User Guide

Software Support Package

AS7050

v1.3 • 2021-Nov-04

Content Guide

1 Introduction

The Software Support Package for the AS7050 sensor solution allows users to quickly prototype their solutions by using the supplied software. This document provides an overview of the system, software components, installation, and user instructions.

The software support package provides the ability to build and use the software components supplied to the users, along with this guide. It provides the necessary documentation, along with the source code for some components, as well as sample code demonstrating the integration of software components. The software support package allows users to access sensor data and other calculations available for use in their prototypes.

1.1 System Requirements

To use the sample code with the software packages, the following items will be required:

- AS7050 EVK Board (with firmware version >= v2.3.0)
- On Windows platform:
  - USB cable to connect AS7050 EVK Board
  - MSYS2 build environment with the "*mingw-w64-x86_64*" toolchain
- On Linux platform:
  - Refer to Chapter 3.4
  - Refer to the LINUX DRIVER [3] documentation

1.2 References

Figure 1: References

Reference	Item
APPLICATION MANAGER [1]	"as7050_appmgr_doc_vx.x.x.pdf"
CHIP LIBRARY [2]	"as7050_chiplib_docu_vx.x.x.pdf"
LINUX DRIVER [3]	"as7050_linux_driver_docu.pdf"
EVK BOARD SCHEMATIC [4]	"e005fa0_AWS AS7050 VitalSignevk_mod_variant standard_eng.pdf"

1.3 Abbreviations

Figure 2: Abbreviations

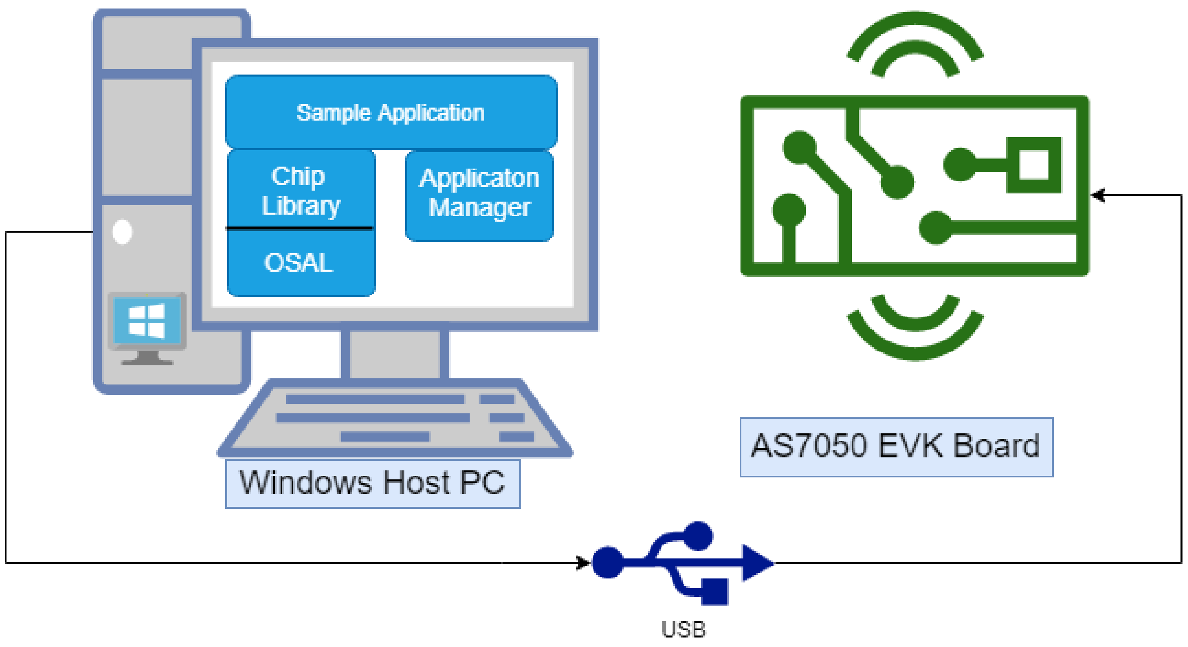
Term	Description
AGC	Auto Gain Control
API	Application Programming Interface
CLI	Command Line Interface
ECG	Electrocardiogram
EVK	Evaluation Kit
HRM	Heart Rate Monitoring.
GSR	Galvanic Skin Resistance
MS	Microsoft Corporation
OSAL	Operating System Abstraction Layer
PRV	Pulse Rate Variability
PPG	Photoplethysmogram
SpO2	Saturation of Peripheral Oxygen

2 System Overview

The Software Support Package consists of software components, documentation, and sample code, to enable users to design and build their custom application(s). The software is divided into various layers that operate at different levels. Chip Library and Application Manager are two such components of this solution, classified as the core components of this software support package, whereas Sample code contains examples and steps, which demonstrate how to glue the individual components i.e. Chip Library and Application Manager, to build a user application. To achieve platform independence, Chip Library introduces an additional layer called Operating System Abstraction Layer (OSAL). This design choice, to introduce an additional layer known as OSAL between Chip Library and hardware, allows users to integrate the provided solution into their design, by making minimal changes to OSAL, as per the underlying hardware architecture.

The software components described in this document apply to two types of platforms - Windows and Linux. **Figure 3** provides an overview of the Windows system. The components are hosted on a host PC and connect with the AS7050 EVK board via USB. Refer to chapters 3.3 and 3.4 for the Linux platform documentation.

Figure 3: System Overview



This solution includes the following components:

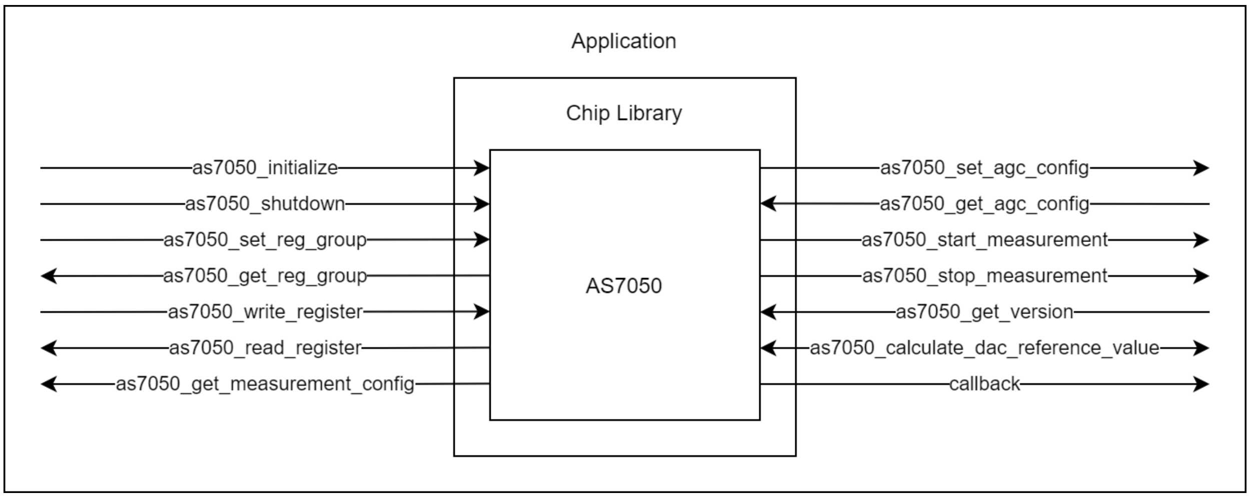
- **Chip Library:** Configures and controls the AS7050 sensor and delivers the output data from the sensor and the AGC status information.
- **Application Manager:** Configures and controls the Vital Signs Application. Also contains the GSR application.
- **Vital Signs Application:** Processes the measurement data from the Chip Library and outputs vital parameters:
  - **HRM/PRV Application:** Heart rate and pulse rate variability monitoring.
  - **SpO2 Application:** Blood oxygen saturation monitoring.
- **Vital Signs Accelerometer:** Interface to an accelerometer. This component includes an implementation of this interface for LIS2DH12.
- **Linux Driver:** Device driver to control the AS7050 sensor on Linux platforms using the Chip Library.
- **Sample Code:** Demonstrates the usage of the Chip Library and the Application Manager. Examples showing HRM, SpO2, GSR, and raw data measurement are available.

3 Software Components Overview

3.1 Chip Library

The Chip Library allows control of the AS7050 vital signs sensor module. The Chip Library interfaces with the OSAL (operating system abstraction layer). The Chip Library APIs allow usage in several different fields of application. The Chip Library is developed in standard C language, can directly be called by the user application, and can be used without adaption. The Chip Library provides APIs to the user application for establishing the communication and controlling the sensors from a higher level.

Figure 4: Block Diagram - Chip Library

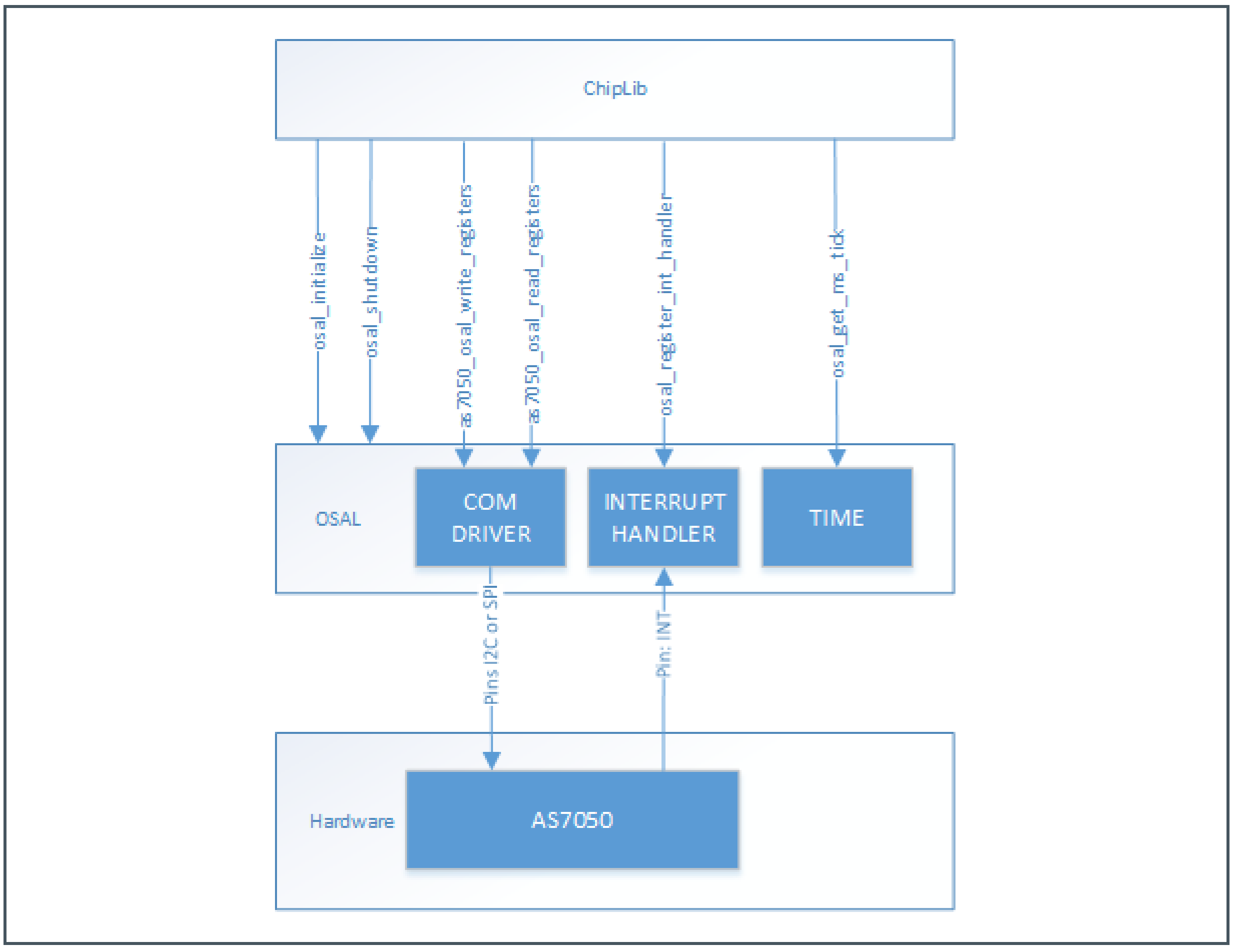


3.1.1 OSAL (Operating System Abstraction Layer)

The OSAL encapsulates platform and operating system specific functions to a separate layer, where one can customize this interface to their requirements. The OSAL reduces the direct dependency between Chip Library and the platform hardware. The OSAL is responsible for handling the sensor communication e.g. reading, writing, enabling data transfer, and interrupt signal. It can be visualized as an interface layer so that the Chip Library can be re-used on any hardware platform when it has a compatible OSAL layer. Its purpose is to hide hardware characteristics from the software.

