Freescale Semiconductor
Application Note

Document Number:AN4325
Rev. 1, 12/2012

Spirometer Demo with Freescale Microcontrollers

by: Jorge González

Contents

1 Introduction

The contents of this application note show how it is possible to use the Kinetis K50 and Flexis MM families of microcontrollers along with the Freescale Tower System to implement a device capable to quantify human respiration capacities, by measuring volumes and flow rates.

The results of the test are sent afterwards to a host computer installed with a graphical user interface (GUI) that shows a graph of the respiration process and the measured results. The data is sent using the Freescale USB stack.

This application note can be useful for those interested in spirometry and for developers of medical devices.

2 Spirometry fundamentals

This section provides a general explanation of the spirometry parameters, and the advantages of being able to measure such parameters. It also gives a general explanation how the spirometer takes flow and volume measurements.

2.1 Spirometry

Spirometry refers to a series of simple tests of a person’s respiratory capacities. Spirometry takes measurements of the quantity of air inhaled and exhaled by the lungs during a certain period of time to determinate the pulmonary capacity. The device used with this purpose is called a spirometer. These measurements are useful when it comes to checking pulmonary function, since results are valuable in diagnosing diseases such as pulmonary fibrosis, asthma, cystic fibrosis and COPD (chronic obstructive pulmonary disease). When the spirometer is a pneumotachograph type, the process to perform a spirometry test usually involves the patient breathing through a hose or tube in which one of the ends contains a sensor that quantifies the air flow. Figure 1 shows an example of a basic spirometer depicting the general idea about spirometry tests.

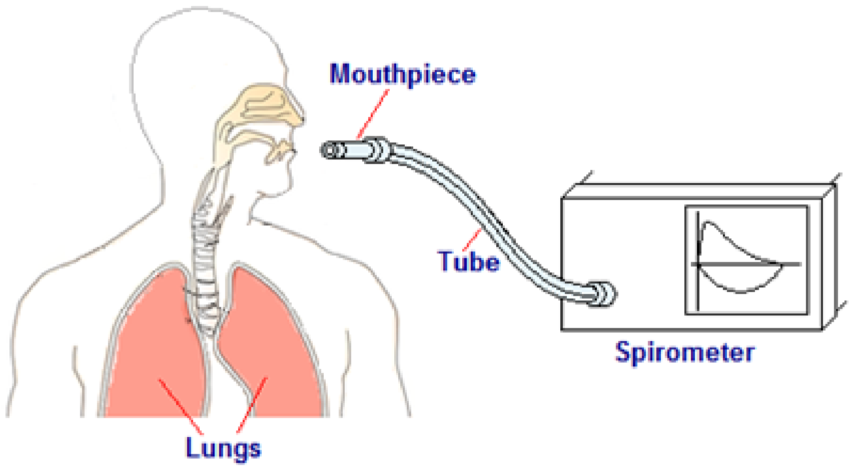


Figure 1. Basic spirometer system

In Figure 1, the basic components of a spirometer are the mouthpiece, a hose or tube, and an electronic device to measure flows and calculate spirometry parameters. The process to perform a test involves the patient to breathe using the mouthpiece to generate an air flow through the tube, which allows its conversion, by a sensor to an electrical signal.

2.2 Common spirometry parameters

The most usual and important spirometry parameters are mentioned and described below:

- Tidal Volume (TV)—The amount of air imhaled or exhaled during a single breath without forced conditions.
- Inspiratory Reserve Volume (IRV)—The maximum additional air that can be inhaled at the end of a normal inspiration.
- Expiratory Reserve Volume (ERV)—Refers to the maximum volume of air that can be exhaled at the end of a normal expiration.
- Vital Capacity (VC)—The maximum amount of air that can be expelled from a person’s lungs after a maximum inspiration. The vital capacity is equal to the sum of IRV, ERV, and TV.
- Forced Vital Capacity (FVC)—The volume of air that can be blown out by a person at a maximal speed and effort after a full inspiration.
- Forced Expiratory Volume in 1 second (FEV1)— Represents the maximum volume of air that can be exhaled in a forced way in the first second, after taking a deep breath.
- Forced Inspiratory Vital Capacity (FIVC)—The maximum air volume that can be inhaled.
- Peak Inspiratory Flow (PIF)—The forced maximum flow that can be achieved during inhalation.
- Peak Expiratory Flow (PEF)—The maximum air flow that can be forced during exhalation.

The first four parameters are taken under normal breathing conditions, while the last five are forced tests, (notice the names). Other measurements related with the lung volumes are the Functional Residual Capacity (FRC) and Residual Volume (RV), but these parameters cannot be assessed by spirometry since they depend on the volume of air that stays inside the lungs, and a spirometer is not capable of measuring it.

Figure 2 is a graph that shows the average values for the parameters FVC and FEV1 depending on sex and age.

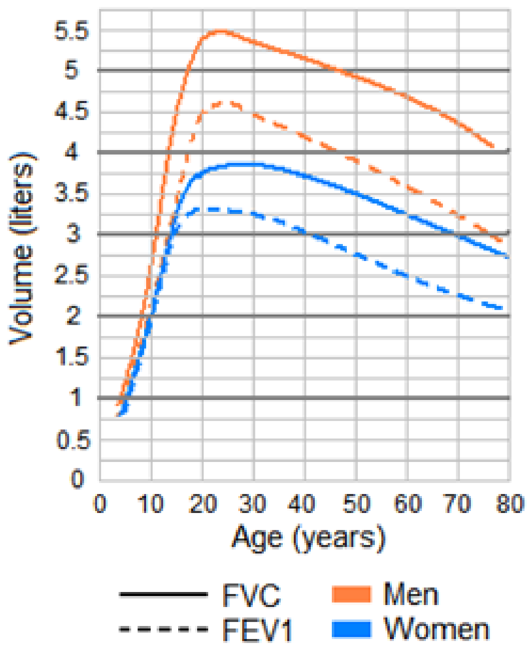


Figure 2. Average values for FVC and FEV1

2.3 Flow-volume loop

When a spirometry test is performed, the results are shown in a graph called a flow-volume loop or spirogram. This graph represents the air volume in liters on the X-axis and the flow rate in liters per second on the Y-axis. Inspiration results are plotted below the horizontal axis while expiration data are plotted above the horizontal axis. The flow-volume loop can be used to pre-diagnose airway obstruction diseases by analyzing the shape and magnitudes of the graph. An example of a spirogram can be seen in Figure 3.

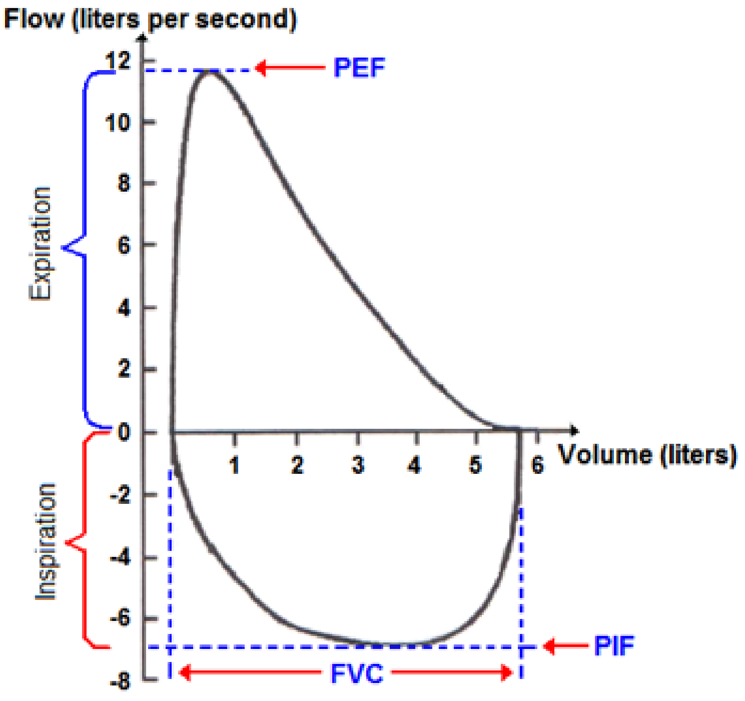


Figure 3. Flow-volume loop

2.4 Flows and volumes measurement

The method to measure flow in this application is based on the fluid dynamics laws. One of these laws is the Venturi effect according to which when a fluid passes from a wider to a narrower section of a pipe, the pressure of the fluid reduces while the velocity increases. The velocity and pressure of the fluid change to satisfy mass conservation is regulated by the next equation, called the “Venturi effect equation”:

$$P_1 - P_2 = \frac{d}{2}\left(v_2^2 - v_1^2\right)$$

Equation 1.

Where P1 and P2 are pressures, p is the density of the fluid and v1-v2 are the velocities before and after entering the pipe constriction respectively.

An example of the Venturi effect is shown in Figure 4.

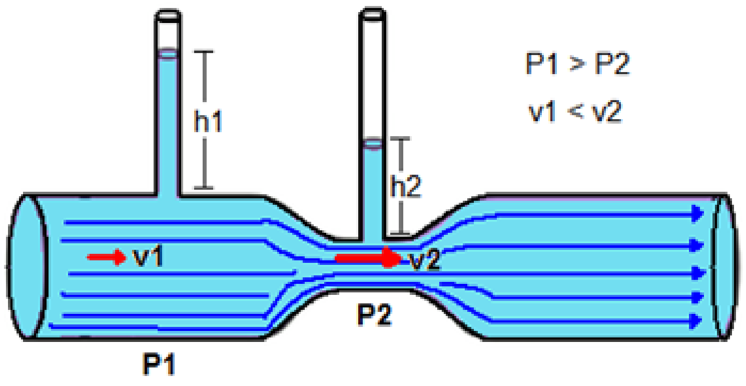


Figure 4. Venturi effect

There are different flow sensors whose characteristics vary. The producer of the sensor provides information such as maximum flows and resistance to the pass of the air.

The flow sensor used in this case to implement the spirometer demo is similar to the Venturi pipe shown previously. The sensor is shown in Figure 5.

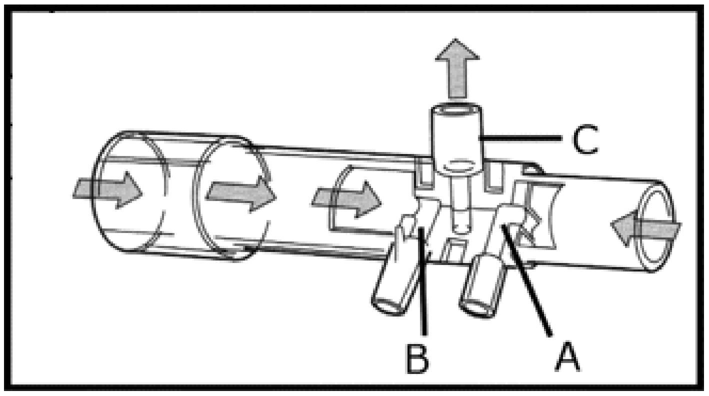


Figure 5. Flow sensor

Port C of the sensor is not used in this application. When connected to pressure sensors, ports A and B measure total or static pressure, depending on the direction of the flow. The difference between the two of them represents the dynamic pressure, which is proportional to the velocity of the fluid passing through the tube.

Considering Q = v_1A_1 = v_2A_2 where Q is flow rate, v is velocity and A is area, there are some operations that can be done:

$$Q = v_1 A_1$$

Equation 2.

$$P_1 - P_2 = \Delta P = \frac{d}{2}\left(v_2^2 - v_2^2\right)$$

Equation 3.

$$v_1 = \sqrt{v_2^2 - \frac{2(P_1 - P_2)}{d}}$$

Equation 4.

$$Q = A_1\sqrt{v_2^2 - \frac{2(\Delta P)}{d}}$$

Equation 5.

Notice with Equation 5 on page 5, it is possible to find the flow rate in the sensor if the differential pressure between the sections of the different diameters and the response of the tube to the changes in velocity, is known.

For the prototype and flow sensor described in this application note, the differential pressure is measured with a special sensor that can be used to quantify it. This application was calibrated with a machine that insufflates a specific quantity of air within a determined period of time, this means there is constant flow rate. Knowing the flow rate that the machine is insufflating and the result that the pressure sensor is delivering, it is possible to calibrate and find a factor to convert from the differential pressure to the flow rate.

3 Hardware description

This chapter presents an explanation about the hardware needed to implement a spirometer demo, and the advantages of using Kinetis K50 and Flexis MM microcontrollers.

3.1 Spirometer block diagram

The MED-SPI demo is formed by the blocks shown in Figure 6.

