Freescale Semiconductor
Application Note

Document Number: AN4025
Rev. 1, 4/2010

Implementing a Glucometer and Blood Pressure Monitor Medical Devices

by: Roxana Suarez and Carlos Casillas
RTAC Americas
Guadalajara
Mexico

1 Introduction

This document describes a combinational medical device designed to integrate both a low-end glucometer and a blood pressure monitor. Nowadays, people suffering from chronic degenerative diseases such as hypertension and diabetes can develop a plurimetabolic syndrome.

This syndrome and both diseases in the same patient share some risk factors such as obesity, hypercholesterolemia, and atherosclerosis.

Medical combinational devices target this new market and not only is power consumption a target, but bringing a better solution for disease control.

This application note addresses medical devices implemented with Freescale technology. By using the MC9S08LL16 this device is energy efficient. It includes a Medical USB Stack programmed into the MC9S08JS16 for communication and the MPXV5050GP pressure sensor.

A glucometer is a device for measuring levels of glucose concentration in the blood. This device is usually portable and is used at home for monitoring diabetic-patients.

A blood pressure monitor is a device that detects systolic and diastolic blood pressure, heart rate, and mean arterial pressure for patients who suffer or are at risk of developing high blood pressure.

Contents

2 Glucometers and Diabetes

Diabetes is one of the most common diseases today. It is essential to produce glucometers whose one of many advantages is to empower diabetics to take care of themselves without the need to visit doctors. Glucometers help to detect and confirm hypoglycemia and infections. High blood sugar may also be a sign of infection or illness that needs to be treated.

2.1 Diabetes Fundamentals

Diabetes mellitus commonly known as “ diabetes ” is a common health problem throughout the world. It prevents the body from producing enough insulin, does not produce insulin, produces defective insulin, or has resistance to the same. Insulin is a hormone produced in the pancreas. According to the World Health Organization statics, the global prevalence of diabetes mellitus is approximately 155 million people and expected to increase to 300 million in the year 2025. *Medical Management of Diabetes and Heart Disease Book*, Marcel Dekker Inc.

Glucometry is a technique that obtains the value of glucose concentration in peripheral or central blood. These values expressed either in mgr/dl or mmol are important clinical values for metabolic disorders such as diabetes mellitus, denutrition, and other consequences like hyperosmolar coma, malabsorption syndrome, and most critical hypoglycemia. A glucometer and proper pharmaceutical treatment is fundamental for glycemic control of diabetic patients. At home, some glucometers include different kinds of strips to monitor other variables such as ketones which are produced when a patient is experiencing hyperglycemia. Figure 1 shows a general diagram of a blood pressure monitor. It shows different peripherals for communication between the user and the device. The most important part is the test strip, this is the sensor to collocate the blood and get a determined measurement with the analog-to-digital converter (ADC) of the microcontroller unit (MCU). The other peripherals depend on the designer.

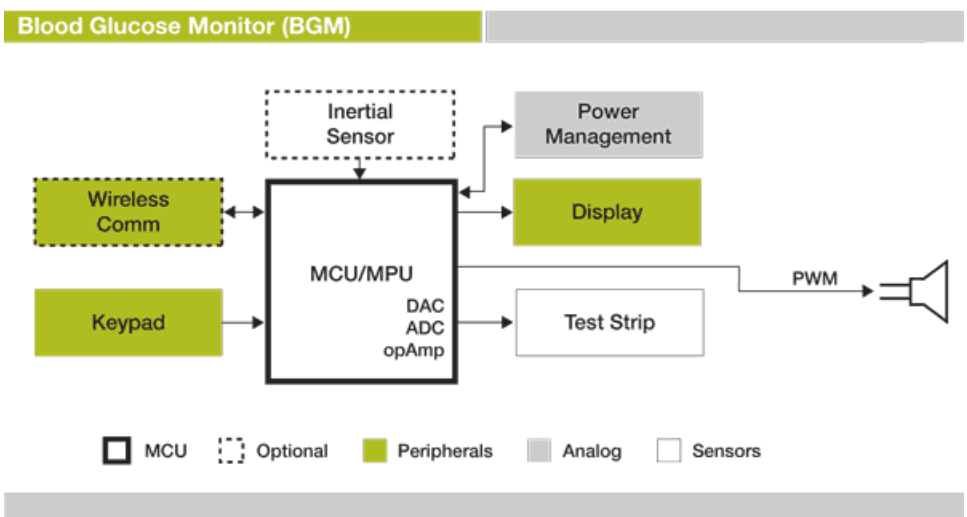

Figure 1. Blood glucose monitor block diagram

2.2 Glucose Sensors

The first step to measure the glucose in the blood is to convert the glucose concentration into a voltage or current signal, this is possible with special sensor strips for amperometry. The sensor uses a platinum and silver electrode to form part of an electric circuit where hydrogen peroxide is electrolyzed. The hydrogen peroxide is produced as a result of the oxidation of glucose on a glucose oxide membrane. The current through the circuit provides a measurement of the concentration of hydrogen peroxide, giving the glucose concentration.

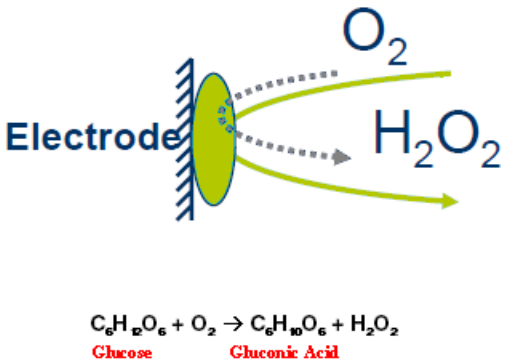

Figure 2. Electrode reactions between glucose and gluconic acid

The sensor used as a blood-glucose meter is based on a glucose oxide electrode. The glucose oxides were immobilized in a platinized activated carbon electrode. The enzyme electrode was used for amperometry determination by using an electrochemical detection of enzymically produced hydrogen peroxide. The sensor is composed of various electrodes; a glucose oxide membrane layer, a polyurethane film that is permeable by the glucose, oxygen, and hydrogen peroxide.

2.3 Amperometry

Amperometry measures electric current between a pair of electrodes that are driving the electrolysis reaction. Oxygen diffuses through the membrane and a voltage is applied to the Pt electrode reducing O_2 to H_2

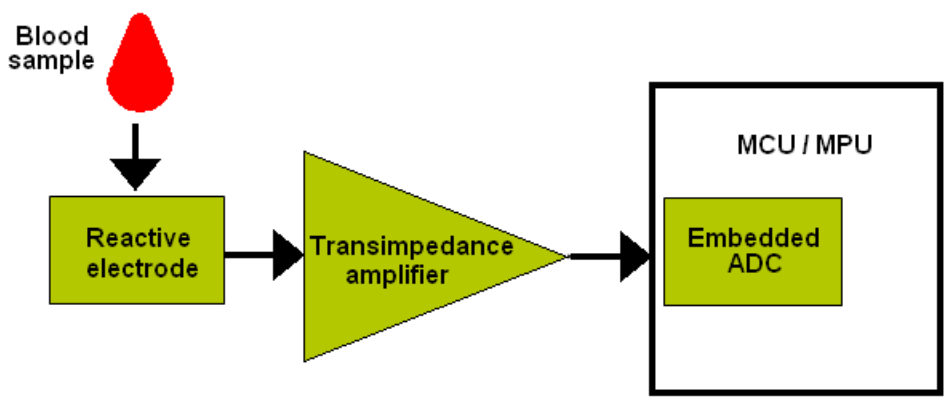

Figure 3. Test strip basic block diagram

These reactive electrodes are amperometric type sensors that use a three electrode design. This approach is useful when using amperometric sensors due to the reliability of measuring voltage and current in the same chemical reaction. Three electrode models use a working electrode (WE), reference electrode (RE), and a counter electrode (CE). After this current is produced this must be changed to voltage for processing by the MCU. This action is performed by the transimpedance amplifier. Finally, the MCU detects and processes this signal with the ADC module. For a transimpedance amplifier and sensor designs for medical applications go to Medical Application User Guide at www.freescale.com.

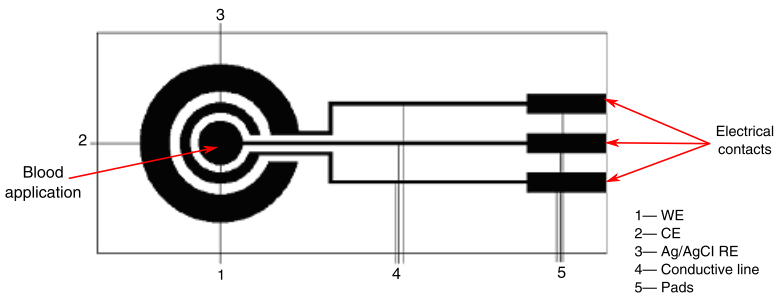

Figure 4. Chip schematic

Use an amperometric determination method with a constant potential of 0.3V used in the portable meter. The current response of the sensor is linear with a glucose concentration in the range of 5 to 30 mmol/ L and a fast response time of about 20 seconds.

3 Blood Glucose Monitor

This section explains how to develop a medical device for measuring the blood glucose level. This device operates placing a relatively small drop of blood on a disposable test strip that interfaces with a digital meter. Within seconds the level of blood glucose is shown on the liquid crystal display (LCD).

3.1 Transimpedance Amplifier

The transimpedance amplifier consists of an operational amplifier and a feedback resistor between the output and the negative input. The positive input can be connected to either GND or used for offset calibration.