This design choice allows to achieve the platform independence as customers can implement their own OSAL (refer to chapter 5.7) to integrate as per their custom hardware. The OSAL interface implementation is simple. There is no use of complex data types and dependencies between functions are as small as possible. Details on how to implement platform specific OSAL are described in the Chip Library documentation [2].

Figure 5: OSAL Overview



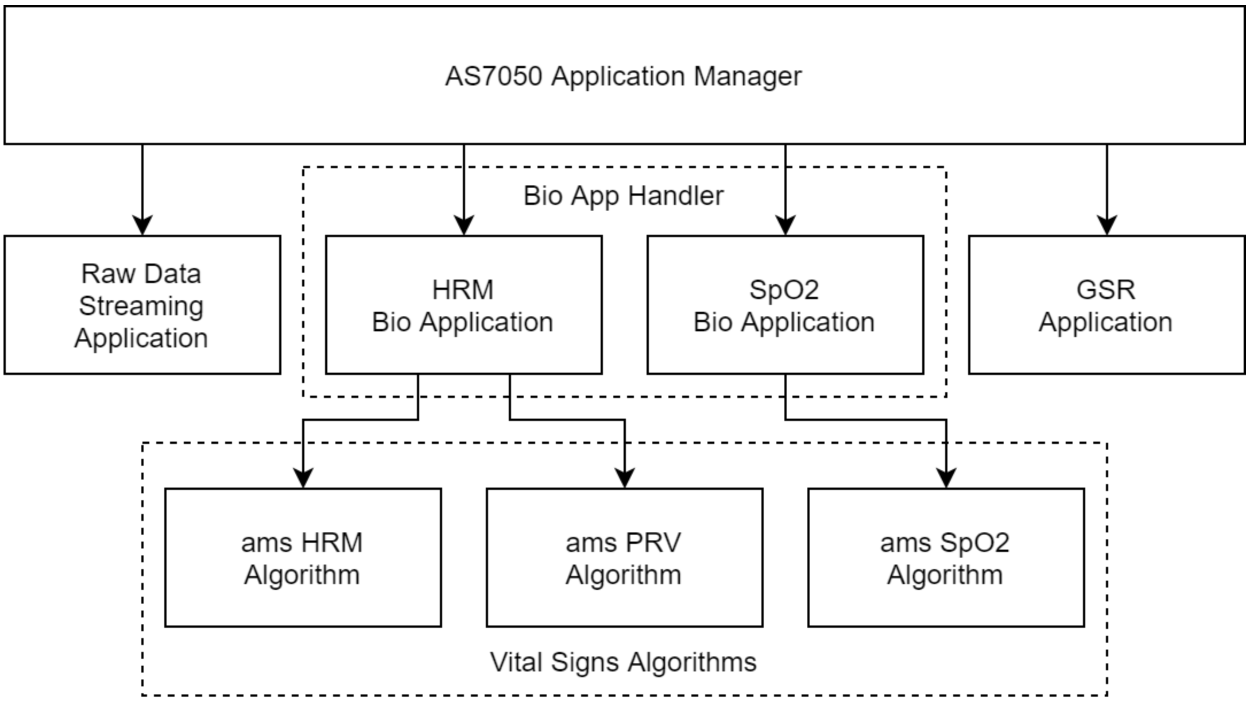
For more information on the OSAL template, please refer to 5.7 in this document.

3.2 Application Manager

The Application Manager handles and executes all Vital Signs Applications in a platform-independent way. The block diagram below gives an overview of the central functions and the interface of the Application Manager. **Figure 6** shows the different sub-blocks of the Application Manager and the corresponding provided functions.

Chapter 5.4 of this document explains the execution scheme.

Figure 6: Application Manager Overview



RAW data application: This application is not a real vital signs application but can be used to collect and distribute measured raw data for development purposes. Additional accelerometer data can be enabled/disabled by the configure-function to reduce the payload if it is not needed. The data collected inside the app is transmitted out only when any one of the below conditions is true:

- The size is greater than 149 bytes.
- New AGC data is available.
- If more than 10 accelerometer data are available.
- Latest after 1 second.

HRM application: This application sends cyclic HRM and PRV data. PRV data is optional. The output structure is bio_hrm_a0_output_t [1].

SpO2 application: This application sends cyclic SpO2 data. The output structure is bio_spo2_a0_output_t [1].

GSR application: This application performs periodic GSR measurements. Note that this application only works with the ECG channel, that the application must be configured using the DAC reference value obtained using as7050_calculate_dac_reference_value [2], and that a special chip configuration is required. The output structure is gsr_output_t [1].

Information

Note that a raw data application can be enabled all the time, but SPO2- and HRM-APP can only be enabled separately.

3.3 Linux Driver

The Linux driver provided by this solution allows access to AS7050 sensor in a Linux-based environment. The driver is implemented as a character device driver and fully integrates the Chip Library [2] of the sensor.

3.3.1 Build the driver source

The driver sources, which are provided as an archive, need to be compiled. The driver can either be built in the kernel or a module, which can be dynamically loaded at runtime. The whole process is done in several steps and finally, the sources are copied in the image present on the SD card of the raspberry pi.

The LINUX DRIVER[3] document contains detailed step-by-step instructions to prepare the Linux kernel and driver sources.

3.4 Hardware Setup for Linux Environment

This section describes the hardware connections required between the EVK board and Raspberry Pi. This setup is required to run the sample code on Linux platform.

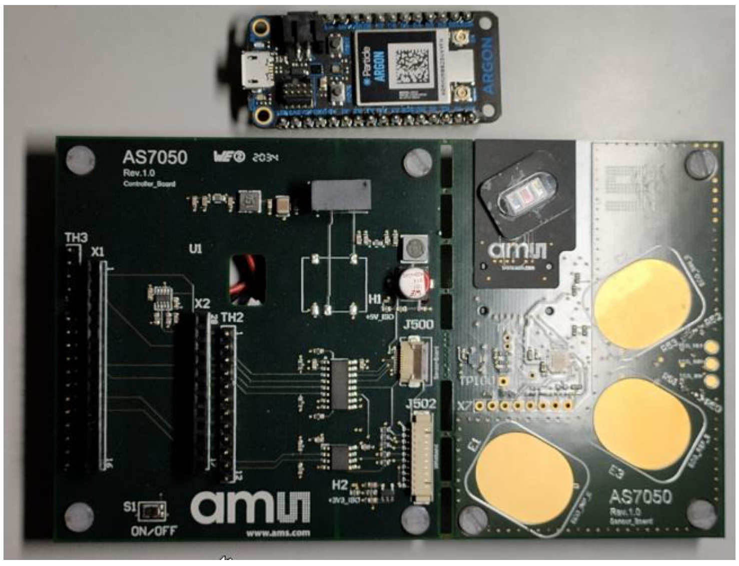
For more information on how to use the existing sample code solution, please refer to LINUX DRIVER[3].

3.4.1 Hardware Setup - AS7050 EVK and Raspberry Pi

For the setup, it requires the following hardware is:

1. AS7050 EVK board (with the Argon microcontroller board detached).

Figure 7: AS7050 EVK Board

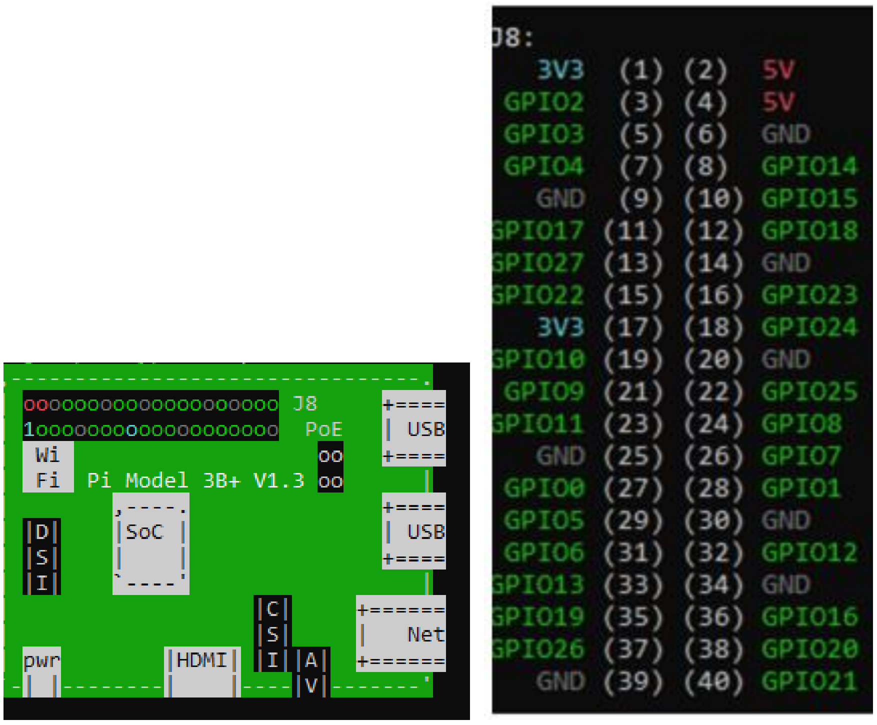


2. Jumper wires (Not included with the EVK)

Figure 8: Jumper cables with Female to Female connectors

2. Raspberry Pi board – Model 3B+

Figure 9: Board Layout and Raspberry Pin Configuration



```
-------------------------------.
oooooooooooooooooooo J8   +====
1ooooooooooooooooooo  PoE | USB
 Wi                    oo +====
 Fi  Pi Model 3B+ V1.3 oo      |
       ,----.               +====
|D|    |SoC |               | USB
|S|    |    |               +====
|I|    `----'                  |
                  |C|     +======
                  |S|     |   Net
pwr        |HDMI| |I||A|  +======
-| |--------|    |----|V|-------'
```

```
J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21
```

