unchanged; refer to section 4

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

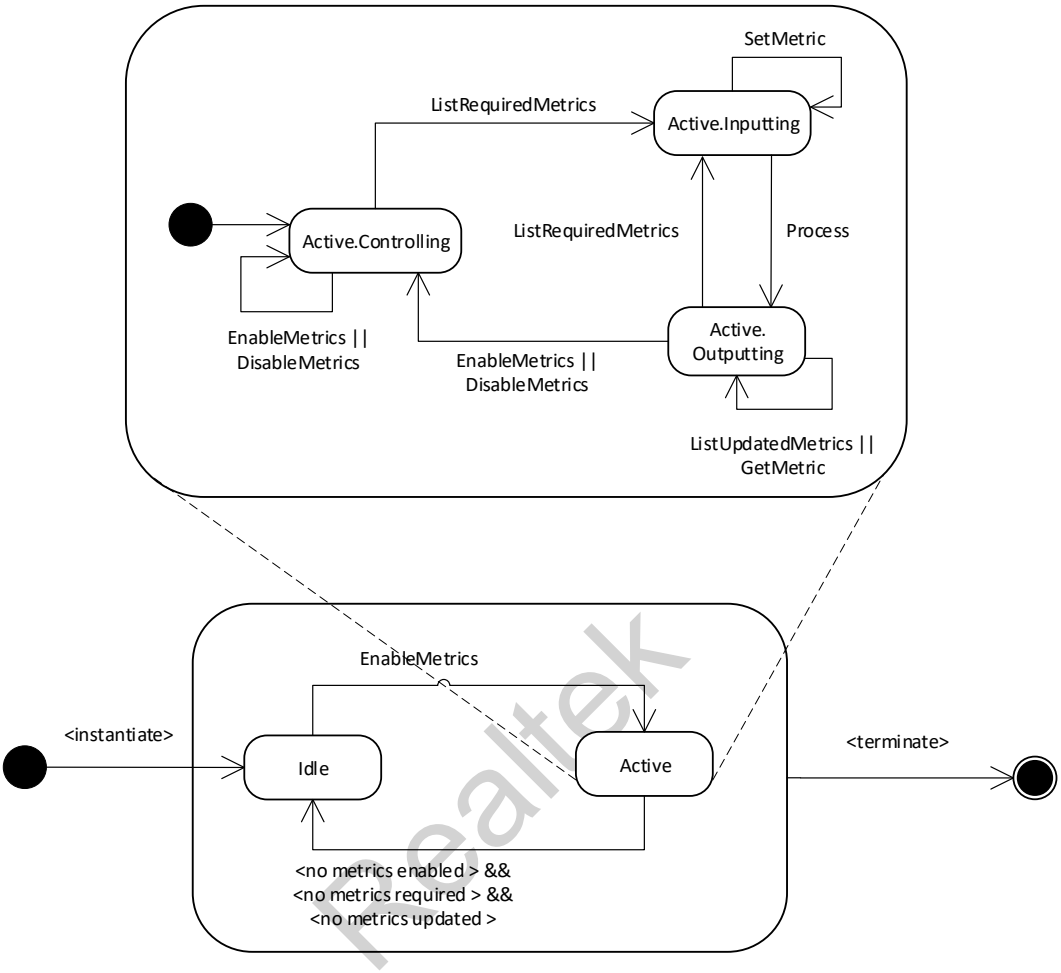

Figure 3 – Protocol state machine

After instantiating PSP, it enters state **Idle** where no output metrics are enabled, hence also no metrics are extracted. In this state, PSP requires no computational resources.

When enabling any one or more output metrics in state Idle, the **Active** state is entered where PSP produces output metrics (i.e. extracts output metrics and propagates input metrics), in accordance to which output metrics are enabled. This enabling may be adapted over time in substate **Active.Controlling**. In substate **Active.Inputting**, all input metrics that PSP requires are set (1 value for each). Subsequent to a Process, PSP provides output metrics in substate **Active.Outputting**.

Whenever *all* output metrics are disabled in state Active, PSP *still* continues processing in order to empty its internal pipeline and to complete and provide any pending output metrics in that pipeline. Meanwhile, it may also still require input metrics. Only after no more input metrics are required and there are no more updated output metrics, state Idle is entered again.