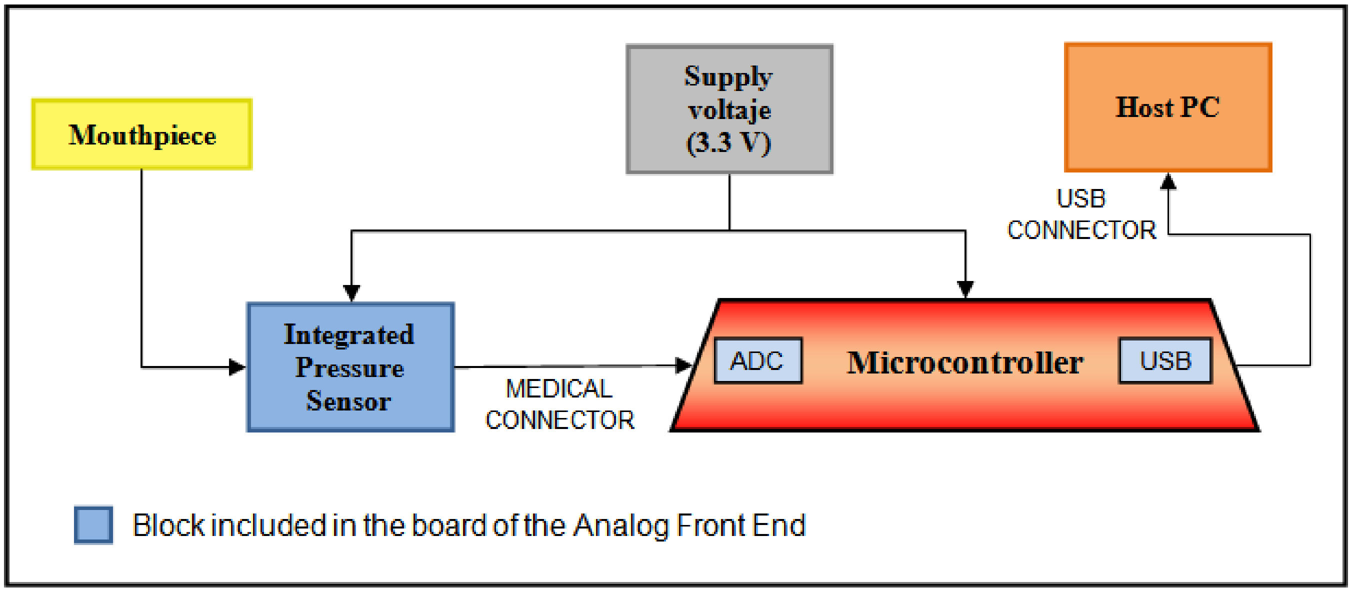


Figure 6. Spirometer system block diagram

3.2 Respiration set

The first step to implement a spirometer is the acquisition of the signal, in this case coming from the breathing process, so there is a way to detect the air volumes and flow rates involved in the process. This can be achieved using a special mouthpiece that consists of a respiration tube and a hose with two air ways to transmit the air pressure to a pressure sensor that handles the conversion of the signal. During a test, the patient breathes using the tube, which contains a flow restriction mechanism that enables the air to flow just through one of the air ways in the hose, depending on whether the person is inhaling or exhaling. The mouthpiece can be observed in Figure 7.

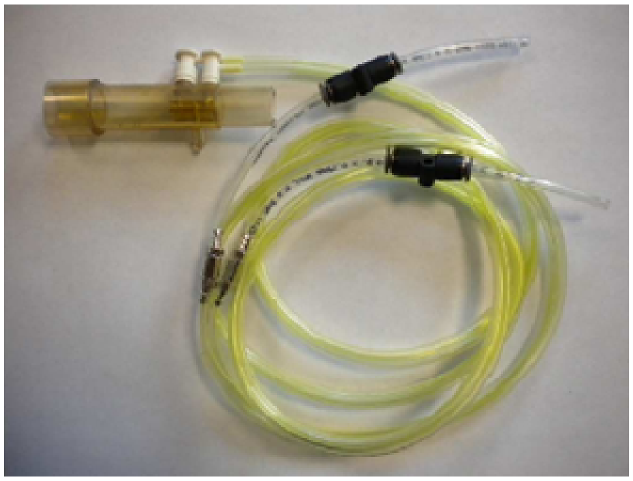


Figure 7. Mouthpiece

3.3 Pressure sensor

A key part when implementing a spirometer is a transducer to convert the air flow into an electrical signal, which can be then processed by electronic means. To achieve this, the MED-SPI demo uses an integrated silicon pressure sensor from Freescale, the MPXV7025DP. It is a differential pressure sensor that delivers an analog output voltage proportional to the applied differential pressure on the sensor.

The key features of this pressure sensor are listed below:

- Pressure range— -25 to 25 kPa (-3.6 to 3.6 psi).
- Output—0.2 to 4.7 V.
- 5.0 % Maximum error over 0° to 85°C.
- Ideally suited for microprocessor or microcontroller systems.
- Temperature compensated over -40° to +125°C.

The sensor is embedded in the AFE board. It contains a pair of pipes in its shell to allow a different pressure on each one of them and being capable of detect the direction of the air flow. Both the sensor and the board are showed in Figure 8.

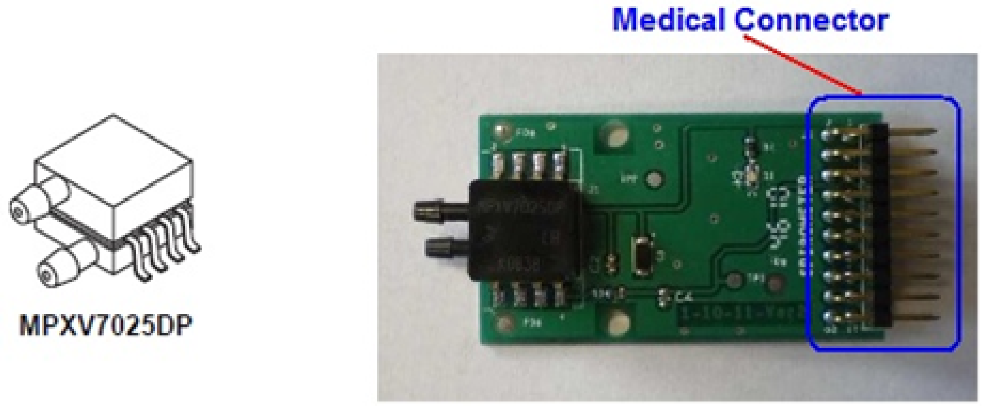


Figure 8. Pressure sensor and AFE

For the pressure sensor to work properly, it has to be connected as shown in Figure 9.

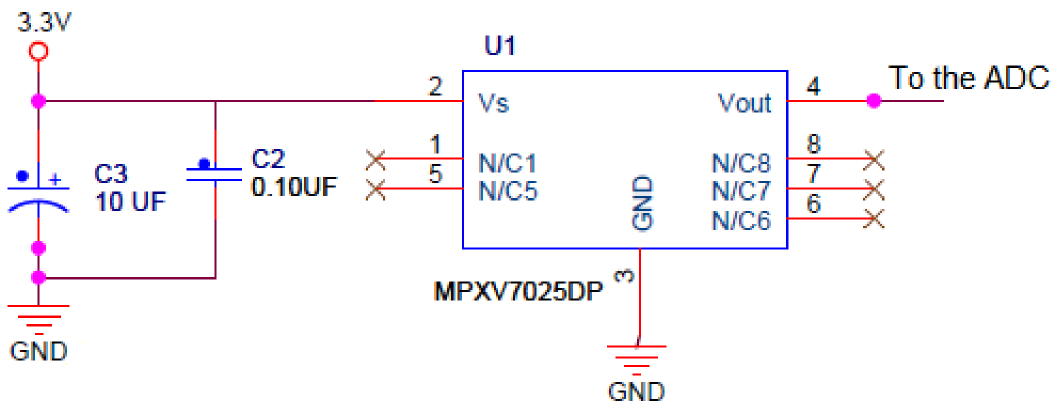


Figure 9. Pressure sensor connections

3.4 Microcontroller

The microcontroller is a fundamental piece of the design. It performs the principal functions of the system, from sampling and converting the signal delivered by the pressure sensor to calculating the spirometry parameters and sending data and results to the GUI.

The Freescale microcontrollers that are used to implement this demo are the Kinetis K53, the MCF51MM256, and the MC9S08MM128.

The Kinetis K50 microcontroller has the following features and peripherals in its integrated measurement engine:

- Ultra low-power operation
- 2x OPAMP
- 2x TRIAMP
- 2x 12-bit DAC
- 2x 16-bit SAR ADC, up to 31 channels with programmable gain amplifiers (PGA)
- Programmable Delay Block (PDB)
- I2C
- USB connectivity
- ARM® Cortex™-M4 core with DSP (Digital Signal Processor) instructions

The features of the Flexis MM microcontrollers (MCF51MM256 and MC9S08MM128) are the following:

- ColdFire V1 and HCS08 cores respectively
- Ultra low-power operation -2 operational amplifiers (OPAMP)
- 2 transimpedance amplifiers (TRIAMP)
- 16-bit SAR analog-to-digital converter (ADC), 4 differential channels and up to 12 external single-ended channels.
- 12-bit digital-to-analog converter (DAC)
- Programmable Delay Block (PDB)
- Inter-integrated circuit (I2C)
- Universal Serial Bus connectivity (USB)
- Multiply-Accumulate Unit (MAC only in MCF51MM)

The activities assigned to the microcontroller module are the following:

- Take samples and convert the analog voltages from the pressure sensor into digital values using the internal ADC.
- Receive requests from the Host PC via USB communication to start and stop measurements; it also sends both confirmation and data.
- Quantify and detect the current state of the breathing process, to check if the patient is inhaling or exhaling and the limits in which the measurement must be paused or stopped.

- Perform spirometry calculations and send volume and flow data to the GUI so it can plot them on the screen of the PC.
- Send the final results to be shown on the GUI.

4 Software architecture

This chapter has a general description of the software for the MED-SPI demo. It explains the purpose of the most important routines, and the operation of the system based on state machines and the Freescale USB Stack.

NOTE

Software corresponding to this application note (AN4325SW.zip) is available on **freescale.com** for TWR-K53, TWR-MCF51MM256 and TWR-S08MM128.

4.1 Software block diagram

The block diagram representing the general scheme of the software is shown in Figure 10.

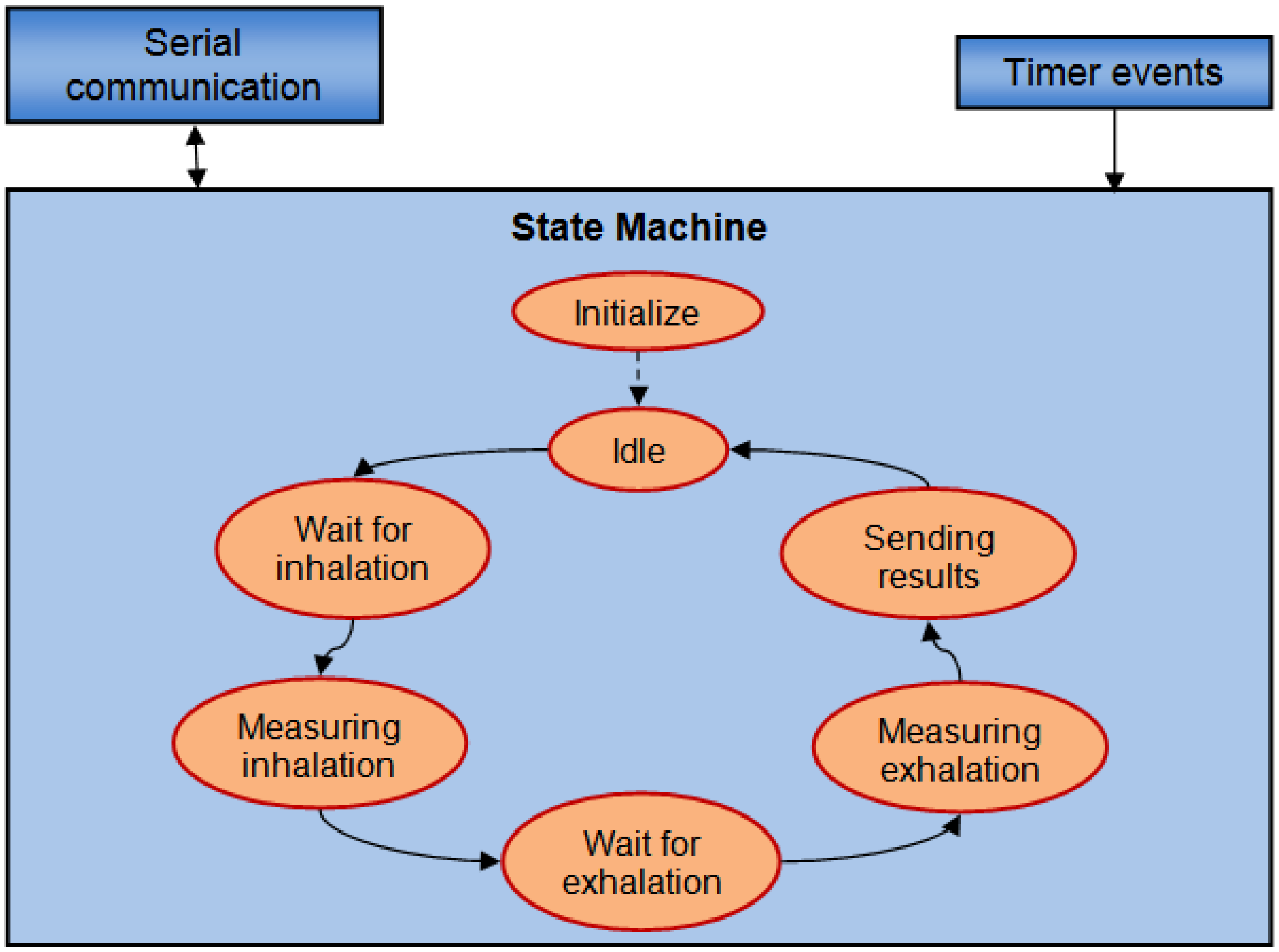


Figure 10. Software block diagram

4.2 Freescale USB stack

Freescale provides a solution for medical applications developers to have a way to communicate the microcontrollers with a Host PC through a USB interface. That solution is the Freescale's USB stack with personal healthcare device class (PHDC), which makes possible to connect medical devices with the PC. It is a portable source code to be implemented in many applications and provide them with an interface to a host device.

In this specific application, the MCU acts like a communication device class (CDC). The graphical user interface (GUI) to be installed in the computer controls the actions of the spirometer demo with some data packet transactions. The three basic types of packets are listed and explained:

- REQ Packet—The host sends a REQUEST packet either to start or to stop a measurement.
- CFM Packet—After the device receives a REQ packet, a CONFIRMATION packet is sent to the host to notify that the command is valid or that there was an error.
- IND Packet—Once either the graph data buffers or the results buffer is full, the microcontroller sends an INDICATION packet and also the data.