Prepare the Hardware Setup

1. Disassemble the Argon microcontroller board from the AS7050 EVK board.

2. Do not connect the plug of the accumulator.

3. Please ensure that I2C on the Raspberry Pi is enabled.

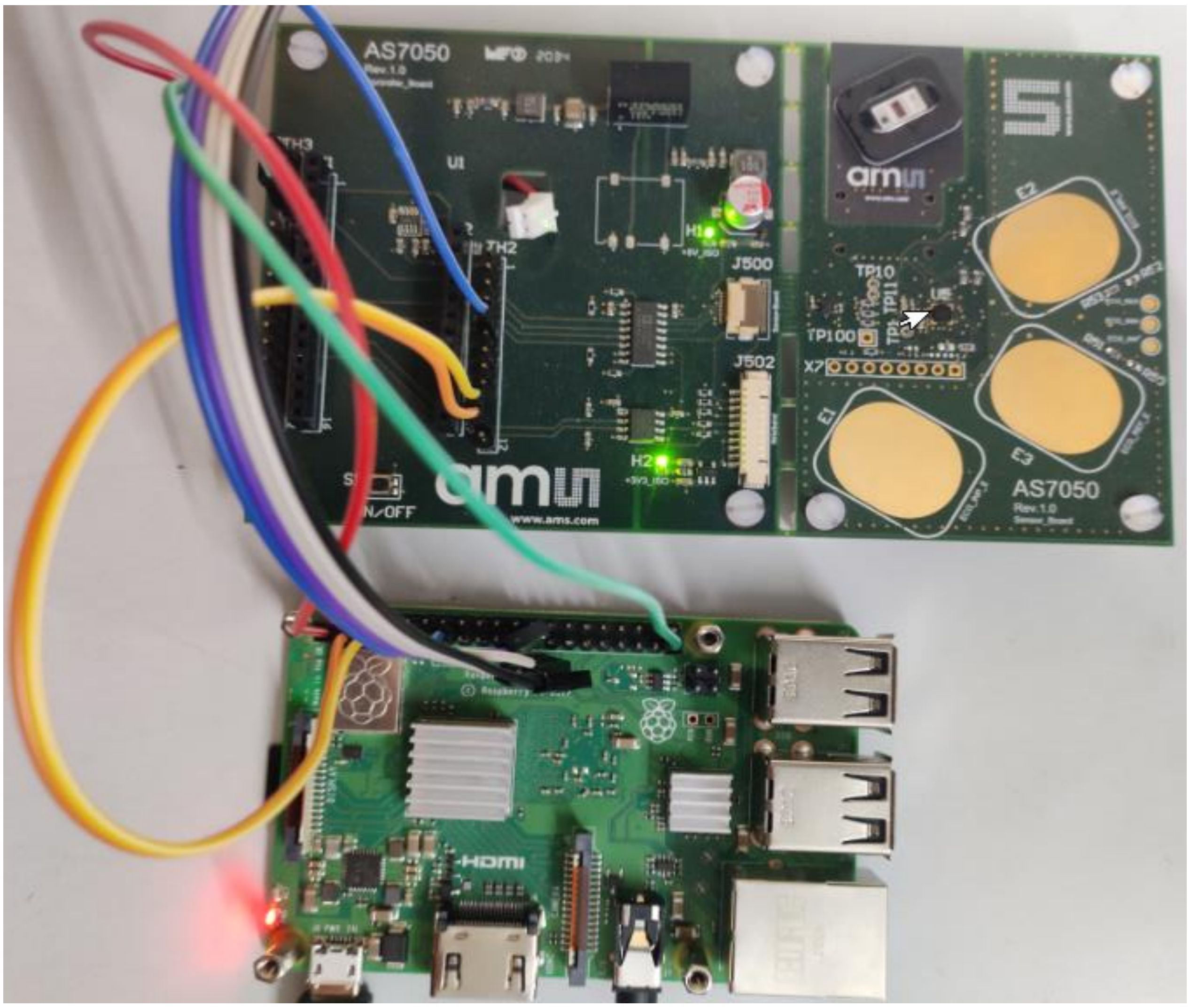
4. Connect the following pins (Cables not included with the EVK):

```
AS7050,TH2-11 <->  RASPI, J8-3  (SDA <-> SDA1)
AS7050,TH2-10 <->  RASPI, J8-5  (SCL <-> SCL1)
AS7050,TH2-5   <->  RASPI, J8-13 (7050_INT <-> GPIO27)
AS7050,TH3-2   <->  RASPI, J8-1  (3V3_BT <-> 3V3)
AS7050,TH3-4   <->  RASPI, J8-39 (GND <-> GND)
```

Attention

Please refer EVK BOARD SCHEMATIC [4] for details on Pin connections.

Figure 10: Integrated setup between AS7050 EVK and Raspberry Pi



3.4.2 Driver Usage

There are two ways to use this driver.

- If the driver is compiled into the kernel i.e using devicetree, no further action is needed and driver will be loaded when the raspberry boots.

- Without the devicetree: If the driver is compiled as a module then the AS7050 driver kernel object file needs to be copied into the raspberry pi in the location below.

  ```
  /lib/modules/$(uname -r)/kernel/drivers/staging/ams-as7050-i2c.ko
  ```

Once the previous step is complete, execute the commands below to load the driver.

```
> Depmod
> insmod ams-as7050-i2c
> echo as7050-i2c 0x55> /sys/bus/i2c/devices/i2c-1/new_device
> echo 27 > /sys/class/misc/as7050/irq
```

Attention

Please note that the commands above require privileged operation i.e. “`sudo/ su`” permissions.

4 Installation

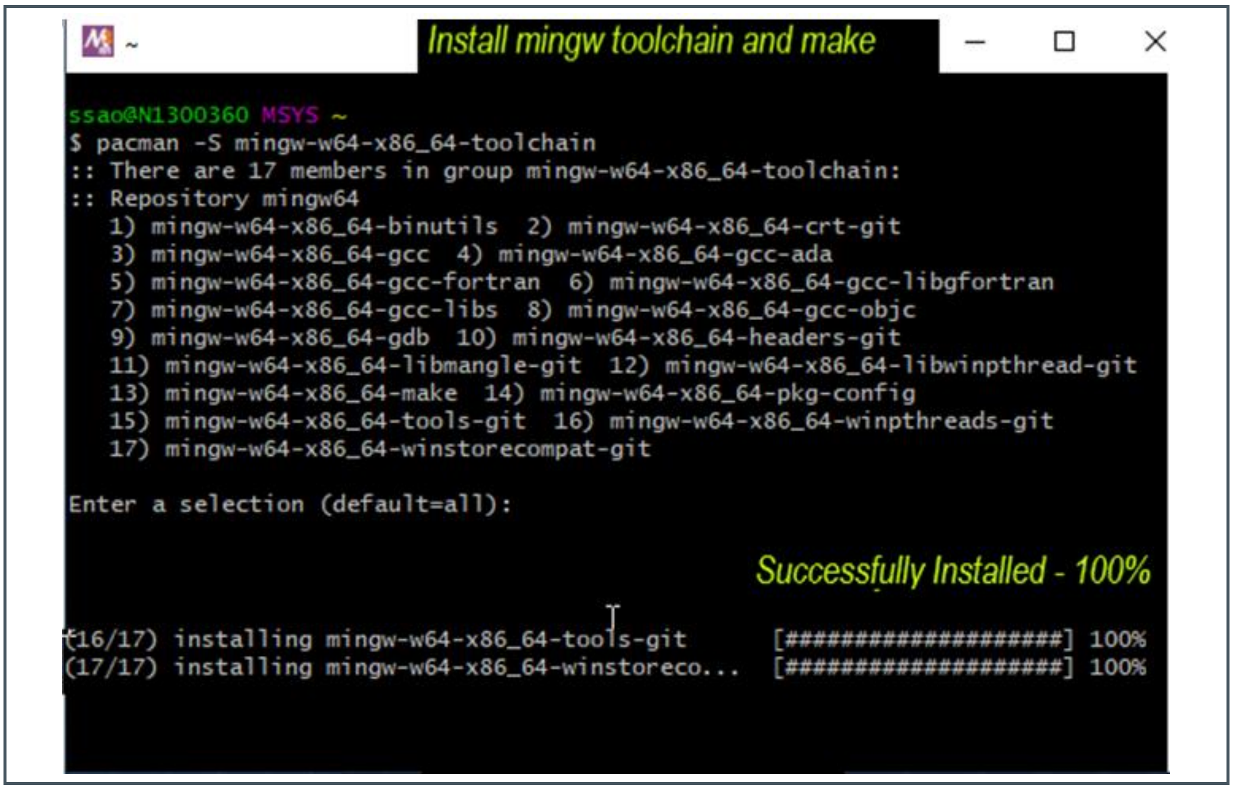
4.1 Requirements

Refer to chapter 1.1 System Requirements in this document.

4.2 Installation of msys2 tool collection

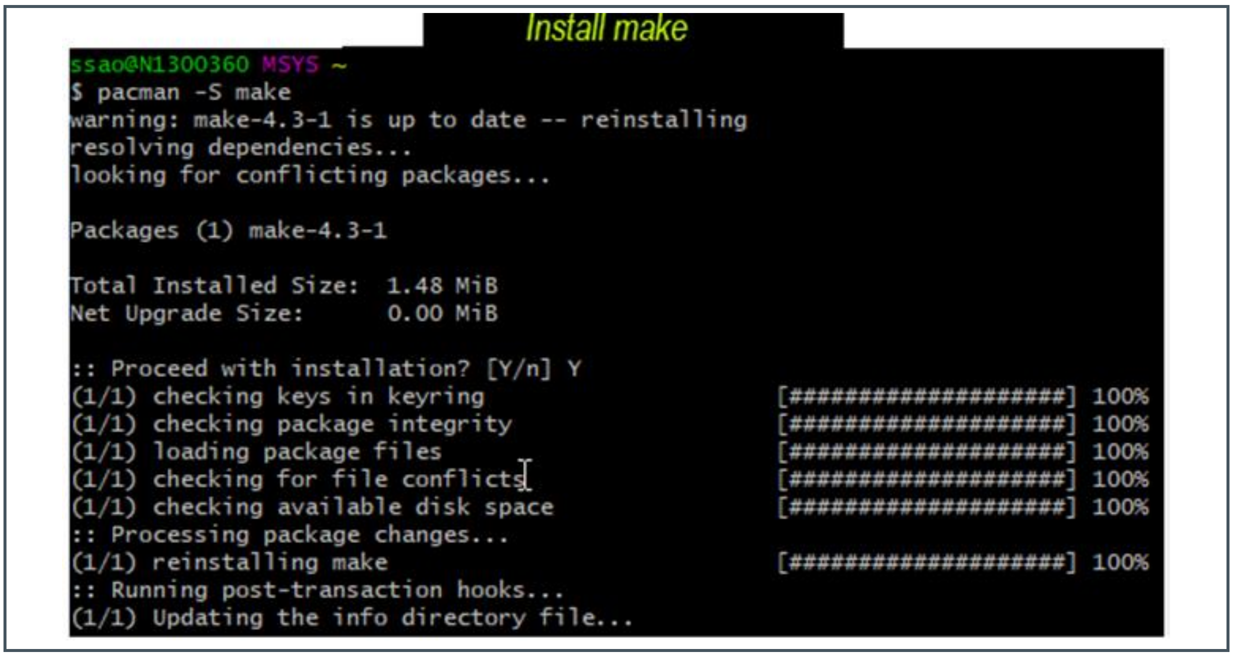
1. Download the MSYS-64 toolchain software. (Please use this link or download from the internet.)
2. When the previous step is complete, locate the msys2-x86_64 executable(.exe) and start the installation procedure. Follow the instructions and complete the setup.
3. For further help on installation, refer to here after installing the msys2 environment.
4. After completing steps 1 -3, launch the MSYS2 shell (default: *C:\msys64\msys2.exe*).
5. Execute the command "`pacman -S mingw-w64-x86_64-toolchain`" and press "Enter" to install all the package updates. Follow onscreen instructions until completion (refer to Figure 11 and Figure 12.
6. Execute the command "`pacman -S make`".
7. Once installation is complete, close the mysys2 shell (optional).