The functions specified in section 3 may be called only in specific (sub)states, as listed in Table 2

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Function	Idle	Active. Controlling	Active. Inputting	Active. Outputting
<instantiate>				
EnableMetrics	x	x		x
DisableMetrics		x		x
ListRequiredMetrics		x		x
SetMetric			x	
Process			x	
ListUpdatedMetrics				x
GetMetric				x
GetVersion	x	x	x	x
<terminate>	x	x	x	x

Table 2 - Functions and protocol states: 'x' denotes that function may be called

The example pseudo-code in Figure 4 adheres to the protocol. The functions are called in a simple, repetitive fashion.

```
"instantiate PSP"          // deployment format specific
…                          // state = Idle

repeat
      if ( " any metrics need to be enabled " )
          EnableMetrics(…)       // enable metrics
      end if
      if ( " any metrics need to be disabled " )
          DisableMetrics(…)    // disable metrics
      end if
      noEnabledMetrics = ( " no metrics enabled "? )
      ListRequiredMetrics ( &requiredMetricsList )
      for ( each metricID in requiredMetricsList ) // PPG, acceleration, …
          "gather metricData"  // exactly 1 value per required metric
          SetMetric(metricID, metricData )
      end for
      Process ( )          // let PSP process anything it can
      ListUpdatedMetrics ( &updatedMetricsList )
      for ( each metricID in updatedMetricsList )
          GetMetric( metricID, &metricData )
          "interpret metricData"
      end for
until (   noEnabledMetrics          &&
     ( requiredMetricsList empty ) &&
     ( updatedMetricsList  empty ) )

…                            // state = Idle
"terminate PSP"          // deployment format specific
```

Figure 4 – Example API calling

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Note that in substate Active.Inputting, exactly 1 value per required input metric must be set before calling Process(). This implies also that the time periods between subsequent Process calls is determined by the smallest update interval of all required input metrics, as specified in [SY1902]. E.g. if Acceleration or PPG are required and these have a 1s update interval, consequently Process() will be called once per second.

3.5 Concurrency limitations

PSP is designed to be used synchronously within an PSP instance: the application **may not make** concurrent function calls for **one PSP instance**. In other words, all functions for that instance must be called one-by-one sequentially, in the order of the protocol specified in section 3.4.

The application **is allowed** to make concurrent PSP calls for **different PSP instances** (e.g. in a multi-user application as described in section 3.1).

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

4 Metrics

As described before, PSP requires and accepts certain metrics as input and from these, extracts output metrics. All metrics comprise:

- an ID, uniquely identifying the type of metric;
- an Index (range 0..255), which is incremented at every metric value update, and enabling the application to discern consecutive values (e.g. for polling purposes);
- a quality indicator (range 0-4), which indicates how reliable the metric value is: from 0 = fully unreliable to 4 = maximally reliable;
- the actual metric value itself.

Detailed syntax and semantics of all metrics are given in [SY1902].

4.1 Input metrics

PSP accepts input metrics as listed in Table 3. Which of these input metrics are actually required for PSP to offer its functionality depends on which output metrics are supported, i.e. on the PSP variant. Detailed syntax and semantics are given in [SY1902]. For input metrics, the index is ignored (any value may be set). Unless stated otherwise (see section 4.2), the quality of input metrics shall be set to 4. After PSP instantiation, all input metrics are by default unspecified.

Realtek

Metric	Notes
Acceleration	Depending on enabled metrics, if Fall Occurrence metric is enabled, format 0x70 must be used otherwise 0x6E is used.
Activity Type	Use also as input to set activity type for heart rate calculation per activity. Note Activity Type is also an output metric.
Age	See also section 4.5
Angular Rotational Velocity	Gyroscope data
Height	
PPG-Ambient	Only sample format 60 supported.
PPG-Green	Only sample format 60 supported.
PPG-Infrared	Only sample format 60 supported.
PPG-Red	Only sample format 60 supported.
PPG-Motion	Only sample format 60 supported.
Pressure	Barometer data
Profile	See also section 4.5
Skin Conductance	
Sleep Preference	
Time	See also section 4.4
Weight	

Table 3 – PSP input metrics

4.2 Output metrics

PSP provides output metrics as listed in Table 4. Detailed syntax and semantics are given in [SY1902]. These output metrics comprise two categories:

© Koninklijke Philips N.V. 2022. All rights reserved.