A general idea of the USB interface is shown in Figure 11. The communication protocol is explained with more details in Communication protocol.

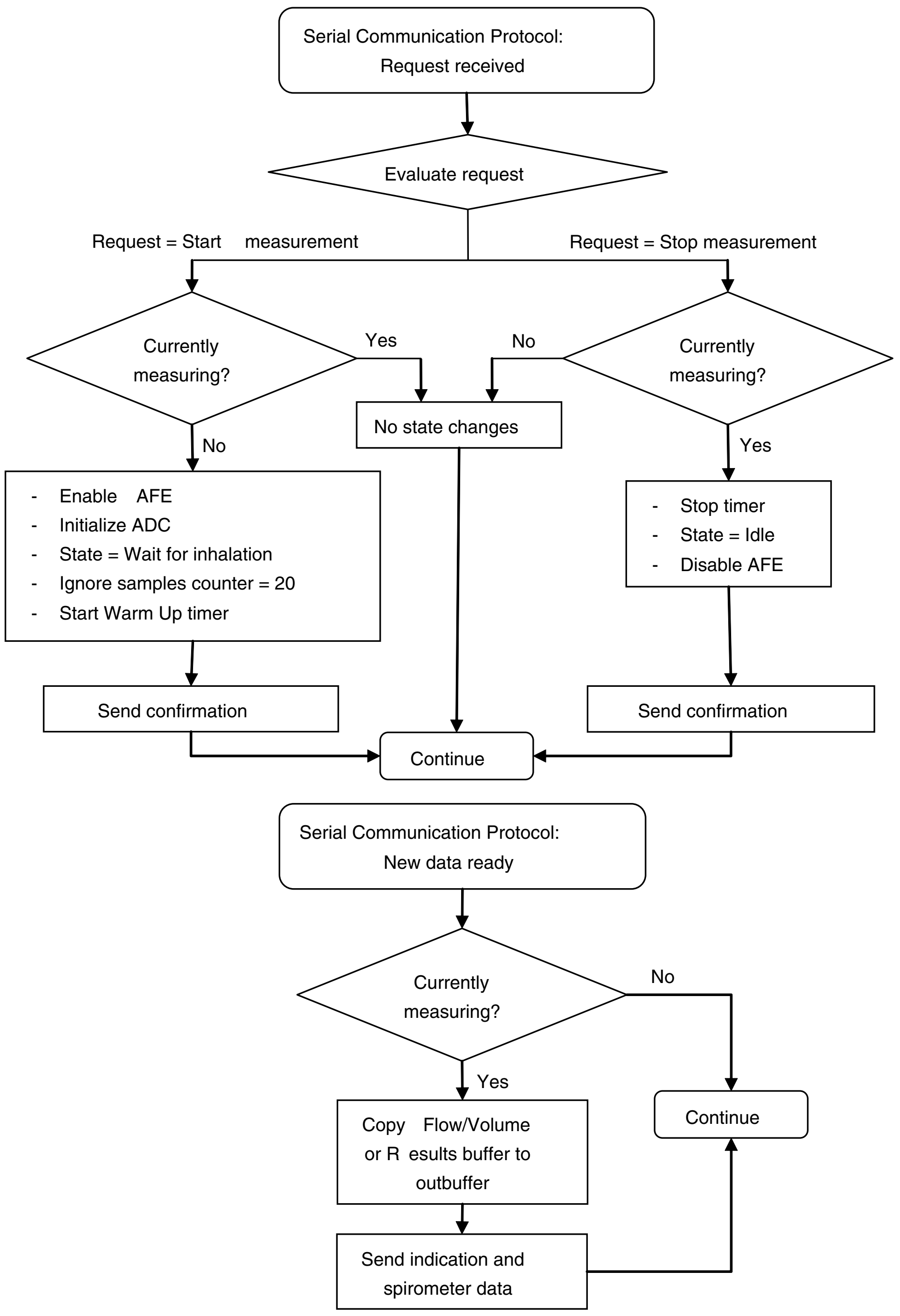


Figure 11. Communication with Host PC

4.3 Spirometer initialization

There are some peripherals of the microcontroller that have to be initialized with some configurations. The flow chart in Figure 12 shows the general initialization stages.

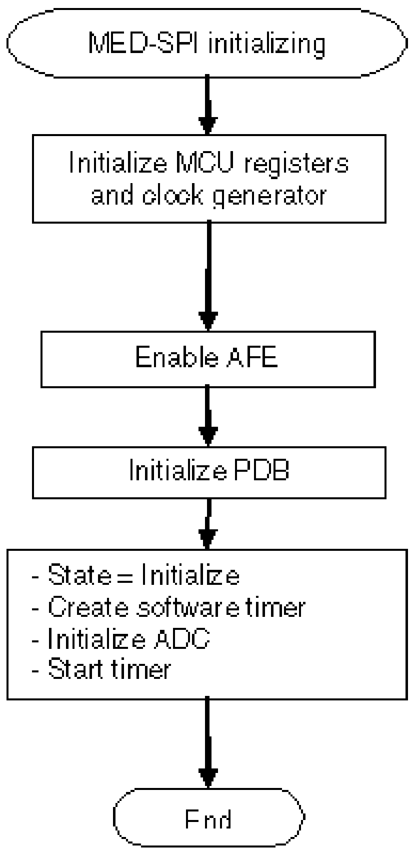


Figure 12. System initializing

4.4 Sampling times and process

The sampling rate of the MED-SPI demo has to be set to 1 sample of the pressure signal every 10 milliseconds. The time intervals are set using the programmable delay block (PDB module) of the microcontroller. The PDB is configured to generate an interrupt every millisecond. With each interrupt service routine (ISR), a variable is incremented to save the number of interrupts generated. To achieve the sampling rate, 10 interrupts have to occur before taking a new sample. This is implemented with a software timer, which through a periodic task, checks the number of interrupts generated by the PDB, and verifies if the count is equal to the sampling period (in this case 10). When the timer elapses and the sampling periods match, an event is executed to take the current analog value of pressure signal and convert it to a digital value, indicating that there is a new spirometer sample available. Figure 13 are the flow charts illustrating the functioning of the timing process.

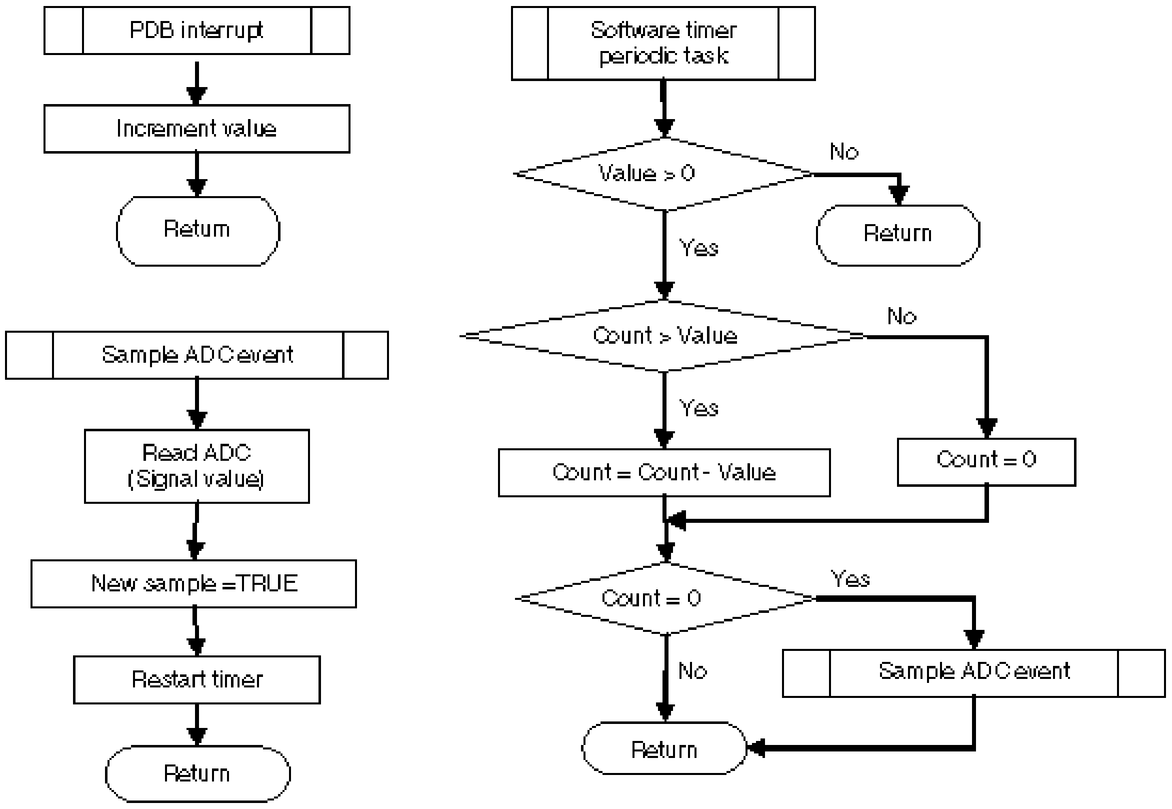


Figure 13. Sampling process

4.5 State machine

An outstanding characteristic of the software is the use of state machines, whose functionality makes it possible to emulate parallelism by switching between several tasks and remembering the state that was being executed before changing or the next state to be executed. This kind of processing gives efficiency to the system and forces it to follow a specific sequence of states. For example, the microcontroller is continuously sending data to the PC while a spirometry test is in process. It is easier to understand the advantages of state machines with Figure 14.

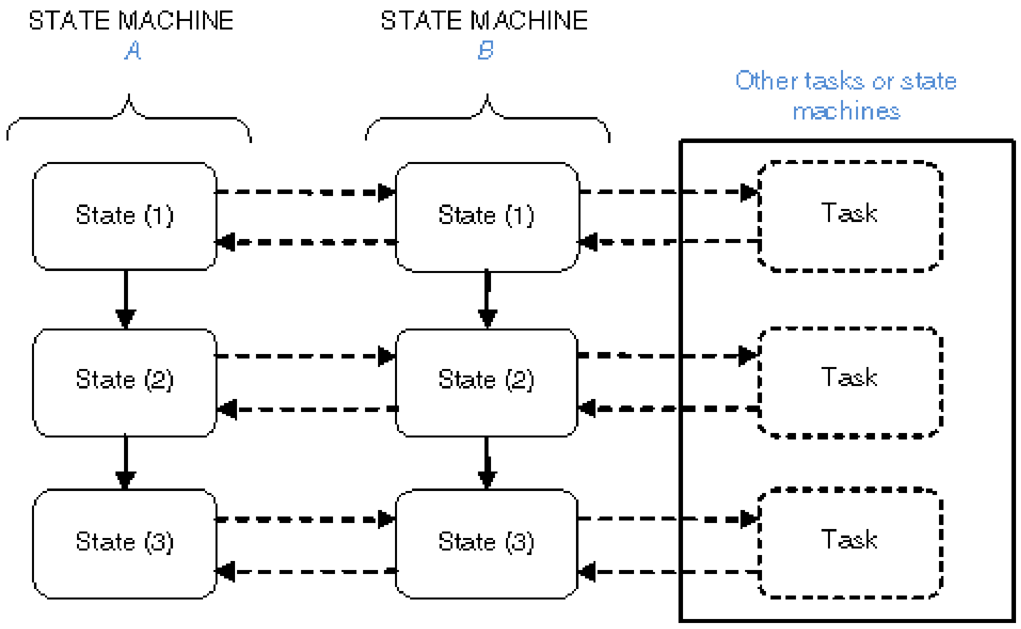


Figure 14. State machines

4.5.1 State—Initialization

The first state to be performed in the state machine is initialization. It acquires an average of the offset value, meaning a baseline. At the beginning of this state, there is an inactive interval of time of 50 milliseconds for the pressure sensor to warm up, and the next 19 samples of the signal are ignored to stabilize the moving average process which is used as a filter for the signal AD conversions. Afterwards, 200 samples are used to calculate the average offset.

This state is run only once when turning on the system, but afterwards, the offset value is held for every measurement start. The flow chart for this state can be seen in Figure 15.