Figure 11: Toolchain installation and make


~
Install mingw toolchain and make
Successfully Installed - 100%

```
ssao@N1300360 MSYS ~
$ pacman -S mingw-w64-x86_64-toolchain
:: There are 17 members in group mingw-w64-x86_64-toolchain:
:: Repository mingw64
   1) mingw-w64-x86_64-binutils  2) mingw-w64-x86_64-crt-git
   3) mingw-w64-x86_64-gcc  4) mingw-w64-x86_64-gcc-ada
   5) mingw-w64-x86_64-gcc-fortran  6) mingw-w64-x86_64-gcc-libgfortran
   7) mingw-w64-x86_64-gcc-libs  8) mingw-w64-x86_64-gcc-objc
   9) mingw-w64-x86_64-gdb  10) mingw-w64-x86_64-headers-git
   11) mingw-w64-x86_64-libmangle-git  12) mingw-w64-x86_64-libwinpthread-git
   13) mingw-w64-x86_64-make  14) mingw-w64-x86_64-pkg-config
   15) mingw-w64-x86_64-tools-git  16) mingw-w64-x86_64-winpthreads-git
   17) mingw-w64-x86_64-winstorecompat-git

Enter a selection (default=all):

(16/17) installing mingw-w64-x86_64-tools-git      [####################] 100%
(17/17) installing mingw-w64-x86_64-winstoreco...  [####################] 100%
```

Figure 12: Install make



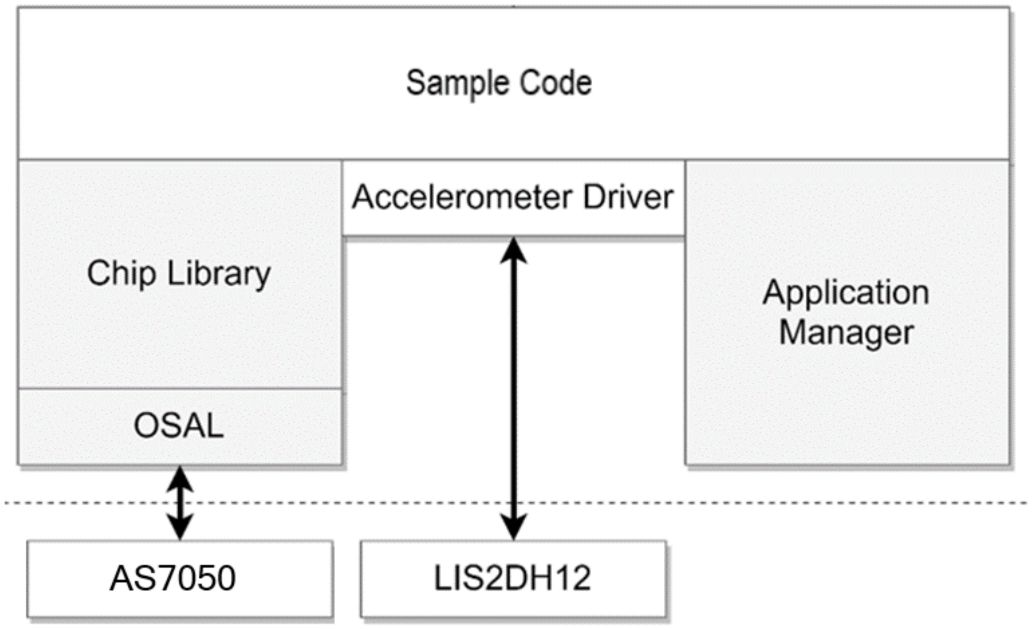
```
ssao@N1300360 MSYS ~
$ pacman -S make
warning: make-4.3-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) make-4.3-1

Total Installed Size:  1.48 MiB
Net Upgrade Size:      0.00 MiB

:: Proceed with installation? [Y/n] Y
(1/1) checking keys in keyring                     [####################] 100%
(1/1) checking package integrity                   [####################] 100%
(1/1) loading package files                        [####################] 100%
(1/1) checking for file conflicts                  [####################] 100%
(1/1) checking available disk space                [####################] 100%
:: Processing package changes...
(1/1) reinstalling make                            [####################] 100%
:: Running post-transaction hooks...
(1/1) Updating the info directory file...
```

5 Getting Started

5.1 Sample Code Overview

The Sample Code provided by this solution shows in a simple manner how to connect Chip Library and the Application Manager. The interfaces to the OSAL and Accelerometer are included in this solution. In this example, the Chip Library, the Application Manager, and the sample code reside and execute on a host PC, and the EVK board functions only as a sensor interface. Refer to **Figure 13** for an overview.

Figure 13: Sample Code Overview



Information

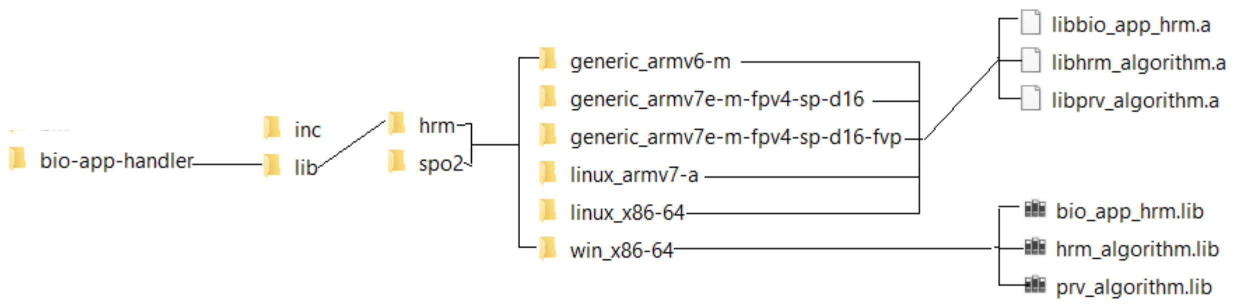
Note: The sample code instructions explained in this section are specific to the MS Windows platform. Before execution, please connect an AS7050 EVK board to a host PC. Chapter 5.6.1 in this document, contains detailed instructions on how to execute the sample code. Please refer to it for further details.

5.2 Extracting the source files

This chapter explains the folder structure and the individual components provided as a part of the software support package. Extract the contents of the ZIP archive to the desired folder, and ensure that all the components are available as listed below.

- **as7050-app-manager:** AS7050 Application Manager, handles the Vital Signs Applications. Also contains the GSR application.
- **as7050-chip-library:** AS7050 Chip Library configures and controls the sensor via I2C.
- **as7050-linux-driver:** AS7050 driver for the Linux environment.
- **bin:** Contains the prebuilt executable files for HRM, SpO2, Chip Library, and GSR sample codes. These executables are available only for Windows x86_64 platform.
- **bio-app-handler:** Shared libraries for the Vital Signs applications HRM and SpO2. Used by the AS7050 Application Manager.
- **css-sw-utilities:** Various helper libraries specific to this AS7050 EVK implementation.
- **doc:** API documents - Application Manager and Chip Library.
- **sample_code:** Sample code showing HRM, SpO2, and GSR measurements using AS7050 Chip Library and AS7050 Application Manager. Also contains an example that uses AS7050 Chip Library only.
- **vital-signs-acc:** Accelerometer interface. This component includes an implementation of this interface for LIS2DH12.
- **vitalsigns-algorithms:** Algorithms for calculating HRM and SPO2 values from the AS7050 raw data, used by the Bio App Handler provided as binary libraries.
- **Makefile:** Used for code compilation.

Figure 14: Platform specific binary libraries



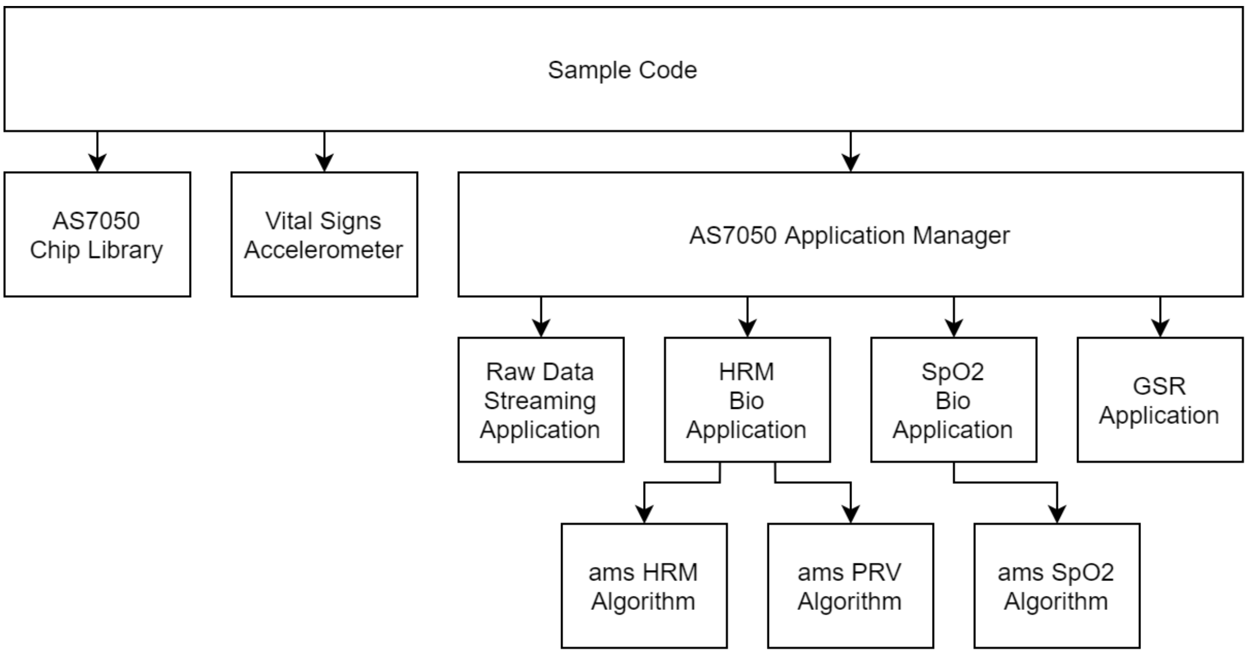
Information

Note: Binary libraries provided for Bio app handler and vital signs algorithms support arm and x86_64 platforms, as shown in Figure 14.

5.3 Integration with Chip Library and Application Manager

This chapter provides an overview of the integration process of Chip Library and Application Manager with the Sample Code, whereas the subsequent sections contain further instructions and explanations required in the process. The interfaces to the OSAL and accelerometer are included. This solution is curated to work specifically with the AS7050 sensor module and EVK board. Customers could also integrate the Chip Library and Application Manager into their products by following the process and steps in this guide. The Sample Code solution contains source files developed in C language. It contains examples to demonstrate the integration process with the available components.