- **extracted metrics**; which subset of these metrics are supported depends on the actual PSP variant. The quality attribute of extracted metrics is determined by PSP: it indicates the reliability of the metrics value as estimated by PSP itself. Extracted metrics require specific input metrics to be set, as indicated in Table 4. Not all input metrics may be required all of the time; function ListRequiredMetrics() indicates dynamically which of the input metrics are required (refer to section 3.3).
- **propagated input metrics**; setting of an input metric results in a corresponding output metric update for that same metric, but only at the moment at which PSP takes over the input metric value, i.e. reflecting when the value become applicable for PSP. This input-to-output propagation functionality is intended mainly for testing, performance analysis and problem solving purposes, and will in many PSP applications not be used. To get the propagated input metrics at output, first enable the metrics and then after PSP library processed the input, the propagated metric ID will be updated in the ListUpdatedMetrics. User can then get the propagated metrics at output.

Which of this list of output metrics is actually supported or not depends on the PSP variant.

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Metric	Notes	Mandatory Input metrics to be set
Propagated input metrics		
Acceleration	See Table 3	n/a
Activity Type[2]		
Age		
Angular Rotational Velocity		
Height[3]		
PPG-Ambient		
PPG-Green		
PPG-Infrared		
PPG-Red		
PPG-Motion		
Pressure		
Profile[3]		
Skin Conductance		
Sleep Preference		
Weight		
Time	See section 2.2 and 4.4	Time
Extracted metrics		
Active Energy Expenditure[4]	See also section 4.14	Acceleration, PPG_G_INPUT[5], Age, Profile.sex, Height, Weight
Activity Count	See also section 4.17	Acceleration
Activity Type		Acceleration[6]
Cadence	See also section 4.11	Acceleration
Cognitive Zone	See also section 4.21	Skin Conductance, Profile.sex, Time (including Time Zone and DST)
Compressed Acceleration	Only sample format 6E supported	Acceleration
Compressed PPG[7]	Only sample format 60 supported	PPG (including PPG_Ambient)
Fall Occurrence	See also section 4.22	Acceleration, Angular Rotational Velocity, Pressure
Fitness Index	See also section 4.8	Acceleration, PPG_G_INPUT, Age, Profile.sex, Height, Weight
Heart Beat Time Stamp	See also section 4.18	Acceleration, PPG_Green
Heart Rate	See also section 4.6	Acceleration, PPG_G_INPUT
Heart Rhythm Type		Acceleration, PPG_Green
Low Power Active Energy Expenditure[8]	See also section 4.16	Time, Acceleration, PPG_G_INPUT, Age, Profile.sex, Height, Weight
Low Power Heart Rate	See also section 4.15	Time, Acceleration, PPG_G_INPUT
Stress Level Heart Rate	See also section 4.19	Acceleration, PPG_G_INPUT
Private Data	See also sections 2.2 and 4.4	Time, Acceleration
Respiration Rate		Acceleration, PPG_Green
Resting Heart Rate	See also section 4.7	Time, Acceleration, PPG_G_INPUT
Skin Proximity	See also section 4.10	Acceleration, PPG_G_INPUT or Skin Conductance
Sleep Stages	Private Data must be enabled in combination with Sleep Stages. This makes PSP extract Private Data for PSP-S, and from that makes PSP-S extract Sleep Stages (see also section 2.2 and 4.13)	PPG_G_INPUT, Sleep Preference
Stress Level Skin Conductance	See also section 4.20	Skin Conductance, Profile.sex, Time (including Time Zone and DST)
Speed	See also section 4.12	Acceleration, Profile.sex, Height

[2] Activity Type is an input as well as output metric. See section 4.6

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Metric	Notes	Mandatory Input metrics to be set
SpO2	See also section 4.9	PPG_Infrared, PPG_Red, PPG_G_INPUT, Acceleration
VO2Max	See also section 4.8	Acceleration, PPG_G_INPUT, Age, Profile.sex, Height, Weight

Table 4 – PSP output metrics

Maximum 2x PPG-Green, 2x PPG_Ambient, 2x PPG-Red and 2x PPG-Infrared streams can be fed to the library for relevant metrics extraction when needed. It is not recommended to combine to one stream outside the PSP library.