3.2 Glucose Software Overview

The voltage source is always at 3.3 V. To start taking ADC samples, the source voltage must go to 0.3 V. The connection between the source and the application contains a voltage regulator of 3.3 V. You can provide voltage to the system by using a 9 V battery.

TakeSample function —The function configures the ADC module and starts conversion. It compares the ADC conversion obtained with `Value1` which is 60. If the ADC conversion is lower than this value, the MCU goes into stop mode, and after 20 seconds sends an error message to the LCD.

```
void TakeSample (void)
{
.
.
.
  ADC_Start();
  Strip_CTRL =StripVoltage300mV;        //0.3V Supply
  CountSec=0;
  ADC_Start_conversion (2);

  Drop=1;
  while(ADC_Get_Newconversion(2)<=Value1)
  {
    ADC_Start_conversion (2);
    _Stop;
    if(CountSec==20)
    {
      Error(1);
      Option=6;
      Drop=0;
```

```
      break;
    }
}
```

If no errors occur. The ADC conversion continues and sets the ranges as shown in the code below. The samples obtained must be the following:

- Range 0— ADC Conversion < 128,
- Range 1— 141 < ADC Conversion <= 292
- Range 2— 239 < Range 2 ADC Conversion<=407
- Range 3— 408 < Range 3 ADC Conversion<= 537
- Range 4— 539 < Range 4 ADC Conversion<= 752
- Range 5— ADC Conversion >752. Indicates a high level of glucose

NOTE

The ADC module resolution is 0.8058 mV/count.

```
while(CountSec<6)
{
  bLCD_CharPosition = 10;
  vfnLCD_Write_Char (0x30+(5-CountSec));
  if(CountSec==1)
  {
    ADC_Start_conversion (2);
    Sample=ADC_Get_Newconversion(2);

    if(Sample<128)
    {
      bLCD_CharPosition = 0;
      vfnLCD_Write_Char ('0');
      Range=0;
    }
.
.
.
}
```

The code below sets the glucose levels for each range. You have to change the information in range 2, range 3, and range 4 in the lines as commented below. Finally, determine the glucose level with the equation:

Glucose = x + midpoint

The x variable for:

- Range 1 = 35
- Range 2 = 86
- Range 3 = 166
- Range 4 = 201

```
if(Range==1) // changes for 2, 3 or 4
  {
    low=0;
    high=51;
    midpoint=0;
    while (low<high)
    {
      midpoint =(low+high)/2;
      if (Sample<Range1[midpoint]) // changes for Range2, Range3 or Range4
{
      high=midpoint-1;
    }
    else
    {
      low=midpoint+1;
    }
   }
   Glucose=35+midpoint;
```

Intialization range arrays:

```
UINT16 Range1[] = {179,..... ,275};

UINT16 Range2[] = {275, ..... ,391};

UINT16 Range3[] = {384, ..... ,545};

UINT16 Range4[] = {545, ..... ,763};
```

The TakeSample function disables the ADC and displays the results on the LCD.

```
DisplayResults();
vfnLCD_All_Segments_OFF ();
TASK=5;
```

4 Blood Pressure Monitor and Hypertension

Because hypertension (high blood pressure) becoming more and more common, technology has had to develop medical devices to help control these diseases. These portable devices allow monitoring blood pressure at home.

4.1 Hypertension Fundamentals

Hypertension or high blood pressure is a condition when the blood pressure in the arteries is chronically elevated. In every heart beat, the heart pumps blood through the arteries to rest of the body. Blood pressure is the force of the blood that is pushing up against the walls of the blood vessels. If the pressure is too high, the heart has to work harder to pump and this can lead to several diseases. A blood pressure monitor is a device used to measure arterial pressure as blood is pumped away from the heart.

Typically, from a user's perspective, a monitor includes an inflatable cuff to restrict blood flow and a manometer (pressure meter) to measure the blood pressure. From a system designer’s perspective a blood pressure monitor is more complex. It consists of a power supply, motor, memory, pressure sensor, and user interfaces that can include a display, keypad, or touchpad, and audio as well as optional USB or wireless communication interfaces. For more information go to the Blood Pressure Monitors webpage.

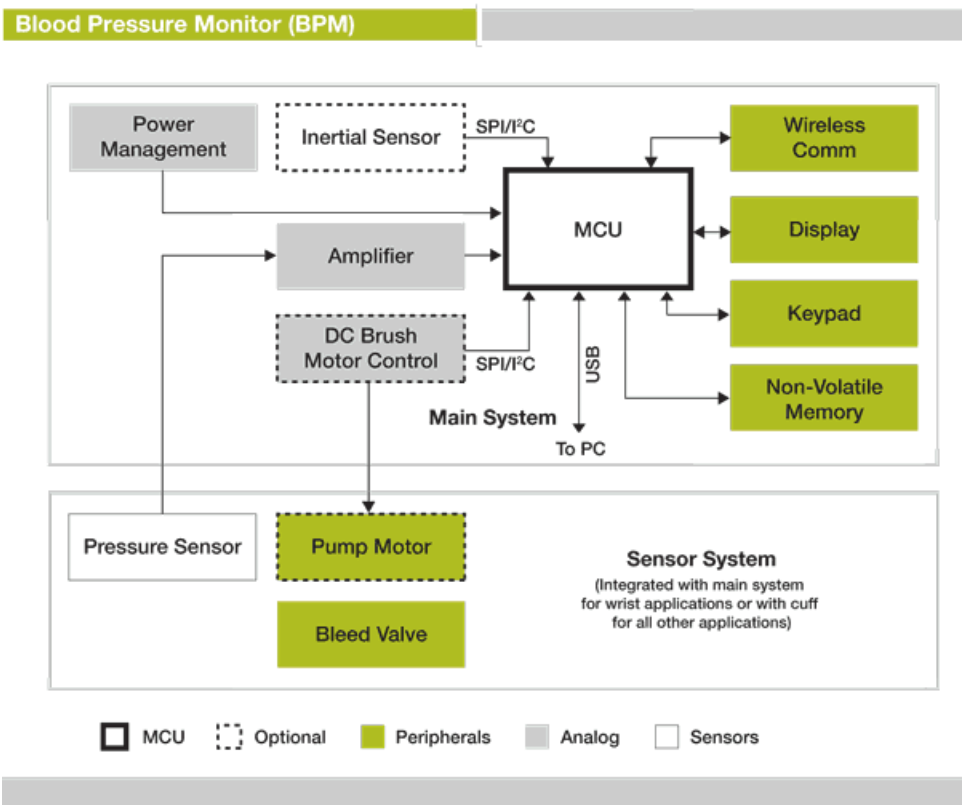

Figure 5. Blood pressure monitor general block diagram

5 Blood Pressure Monitor

This section explains the hardware background to develop a blood pressure monitor. The functionality of the MPXV5050GP pressure sensor, the hardware, and software developed are also described.

5.1 Pressure Sensor

This application implements the motor control mentioned in the Figure 5 driven by a pulse width modulation (PWM). The principal device for this application is the MPXV5050GP pressure sensor. This is because it depends on the pressure detected by the MCU so that the motor either turns on or turns off. Also important is an air compressor controlled by the motor and valve. The pressure sensor provides a signal that splits in two. One without a filter, and the other with a filter that removes noise.

MPXV5050GP medical features:

- Patented silicon shear stress strain gauge
- Pressure range up to 300 mm Hg (consult the datasheet)
- Polysulfone case material (medical, class V approved)

5.2 Hardware—Blood Pressure Monitor

The motor must be connected to the mini air pump valve to generate air for the air chamber, at the same time, the mini air pump has to be connected to a cuff, an air reference, and to the valve. Therefore when the motor is disabled and the valve enabled, the cuff starts to deflate. The pressure sensor is also connected to the cuff for taking in every moment and measuring the pressure. Figure 6 shows how to connect these elements. A hose can be used.

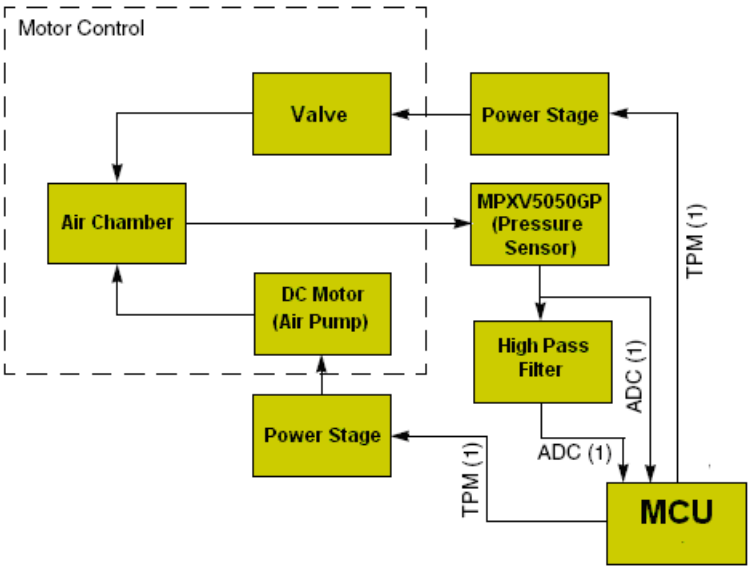

Figure 6. Pressure gauge block diagram

Figure 7. Motor, mini air pump, valve

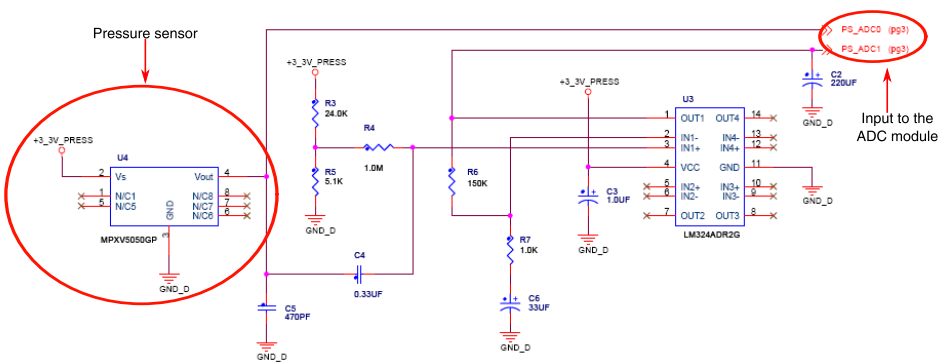

Figure 8. Pressure sensor connection

5.3 Software—Blood Pressure Monitor

BloodPressureMonitor.c

In this file are the determined incremented and decremented states to start to measure blood pressure. The PWM configuration controls the motor and the valve. The timer/pulse width modulator (TPM) configuration as PWM edge-aligned using channel 0 for motor control and channel 1 for valve control is described below. There are also some values used for the motor and valve control. The values can be changed depending on the hardware implementation.