Figure 15: Sample Code Integration



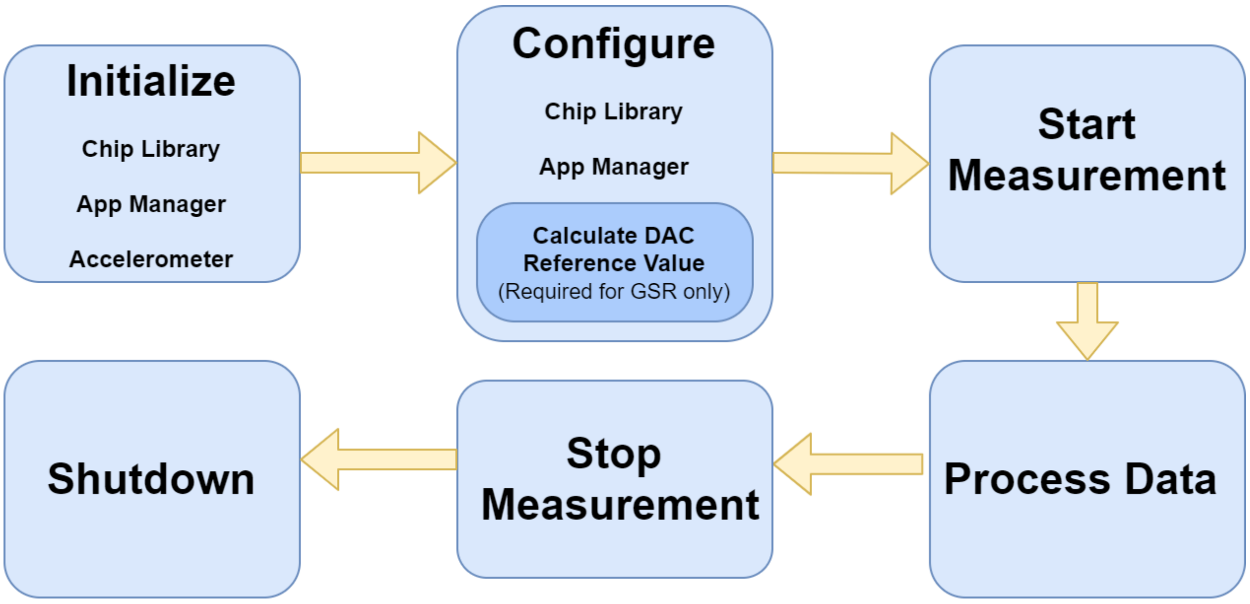
5.3.1 Examples

The software support package contains the source files and pre-built executables for sample code. These executables can be located in bin folder and are available only for Windows x86_64 platform.

5.4 Execution Flow

This chapter provides an overview on the code flow. The steps in **Figure 16** describe the general code flow of the sample code and the chapters below contain the code snippets from the source code.

Figure 16: Execution Flow



Information

Note: The code snippets provided below are only a demonstration of the code flow. Please refer to the source files provided in chapter 5.2 for the complete solution.

5.4.1 Initialize

Initialize the components first by invoking the Initialization functions:

- “as7050_initialize”
- “as7050_appmgr_initialize”
- “vs_acc_initialize”

5.4.2 Configuration

Once the initialization steps are complete:

1. Configure Chip Library registers by calling the "as7050_set_reg_group" function. A code snipped configuring the GPIO register group can be found below. Similarly, other register groups shall be configured. Refer to [2] for the API details and instructions.

```
const as7050_config_gpio_t gpio_config = {
    .reg_vals.gpio1_cfg = 0x00,
    .reg_vals.gpio2_cfg = 0x00,
    .reg_vals.gpio1_cfgb = 0x00,
    .reg_vals.gpio2_cfgb = 0x00,
    .reg_vals.gpio_io = 0x00,
};

result = as7050_set_reg_group(AS7050_REG_GROUP_ID_GPIO, (uint8_t *)gpio_config.
reg_buffer, sizeof(as7050_config_gpio_t));
```

2. Configure the Application Manager by calling the functions "as7050_appmgr_set_signal_routing" and "as7050_appmgr_enable_apps" as shown in the code snippet below.

```
as7050_appmgr_channel_id_t hrm_channel = AS7050_CHANNEL_PPG_1;
result = as7050_appmgr_set_signal_routing(AS7050_APPMGR_APP_ID_HRM_A0,
&hrm_channel, 1);

result = as7050_appmgr_enable_apps(AS7050_APPMGR_APP_FLAG_HRM_A0);
```

3. Before the start of a measurement, the Application Manager and its applications must be prepared for calculation by calling the function "as7050_appmgr_start_processing" as shown in the code snippet below. Note that the configuration of the Application Manager or its applications cannot be changed until "as7050_appmgr_stop_processing" is called.

```
as7050_meas_config_t meas_config;
result = as7050_get_measurement_config(&meas_config);

result = as7050_appmgr_start_processing(meas_config, ACC_SAMPLE_PERIOD_US,
&agc_config.channel[AS7050_CHANNEL_GROUP_A], 1);
```

5.4.3 Start and stop the measurements

To start and stop measurements using AS7050 call "as7050_start_measurement" and "as7050_stop_measurement". Call "vs_acc_start" and "vs_acc_stop" to control accelerometer measurements.

5.4.4 Process Data

The execution scheme of the applications follows a simple principle:

1. Set a new data input with the function "as7050_appmgr_set_input" and dispatch the new input data to the enabled applications. The applications then respond to whether they are ready for execution. The aggregated "ready for execution" status will be returned.
2. Execute applications by calling the function "as7050_appmgr_execute". All applications that are ready to execute will be executed. The applications then respond to whether new output data was generated. Combined flags indicating the applications with new output data available will be returned.
3. Get output data directly from the application with the functions "as7050_appmgr_get_output": The available output data can be fetched from the applications.

In the provided sample code, the measured raw data is provided to the Application Manager when the Chip Library's callback function is invoked. After providing raw data chunks to the Application Manager, the Application Manager indicates that it is ready for execution. Execution of the applications and getting output data occurs periodically in the main thread, as shown in the code snippet below.

```
uint32_t app_data_available;
result = as7050_appmgr_execute(&app_data_available);

if (AS7050_APPMGR_APP_FLAG_HRM_A0 & app_data_available) {
    bio_hrm_a0_output_t hrm_output;
    uint16_t output_size = sizeof(bio_hrm_a0_output_t);

    result = as7050_appmgr_get_output(AS7050_APPMGR_APP_ID_HRM_A0, &hrm_output,
    &output_size);
}
```

Information

Note: Providing raw data from the Chip Library and executing the Application Manager is a cyclic process. Do not execute Application Manager from the Chip Library's callback function.

5.4.5 Shutdown

To de-initialize the system, start the app shutdown process. This is the final step in the execution. Invoke these functions to begin the shutdown process:

- "vs_acc_stop"
- "vs_acc_shutdown"
- "as7050_appmgr_shutdown"
- "as7050_shutdown"

5.5 Compile and Build the Code

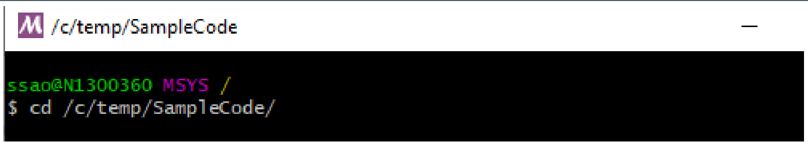
5.5.1 Target Platform – MS Windows

Information

Note: The solution provided in the ZIP archive already contains the pre-built executables for the HRM and SpO2 applications. In case the user wishes to build the sample source code, please follow the instructions below.

1. Launch the msys2 Mingw terminal and navigate to the folder where contents of the ZIP folder were extracted. (hint: `cd  folder_location`)

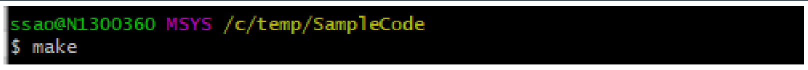
Figure 17: Open file location



```
/c/temp/SampleCode

ssao@N1300360 MSYS /
$ cd /c/temp/SampleCode/
```

2. Execute make.

Figure 18: Executing make



```
ssao@N1300360 MSYS /c/temp/SampleCode
$ make
```

3. This builds the sample code and places the generated executables (*.exe) in the directory where the contents of the archive were extracted.

Information

Note: In case the above steps do not work, please check chapter 6.1 Msys2 build process issue.

5.5.2 Target Platform – Linux

The source code for the sample code provided in the ZIP archive contains the Makefile, which can be used to build the executable for Linux platform. To build the sample code provided in the form of a ZIP archive, please follow the steps described below:

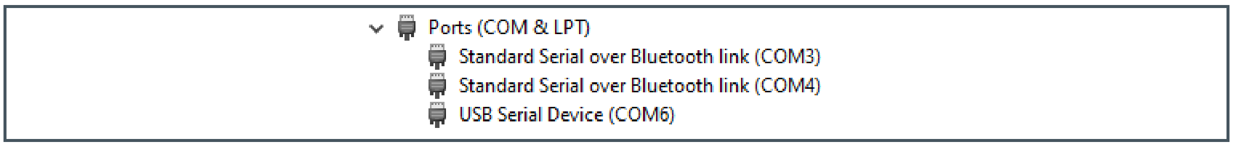
1. Navigate to the directory where the contents of the ZIP archive are extracted, and issue command "`make TARGET_OS=linux TARGET_ARCH=armv7_a`".
2. Once the build generation is successful, the executables are created in the same directory where the `make` command was issued.

5.6 Execute the Sample Code

5.6.1 On Target platform – MS Windows

1. Connect the AS7050 EVK to the PC.
2. Check the correct PORT number from the device manager.

Figure 19: COM Port in Device Manager



```
Ports (COM & LPT)
    Standard Serial over Bluetooth link (COM3)
    Standard Serial over Bluetooth link (COM4)
    USB Serial Device (COM6)
```

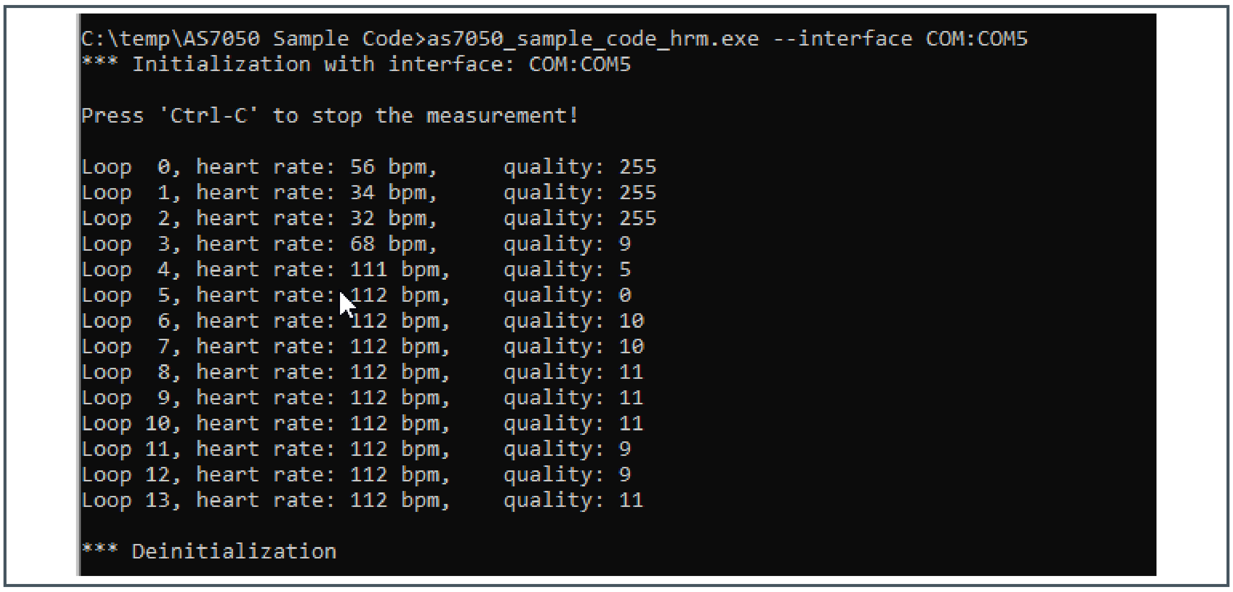
3. Open the windows command prompt.
4. Navigate to the directory where the “as7050_sample_code_hrm.exe”, “as7050_sample_code_spo2.exe”, “as7050_sample_code_chip_lib.exe”, and “as7050_sample_code_gsr.exe” sample codes were generated.