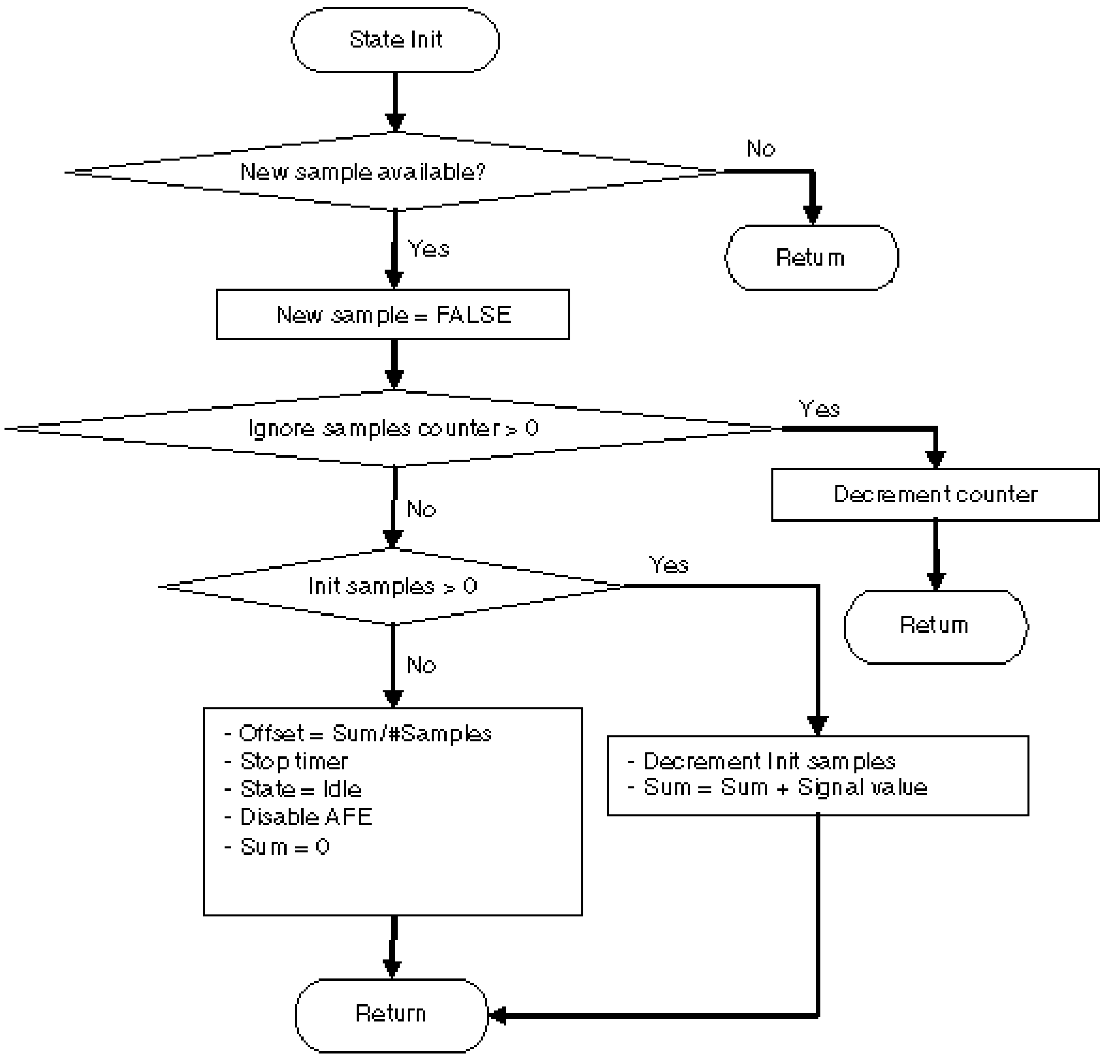


Figure 15. State Init flow chart

4.5.2 State—Wait for inhalation

After receiving a request from the host PC to start a new spirometry measurement, the device has to change its state from idle to wait for inhalation. In this state, the microcontroller is continuously checking the value of the signal and comparing it with a threshold value, to know when the patient starts inhaling. This state, just like the initialization state, ignores some of the first samples to wait for stabilization. The flow chart that corresponds to this state of the device is Figure 16.

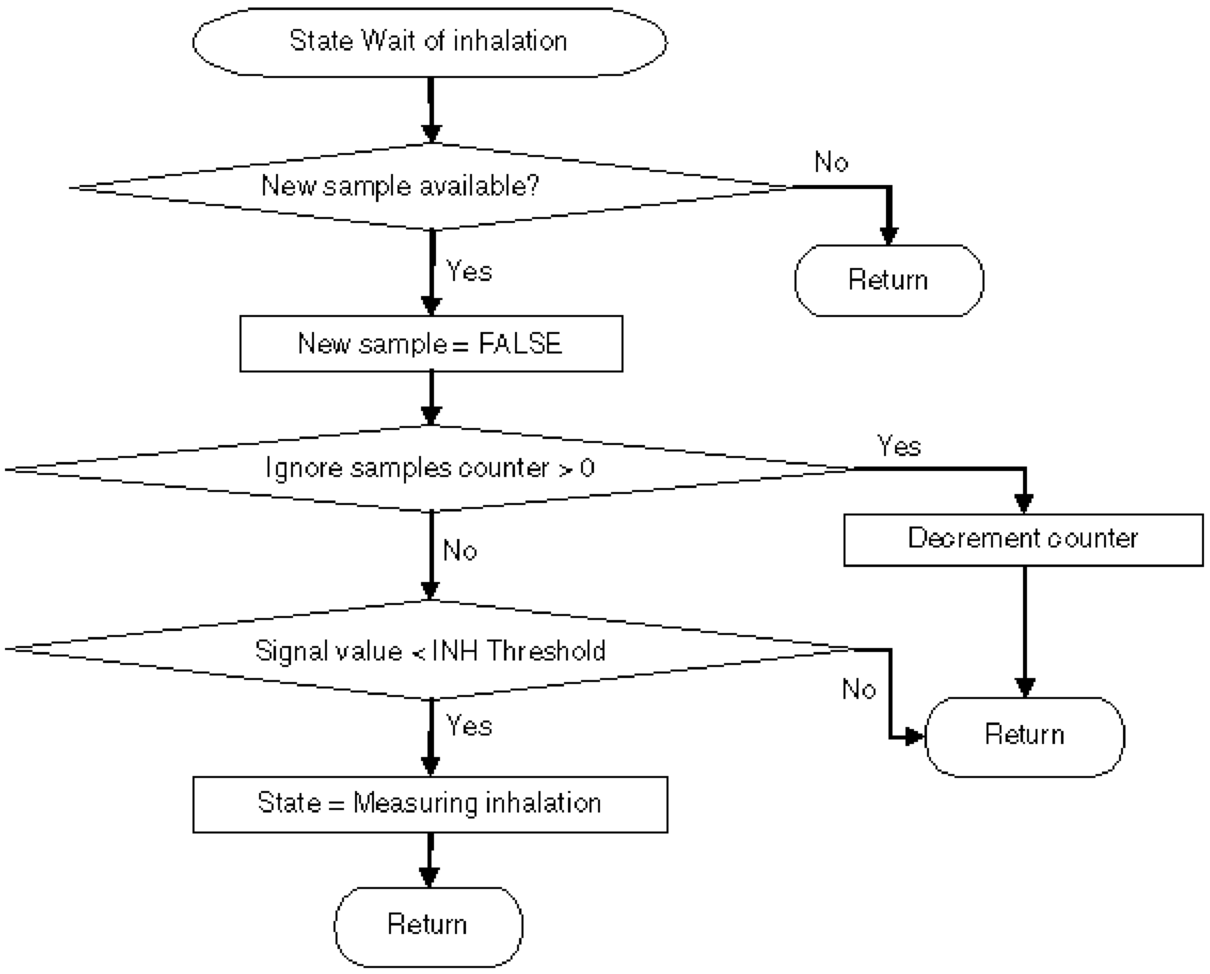


Figure 16. State Wait for inhalation flow chart

4.5.3 State—Measuring inhalation

After the result of the ADC is less than the inhalation threshold, the next state taken by the state machine is the measuring inhalation state. This state handles data processing while the patient is inhaling. With each new sample, the microcontroller calculates and updates the spirometry parameters, including flow and volume, which are sent to the host PC via USB communication. The value delivered by the ADC is continuously compared with the baseline; this is to know when inhalation has finished. Figure 17 is the flow chart.

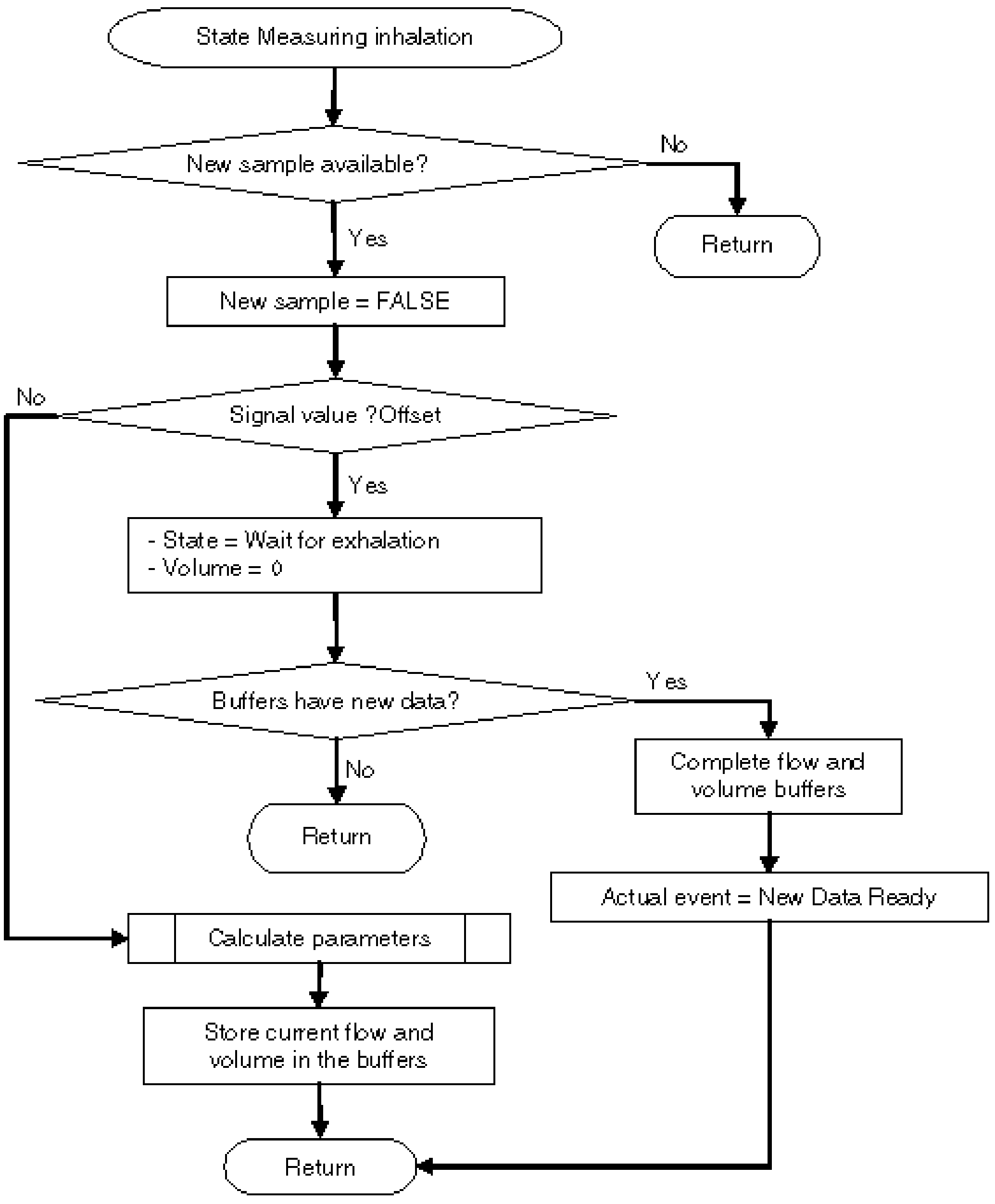

State Measuring inhalation
New sample available?
No
Return
Yes
New sample = FALSE
Signal value ?Offset
No
Yes
- State = Wait for exhalation
- Volume = 0
Buffers have new data?
Yes
Complete flow and volume buffers
No
Return
Actual event = New Data Ready
Calculate parameters
Store current flow and volume in the buffers
Return

Figure 17. State measuring inhalation flow chart

4.5.4 State—Wait for exhalation

When the microcontroller detects that the patient is no longer inhaling, it executes the next step in the state machine, which is wait for exhalation. This state is to wait for the signal to reach an exhalation threshold value above the baseline. When that occurs, a variable called Data_type changes to indicate to the GUI that the next information packets are exhalation values. There is no spirometer data transfer in the meantime. The duration of this state is short, because the patient needs to exhale right after maximum inhalation. A graphical description of this state can be seen in Figure 18.

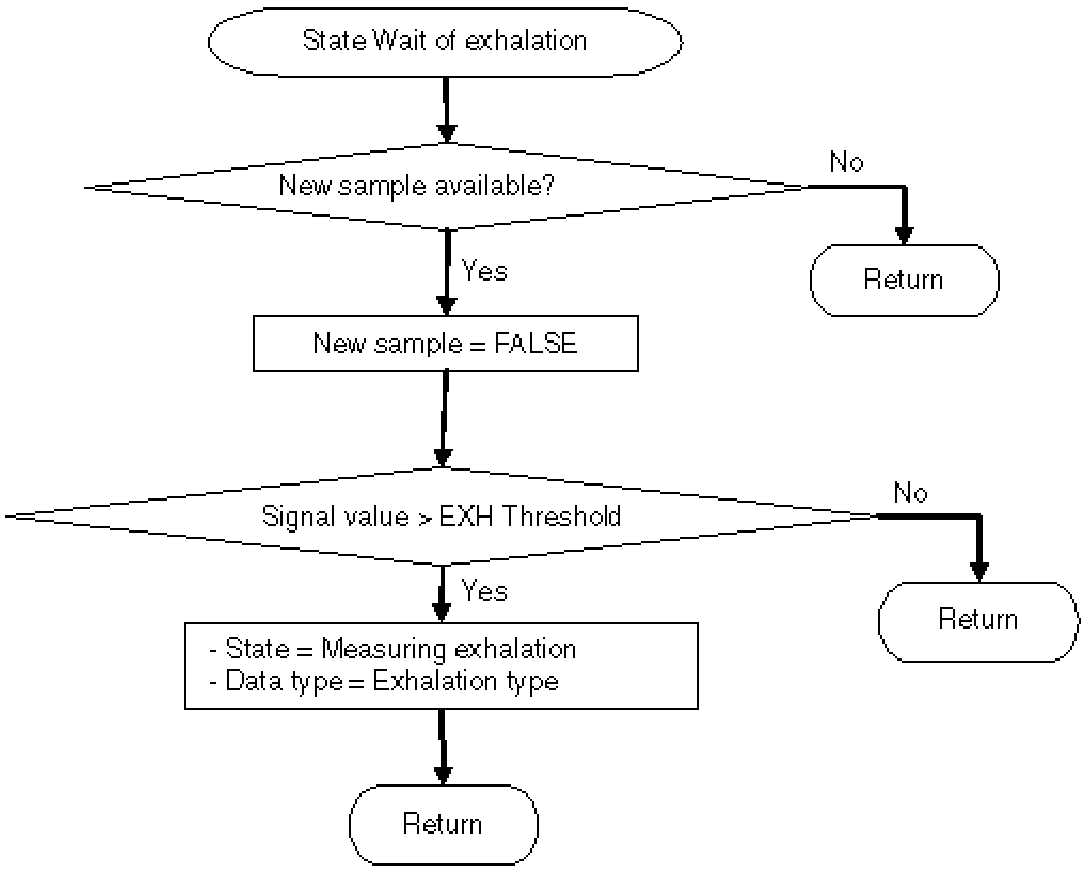


Figure 18. State Wait for exhalation flow chart

4.5.5 State—Measuring exhalation

The measuring exhalation state is analogous to the measuring inhalation state. The MCU calculates the exhalation parameters, filling the data buffers and comparing the signal with the baseline to know when the patient is no longer exhaling. Spirometer data is also sent to the computer during this state. When the signal value is equal to the baseline; the state changes to the sending results state. Flow chart for this state is shown in Figure 19.