Clock configuration:

- Clock source—Bus clock/64 = 10 MHz/64 = 156.250 kHz
- Clock cycle—6.4 us

PWM configuration:

- Counter Value—0x200 (HEX) = 512 (DEC) = 3.27 ms per duty cycle
- Motor PWM—0x80 (HEX) = 128 (DEC) = 25% of duty cycle
- Valve PWM—0x 100 (HEX) = 256 (DEC) = 50% of duty cycle

NOTE

This value is used to generate the PWM in the TPMxMOD register.

Although the motor and valve PWM have the same clock source they are never enabled at the same time.

5.4 Obtaining Blood Pressure Measurements

Shown below is the maximal value that corresponds to 180 mm Hg. When the MCU detects this pressure the system turns off the motor and starts to stabilize. How blood pressure and voltage relate and compare to each other is explained below. A pressure of 180 mm Hg is taken as a maximal value. First, it is necessary to make the conversion from mm Hg to kPa, because the sensor datasheets show values in kPa.

Conversion of mm Hg to kPa—1 kPa=7.50061505043 mm Hg then 180 mm Hg=24 kPa. It is possible with this conversion to get an approximate voltage.

According to the graphic and transfer function that the pressure sensor datasheet provides, it is possible to know the voltage present when the pressure sensor detects 180 mm Hg.

Transfer Function—3.3*[(0.018*24) + 0.04] = 1.55 V (This value is just an approximation without error)

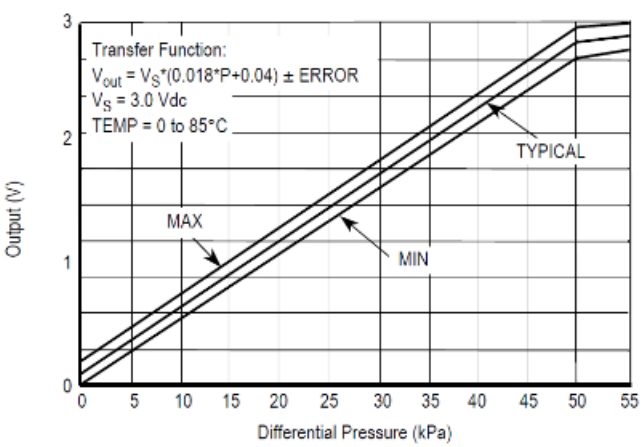

Figure 9. Voltage output versus pressure

The ADC module was configured in a 12-bit mode and can have counts of up to 4095. Taking the voltage supply of 3.3 V the ADC resolution is approximately 0.8058 mV/count . It is necessary to know when the pressure reaches 180 mm Hg which according to the transfer function the sensor delivers 1.55 V. This data lets you know how many counts are necessary to detect 180 mm Hg=1.55 V/0.8058 mV=1923 counts to provide a range of error. The systems uses 1900 counts to have a margin of error.

```
if (gu16Pressure>1900) //180 mm Hg
```

5.5 Motor and Valve Control

After the pressure reaches the maximal value of 180 mm Hg the program starts to stabilize the system. It establishes a shorter duty cycle for motor control and a longer duty cycle for valve control and starts to measure the blood pressure. The flowchart below shows how to implement the motor control.

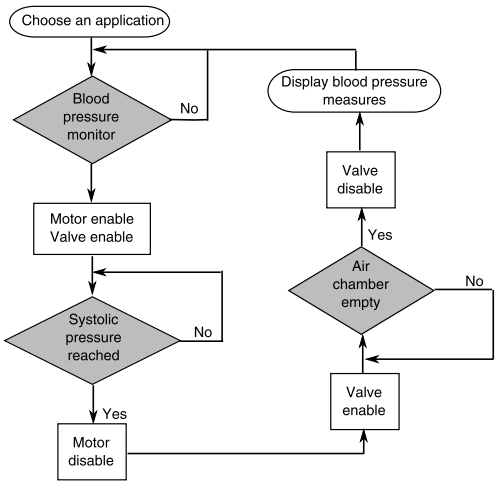

Figure 10. Motor control flowchart

The flowchart shows both pressures, the systolic which represents the maximum pressure exerted (cuff inflates) when the heart contracts and diastolic which represents the pressure in the arteries when the heart is at rest (cuff deflates).

5.6 Obtaining Heart Beats at BloodPressure.c

Blood pressure is one of the principal vital signs. Blood pressure is a force on the walls of blood vessels exerted by the blood circulating. During each heartbeat, blood pressure varies between systolic pressure and diastolic pressure. For a better diagnose and control of hypertension it is important to obtain the heart rate.

5.6.1 Obtaining the Heart Rate

The pressure sensor has one output split in two. The two outputs are taken from the pressure sensor to remove noise to get a better measurement. One output is filtered and obtains the specific frequency of the heart. The other output is for sensing the pressure receiver. When the cuff is attached to a person’s arm and it is deflating you can see slight variations in the overall pressure on the cuff. This variation in the pressure from the cuff is actually due to the pressure change from blood circulating. This variation is amplified through a high-pass filter designed at 1 Hz and set to an offset. The heart rate is calculated with the resulting signal.

5.6.2 Oscillometric Method

This method consists in taking samples from the oscillations caused by the blood flow. While the cuff is deflated from the systolic level, the blood starts to circulate through the obstructed artery producing vibrations. When the pressure cuff is decreasing, the oscillations increase up to a maximal amplitude; after decreasing, the blood flow is normal. The pressure in the cuff in the maximal oscillation is known as mean arterial pressure (MAP). When oscillation starts to increase quickly, this is the systolic pressure (SBP)(top number). When oscillations decrease quickly, this is the diastolic pressure (DBP). Figure 11 shows the pressure and the heart rate.

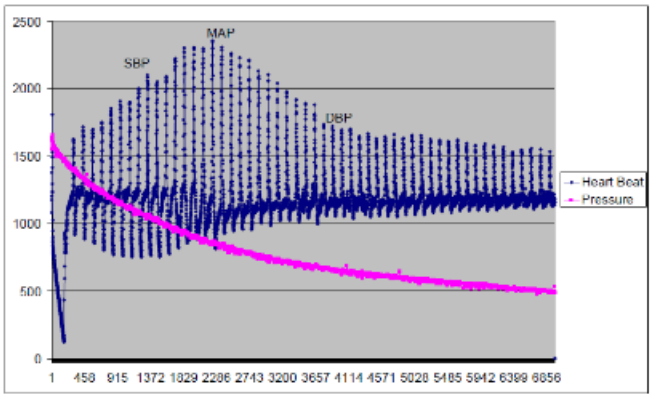

Figure 11. Oscillatory pressure curve

5.6.3 Mean Arterial Pressure

Mean arterial pressure (MAP) is a term used in medicine to describe an average blood pressure in an individual. It is defined as the average arterial pressure during a single cardiac cycle.

At normal resting heart rates the MAP can be approximated using the more easily measured systolic pressure (SP) and diastolic pressures (DP):

$$MAP \simeq DP + \frac{1}{3}(SP - DP)$$

6 Technology and Medical Devices

Medical diagnostics are now made with high technology avoiding continuous visits to the hospital. Making remote control LCDs can be used as an interface to communicate with the patient at different stages of the disease and generate alarms in chronic situations. The application also has different ways to communicate with external devices such as Bluetooth, SD Card, and USB.

7 LCD Driver

This application was developed using the MC9S08LL16 series. This device provides an LCD module that controls up to 192 segments and generates the waveforms necessary to drive an LCD. The LCD used for this application is a glass with 29 segments.

7.1 Modes of Operation and Power Supply

In the MC9S08LL16 series the LCD module can be configured to operate in Stop and Wait modes. The LCD module can operate in stop2 with all clocks turned off.

According to the LCD glass specifications this must be supplied at 3.3 V. For this configuration V_{IREG} is connected to V_{LL1} internally and a range of 1V for the V_{IREG} is chosen. V_{LL2} and V_{LL3} are generated by a charge pump. Figure 12 shows the connections between the LCD glass and the MCU.

7.2 LCD Hardware

Figure 12 shows how to connect the LCD pins and capacitors to V_{LL1}, V_{LL2}, and V_{LL3}, a charge pump source is enabled and it is necessary to connect a capacitor in Vcap1 and Vcap2 pins. It also describes the general connection between the LCD and the MCU.