Execute the HRM, SpO2, and Chip Library Sample Code

In a command prompt, run the executable of the sample code you wish to execute and pass “`--interface COM:<COM-Port>`” as argument. You need to replace `<COM-Port>` with the name of the port the device is assigned to. For example, to run the HRM sample code with an AS7050 EVK assigned to serial port COM6, run “`as7050_sample_code_hrm.exe --interface COM:COM6`”.

The sample code can be stopped by pressing “Ctrl – C”

Figure 20: HRM Code Execution for MS Windows

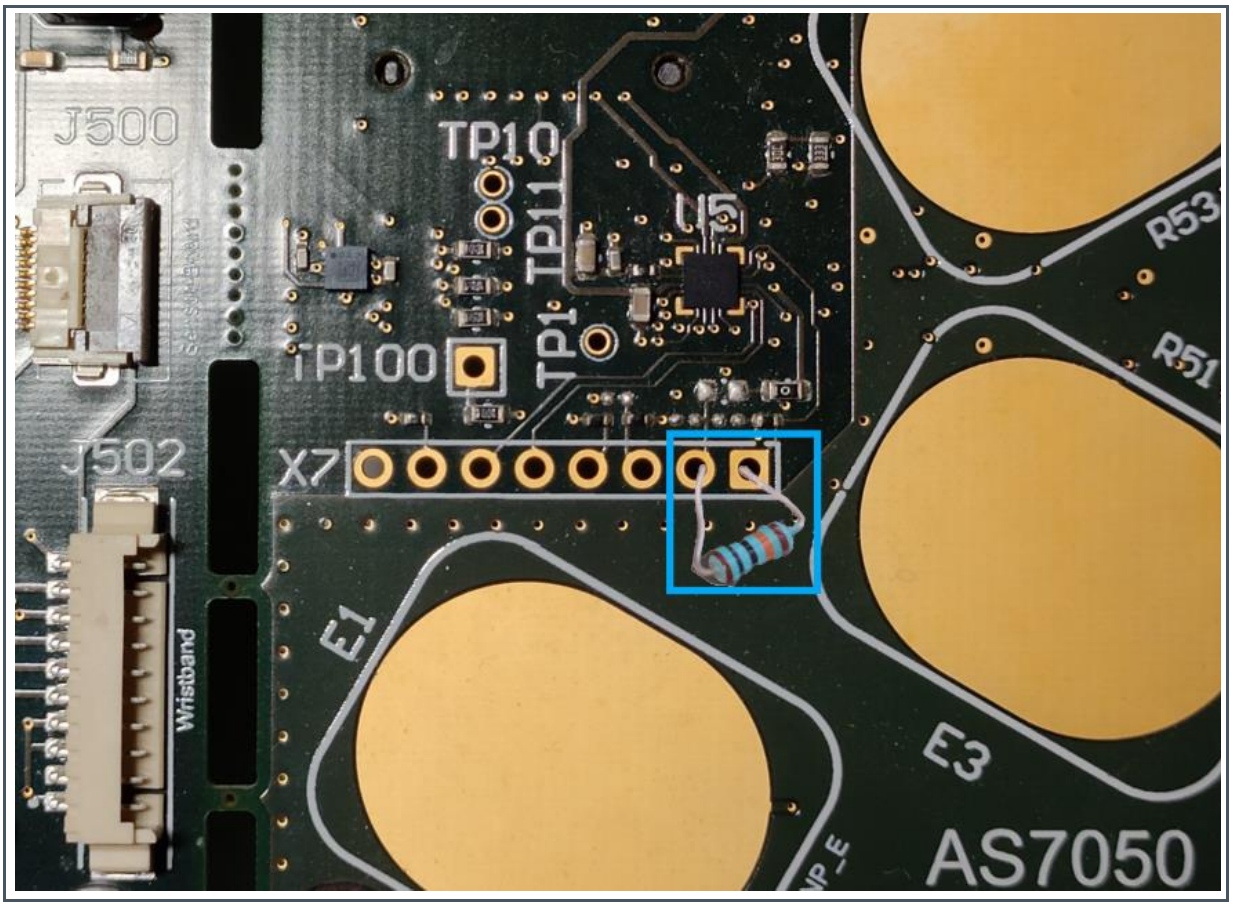


```
C:\temp\AS7050 Sample Code>as7050_sample_code_hrm.exe --interface COM:COM5
*** Initialization with interface: COM:COM5

Press 'Ctrl-C' to stop the measurement!

Loop  0, heart rate: 56 bpm,     quality: 255
Loop  1, heart rate: 34 bpm,     quality: 255
Loop  2, heart rate: 32 bpm,     quality: 255
Loop  3, heart rate: 68 bpm,     quality: 9
Loop  4, heart rate: 111 bpm,    quality: 5
Loop  5, heart rate: 112 bpm,    quality: 0
Loop  6, heart rate: 112 bpm,    quality: 10
Loop  7, heart rate: 112 bpm,    quality: 10
Loop  8, heart rate: 112 bpm,    quality: 11
Loop  9, heart rate: 112 bpm,    quality: 11
Loop 10, heart rate: 112 bpm,    quality: 11
Loop 11, heart rate: 112 bpm,    quality: 9
Loop 12, heart rate: 112 bpm,    quality: 9
Loop 13, heart rate: 112 bpm,    quality: 11

*** Deinitialization
```

Execute the GSR Sample Code

To perform GSR measurements the load needs to be connected to pins GPIO1 and GPIO2 of the AS7050 device. Pins GPIO1 and GPIO2 are exposed via pins 1 and 2 of connector X7 on the AS7050 EVK.

Figure 21: Resistor connected to AS7050 pins GPIO1 and GPIO2



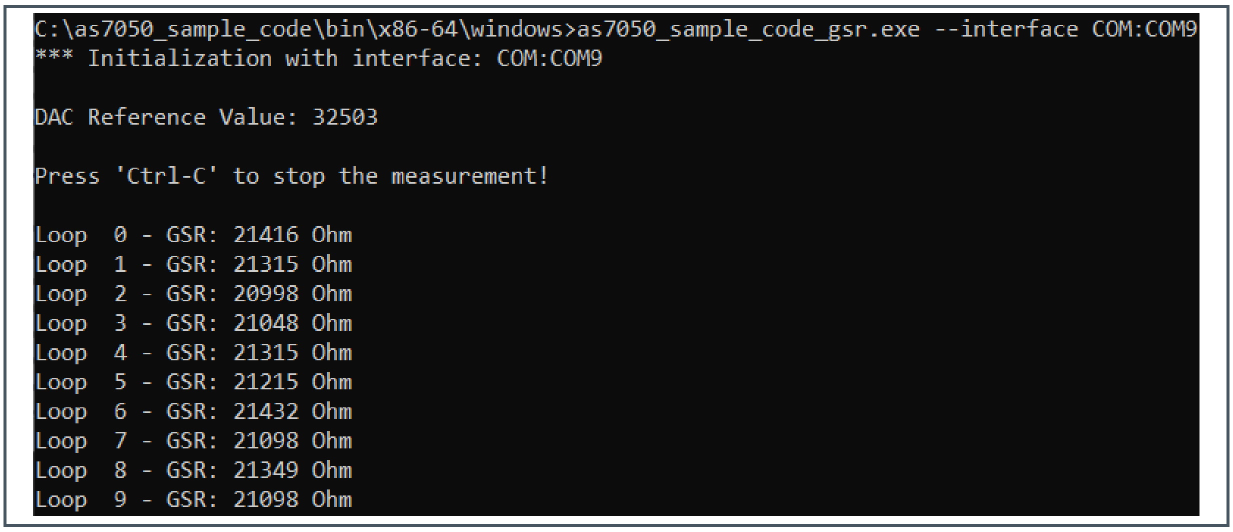
Information

By default, the ECG electrodes on the AS7050 EVK Board are not connected to GPIO1 and GPIO2, i.e. they cannot be used for GSR measurements. If you want to use the on-board ECG electrodes for GSR measurements, hardware modifications are required. Please reach out to our support team for more information.

To execute the GSR sample code, run "`as7050_sample_code_gsr.exe –interface COM:<COM-Port>`" in a command prompt. You need to replace `<COM-Port>` with the name of the port the device is assigned to.

The sample code can be stopped by pressing "Ctrl – C"

Figure 22: GSR Code Execution for MS Windows



```
C:\as7050_sample_code\bin\x86-64\windows>as7050_sample_code_gsr.exe --interface COM:COM9
*** Initialization with interface: COM:COM9

DAC Reference Value: 32503

Press 'Ctrl-C' to stop the measurement!

Loop  0 - GSR: 21416 Ohm
Loop  1 - GSR: 21315 Ohm
Loop  2 - GSR: 20998 Ohm
Loop  3 - GSR: 21048 Ohm
Loop  4 - GSR: 21315 Ohm
Loop  5 - GSR: 21215 Ohm
Loop  6 - GSR: 21432 Ohm
Loop  7 - GSR: 21098 Ohm
Loop  8 - GSR: 21349 Ohm
Loop  9 - GSR: 21098 Ohm
```