4.3 PPG and Acceleration sample rate synchronization

Whenever PSP requires PPG and Acceleration inputs simultaneously, then as a consequence of the calling protocol (section 3.4) acceleration and PPG (feed acceleration data before PPG data, as in the order of list returned by function ListRequiredMetrics) will be input at a constant number of samples per second, and their sample rates will also have an exact ratio. For example:

- 32 PPG samples are requested once per second, or at a total rate of 32 x 1 = 32 Hz;
- 32 Acceleration samples are requested once per second, or at a total rate of 32 x 1 = 32 Hz;
- then the ratio of the Acceleration and PPG sample rates is also exactly 32/32 = 1.

So the application has to provide both PPG and Acceleration at constant, locked rates.

If the required rates and/or their locking is not guaranteed (e.g. in a wearable device where the hardware sample frequencies of PPG and/or acceleration sensors are different from the requirement, and/or they are not hard-locked in hardware), then the application must adapt the PPG and Acceleration sampling rates (e.g. resample the data) such that they get the correct sample rates and become frequency locked.

Realtek

4.4 Handling of Time for PSP-S extraction

As described in section 2.2, the PSP-S software library extracts additional metrics from Private Data generated by PSP. Typically, this PSP-S extraction occurs at a later, indeterminate point in time than when PSP generates the Private Data. Therefore, PSP-S accompanies its output metrics with time information, so that the application can know at which point in time those metrics values were valid.

Initially, PSP nor PSP-S have a notion of the absolute time. Therefore, this information has to come from the PSP application via a SetMetric() of Time. This required Time will also be indicated by ListRequiredMetrics(). PSP will take over the set time instantaneously. From then on, PSP keeps track of elapsing time via counting of Acceleration and/or PPG samples, and subsequent GetMetric() calls for Time will return the absolute time in seconds.

PSP maintains Time as long as any metrics other than Time itself are enabled. When for some period all metrics are disabled, then during this period also the Time holds, i.e. any GetMetric() calls for Time yield the same 'frozen' time. To

[3] It is advised to set Profile.sex and Height, accompanied by quality 4. This will under certain circumstances lead to highest accuracy for Speed. But it is allowed to set Profile.sex and Height to unspecified values, by setting their quality to 0; Speed will anyway be extracted but may have slightly lower accuracy

[4] Active Energy Expenditure quality will be low if Age < 10 years.

[5] PPG_G_INPUT means PPG_Ambient and PPG_Green.

[6] Body position index must equal either Left Wrist or Right Wrist

[7] Compressed PPG reports PPG_Green+PPG_Ambient.

[8] Low Power Active Energy Expenditure quality will be low if Age < 10 years.

© Koninklijke Philips N.V. 2022. All rights reserved.

change the time to the correct value again, the application is free to set the absolute time at any one or more moments, e.g. after (re)enabling of any output metrics.

Clearly, the accuracy of Time has a direct relation to the accuracy of the PPG and Acceleration sample rates: if the rates are not exact (slightly high or low), the PSP Time may drift away from true time. This will result in inaccuracy of the time information in the metrics output by PSP-S. **The application may compensate for this drift by setting the PSP Time regularly to the correct value, e.g. updating the PSP Time with the time obtained from an NTP server.**

4.5 Age versus date of birth

For certain output metrics, input metric Age is required to indicate the current age of the monitored person.

Additionally, input metric Profile comprises the date (i.e. year, month and day) of birth. However, at present, this information is not used by PSP. The application may decide to leave year, month and date unspecified (set to 0). For future compatibility, it is advised to set the actual date of birth.

4.6 Heart Rate Extraction

Input metrics required for the heart rate extraction are the acceleration, PPG-Green and PPG-Ambient metrics. Note that the ambient signal is a separate metric, not part of the PPG-Green metric. If the ambient signal is not used, set all the ambient signal to zero before feeding to the library.

The Heart Rate metric should be enabled in order to get the heart rate calculation. Then the library will request the system to provide the required metrics in the ListRequiredMetrics(). **System only needs to provide the required metrics when requested and does not need to specifically control the PPG/ACC on off time**.

The Activity Type metric can also be used (besides as an output metric) as an input to the heart rate calculation. When the Activity Type is enabled and set (e.g. running) before enabling the Heart Rate metric, the heart rate can be optimized for the activity specified.

When Activity Type metric is enabled without specifying any activity type, the heart rate is calculated and optimized automatically based on internal estimated activity type. When the Activity Type metric is not enabled or enabled but Activity Type set to 0 (unspecified), the heart rate calculation will NOT specifically be optimized for any activity type.