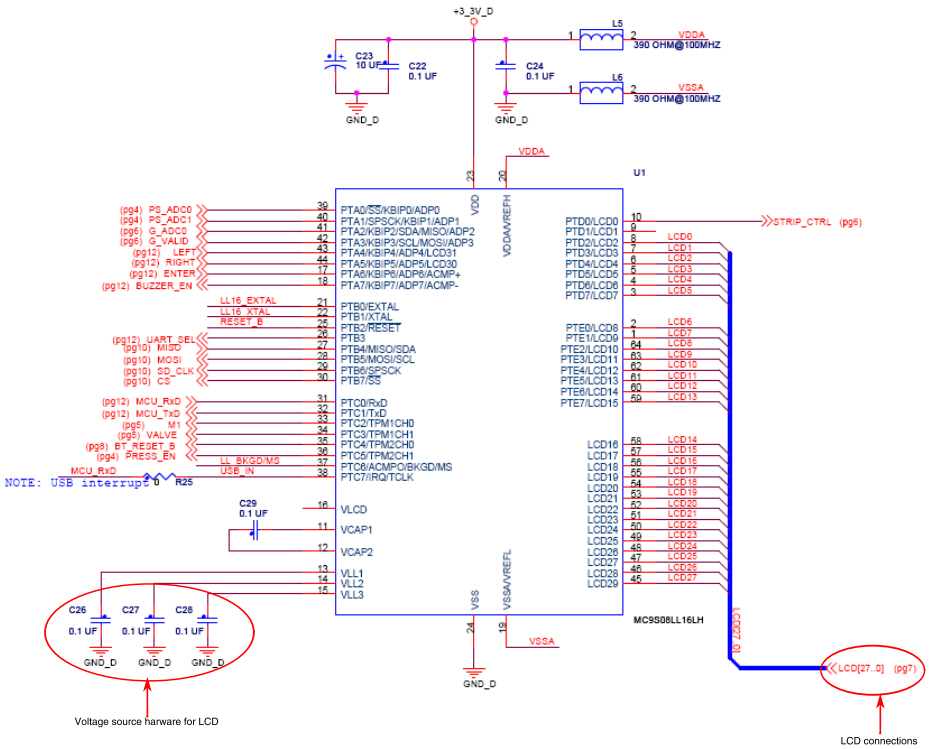

Figure 12. Connection to LL16

7.2.1 LCD Segment Specs

This section provides information of the LCD driver developed with the LL16 microcontroller that allows configuration, customization of different segments, and special symbols depending on the LCD glass used. The application note titled *LCD Driver Specifications* (document AN3796) is available at the Freescale website.

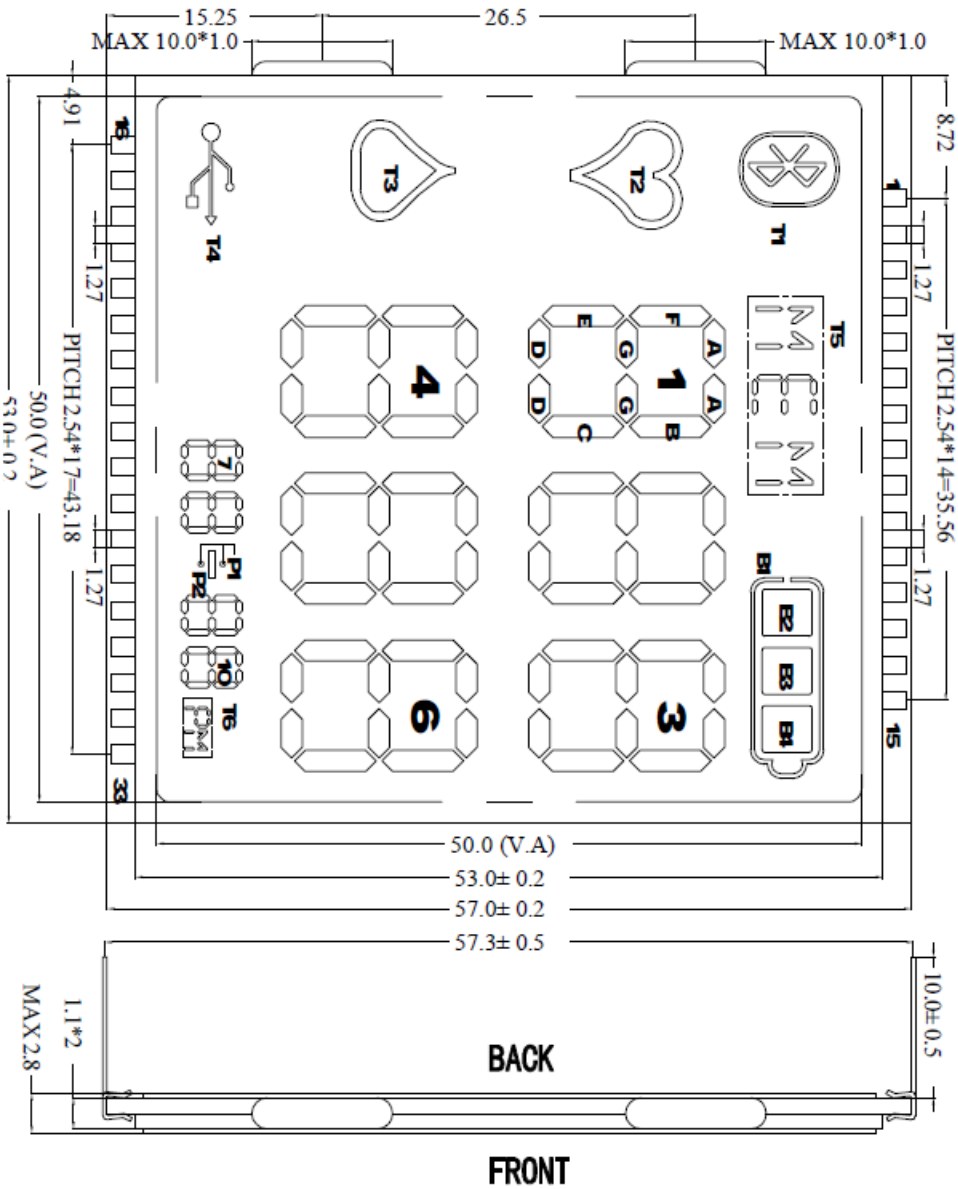

Figure 13. LCD glass

Table 1 shows the custom glass worksheet.

Table 1. LCD Specs

PIN	COM1	COM2	COM3	COM4
1				COM4
2			COM3	
3		COM2		
4	COM1			
5	B1	B2	B3	B4
10	1A	1F	1E	1D
11	1B	1G	1C	

PIN	COM1	COM2	COM3	COM4
12	2A	2F	2E	2D
13	2B	2G	2C	
14	3A	3F	3E	3D
15	3B	3G	3C	
16	T1	T2	T3	T4
17	4A	4F	4E	4D
18	4B	4G	4C	
19	5A	5F	5E	5D
20	5B	5G	5C	
21	6A	6F	6E	6D
22	6B	6G	6C	
23	7A	7F	7E	7D
24	7B	7G	7C	
25	8A	8F	8E	8D
26	8B	8G	8C	
27		P2	P1	
28	9A	9F	9E	9D
29	9B	9G	9C	
30	10A	10F	10E	10D
31	10B	10G	10C	
32	T6	T5		

7.3 LCD Software

General LCD software flowchart

Figure 14 shows the LCD sequence. It is in an infinite loop that always returns to the Task Management switch. To show the different options of the application this function controls a variable named TASK. This variable is configured with a 1 to enter in the glucometer's principal menu, 2 for the blood pressure monitor device (choose an application), 3 to save the measurement, 4 for Bluetooth communication, 5 for USB communication, and 6 to enter stop mode. To change the status of the TASK variable assign the value necessary.

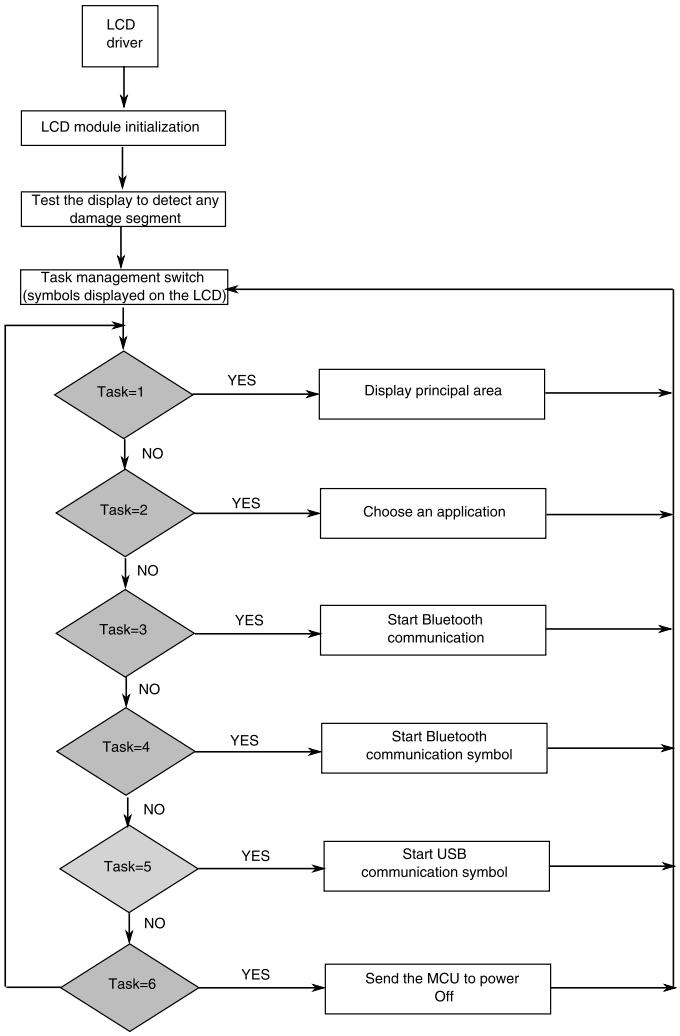

Figure 14. LCD flowchart

LCD.c

The principal function of this file is to configure the LCD module, set the frontplanes, backplanes, the power supply, the clock source, and the duty cycle for the waves to generate the messages in the LCD glass.