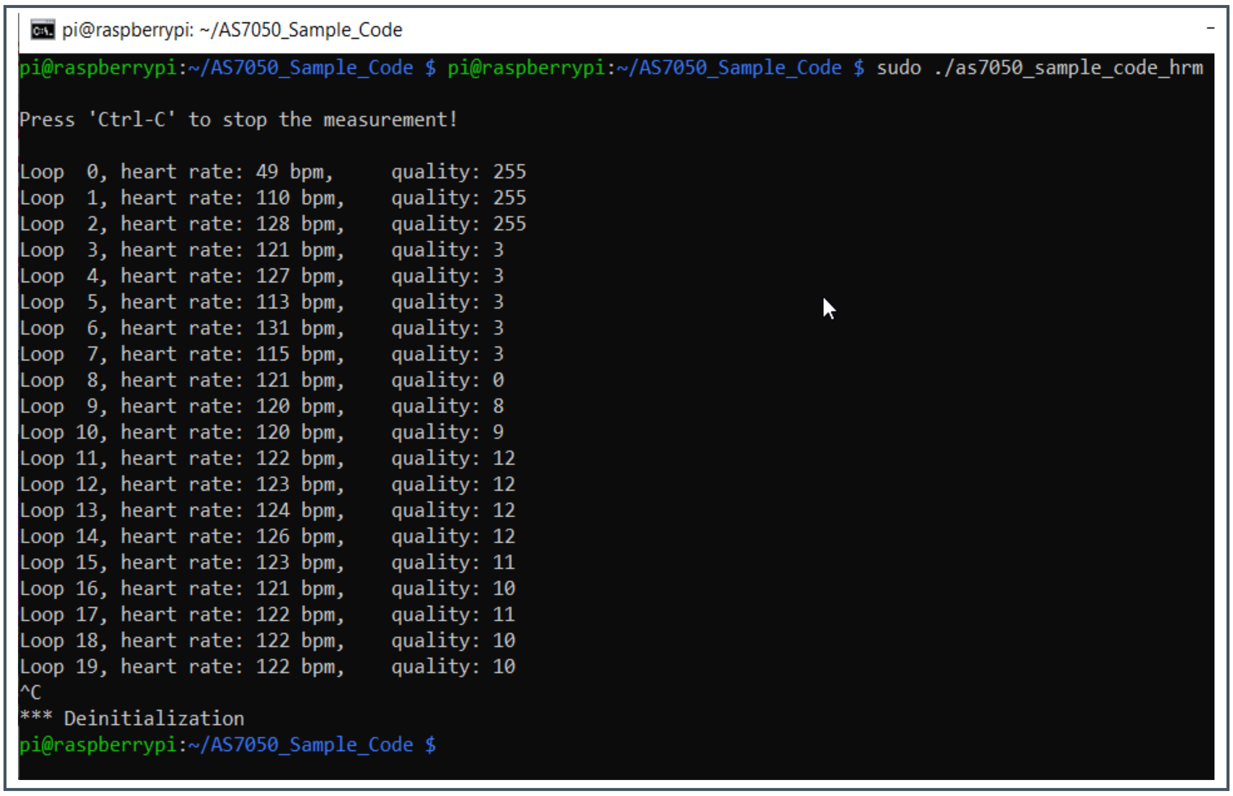
Information

The data printed on the console can be logged to a file. Use command "`<Executable> --interface COM:<COM-Port> > <filename.log>`", e.g. "`as7050_sample_code_gsr.exe --interface COM:COM5 > SampleCodeData.log`"

5.6.2 On Target platform – Linux

To run a sample code executable on Linux, simply run the executable without any additional argument. There are no other differences between the Linux and Windows variants of the sample code, i.e. most of the information provided in chapter 5.6.1 applies to the Linux sample code as well.

Figure 23: HRM Code Execution for Linux



```
pi@raspberrypi:~/AS7050_Sample_Code $ pi@raspberrypi:~/AS7050_Sample_Code $ sudo ./as7050_sample_code_hrm

Press 'Ctrl-C' to stop the measurement!

Loop  0, heart rate: 49 bpm,     quality: 255
Loop  1, heart rate: 110 bpm,    quality: 255
Loop  2, heart rate: 128 bpm,    quality: 255
Loop  3, heart rate: 121 bpm,    quality: 3
Loop  4, heart rate: 127 bpm,    quality: 3
Loop  5, heart rate: 113 bpm,    quality: 3
Loop  6, heart rate: 131 bpm,    quality: 3
Loop  7, heart rate: 115 bpm,    quality: 3
Loop  8, heart rate: 121 bpm,    quality: 0
Loop  9, heart rate: 120 bpm,    quality: 8
Loop 10, heart rate: 120 bpm,    quality: 9
Loop 11, heart rate: 122 bpm,    quality: 12
Loop 12, heart rate: 123 bpm,    quality: 12
Loop 13, heart rate: 124 bpm,    quality: 12
Loop 14, heart rate: 126 bpm,    quality: 12
Loop 15, heart rate: 123 bpm,    quality: 11
Loop 16, heart rate: 121 bpm,    quality: 10
Loop 17, heart rate: 122 bpm,    quality: 11
Loop 18, heart rate: 122 bpm,    quality: 10
Loop 19, heart rate: 122 bpm,    quality: 10
^C
*** Deinitialization
pi@raspberrypi:~/AS7050_Sample_Code $
```

Information

The Linux sample code uses the AS7050 Linux Driver and requires that the Linux target has been configured to use this driver. Please refer to chapter 3.4 for more information.

5.7 OSAL Template Guide

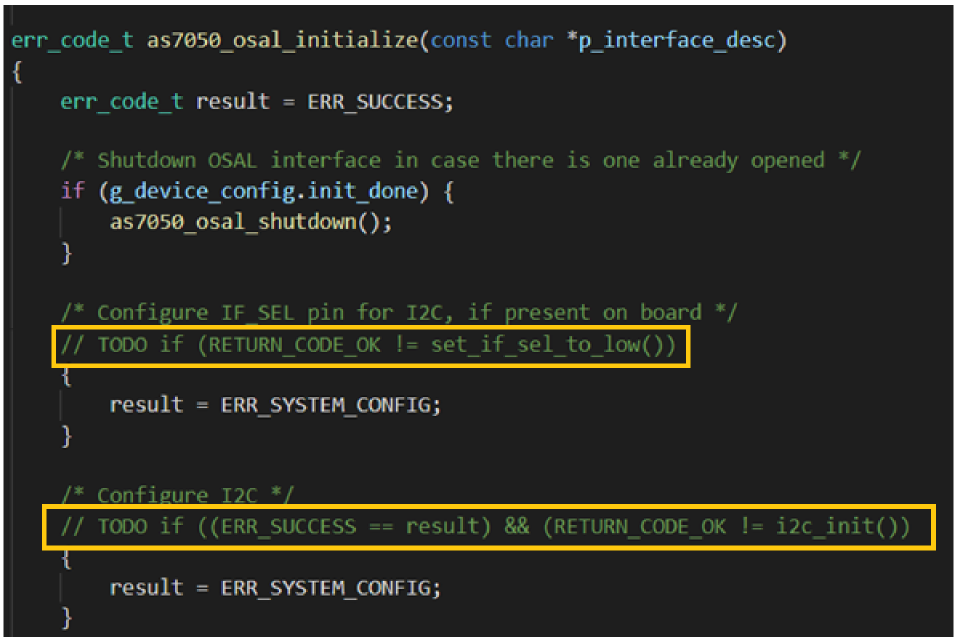
5.7.1 What is OSAL Template?

The AS7050 sample code archive (explained in chapter 5.2) also contains OSAL templates for the AS7050 Chip Library and the accelerometer interface. The purpose of the template files is to serve as a starting point for custom OSAL implementations. The OSAL template file for the AS7050 Chip Library can be found at path “as7050-chip-lib/osal/as7050_osal_chiplib_template.c” inside the archive. The accelerometer OSAL template file can be found at path “vital-signs-acc/src/ vital_signs_acc_osal_template.c”.

5.7.2 OSAL template usage

OSAL functions need to be implemented to support the platform-specific targets. It is a mandatory step in the process for customers who wish to use this for their specific target platform. The customers can choose this template as a base and replace the TODO comments with their code. For example, this code snippet is extracted from the template file and highlights the TODO entries, which are required to be filled/updated by the customer. This is to help the customers to visualize the OSAL code interface aiming to simplify the whole integration process for their product development.

Figure 24: OSAL code



```
err_code_t as7050_osal_initialize(const char *p_interface_desc)
{
    err_code_t result = ERR_SUCCESS;

    /* Shutdown OSAL interface in case there is one already opened */
    if (g_device_config.init_done) {
        as7050_osal_shutdown();
    }

    /* Configure IF_SEL pin for I2C, if present on board */
    // TODO if (RETURN_CODE_OK != set_if_sel_to_low())
    {
        result = ERR_SYSTEM_CONFIG;
    }

    /* Configure I2C */
    // TODO if ((ERR_SUCCESS == result) && (RETURN_CODE_OK != i2c_init())
    {
        result = ERR_SYSTEM_CONFIG;
    }
```

5.8 Conclusion / Next Steps

The solution provided in this example is built for the AS7050 sensor. However, it allows flexibility for customers/users to reuse the core components.

Key Points for Customers/Users:

- To reuse the core components on different hardware, modify the OSAL for platform-specific changes to include support for custom hardware. It is recommended to use the OSAL template provided in the sample code archive. (Refer to chapter 5.7 of this document.)
- Application Manager is an optional software component. Customers can choose to implement their solution as per their needs.
- Choose your improved algorithms for HRM and SpO2. **ams** vital signs algorithms (HRM/PRV, SpO2) provided in the form of binary libraries are also an optional component. Customers can choose to replace with their algorithms.

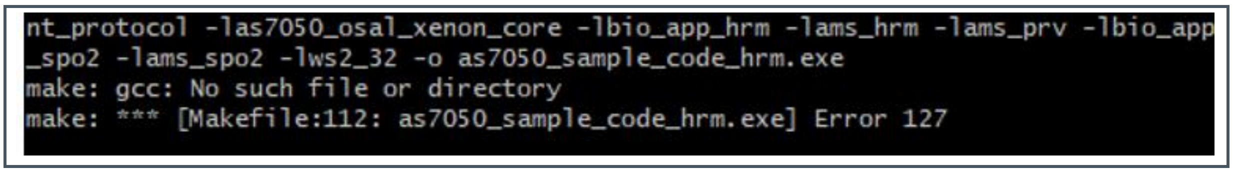
6 Limitations

6.1 Msys2 build process issue

At times, the compilation process with msys2 does not succeed using the above steps. It is a known limitation of msys2 where the toolchain is not able to recognize the files required to compile successfully. In such scenarios, use the Windows command prompt and follow the same steps as above. Refer to Figure 26 and Figure 27 for the issue and solution respectively.

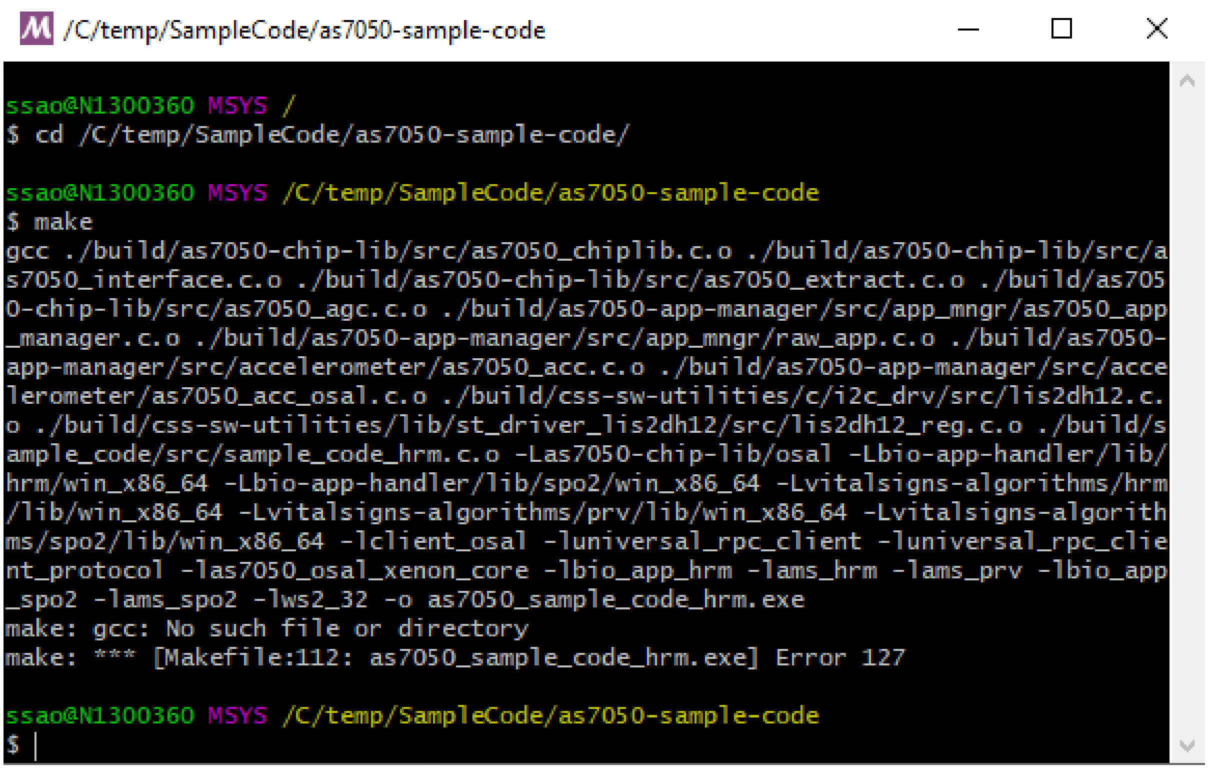
Issue: `make` error on msys2.