Be precautious that when the activity set does not match the actual activity or the body position is not set correctly, the heart rate calculated will suffer.

4.7 Resting Heart Rate extraction

The Resting Heart Rate metric is extracted by collecting information during periods of relative physical inactivity. The information is aggregated and retained for up to seven days.

After first enabling Resting Heart Rate, its initial value is available after the metric's update interval. Typically, the quality of initial value is still low, due to a limited amount of relevant information. The quality of subsequent values will increase when cumulatively more relevant information is aggregated, depending on the monitored person's lifestyle pattern.

If Resting Heart Rate is disabled, no new information is collected anymore, and the aggregated information is gradually discarded as time progresses. When re-enabling Resting Heart Rate within less than seven days, the remaining retained information still contributes to the metric extraction but due to mentioned data discarding, the quality may be lowered.

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

After seven or more days of disabling, all retained information is lost, and upon re-enabling, complete information will have to be collected anew.

4.8 VO2Max and Fitness Index extraction

The VO2Max and Fitness Index metrics are extracted by collecting information during periods of physical activity, in particular related to walking or running. The information is cumulatively aggregated as long as VO2Max or Fitness Index are enabled.

After enabling the VO2Max or Fitness Index metrics, their initial values are available after their respective update intervals. Typically, the quality of initial value is still low, due to a limited amount of relevant information. The quality of subsequent values will increase when cumulatively more relevant information is aggregated, depending on the monitored person's lifestyle pattern.

When VO2Max or Fitness Index metrics are disabled, the collected information is discarded. Upon re-enabling, complete information will have to be collected anew.

4.9 SpO2 Extraction

Input metrics required for the SpO2 extraction are the acceleration, PPG-Green, PPG-Ambient, PPG-Red and PPG-Infrared metrics.

Realtek

For SpO2 calculation, the device is expected to be wearing reasonably tight on the wrist of the user. The measurement should be taken with less or no motion on the wrist/finger while the palm of the user is facing downward.

To obtain an accurate SpO2 measurement, calibration coefficients should be passed to the PSP library during initialization phase before the library is executed. Typically the ratio (R) between Red and Infrared is used.
SpO2 estimation is using linear or second order polynomial to come to actual percentage, the formula is described as below:

$$SpO2 = a * R^2 + b * R + c \qquad (\text{linear} = \text{factor a equals 0})$$

The values a/b/c are called the SpO2 calibration coefficients and are determined with a calibration process (out of scope of this document). These values have to be passed to the PSP library, this is done via the PSP_INST_PARAMS struct in psp.h with element calCoefSpO2[3] defining an array of coefficients in the order of a, b, c, i.e. calCoefSpO2[0] = a, calCoefSpO2[1]=b, calCoefSpO2[2]=c.

To allow for better resolution of the a/b/c coefficients produced by calibration process, they are scaled with a factor of 100 before feeding to PSP library: e.g. 110 becomes 11000.

SpO2 takes approximately 20 - 30s to output first SpO2 value after feeding valid PPG signals.

4.10 Skin Proximity & wearing detection

The main objective of metric Skin Proximity is to indicate whether or not the input signals (PPG, acceleration and/or skin conductance) can be considered to stem from contact with the user's skin, which in wearable device applications translates to whether or not the device is worn by the user.

Skin Proximity analyses Acceleration, PPG and/or skin conductance signals. While Skin Proximity is enabled, PSP can dynamically require or not require the input metric, if it is not already required because of other enabled metrics.

© Koninklijke Philips N.V. 2022. All rights reserved.

PSP's output metrics are extracted with algorithms that maintain some state (i.e. history). This memory could lead to delays in re-establishing accurate metrics after a period of non-wearing, Therefore, it is advised to disable PSP's output metrics in non-wearing periods, which will reset the extraction functionality to an initial state.

The application may use the Skin Proximity metric to disable other output metrics during such non-worn periods (the Skin Proximity metric itself should stay enabled all the time, also during non-wearing). Note that PSP itself does not enable or disable metrics in relation to Skin Proximity; this is the responsibility of the application.

Depending on which input sensor signals are available and which output metrics are enabled, Skin Proximity provides different characteristics. If Stress Level Skin Conductance metric and/or Cognitive Zone is enabled, then Skin Proximity will be based on input metric Skin Conductance, otherwise based on ACC/PPG.