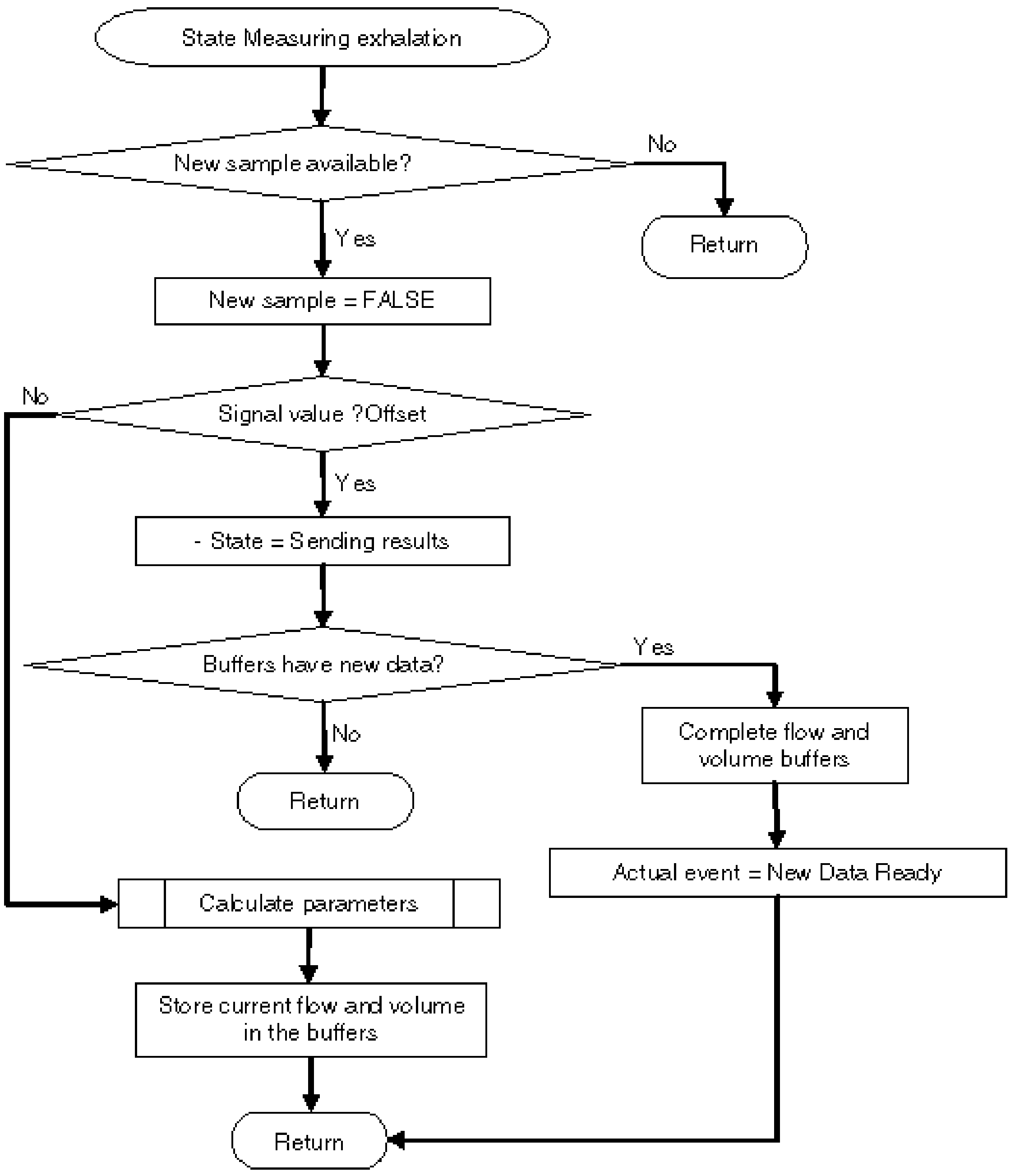

State Measuring exhalation
New sample available?
No
Return
Yes
New sample = FALSE
No
Signal value ?Offset
Yes
- State = Sending results
Yes
Buffers have new data?
Complete flow and volume buffers
No
Return
Actual event = New Data Ready
Calculate parameters
Store current flow and volume in the buffers
Return

Figure 19. State Measuring exhalation flow chart

4.5.6 State—Sending results

Once the end of exhalation is detected, the next state is the sending results state. It consists of filling the results buffer with the final results obtained with the calculation routines, and generates an event to send that information to the computer. Afterwards, the measurement is aborted by stopping the timer, changing the state to idle, and turning the MED-SPI board off. Figure 20 shows the flow chart.

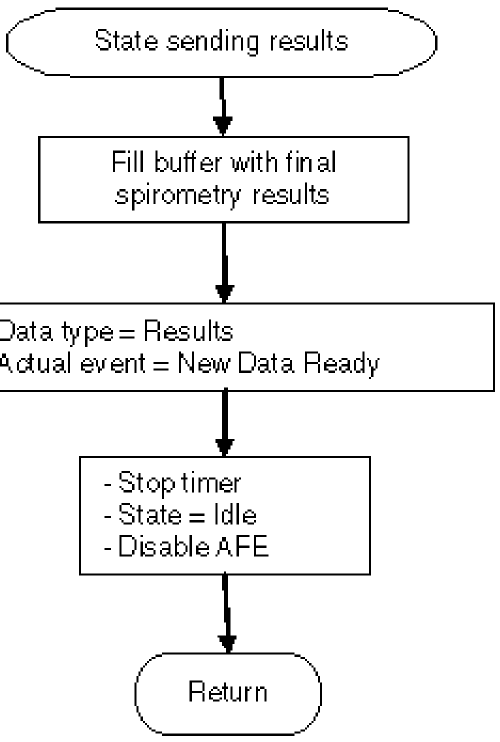


Figure 20. State Sending results flow chart

4.6 Calculating parameters

This section provides a summary of the formulas used to calculate the spirometry parameters. These formulas are embedded in the MCU software.

With each new sample taken by the ADC, the next calculations are performed:

If ADC Value > Signal offset:

$$\text{FlowX} = (\text{ADCValue} - \text{Signaloffset}) * \text{Conversionfactor}$$

If ADC Value < Signal offset:

$$\text{FlowX} = (\text{Signaloffset} - \text{ADCValue}) * \text{Conversionfactor}$$

If (FlowX >4.7), then *flowfactor = 0*

If (FlowX ≤4.7), then *flowfactor = (4.7–FlowX) / 2.7*

*Instant flow = FlowX +(FlowX*flowfactor)*

*Instant volume = Instant flow * sampling period*

Volume = Volume + instant volume

- ADC Value—The value read from the ADC.
- Conversion factor—This is a factor that converts the pressure measured with the sensor to flow units. It depends on the mouthpiece used. Using the sensor mentioned in Spirometry, this factor is 0.0136.
- Signal offset—The baseline of the measurement. This value corresponds to an air flow of 0 (no differential pressure on the sensor).
- Sampling period—The time between consecutive samples (seconds).
- Instant flow—The corresponding instantaneous flow of the current sample (liters/second).

- Instant volume—The volume of air acquired by the spirometer during the current sample (liters).
- Volume—The sum of instant volumes obtained with each measurement.

The formulas to calculate the spirometry parameters are shown:

PIF = The maximum instant flow detected during inhalation

PEF = The maximum instant flow detected during exhalation

$$FIVC = \sum_{1}^{\#INsamples} INSTANTvolume$$

$$FVC = \sum_{1}^{\#EXsamples} INSTANTvolume$$

$$FEV1 = \sum_{1}^{\#1Ssamples} INSTANTvolume$$

FET (Forced Expiratory Time) = #EXsamples * sampling period

5 Getting started and running the MED-SPI demo

This section of the document explains how to start the spirometer demo and use the GUI, available on the Freescale website.

It was already mentioned that the GUI communicates with the microcontroller through the USB port, using the Freescale USB stack that contains an algorithm to create a virtual serial port, serving as the communication channel to send and receive data. For more information about the Freescale USB stack with PHDC, see **freescale.com.**

5.1 Hardware settings

This part explains the steps needed to follow to run the MED-SPI demo with the microcontroller modules. Depending on the choice, the user can refer to either the Kinetis K53 or Flexis MM instructions given in the following subsections.

5.1.1 MED-SPI demo with Kinetis K53

The necessary steps when using a Kinetis K53 microcontroller are:

1. Download the software (AN4325SW.zip) for the demo and the medical GUI, from **freescale.com**.
2. Download and install the IAR embedded workbench from the IAR webpage, **iar.com**. It is used to program the MCU, because the software was developed using that platform.
3. Assemble all the pieces of the Tower System. Primary and secondary sides of all the boards have to match the respective primary and secondary elevators. Figure 21 shows all the boards that have to be connected to implement the system.

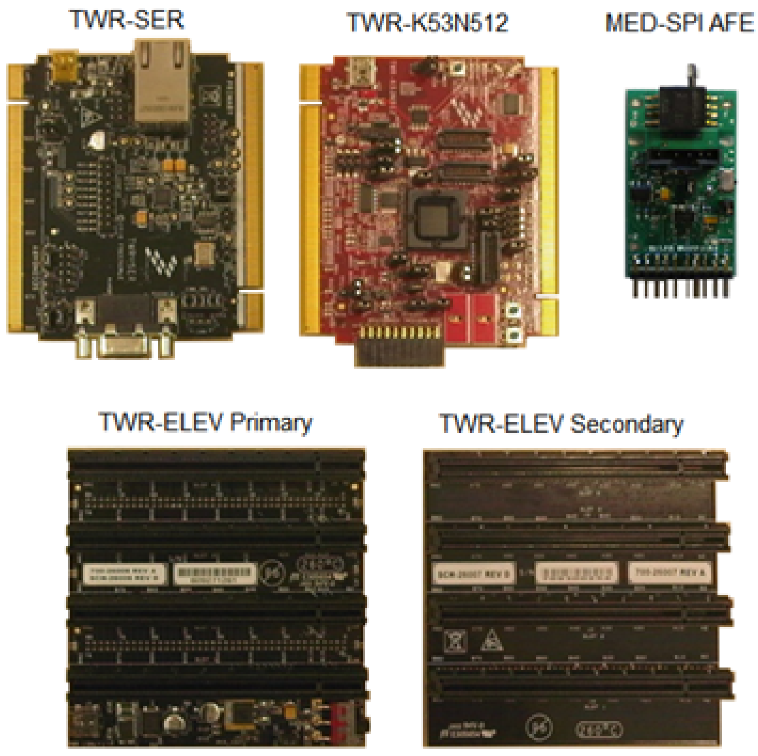
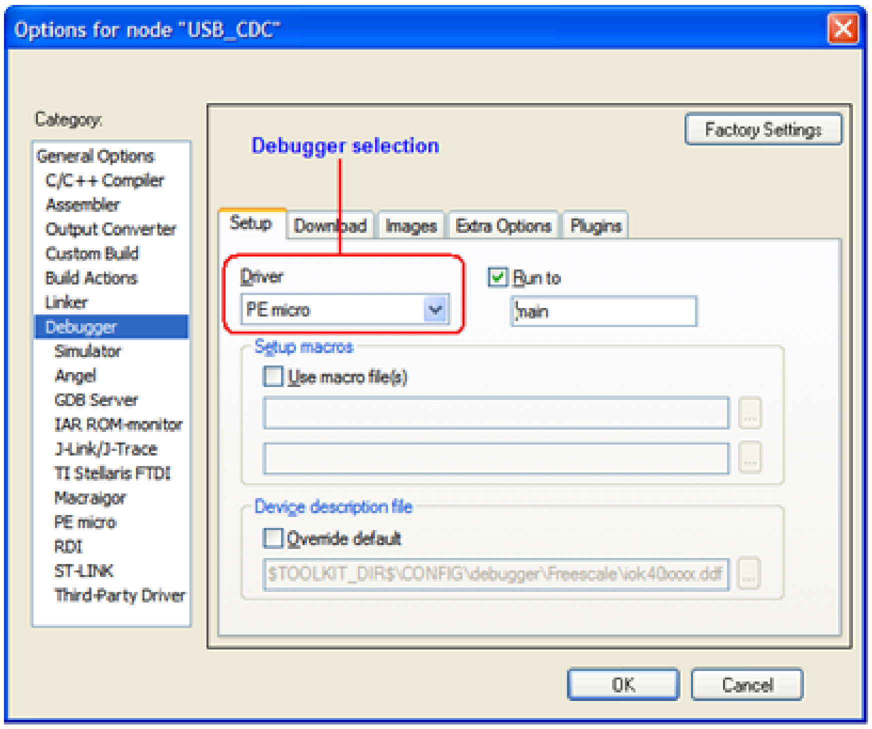


Figure 21. Tower System components

4. With the IAR embedded workbench, open the project of the demo contained in the software folder. The name of the project is MED-SPI K53.eww, and locate can find it in the following path, *MED-SPI_K53\app\cdc\iar_ew\kinetis \MED-SPI K53.eww*.
5. Using a USB cable, connect the MCU board to the host PC. The computer must recognize the Open Source BDM debug port. After the OSBDM has been recognized, install the driver in the IAR folder.
6. The selected debugger has to be PE micro. Verify this in the project options panel (Project > Options> Debugger) as shown in Figure 22.