Below you can see how to enable the LCD pins, and set which LCD pins will be backplanes and COMS.

Table 2. LCD pins and backplane configurations

Function	Parameters	Functionality
`EnableLCDpins (RegNum,Mask)`	RegNum—Number of the register to write. Mask—Mask to habilitate LCD pins	Enable the corresponding LCD pin for the LCD operation.
`EnableBackplane (RegNum, Mask)`	RegNum—Number of the register to write. Mask—Mask to select backplanes	Enable the corresponding LCD pin for the backplane operation.

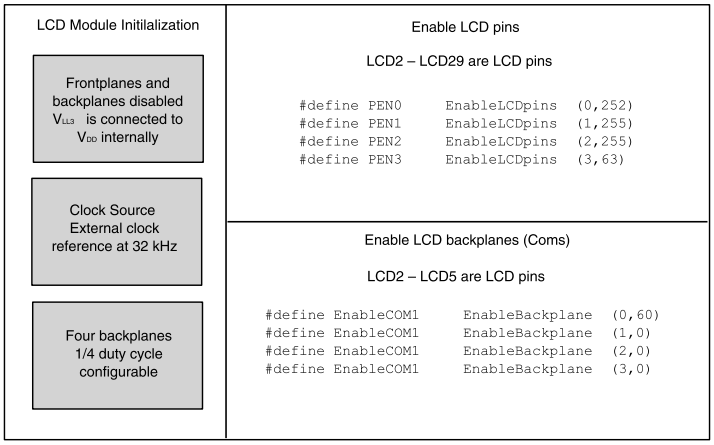

Figure 15. LCD.c structure

LCDMessages.c

This application depends on the function and option chosen. LCDMessages.c contains functions that determine actions taken by the LCD. For more information consult the LCDMessages.c file directly. This file contains a message that informs communication status of the Bluetooth and the USB. It also provides a message to inform when there is an error and level voltages.

Table 3. Communications status

Function	Parameters	Functionality
`Application_Menu(UINT8 appli)`	appli—Indicator to each application	Shows the application options that the system has
`Configure_Menu(UINT8 config)`	config—Indicator for different options of configuration	Shows the configure options that the system has
`ChooseMemory(bool sel)`	sel—Indicates if it is the memory for the blood pressure monitor or for glucose	Shows the memory options
`MemoryDisplay(UINT8 Mem)`	Mem—Indicates what information shows glucose or blood pressure measurements	Shows on the display the information stored on the SD card.
`ReadyBP(void)`	None	Message before the blood pressure measurement
`DisplayResults()`	None	Shows on the display the current information measured
`BP_Configure(UINT16 Inf)`	Inf—saves the values taken from the ADC	Shows the current value of MaxPressure and SensorCalibration on the display
`Select_Memory_Configure(void)`	None	Shows the memory selected on the display
`DisplayMemoryFunction(UINT8 C)`	C—Indicates what actions the memory is taking	Shows memory function options on the display
`Select_Communication_Configure(void)`	None	Shows communication option selected on the display.
`Display_Communication(void)`	None	Shows the option selected—USB enable, Bluetooth enable, or both disabled

8 Bluetooth Connectivity

This communication helps short range transmission of data from medical devices to mobile phones and computers. This feature in medical devices helps to have contact between the patient and the doctor without having to go to the doctor's office.

8.1 Bluetooth Theory

Bluetooth is an open wireless protocol that creates personal area networks (PANs) and exchanges data over short distances between fixed or mobile devices. It can connect several devices and overcome problems of synchronization. Bluetooth provides 10 meters of distance to set communication at a speed of up to 1 Mb/s, high compatibility to most computers, and it is not necessary to implement a special network for communication.

8.1.1 Service Discovery Application Profile (SDAP)

Service discovery protocol (SDP) provides a means for applications to discover what services are available and to determine characteristics of those available services. A specific service discovery protocol is needed in a Bluetooth environment. The service discovery protocol defined in Bluetooth specification is intended to address unique characteristics of a Bluetooth environment.

8.2 Bluetooth Hardware

In this application the LMX9838 Bluetooth serial port module is used. It integrates the Bluetooth 2.0 baseband controller with 2.4 GHz radio, crystal, antenna, and other features. This section explains how to connect this device with the MCU. To power up the device some filters were implemented to the VCC, VCC_CORE, and VCC_IO pins as shown in Figure 16. It is important to connect the LMX9838 with adequate ground planes and a filtered power supply. To provide better performance and low power consumption an external crystal was placed at 32 kHz. The Bluetooth device also provides two pins to connect LEDs that indicate link status and RF traffic.

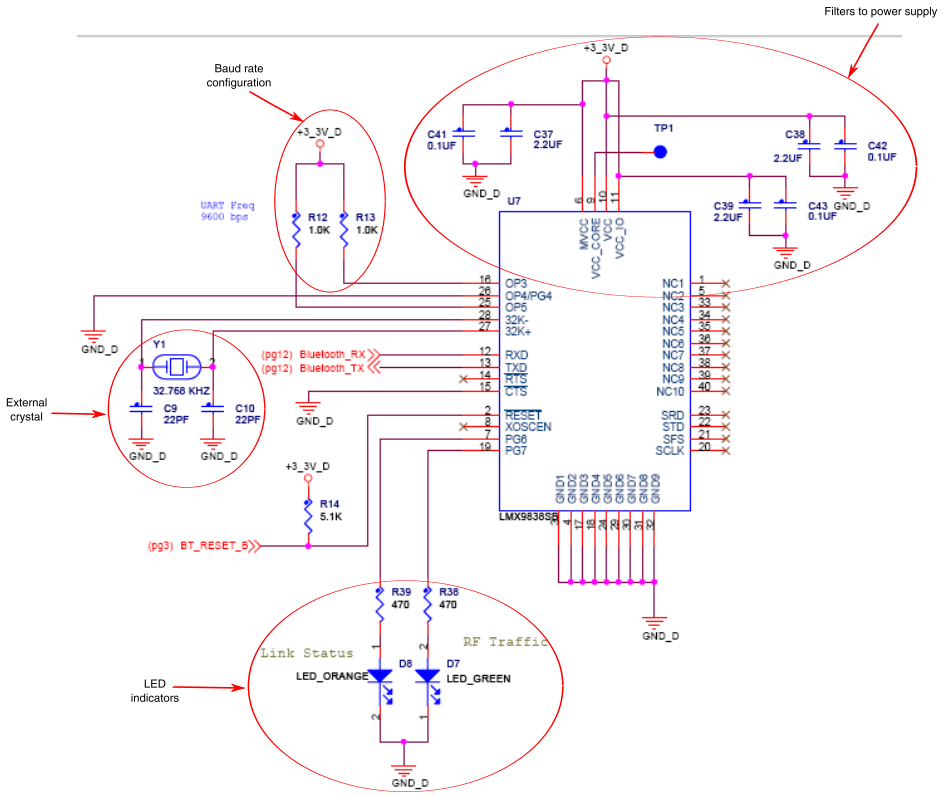

Figure 16. Bluetooth driver

Because this application uses USB and Bluetooth communication both through the serial communication interface (SCI) module, a QS3VH253Q multiplexor is used to control what communication is used.

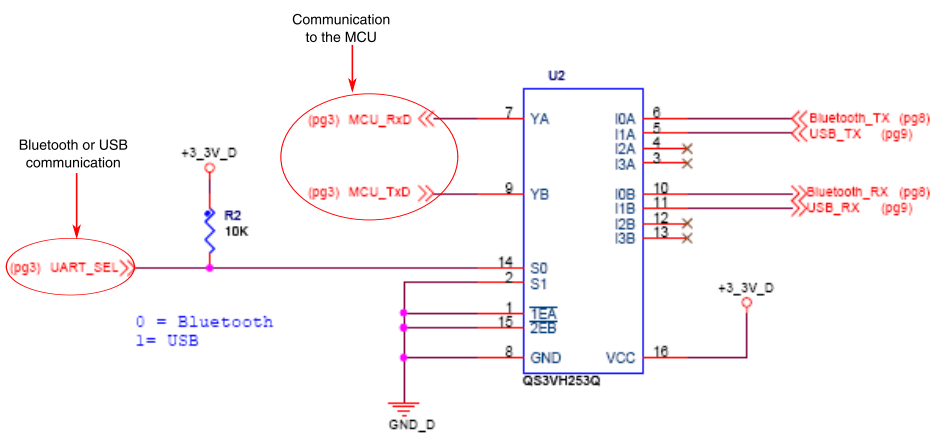

Figure 17. Communication multiplexor

8.2.1 Baud Rate Configuration

The LMX9838 provides a UART interface to communicate with the MCU through the SCI module. The UART interface supports formats of 8-bit data with or without parity with one or two stop bits. It can operate at standard baud rates from 2400 bits/s up to a maximum baud rate of 921.6 kbits/s. The UART baud rate is configured during startup by checking option pins OP3, OP4, and OP5. Table 4 shows different UART frequency settings.

Table 4. UART frequency settings

OP3	OP3	OP3	Baud Rate
1	0	0	Read from NVS [1]
1	0	1	9.6 kbps
1	1	0	115.2 kbps
1	1	1	921.6 kbps

1. System parameters in non-volatile storage

When the value is at 1 this means a 1 K pull-up resistor must be placed.