Skin Proximity characteristics using PPG and ACC signals
Skin Proximity transitions from off-skin to on-skin are typically signaled with a latency of around 30-35 seconds. This lengthy period serves to reduce false on-skin values.

Skin Proximity transitions from on-skin to off-skin are typically signaled with a latency from as little as 1-2 seconds, up to around 15-20 seconds, depending on various acceleration and PPG signal conditions.

In general, Skin Proximity provides reliable values in case the wearable device applying PSP is either worn by the user (where the user may be moving or stationary), or stored in a stable, stationary position (e.g. on a bedside table or charging cradle). But it is important to notice that Skin Proximity has limitations under these known conditions:

- false on-skin may occur in case the device is in motion and not worn but still facing a reflective surface, e.g. in a handbag;
- false off-skin may occur e.g. if the user is asleep in a posture hampering the detection of blood flow (e.g. laying on the wearing wrist).

Realtek

Skin Proximity characteristics using skin conductance
Skin Proximity transitions from off-skin to on-skin or on-skin to off-skin typically take 10 seconds. In general, Skin Proximity provides reliable values in case the wearable device applying PSP as long as the electrodes are in good contact with skin for on skin detection.

How Skin Proximity can be used strongly depends on the wearable device's intended use cases. If the Skin Proximity response times are too long, or if aforementioned non-stationary non-worn situations or sleeping conditions can occur, then it is advisable to equip the device with other means (such as other sensors) to detect wearing/non-wearing, or to ask the user to manually switch the device's monitoring function on or off via a user interface (e.g. button). Also, in certain use cases, combinations are conceivable, e.g. let the user switch on the monitoring function with a button, and use Skin Proximity off-skin value to switch the function off again.

In certain applications, it might also be possible to infer device wearing or non-wearing from other (combinations of) output metrics of PSP, such as Activity Count (to detect certain amounts of motion or non-motion). The detection algorithm needs to be designed, implemented and validated by the applicant. How feasible and useful this approach is, depends on the application use case specifics, and therefore differs per application.

4.11 Cadence

Cadence actually is the rate of the predominant motion during an activity. It typically corresponds to the number of strides per minute while walking/running, and to the number of crank revolutions per minute while cycling.

© Koninklijke Philips N.V. 2022. All rights reserved.

4.12 Speed

This metric is based on the cadence and characteristics of the subject (like height and gender) and the speed is approximated. Typically this metric has 10-second latency. When cycling the speed is not calculated.

4.13 Sleep extraction

To generate the sleep output, the system has to consist of two parts, one is on the wearables and another one is on a mobile device.

On the wearable side, both Private Data (metric ID 0x2F) and Sleep Stages (metric ID 0x30) must be enabled/disabled together. If enabled, then the library on wearable takes raw PPG (metric ID 0x7E and 0x7F) & ACC (metric ID 0x2B) as input. Intermediate data packets (Private Data, metric ID 0x2F) are calculated based on the sensor inputs from the PPG and the accelerometer. These packets with update interval of 1-minute need to be sent, or stored first on the wearable and then sent to the mobile device for the Sleep Stages calculation.

Only Private Data is reported in the output of the wearable, not for Sleep Stages.

The Start/Stop of Sleep Session is detected automatically. Sleep Preference (metric ID 0x09, optional) can also be used as a trigger for the start/stop. Note that Sleep Preference is not used for enabling/disabling sleep function. Please use Sleep Preference with following guideline.

- If one is enforced to set Sleep Preference (after enabling Sleep Stages/Private Data) then “unspecified” can be used.
- Only set once Sleep Preference = yes when going to sleep
- Only set once Sleep Preference = no when you are awake

For details, please refer to [SY1903].

Max 300 bytes per minute sleep data will be sent to the mobile device for the Sleep Stages calculation (or the Apps level in the watch running Android).

So in case of 8hr sleep: 8 * 60 * 300 = 144000 bytes (Max). On average we see approximately 100K per sleep session.

On the mobile side, a Sleep Library is running in the Apps, in which Sleep Stages is enabled. The library on mobile side takes Private Data as input and reports the sleep stages in the Sleep Session.