Driver: PE micro

[✓] Run to: main

$TOOLKIT_DIR$\CONFIG\debugger\Freescale\iok40xxxx.ddf

Figure 22. Selecting PE micro debugger

7. The next step is to download the software to the microcontroller. To do this, click the “Download and debug” icon. Figure 23 shows the IAR screen.

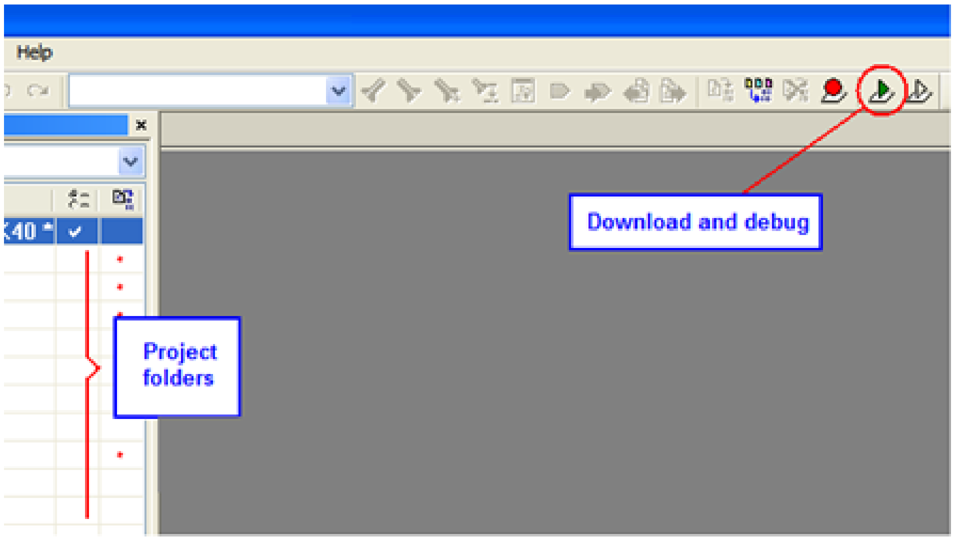


Figure 23. Programming the microcontroller

8. Once the Kinetis microcontroller is programmed, see Graphical user interface to follow the instructions related to the GUI.

5.1.2 MED-SPI demo with Flexis MM

The following steps have to be executed when using the TWR-MCF51MM256 or TWR-S08MM128 to run the MED-SPI demo:

1. Enter the Freescale webpage, **freescale.com**. Download the software and the GUI.
2. If it is not already done, install the CodeWarrior v6.3 on the computer and the latest service pack for MCF51MM256 (if necessary); both are available online.
3. The modules shown in Figure 24 correspond to the Tower System modules and the AFE board. Connect them and ensure the primary and secondary elevators match the respective sides of the tower boards.

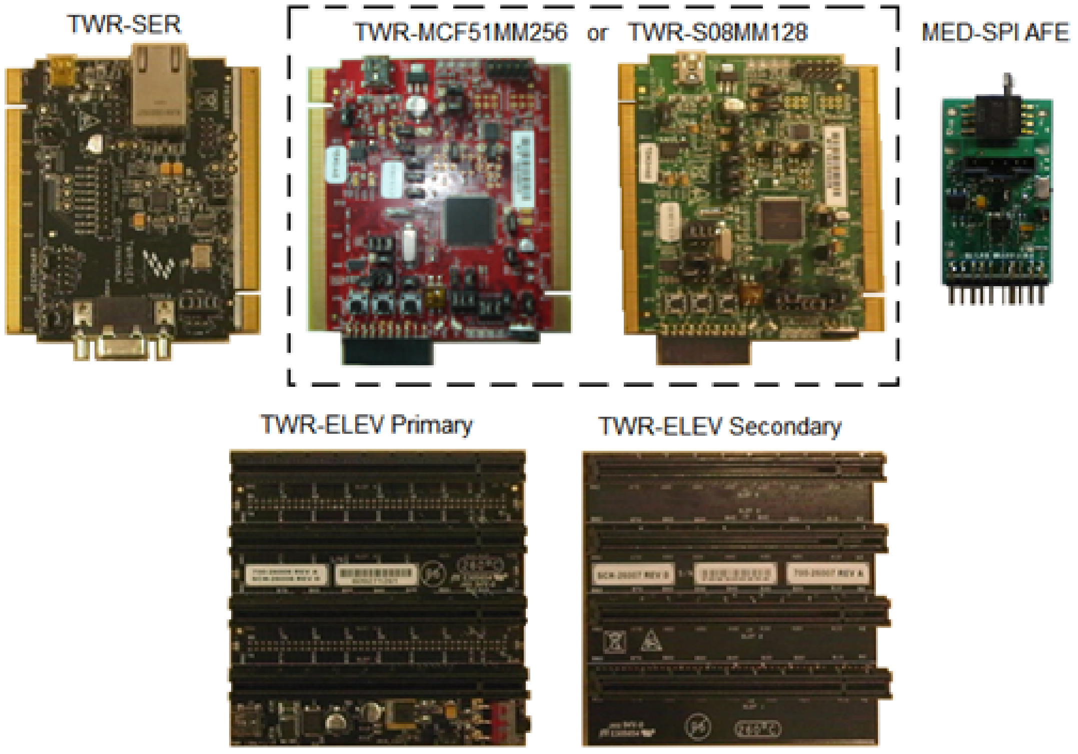


Figure 24. Tower System and Analog Front End

4. Connect the MCU USB ports and serial tower modules to the computer. When connecting the Tower System for the first time, it is necessary to install the driver for the Open Source BDM. There are two options—Install the software automatically, or specify the path where the driver is. By default, the path is—*C:\Program Files\Freescale \CodeWarrior for Microcontrollers V6.3\Drivers\Osbdm-jm60*.

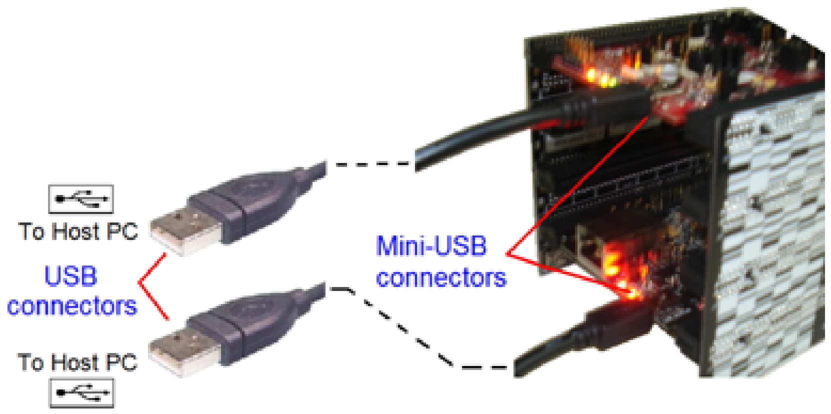
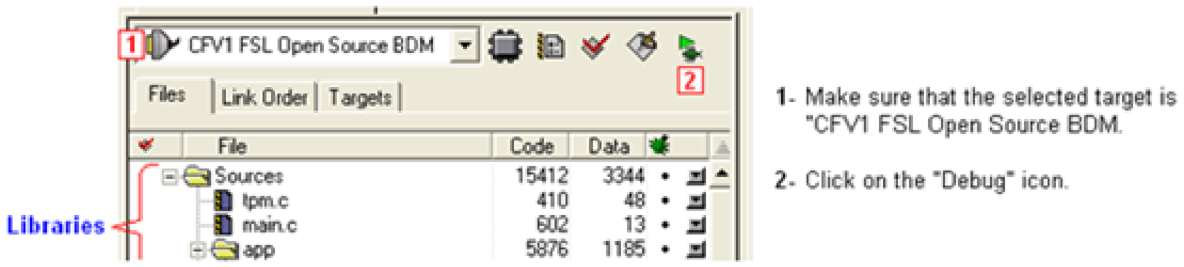


Figure 25. Tower-PC connections

5. Open CodeWarrior v6.3. Load the project for the corresponding MCU, whose paths are the following:
   - *SPR_for_S08MM\MED-SPR_S08MM\SPR_for_S08MM.mcp*
   - *SPR_for_MCF51MM\Source\device\app\cdc\codewarrior\cfv1usbmm256\SPR_for_MCF51MM.mcp*

6. Program the MCU as shown in the example; see Figure 26. The figure shows the target when programming the MCF51MM256. If the microcontroller used is MC9S08MM128, the target must be the HCS08 FSL Open Source BDM.



File	Code	Data
Sources	15412	3344
tpm.c	410	48
main.c	602	13
app	5876	1185

1- Make sure that the selected target is "CFV1 FSL Open Source BDM.

2- Click on the "Debug" icon.

Figure 26. Programming the microcontroller

7. If the microcontroller was successfully programmed, go to Graphical user interface to follow the instructions concerning the GUI.

5.2 Graphical user interface

The graphical user interface helps the user to visualize the spirometry results and the corresponding graph generated when performing a test.

To run a test using the GUI, follow the steps explained below:

1. After the microcontroller has been programmed, connect the TWR-SER module of the tower to the computer as shown in Figure 27. Press the Reset button on the microcontroller board. If it is the first time that the demo is connected, the computer recognizes the device as a Virtual Com Port and the driver must be installed.

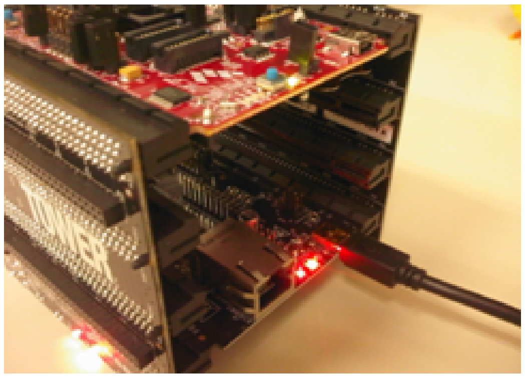
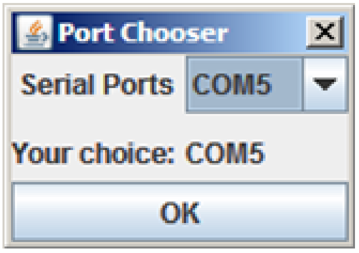


Figure 27. Connecting the USB cable

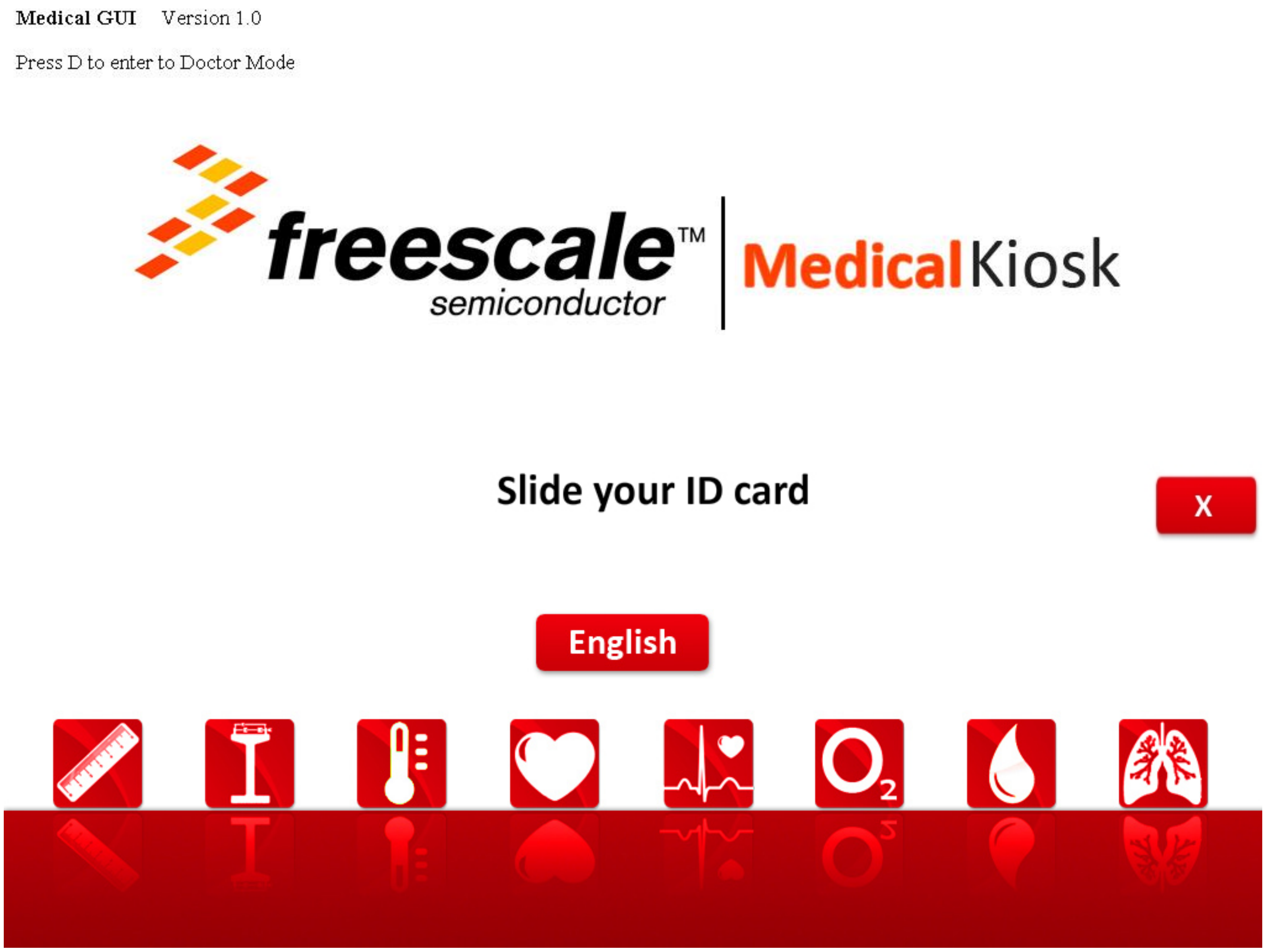
2. Open the Windows Device Manager and search for the port assigned to the USB CDC virtual com. This information is requested by the GUI.
3. Open the graphical user interface (GUI). It will ask for a port number, obtained in step 2. Select the correct port and click OK. This is shown in Figure 28.