This configuration is with the LMX9838 Bluetooth® Serial Port Module. In the MC9S08LL16 the SCI baud rate must also be configured in the `SCI_Init (void)` function.

```
void SCI_enable(void)
{
    SCIBD = BR;      /* SCI baud rate = 10MHz/(16*65) = 9615 bps */
    SCIC1_LOOPS=0;
    SCIC2_TE=1;
    SCIC3_TXDIR=1;
}

*****************************************************************************

#ifndef __SCI_H_
#define __SCI_H_

#include "MyTypes.h"

#define BR    54      /* BAUD RATE = 9600 (Baud rate mismatch = 0.160 %)  */
#define ERROR 0x07

void SCI_enable(void);
void SCI_Send_byte(UINT8);

#endif /* __SCI_H_ */
```

8.3 Bluetooth Software

Bluetooth.h

In the Bluetooth.h file are all the declared necessary functions to implement Bluetooth communication. The functions help to connect and disconnect the system. It also configures the serial port and when to reset.

Function declarations:

```
Set_Bluetooth_Communication(void)
BT_SFW_Reset(void)
BT_Send_Data(void)
BT_SDAP_Connect(void)
BT_SDAP_SPP_Service_Browse(void)
BT_SDAP_Disconnect(void)
BT_Create_SPP_Connection(void)
BT_Send_Data(void)
```

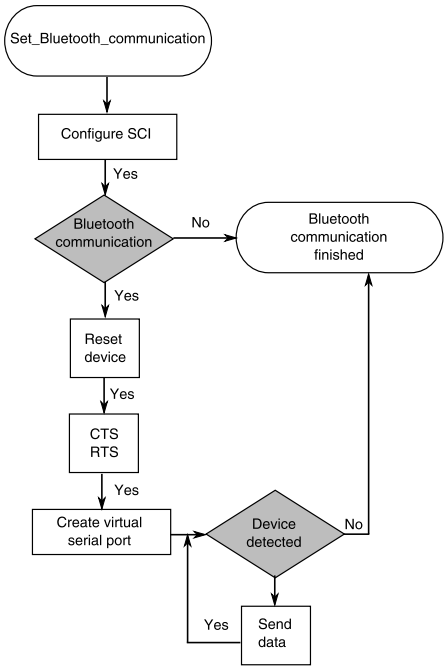

Figure 18. Bluetooth software flowchart

Bluetooth.c

Table 5. Bluetooth functions

Function	Parameters	Functionality
`UART_SEL= UART_Bluetooth`	None	Reset and choose between Bluetooth or USB communication
`Set_Bluetooth_Communication(void)`	None	Set configuration to communicate the application with remote devices using Bluetooth. First, the hardware reset is generated to activate module. The SCI module is configured to function with Bluetooth through a virtual serial port. After configuring the device is ready to set communication with other devices.
`BT_Create_SPP_Connection()`	None	Create virtual serial port with the SCI
`BT_Send_Data()`	None	Send information

9 Micro SD Card

For this application it is necessary to have an interface to save and transport information easily and efficiently. This section explains the connections and software for an SD Card drive.

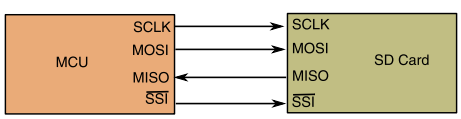

Figure 19. SD Card connection diagram

An SD Card is connected to the microcontroller through the serial peripheral interface SPI. The SD Card functions as a slave and the MCU as master.

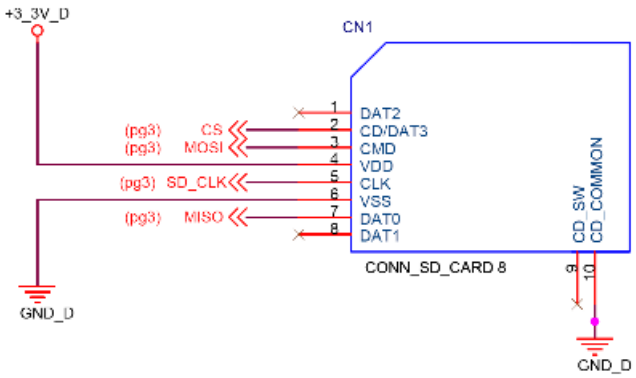

Figure 20. CONN_SD_CARD 8

9.1 Software—Micro SD Card

SD.h file

The SD Card communication is a command protocol, these commands are declared at the SD.h file. Functions are located in the SD.c file. The following commands are the principal commands.

Table 6. SD Card Command

Command	Description
SD_CMD0	Resets the SD memory card
SD_CMD1	Initialization process
SD_CMD6	Check mode 0 and mode 1
SD_CMD8	Host supply voltage
SD_CMD9	CSD
SD_CMD10	CID
SD_CMD12	Stop transmission
SD_CMD13	Sends status register
SD_CMD16	Sets a block length
SD_CMD17	Reads a block
SD_CMD18	Continuously transfers
SD_CMD24	Writes a block
SD_CMD25	Continuously writes up to Stop mode
SD_CMD27	Programs programmable bits of the CSD
SD_CMD28	Sets the write protection bit
SD_CMD29	Clears the write protection
SD_CMD30	Sends the status of the write protection
SD_CMD32	Sets the address of the first write block
SD_CMD33	Sets the address of the last write block
SD_CMD38	Erases all previously selected write blocks
SD_CMD42	Used to Set or Reset the password or lock and unlock the card

Command	Description
SD_CMD55	Defines the next command
SD_CMD56	Used either to transfer a Data Block or to get a Data Block
SD_CMD58	Reads the OCR register of a card
SD_CMD59	Turns the CRC option on or off

9.2 SD Card Initialization

To use an SD Card it is necessary to follow a specific sequence for initialization. The first step is to set the SPI clock. The MCU must then send 80 clock signals before it starts communication. This allows initialization of all the registers. The slave pin is configured at zero with this value the MCU enters SPI mode and a CRC byte is sent. After the configuration, the MCU starts to check the block length and place the clock to maximum.

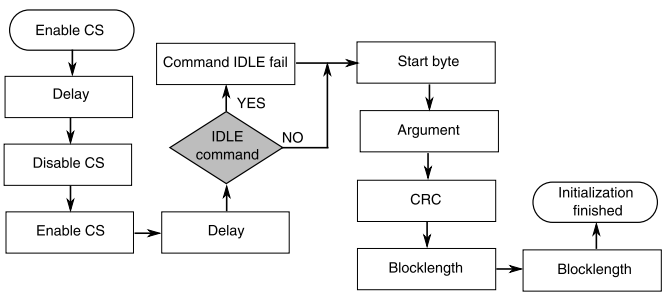

Figure 21. Initialization flowchart

SD.c file

As explained in Figure 21, the SD Card initialization is executed with the SD_init function. The slave is enabled and disabled. Delays are generated prior verifying the IDLE command. The MCU sends a byte that contains information to send the argument and the CRC waits for a response from the slave device.

The card is continuously polled with initialization and block-length commands until the idle bit becomes clear, indicating that the card is fully initialized and ready to respond to general commands.

For more information about SD Card communication, Freescale provides the design reference manual titled *SD Card Reader Using the M9S08JM60 Series Designer Reference Manual* (document DRM104).

Table 7. SD Card function

Functions	Parameters	Functionality
`SD_Read_Block(UINT32 u16SD_Block,UINT8 *pu8Data-Pointer)`	u16SD_Block—Indicates the block to read. *pu8 DataPointer—Base pointer to store the data from the SD Card.	Reads an SD Card block and if an error occurs a code returns
`SD_Write_Block(UINT32 u16SD_Block,UINT8 *pu8Data-Pointer)`	UINT32 u16SD_Block—Indicates the block to b write. UINT8 *pu8DataPointer—Base pointer to store the data to SD Card.	Writes one SD Card block and if an error occurs a code returns.

9.3 Stored Information

The code below shows how to use the SD.c functions that store diagnostic information (glucose levels and blood pressure) with different data. Each diagnose is saved differently. For example, the blood pressure diagnostic saves the systolic pressure, diastolic pressure, and the heart beats taken.

```
void StoreData (void){

if(SD_Init()==1){
Error(9);
}

  if (App_ID==1){                   //Save Glucose Information
    GMem[1]=Glucose;
    GMem[0]=Glucose>>8;
    GMem[2]=Hr;
    GMem[3]=Min;
    GMem[4]=Day;
    GMem[5]=Month;

    ptrG=&GMem[0];

    if(SD_Write_Block(RG,ptrG)!=0){
    Error(9);
    }

  }

  if (App_ID == 2){               //Save Blood Pressure Information
    BPMem[1]=SystolicPressure;
    BPMem[0]=SystolicPressure>>8;
    BPMem[3]=DyastolicPressure;
    BPMem[2]=DyastolicPressure>>8;
    BPMem[5]=HeartBeat;
    BPMem[4]=HeartBeat>>8;
    BPMem[6]=Hr;
    BPMem[7]=Min;
    BPMem[8]=Day;
    BPMem[9]=Month;

    ptrBP=&BPMem[0];

    if(SD_Write_Block(RP,ptrBP)!=0){
    Error(9);
    } }

void BPReadMemory (void){
ptrBP=&BPMem[0];

  if(!SD_Read_Block(RP,ptrBP)){
   SystolicPressure=(BPMem[0]<<8)+BPMem[1];
   DyastolicPressure=(BPMem[2]<<8)+BPMem[3];
   HeartBeat=(BPMem[4]<<8)+BPMem[5];
   Hr=BPMem[6];
   Min=BPMem[7];
   Day=BPMem[8];
   Month=BPMem[9];
   MemoryDisplay(2);
  } else{
  Error(9);
  }

}
```

9.4 Read Information

This code is to read the information provided by the SD Card.