4.14 Active Energy Expenditure

The Active Energy Expenditure (AEE) metric estimates the part of a user’s energy expenditure related to physical activity (so NOT the metabolic part). When talking about Total Energy Expenditure (TEE) that means the AEE + Basal Metabolic Rate (BMR; determined from a person’s characteristics):

TEE = AEE + BMR

Basal metabolic rate is the number of calories one’s body needs to accomplish its most basic (basal) life-sustaining functions (so when in total rest/sleeping). PSP library combines measured activity levels, activity type and heart rate information into an accurate estimation of activity related energy expenditure.

The unit of AEE is Kcal/hour.

© Koninklijke Philips N.V. 2022. All rights reserved.

The update interval for AEE is every second, so it is necessary to read the metrics every seconds and then divide them by 3600 to get a per-second measurement. After that accumulate them over the total period to get total amount of AEE burnt.

4.15 Low Power Heart Rate

Low Power Heart Rate (LPHR) is a metric introduced to save power when a lower update rate suffices.

In case less power consumption is desired, the LPHR heart rate metric is selected. The LEDs are the most consuming part of the system and therefore the LEDs are only turned on for a short interval with a maximum of 15 seconds, at a cycle of 60 seconds. An estimated heart rate is provided every cycle. For this metric, the heart rate estimation relies more on prediction than for the HR metric, since the measurement update is not always available. The accelerometer is running continuously though, which makes the prediction update possible. The library calculates one single Heart Rate value per minute. This metric is intended for the Daily Life use case not for sports use case.

4.16 Low Power Active Energy Expenditure

For this metric Low Power Heart Rate (LPHR) is used instead of the Heart Rate to save power, the value is, therefore, less accurate than AEE.

To provide a low power solution, Low Power Active Energy Expenditure was introduced abbreviated as LPAEE.
In case AEE was requested as well, the AEE value is copied as LPAEE value at the time LPHR becomes available. This is similar to LPHR, in which LPHR is copied from HR when LPHR is active as well.

Only when LPAEE is requested without AEE or HR, LPAEE is calculated only once per minute only at the time when LPHR becomes available.

4.17 Activity Count

Activity Counts are calculated separately for each consecutive 1-second block of acceleration signals. For each of the axes d = {x, y, z}, the deviations between the N samples and their mean are summed (inner summation). The contributions from the axes are also summed (outer summation) and scaled (factor C). The scaling factor is rather arbitrary.

4.18 Heart Beat Time Stamp

Heart Beat Time Stamps are reported in sequence. The sequence of heart beats may also be temporarily interrupted when no beats could be detected e.g. due to motion influences.

This is signaled by the heart beat type (Type = 0 = normal beat in sequence, Type = -1 = Last beat in sequence). The time between two consecutive heart beats in a sequence can be determined by subtracting their times (in milliseconds).

So in following example, you can subtract all the times except 32270 – 26380 (see **bold** marking below).

quality	time	type
4	24051	0
4	25146	0
4	**26380**	**-1**
4	**32270**	**0**
4	33105	0

© Koninklijke Philips N.V. 2022. All rights reserved.

4.19 Stress Level Heart Rate

This metric measures the mental stress based on the heart rate. The output is ranging from 0 to 1000 (arbitrary unit). It takes approximately 52 – 60s to produce the first Stress Level value after first valid PPG sample captured. Another concept is the 'lock-in time' which is the delay from initial system start-up (with valid inputs) to first functionally valid stress level output. Now it is roughly less than 5 minutes. Currently the system reports high stress level during this lock-in time.

4.20 Stress Level Skin Conductance

This metric measures the stress based on the skin conductance. Besides the Skin Conductance metric, TIME (including Time zone and DST information) and Profile.sex are required to feed to the library in order to produce stress level. The output range is ranging from 0 to 1000 (arbitrary unit).

4.21 Cognitive Zone

This metric is a collection of output including the Cognitive Zone, Predictive Cognitive Zone, Predictive Cognitive Zone Transition Time, Cognitive Zone in 1 hour and Cortisol Contribution. Note that the unit for Cortisol Contribution is arbitrary.

In order to get the Cognitive Zone metric output, Skin Conductance metric, TIME (including Time zone and DST information) and Profile.sex are required to feed to the library. The first valid output is typically 30 - 45 minutes after wearing the device.

Realtek

4.22 Fall Occurrence

Fall Occurrence provides the estimation of falls. It requires accelerometer (ACC), gyroscope and barometer signals as input. Therefore Accelerometer, Angular Rotational Velocity and Pressure metrics must be feeding to the library per request. If the hardware does not have all three sensor signals available, specific library has to be built on request.