Your choice: COM5

Figure 28. Selecting port

4. The main screen of the GUI (Figure 29) appears. Make sure that Caps Lock key is not activated on the keyboard and press Shift + D. This will start the GUI doctor mode.


Medical GUI Version 1.0

Press D to enter to Doctor Mode

Figure 29. Main screen of the GUI

5. In the doctor mode screen shown in Figure 30, click the SPI area on the GUI.

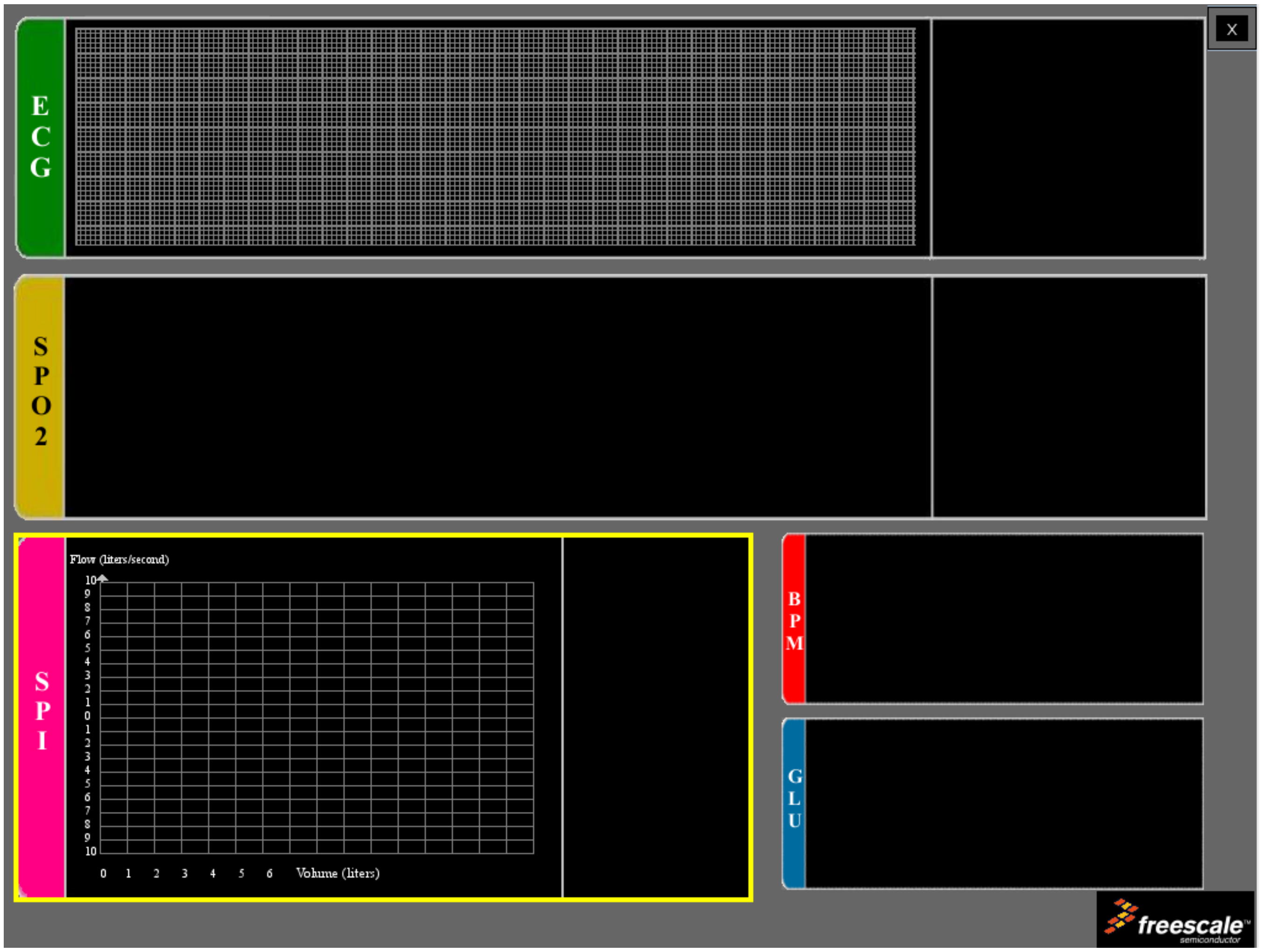


Figure 30. Doctor mode

6. Slowly exhale and take the spirometer tube. The correct form to hold the tube in the hand is shown in Figure 31.

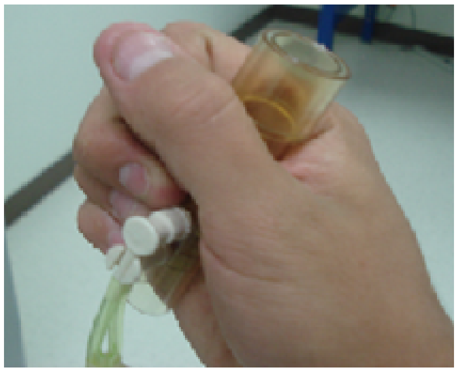


Figure 31. Holding the spirometer tube

7. Place the tube near the mouth as in Figure 32. Make sure that no air leaks out of the tube and take a deep breath, then exhale as hard and as fast as possible.

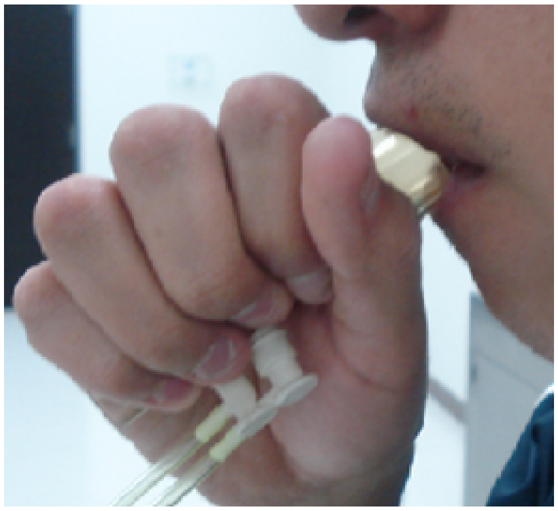
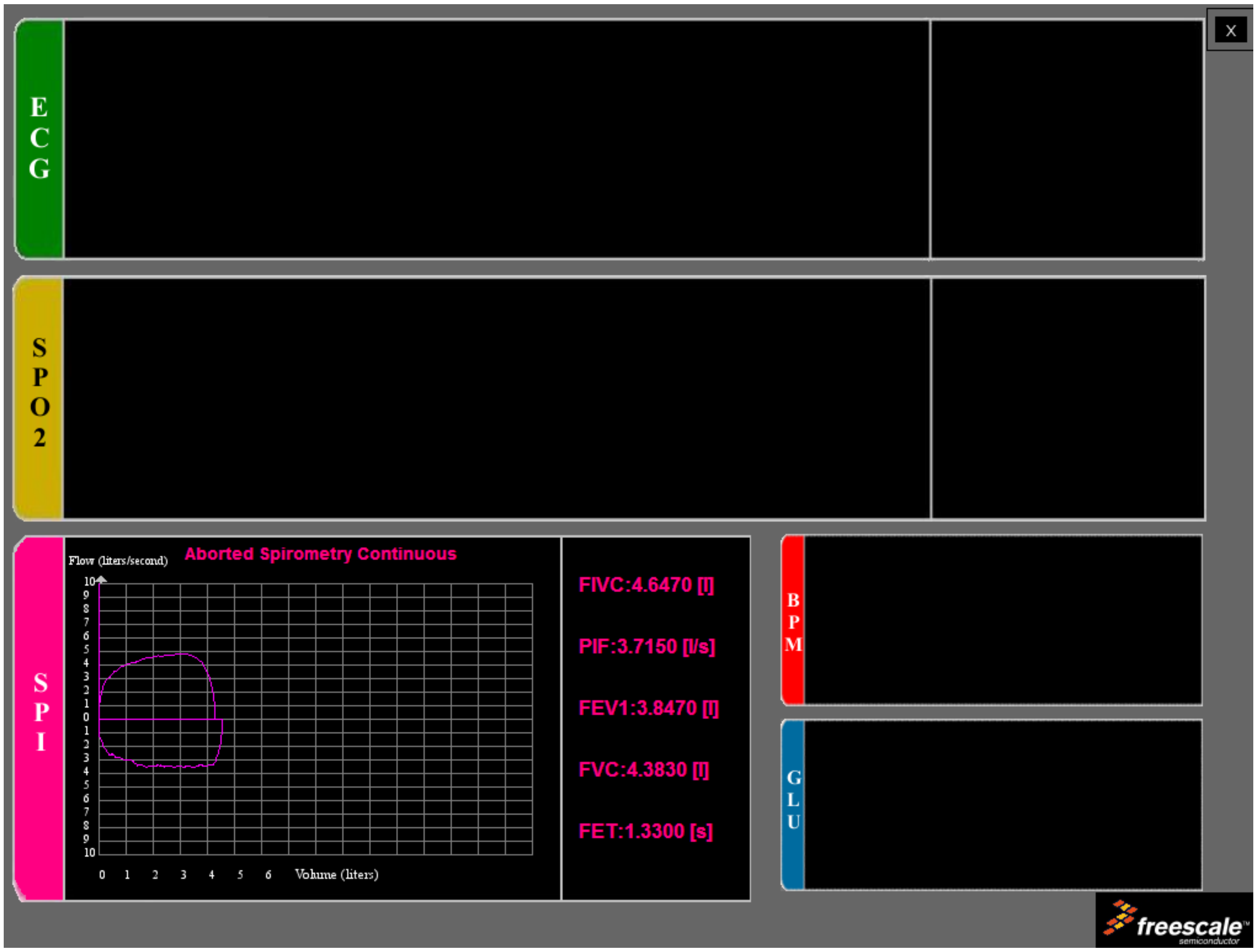


Figure 32. Breathing with the tube

8. The spirometry parameters and the graph are shown on the GUI. Figure 33 is an example of the results of a spirometry test.



Aborted Spirometry Continuous

FIVC:4.6470 [l]

PIF:3.7150 [l/s]

FEV1:3.8470 [l]

FVC:4.3830 [l]

FET:1.3300 [s]

Figure 33. Spirometry results and graph

6 Reference documents

There is more information available on the Freescale website. These are some of the documents:

- Kinetis MK53N512 Reference Manual, on **freescale.com/k50**
- MCF51MM256RM: MCF51MM256 Reference Manual, on **freescale.com/mcf51mm**
- MC9S08MM128RM: MC9S08MM128 Reference Manual, on **freescale.com/s08mm**
- USBUG: USB Stack Users' Guide, on **freescale.com/medicalusb**

Appendix A Communication protocol

The application communicates with the GUI (Graphic User Interface) in the PC using the Freescale USB Stack with PHDC with the device acting as a CDC (Communication Device Class). The device communicates via a serial interface similar to the RS232 communications standard, but emulating a virtual COM port.

After the device has been connected and a proper driver has been installed, the PC recognizes it as a Virtual COM Port and it is accessible, for example using HyperTerminal. Communication is established using the following parameters.

- Bits per second—115,200
- Data bits—8
- Parity—None
- Stop Bits—1
- Flow Control — None

Communication starts when the host (PC) sends a request packet indicating to the device the action to perform. The device then responds with a confirmation packet indicating to host that the command has been received. At this point, the host must be prepared to receive data packets from the device and show the data received on the GUI. Communication finishes when the host sends a request packet indicating the device to stop. The following block diagram describes the data flow.