Blood pressure and glucose information.

```
void BPReadMemory (void){
ptrBP=&BPMem[0];
```

```
  if(!SD_Read_Block(RP,ptrBP)){
   SystolicPressure=(BPMem[0]<<8)+BPMem[1];
   DyastolicPressure=(BPMem[2]<<8)+BPMem[3];
   HeartBeat=(BPMem[4]<<8)+BPMem[5];
   Hr=BPMem[6];
   Min=BPMem[7];
   Day=BPMem[8];
   Month=BPMem[9];
   MemoryDisplay(2);
  } else{
  Error(9);
  }

}

*****************************************************************************************

void GReadMemory (void){
ptrG=&GMem[0];

  if(!SD_Read_Block(RG,ptrG)){
   Glucose=(GMem[0]<<8)+GMem[1];
   Hr=GMem[2];
   Min=GMem[3];
   Day=GMem[4];
   Month=GMem[5];
   MemoryDisplay(1);
  } else{
  Error(9);
  }
}
```

10 Power Management

It is important to correctly implement the power source to control the analog and digital voltage for different stages in the system. Even when the whole system is supplied with the same voltage, the motor and the control consume more electrical current. It is necessary to separate the voltage source and implement the circuit's isolation voltages and GNDs. Figure 22 shows the principal circuit to supply voltage. The system receives 9 V and depending if the switch is ON or OFF, allows passing the regulated voltage of 3.3 V with a capacitor coupling to ensure the voltage level and improve the signal integration.

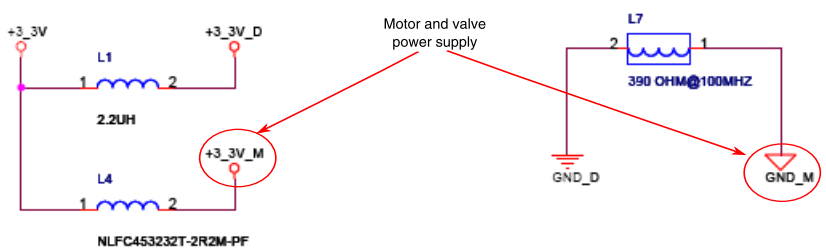

Figure 22. Voltage and GND isolations

10.1 Power Management and the MC9S08LL16

The MC9S08LL16 microcontroller has an operating voltage of 1.8 V to 3.6 V that allows the LCD, SCI ,and SPI to work with low power consumption. This device provides wait and stop modes.

The system provides three power sources from the principal voltage source.

- Integrated circuit voltage
- Motor and valve voltage
- Pressure sensor supply voltage

A switch with transistors is implemented for the pressure sensor voltage source. The MCU sends a signal to the Q2 base. The principal voltage source connects to GND and the pressure sensor voltage source is then generated.

Table 8. Power source

Source	Device
+3_3V_D	MC9S08LL16L, MAX4734, LMX9838SB, MC9S08JS16, SD Card, buzzer, QS3VH253Q, push buttons. (Control or digital section)
+3_3V_M	Motor and valve
+3_3V_PRESS	Pressure sensor

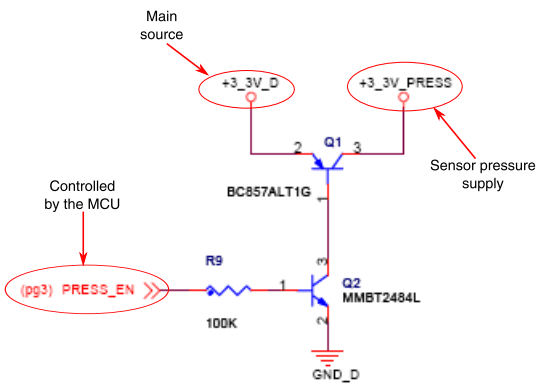

Figure 23. Pressure block on/off power

11 Errors System

The system application generates errors in different situations. To prevent any damage in the system and for patient health alerts an error is generated. The errors are identified with a number and each number of errors alert about different situations.

- Error(1)—Waits a few seconds and if a strip is not detected, the system considers that the strip is used.
- Error(2)—Waits a few seconds and if a strip is detected, but does not have a reaction, the system considers that there is no blood sample.
- Error(3)—Detects a strip and reaction but there is not enough blood to sample.
- Error(4)—When the pressure detected is higher than 180 mm Hg it is possible that there is an escape of air.
- Error(5)—Heart pulse not detected.
- Error(6)—The diastolic pressure is lower than 30 mm Hg.
- Error(7)—Bluetooth is enabled but cannot make a connection.
- Error(8)—The USB module is enabled but cannot make a connection.
- Error(9)—Connection error with the SD Card.

12 Personal Healthcare Device Class and Medical USB Stack Applications

Personal Healthcare Devices Class (PHDC)— It is required to implement methods for sending personal healthcare data to a USB host. The PHDC is to enable seamless interoperability between personal healthcare devices and USB hosts.

Medical USB Stack—Is based on USB PHDC and the IEEE–11073. It is compatible with Continua host emulator software. This Medical USB Stack enables microcontrollers with guidelines for Continua Health Alliance connectivity and is a first step to test your application prior official Continua certification.

Freescale provides implementations to the PHDC using the MC9S08JS16.

Featured documentacion:

- MEDUSBAPIRM—*Medical Applications USB Stack API Reference*
- MEDCONLIBAPIRM—*USB Device API Reference*
- MEDUSBUG—*Medical Applications USB Stack User Guide*
- MEDCONLIBUG—*Medical Connectivity Library Users Guide*
- MDCLUSBSFTWRFS—*Medical Applications USB Stack Fact Sheet*

USB hardware connections:

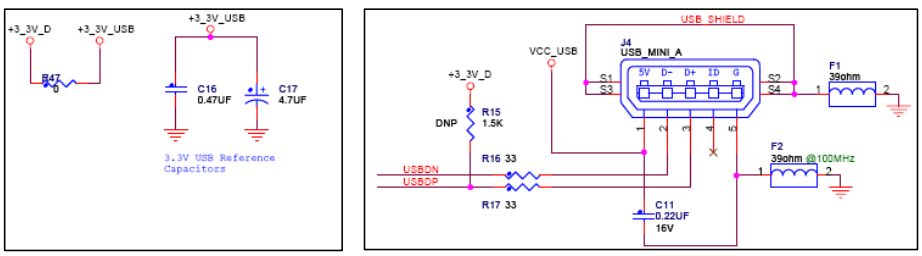

Figure 24. MC9S08JS16 connection and USB connector

13 User Guide

After the glucometer and blood pressure monitor are developed, it is time to test the system. Chapter 5. Blood Pressure Monitor explains how to interconnect these devices and this section explains how to use the system. First, it is important to localize a connection for the battery, motor, valve, glucose connector, and pressure sensor. Connection of a 9 V power supply to the system is required.

NOTE

Without a J7 jumper the system remains off. Turn the system on to start.

System:

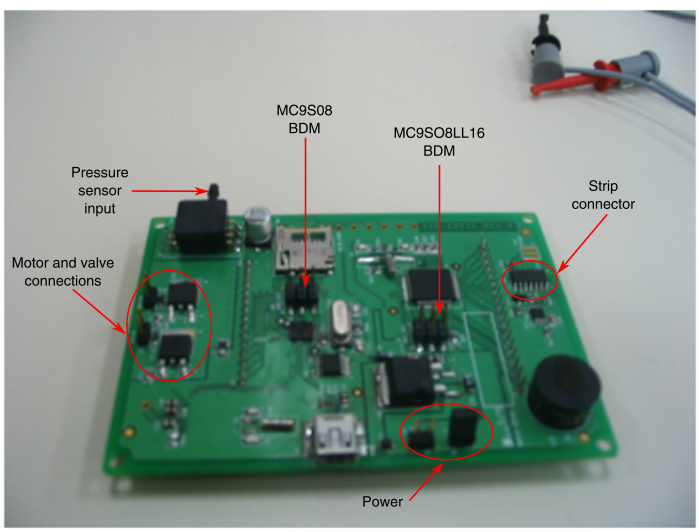

Figure 25. Evaluation board

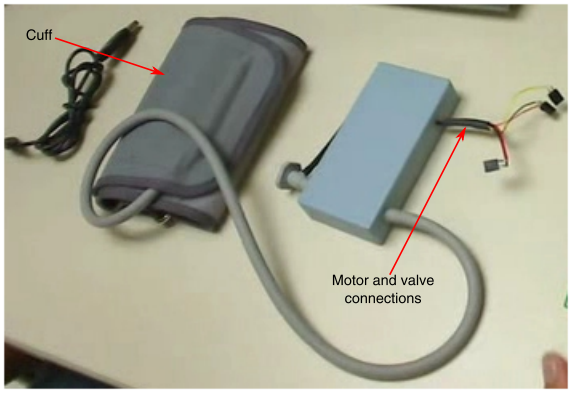

Figure 26. External equipment

Board connection:

J1—Turns on the motor
J2—Turns on the valve
J3—9S08LL16 BDM connections
J5—BAT 9 V battery
J6—MC9S08JS16 BDM connections
J7—SW1 turns on the system
Pressure sensor input—Receives air pressure from the cuff

For the complete application you must load two programs.

The MC9S08LL16 controls the LCD, pressure sensor, motor, and communication peripherals.
The MC9S08JS16 contains the drive for the Medical USB stack and operates as a bridge from the SCI to USB.