When initializing the library, Fall Occurrence false alarm rate threshold (predicted false occurrence, once a day/week/month) has to be passed to the library via initialization parameters, this is done via the PSP_INST_PARAMS struct in psp.h.

For the Accelerometer metric, only format 0x70 is supported for Fall Occurrence. For the moment, this Fall Occurrence is exclusive with other extracted metrics in Table 4 – PSP output metrics.

© Koninklijke Philips N.V. 2022. All rights reserved.

5 Deployment formats

By design, PSP targets a variety of execution platforms, now and in the future. The actual ways to call PSP are platform and programming language specific. The available deployment formats are listed below.

5.1 C library image for ARM Cortex M3/M4[F]

This deployment format provides the following deployment format specific functions.

Function	Semantics
lifecycle control	
GetDefaultParams	Get the default instance parameters before creating an instance, comprising • minimum size of the instance memory buffer that the application must provide to Initialise
Initialise	Initialise and claim the provided instance memory, so as to create a new instance
Terminate	Terminate the specified PSP instance and release the instance buffer memory.

Table 5 – C library image specific functions

5.1.1 Lifecycle control

In this deployment format, memory for the instance must be provided by the application, i.e. PSP does not allocate memory on its own. The application creates an instance by allocating memory of the minimum size indicated by GetDefaultParams() or bigger, and passing it to Initialise().

During the entire PSP lifecycle, the application maintains and does not modify the instance memory. The application passes the instance with any PSP function calls.

At the end of the PSP lifecycle, i.e. at the definitive end of the extraction, Terminate() should be called.

In a multi-instance setting (such as for a device shared by multiple users), the application could keep multiple PSP instances (one per user) in memory. Alternatively, the application could e.g. swap PSP instance contents between memory and any persistent storage in between calls to PSP.

Realtek

5.1.2 Code architecture

In this deployment format, the software is supplied as a pre-built binary library for a specific, predetermined ROM memory area. The software consists of several files (Table 6).

File	Contents
psp.h	PSP functions & definitions
psp_map.h	PSP functions physical memory addresses
psp.c	PSP function implementations, connecting the prototypes in psp.h to the addresses in psp_map.h
fx_datatypes.h	datatype definitions
psp.bin	executable binary code (binary file format)
psp.hex	executable binary code (Intel Hex file format)

Table 6 - C library image code architecture

In order to use the library, these steps need to be followed:

1. ensure that the code memory range, identified in psp_map.h from PSP_ROM_CONTENT$$Base_Address to PSP_ROM_CONTENT$$Limit_Address, is not used for any application code;

© Koninklijke Philips N.V. 2022. All rights reserved.

2. in the application project, define the macro for the desired platform:
   #define FX_PLATFORM_ARM_M3 - or -
   #define FX_PLATFORM_ARM_M4
3. add the correct path to **psp.h** / **fx_datatypes.h** / **psp_map.h** to the application project, so that the compiler can find these header files;
4. use one of these two methods to get the PSP executable binary code into the processor:
   a. incorporate **psp.bin** in the application project (procedure for this is development environment/tool specific), or
   b. load **psp.hex** in code memory separately with programmer
5. incorporate **psp.c** into the application code;
6. call the PSP functions in **psp.h** in the application code.

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.

Legal information

Disclaimers

This document may be subject to change without notice. Philips Electronics Nederland B.V. does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

This document is provided “as is” with no warranties, expressed or implied, including but not limited to any implied warranty of merchantability, fitness for a particular purpose, or freedom from infringement. Koninklijke Philips N.V. or its Affiliates may have patents or pending patent applications, or other intellectual property rights that relate to the described subject matter. Philips and its logo are trademarks of Koninklijke Philips N.V. All other names are the trademark or registered trademarks of their respective holders. The furnishing of this document does not provide any license, expressed or implied, by estoppels or otherwise, to any such patents, trademarks or copyrights, or other intellectual property rights. Philips Electronics Nederland B.V. assumes no responsibility for errors or omissions in this document; nor does Philips Electronics Nederland B.V. make any commitment to update the information contained herein. This document is subject to change without notice.

Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

Philips has been awarded a number of patents on the technology used in the Philips Sensing Platform offering in Europe, Asia and the United States.

Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

Realtek

© Koninklijke Philips N.V. 2022. All rights reserved.