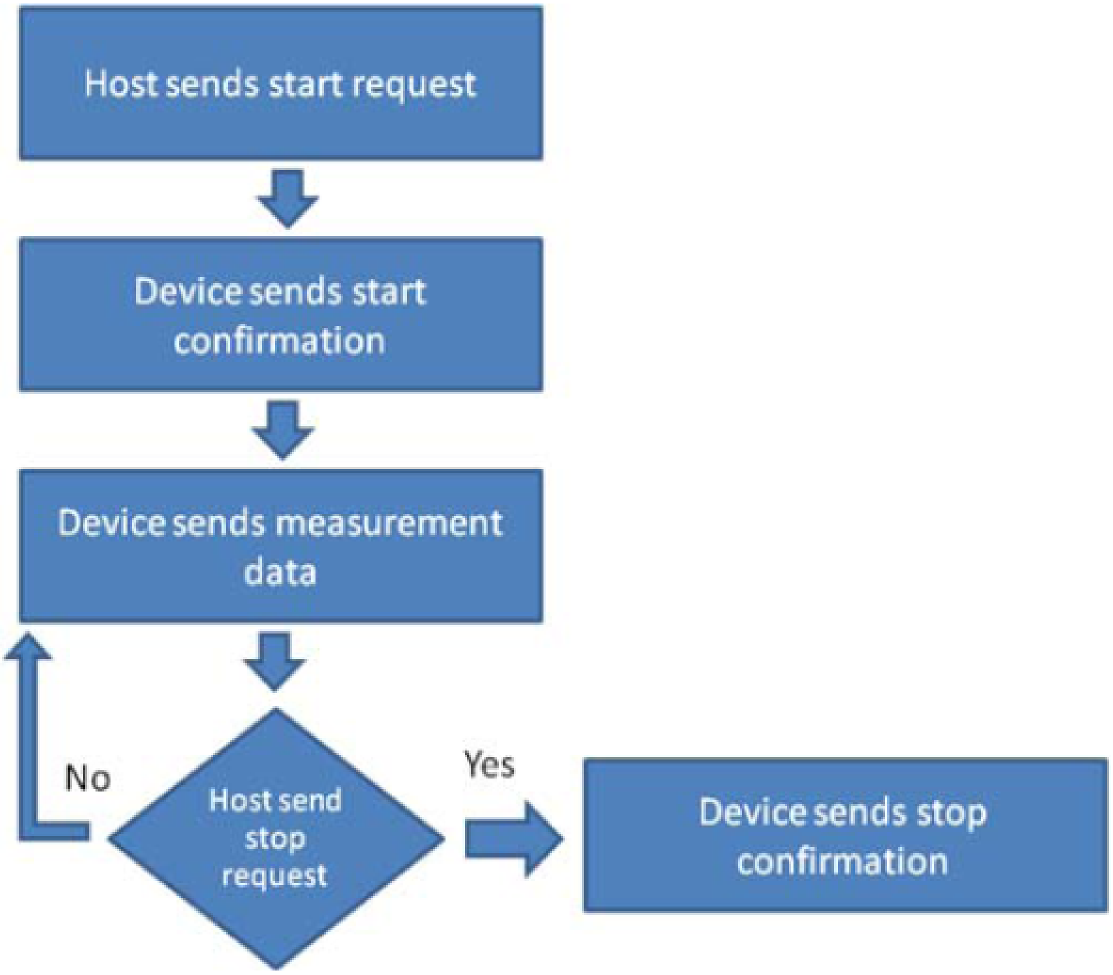

Host sends start request
Device sends start confirmation
Device sends measurement data
No
Host send stop request
Yes
Device sends stop confirmation

Figure A-1. Communication protocol data flow

Packets sent between host and device have a specific structure. The Packet is divided in four main parts:

- Packet Type
- Command Opcode
- Data length
- Data

The image below shows the packet structure.

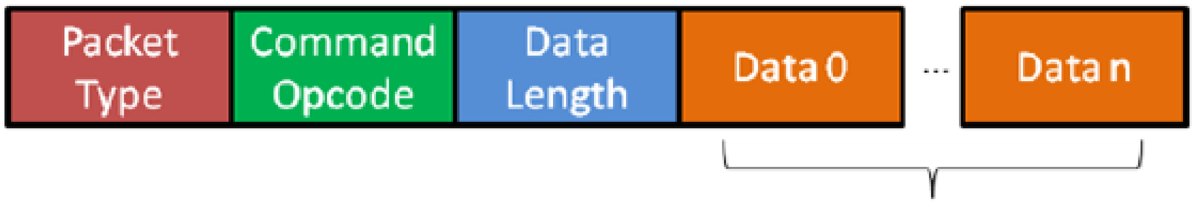


Figure A-2. Packet structure

A.1 Packet type

The Packet Type byte defines the kind of packet sent. There are three kinds of packets that can be sent between host and device.

A.1.1 REQ packet

This is a request packet, this kind of packet is used by the host to request to the device to perform some action like a start or stop measurement. A REQ packet is usually composed of 2 bytes, Packet Type and Command Opcode. Data Length and Data Packet bytes are not required.

A.1.2 CFM packet

This is a confirmation packet. This kind of packet is used by the device to confirm to the host that a command has been received, and sends a response indicating if the command is accepted, or if the device is busy.

A.1.3 IND packet

This is an indication packet. This kind of packet is used to indicate to the host that an event has occurred in the device and data needs to be sent. For example, this is used when the device has a new data to be sent to the GUI.

The following table shows the HEX codes for every Packet Type.

Table A-1. HEX codes

Packet type	Hex codes
REQ	0x52
CFM	0x43
IND	0x69

A.2 Command opcode

The Command Opcode byte indicates the action performed for a REQ packet, and the kind of confirmation or indication, in case of CFM and IND packet types. There are different Opcodes for every Packet Type. The following table shows the different Opcodes (See Note below).

Table A-2. Opcodes

Opcode	REQ	CFM	IND	Opcodes (Hex)
Glucose meter				
GLU_START_MEASUREMENT	X	X		0x00
GLU_ABORT_MEASUREMENT	X	X		0x01
GLU_START_CALIBRATION	X	X		0x02

Table continues on the next page...

Table A-2. Opcodes (continued)

Opcode	REQ	CFM	IND	Opcodes (Hex)
GLU_BLOOD_DETECTED			X	0x03
GLU_MEASUREMENT_COMPLETE_OK			X	0x04
GLU_CALIBRATION_COMPLETE_OK			X	0x05
Blood Pressure Meter				
BPM_START_MEASUREMENT	X	X		0x06
BPM_ABORT_MEASUREMENT	X	X		0x07
BPM_MEASUREMENT_COMPLETE_OK			X	0x08
BPM_MEASUREMENT_ERROR			X	0x09
BPM_START_LEAK_TEST	X	X		0x0A
BPM_ABORT_LEAK_TEST	X	X		0x0B
BPM_LEAK_TEST_COMPLETE			X	0x0C
BPM_SEND_PRESSURE_VALUE_TO_PC			X	0x28
Electro Cardiograph Opcode				
ECG_HEART_RATE_START_MEASUREMENT	X	X		0x0D
ECG_HEART_RATE_ABORT_MEASUREMENT	X	X		0x0E
ECG_HEART_RATE_MEASUREMENT_COMPLETE_OK			X	0x0F
ECG_HEART_RATE_MEASUREMENT_ERROR			X	0x10
ECG_DIAGNOSTIC_MODE_START_MEASUREMENT	X	X		0x12
ECG_DIAGNOSTIC_MODE_STOP_MEASUREMENT	X	X		0x13
ECG_DIAGNOSTIC_MODE_NEW_DATA_READY			X	0x14
Thermometer				
TMP_READ_TEMEPRATURE	X	X		0x15
Height scale				
HGT_READ_HEIGHT	X	X		0x16
Weight scale				
WGT_READ_WEIGHT	X	X		0x17
Spirometer				
SPR_DIAGNOSTIC_MODE_START_MEASURMENT	X	X		0x0C
SPR_DIAGNOSTIC_MODE_STOP_MEASURMENT	X	X		0x0D
SPR_DIAGNOSTIC_MODE_NEW_DATA_READY			X	0x0E
Pulse oximetry				
SPO2_START_MEASURMENT	X	X		0x21
SPO2_ABORT_MEASURMENT	X	X		0x22
SPO2_MEASURMENT_COMPLETE_OK			X	0x23
SPO2_MEASURMENT_ERROR			X	0x24
SPO2_DIAGNOSTIC_MODE_START_MEASURMENT	X	X		0x25
SPO2_DIAGNOSTIC_MODE _STOP_MEASURMENT	X	X		0x26

Table continues on the next page...

Table A-2. Opcodes (continued)

Opcode	REQ	CFM	IND	Opcodes (Hex)
SPO2_DIAGNOSTIC_MODE_NEW_DATA_READY			X	0x27
				0x21
System commands				
SYS_CHECK_DEVICE_CONNECTION	X	X		0x29
SYS_RESTART_SYSTEM	X			0x2A

NOTE
Software related with this application note does not respond to all of these commands.

A.3 Data length and data string

The data length and data string bytes are the data quantity count and the data itself. The data length byte represents the number of bytes contained into the data string. The data string is the information sent, just the data, therefore the Data Length byte must not count the Packet Type byte, the Command Opcode byte or itself.

A.4 Functional description

Communication starts when the host sends a REQ packet indicating to the device to start a new measurement. The host must send a REQ Packet Type to start transactions (Figure A-3).

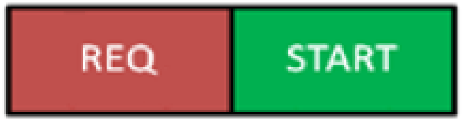


Figure A-3. Start packet sent by host

The Start Opcode can be any Opcode related with start a measurement, for example, if we wanted to start the ECG in diagnostic mode, the Data Packet will look like Figure A-4.

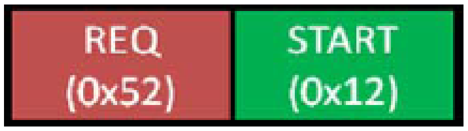


Figure A-4. Starting ECG in diagnostic mode

0x52 is the HEX code for a request (REQ) Packet Type, 0x12 corresponds to ECG_DIAGNOSTIC_MODE_START_MEASUREMENT. After sending the REQ packet, a CFM packet must be received indicating the status of the device. The received packet must look like Figure A-5.

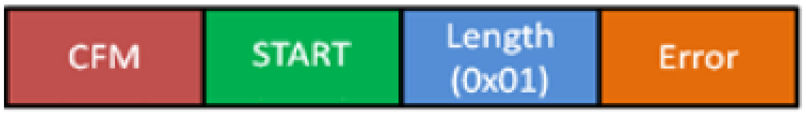


Figure A-5. Confirmation packet structure

The error byte indicates the device status. The table below shows possible error codes.

Table A-3. Error codes

Error	HEX code
OK	0x00
BUSY	0x01
INVALID OPCODE	0x02

If the error byte received corresponds to OK, the device starts sending data as soon as a new data packet is ready. If BUSY error is received, the host must try to communicate later. If the error received is INVALID OPCODE, data sent and transmission lines must be checked.

If a CFM packet with an OK error has been received, the device starts sending Information related with the measurement requested. This is performed using indication packets (IND). Indication packet structure is shown in Figure A-6.

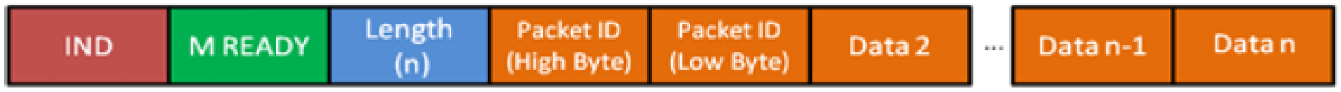


Figure A-6. Indication packet structure

The first byte contains the HEX code for an Indication Packet type. The second byte contains the Opcode for the kind of indication, for example if the device is sending an Indication Packet for ECG_DIAGNOSTIC_MODE_NEW_DATA_READY, the HEX code read in this position is 0x14 because this is the Indication Opcode for a new set of data from the ECG diagnostic mode. The next byte is the Length which indicates the quantity of data sent.

The first couple of bytes after the Length byte are the Packet ID bytes. The Packet ID is a 16-bit data divided in 2 bytes to be sent and contains the number of packets sent. The Packet ID number of a data packet is the Packet ID of the previous packet + 1. For example, if the Packet ID of the previous packet sent was 0x0009, the Packet ID of the next packet must be 0x000A. This allows the GUI to determine if a packet is missing.

The following data bytes are the Data String and contain the information of the measurement requested. The Data quantity is determined by the Data Length byte and data is interpreted depending on the kind of measurement. For example, for the MED-EKG Demo from Data 2 to Data n-1 contains the data graphed. Every point in the graph is represented by a 16-bit signed number, this means that every 2 data bytes in the packet, means it is one point in the graph. The first byte is the most significant part of the long (16-bits) and the second byte is the less significant part. The long is signed using 16-bit complement. The last byte contains the Heart Rate measurement. This byte must be taken as it is, an unsigned char data that contains the number of beats per minute. Figure A-7 shows a typical MED-EKG demo indication packet.

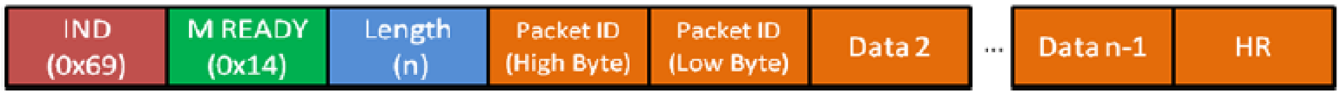


Figure A-7. MED-EKG IND packet

When a Stop request is sent by the host, the device stops sending data and waits for a new Start request command. Figure A-8 shows the Stop Command structure.

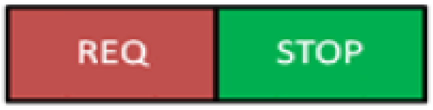


Figure A-8. Stop command structure

Immediately after this, the device must acknowledge with a CFM packet shown in Figure A-9.

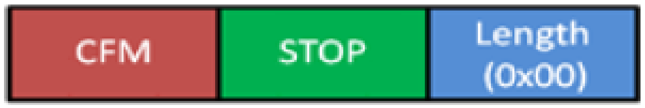


Figure A-9. Stop CFM packet

The CFM packet for stop does not require an error code, it just must be received. If this packet has not been received, the request has been rejected or not taken and must be sent again to stop the measurements.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4325
Rev. 1, 12/2012