The board provides two BDM connections. The system starts with a glucometer application it is necessary to have new strips for testing. The images below show where to place the blood or the glucose sample and how to connect it to the board.

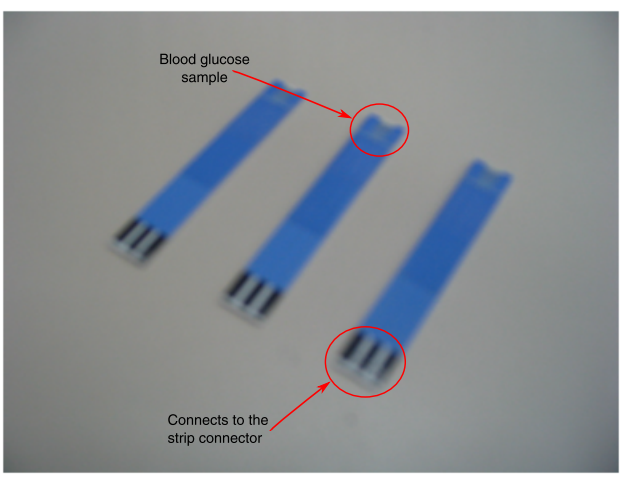

Figure 27. Test strip

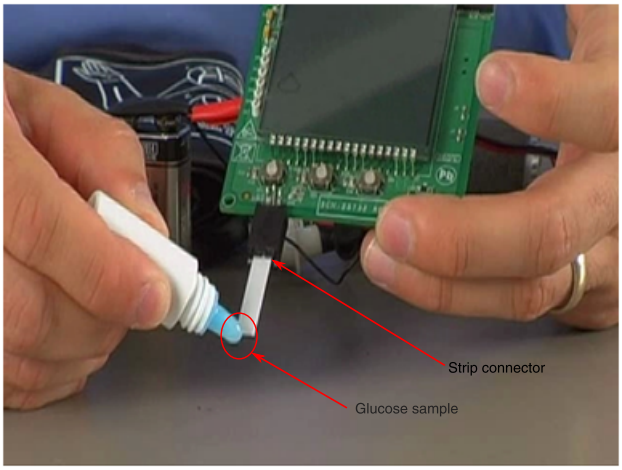

Figure 28. Using the strip

To start to measure the glucose press the ENTER button. The system displays a blood drop blinking on the LCD that indicates the system is measuring. If a strip is not detected the system displays a message of Error 1, 2, 3 and so on, depending on the error number. If there is no action the system changes to low power mode and turns off the display. If the functionality is optimal the result is displayed.

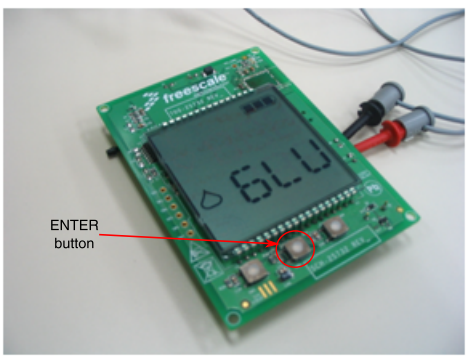

Figure 29. Start glucometer

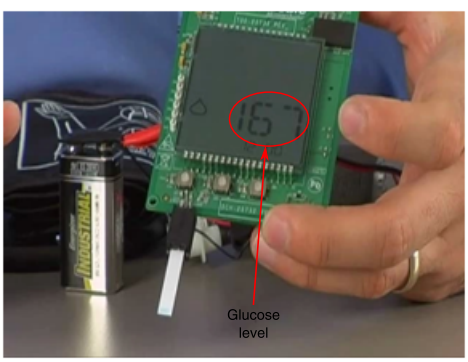

Figure 30. Glucose level

The LEFT button sends the MCU to a low power mode and shows the OFF message on the LCD. To turn off the system completely, press the ENTER button. To send the system to the blood pressure monitor application, press the RIGHT button. To start to measure the blood pressure press the ENTER button. It is important to prevent any error in the blood pressure measurement to ensure that the motor, valve, and cuff are connected to the system. After the system measures the systolic and diastolic pressure. The system displays a heart symbol for the systolic and diastolic pressure, the heart rate values.

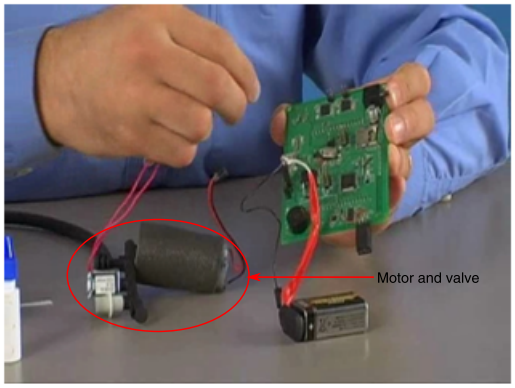

Figure 31. Connect blood pressure monitor

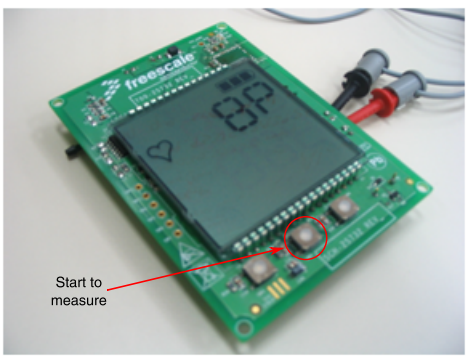

Figure 32. Start blood pressure monitor

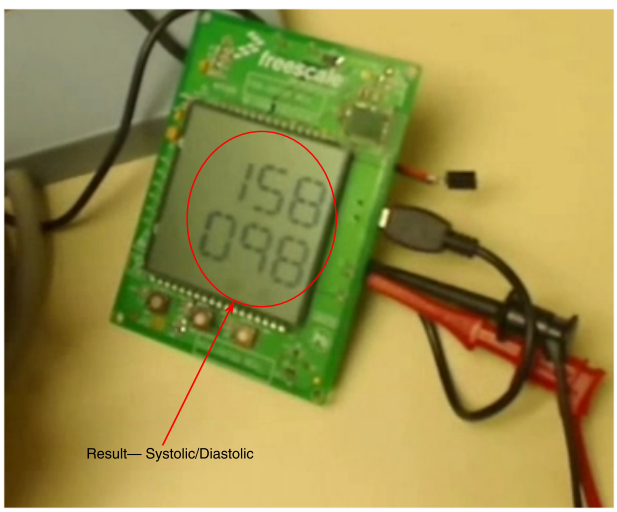

Figure 33. Systolic and diastolic pressure

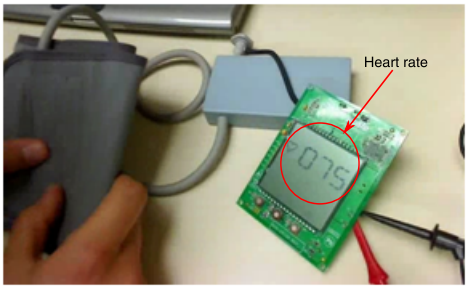

Figure 34. Heart beats per minutes

When the system is powered on, all the LCD segments turn on and detect any damage.

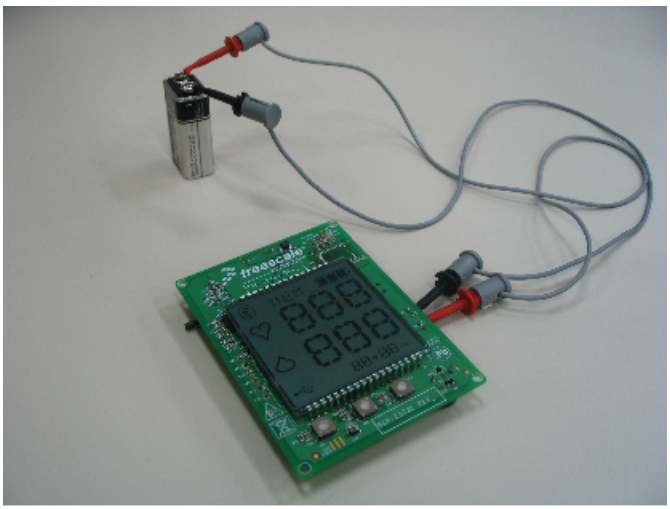

Figure 35. Display test

14 Conclusion

Freescale provides a variety of devices with low power consumption and high performance to develop medical applications. Flexibility in selecting the right communication interface makes Freescale an adequate solution to use USB or Bluetooth. Freescale technology develops medical devices such as glucometers and blood pressure monitors to serve as key parts of intelligent hospitals, telehealth solutions, or single end-user monitoring devices.

15 References

Freescale provides full schematics and source codes for customizing your application and a faster development. See available videos at www.freescale.com/medical. Videos at the Freescale channel on www.youtube.com/freescale.

- Ultra Low Power Medical Device Demo by Freescale— http://www.youtube.com/watch?v=eQTT_sjCMPo
- Blood Glucose Meter with Blood Pressure Monitor—http://www.youtube.com/watch?v=NuoRK7DgJkM
- Continua USB PHDC Demo—http://www.youtube.com/watch?v=Z47ILv0eSLQ
- *Electrocardiograph and Heart Rate Monitor Fundamentales* (document AN4059) at the Freescale Medical webpage—www.freescale.com/medical
- *Medical Management of Diabetes and Heart Disease Book*, Marcel Dekker Inc, Authors: Burton E. Sobel and David J. Schneider.
- Blood Pressure Monitors A Freescale Reference Design at www.freescale.com

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

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4025
Rev. 1, 4/2010