Figure 25: make error on msys2



```
nt_protocol -las7050_osal_xenon_core -lbio_app_hrm -lams_hrm -lams_prv -lbio_app
_spo2 -lams_spo2 -lws2_32 -o as7050_sample_code_hrm.exe
make: gcc: No such file or directory
make: *** [Makefile:112: as7050_sample_code_hrm.exe] Error 127
```

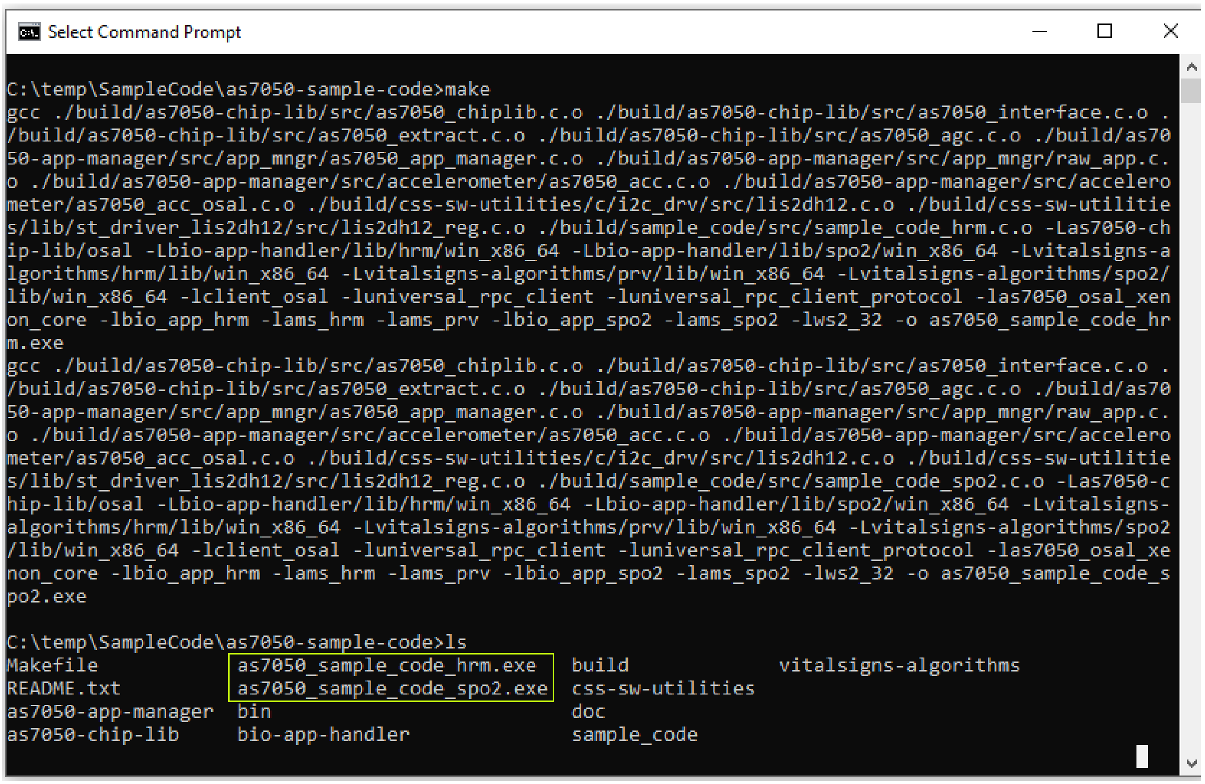
Figure 26: Build issue with msys2



```
ssao@N1300360 MSYS /
$ cd /C/temp/SampleCode/as7050-sample-code/

ssao@N1300360 MSYS /C/temp/SampleCode/as7050-sample-code
$ make
gcc ./build/as7050-chip-lib/src/as7050_chiplib.c.o ./build/as7050-chip-lib/src/a
s7050_interface.c.o ./build/as7050-chip-lib/src/as7050_extract.c.o ./build/as705
0-chip-lib/src/as7050_agc.c.o ./build/as7050-app-manager/src/app_mngr/as7050_app
_manager.c.o ./build/as7050-app-manager/src/app_mngr/raw_app.c.o ./build/as7050-
app-manager/src/accelerometer/as7050_acc.c.o ./build/as7050-app-manager/src/acce
lerometer/as7050_acc_osal.c.o ./build/css-sw-utilities/c/i2c_drv/src/lis2dh12.c.
o ./build/css-sw-utilities/lib/st_driver_lis2dh12/src/lis2dh12_reg.c.o ./build/s
ample_code/src/sample_code_hrm.c.o -Las7050-chip-lib/osal -Lbio-app-handler/lib/
hrm/win_x86_64 -Lbio-app-handler/lib/spo2/win_x86_64 -Lvitalsigns-algorithms/hrm
/lib/win_x86_64 -Lvitalsigns-algorithms/prv/lib/win_x86_64 -Lvitalsigns-algorith
ms/spo2/lib/win_x86_64 -lclient_osal -luniversal_rpc_client -luniversal_rpc_clie
nt_protocol -las7050_osal_xenon_core -lbio_app_hrm -lams_hrm -lams_prv -lbio_app
_spo2 -lams_spo2 -lws2_32 -o as7050_sample_code_hrm.exe
make: gcc: No such file or directory
make: *** [Makefile:112: as7050_sample_code_hrm.exe] Error 127

ssao@N1300360 MSYS /C/temp/SampleCode/as7050-sample-code
$ |
```

Solution: Using the Windows command prompt solves this.

Figure 27: Build successful on 'cmd'


Select Command Prompt

```
C:\temp\SampleCode\as7050-sample-code>make
gcc ./build/as7050-chip-lib/src/as7050_chiplib.c.o ./build/as7050-chip-lib/src/as7050_interface.c.o .
/build/as7050-chip-lib/src/as7050_extract.c.o ./build/as7050-chip-lib/src/as7050_agc.c.o ./build/as70
50-app-manager/src/app_mngr/as7050_app_manager.c.o ./build/as7050-app-manager/src/app_mngr/raw_app.c.
o ./build/as7050-app-manager/src/accelerometer/as7050_acc.c.o ./build/as7050-app-manager/src/accelero
meter/as7050_acc_osal.c.o ./build/css-sw-utilities/c/i2c_drv/src/lis2dh12.c.o ./build/css-sw-utilitie
s/lib/st_driver_lis2dh12/src/lis2dh12_reg.c.o ./build/sample_code/src/sample_code_hrm.c.o -Las7050-ch
ip-lib/osal -Lbio-app-handler/lib/hrm/win_x86_64 -Lbio-app-handler/lib/spo2/win_x86_64 -Lvitalsigns-a
lgorithms/hrm/lib/win_x86_64 -Lvitalsigns-algorithms/prv/lib/win_x86_64 -Lvitalsigns-algorithms/spo2/
lib/win_x86_64 -lclient_osal -luniversal_rpc_client -luniversal_rpc_client_protocol -las7050_osal_xen
on_core -lbio_app_hrm -lams_hrm -lams_prv -lbio_app_spo2 -lams_spo2 -lws2_32 -o as7050_sample_code_hr
m.exe
gcc ./build/as7050-chip-lib/src/as7050_chiplib.c.o ./build/as7050-chip-lib/src/as7050_interface.c.o .
/build/as7050-chip-lib/src/as7050_extract.c.o ./build/as7050-chip-lib/src/as7050_agc.c.o ./build/as70
50-app-manager/src/app_mngr/as7050_app_manager.c.o ./build/as7050-app-manager/src/app_mngr/raw_app.c.
o ./build/as7050-app-manager/src/accelerometer/as7050_acc.c.o ./build/as7050-app-manager/src/accelero
meter/as7050_acc_osal.c.o ./build/css-sw-utilities/c/i2c_drv/src/lis2dh12.c.o ./build/css-sw-utilitie
s/lib/st_driver_lis2dh12/src/lis2dh12_reg.c.o ./build/sample_code/src/sample_code_spo2.c.o -Las7050-c
hip-lib/osal -Lbio-app-handler/lib/hrm/win_x86_64 -Lbio-app-handler/lib/spo2/win_x86_64 -Lvitalsigns-
algorithms/hrm/lib/win_x86_64 -Lvitalsigns-algorithms/prv/lib/win_x86_64 -Lvitalsigns-algorithms/spo2
/lib/win_x86_64 -lclient_osal -luniversal_rpc_client -luniversal_rpc_client_protocol -las7050_osal_xe
non_core -lbio_app_hrm -lams_hrm -lams_prv -lbio_app_spo2 -lams_spo2 -lws2_32 -o as7050_sample_code_s
po2.exe

C:\temp\SampleCode\as7050-sample-code>ls
Makefile            as7050_sample_code_hrm.exe   build            vitalsigns-algorithms
README.txt          as7050_sample_code_spo2.exe  css-sw-utilities
as7050-app-manager  bin                          doc
as7050-chip-lib     bio-app-handler              sample_code
```

7 Revision Information

Changes from previous version to current revision v1.3	Page
Replaced resistor sample code with GSR sample code	
Updated for Application Manager and GSR-related interface changes	

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

8 Additional Documents

For further information, please refer to the following documents:

1. ams AG, AS7050 Application Manager Library Documentation.
2. ams AG, AS7050 ChipLib API Documentation.
3. ams AG, AS7050 Linux Sample Driver Documentation.
4. ams AG, AS7050 EVK Board Schematic Documentation

9 Legal Information

RoHS Compliant & ams Green Statement

RoHS Compliant: The term RoHS compliant means that ams AG products fully comply with current RoHS directives. Our semiconductor products do not contain any chemicals for all 6 substance categories plus additional 4 substance categories (per amendment EU 2015/863), including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes.

ams Green (RoHS compliant and no Sb/Br/Cl): ams Green defines that in addition to RoHS compliance, our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material) and do not contain Chlorine (Cl not exceed 0.1% by weight in homogeneous material).

Important Information: The information provided in this statement represents ams AG knowledge and belief as of the date that it is provided. ams AG bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ams AG has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ams AG and ams AG suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

Headquarters	Please visit our website at www.ams.com
ams AG	Buy our products or get free samples online at www.ams.com/Products
Tobelbader Strasse 30	Technical Support is available at www.ams.com/Technical-Support
8141 Premstaetten	Provide feedback about this document at www.ams.com/Document-Feedback
Austria, Europe	For sales offices, distributors and representatives go to www.ams.com/Contact
Tel: +43 (0) 3136 500 0	For further information and requests, e-mail us at ams_sales@ams.com