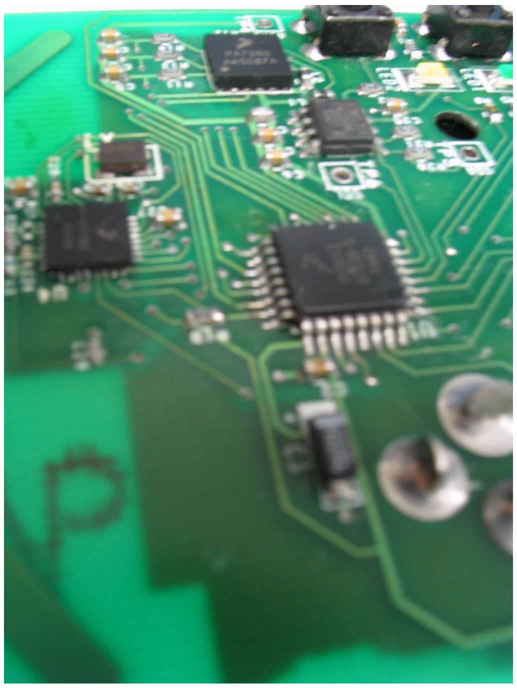
Human Fall Detection Using 3-Axis Accelerometer

Reference Manual

Developed by:
Rogelio Reyna
Edgard Palomera
Rogelio González
Sergio García de Alba
Michelle Clifford



PREFACE

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to: http://www.freescale.com

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

CONVENTIONS

This document uses the following conventions:

Term or Value	Definition	Examples
Terminal Names	Terminal names are the physical connections and are shown in text as all upper case.	... the external supply voltage VSUP1.
Terminal Values	Terminal values are the currents to/from a terminal and are shown as upper and subscripted text.	In Stop Mode the voltage regulator still supplies the MCU with V_{DD}
Decimal Values	No special symbol attached to the number	1.0 34
Numbers	Considered positive unless specifically noted as a negative value	5.0 -10
Blue Text	Linkable on-line	... refer to Table 1, page 2

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Application Intended Functionality

This reference design of the Human Fall Detection using 3-axis Accelerometer provides an implementation of human activity/fall detection mainly targeted for medical and security applications. This reference design demonstrates the application of the 3-axis low g accelerometers (MMA7260Q), 2.4 GHz RF transceiver data modem for 802.15.4 applications (MC13192) and the Digital Signal Controllers from Freescale (MC56F8013).

1.2 Benefits of Our Solution

The Human Fall Detection using 3-axis Accelerometer is a modular architecture. The user is able to use Digital Signal Processing capability, wireless/serial communication interfaces, 3-axis sensing, external memory for data storage, plus the ability to reprogram the board with different applications with a JTAG interface.

Chapter 2 Quick Start

2.1 Introduction

This section describes the main procedures required to set up and start the Human Fall Detection using 3-axis Accelerometer demo board. The demo is designed to show basic and advanced functionalities of the Human Fall Detection using 3-axis Accelerometer. This document also describes the specific steps for startup and provides additional reference information.

The Human Fall Detection using 3-axis Accelerometer can operate in two different modes which are selected based on the push-button's start-up conditions:

- Normal mode (power-up while the Application push-button is released)
- Memory Dump mode (power-up while the Application push-button is pressed)

In the normal mode, the device will be reading the accelerometer signals to determine whether the person with the device attached has fallen or not. If he/she has fallen, then it will report to a remote terminal (host) the event so it can react accordingly. The device also writes to a non volatile memory the energy expended by the person as well as the person state itself.

The Memory Dump mode will dump all the content of the memory to the RS232 interface of the device. The user can connect the device to a PC and open Hyperterminal so they can retrieve the information from the memory. After the dumping is completed, the application is halted. A reset is required to return to normal operation.

2.2 System Requirements

The Application Software for the Human Fall Detection using 3-axis Accelerometer is placed in the flash memory of the DSC, and then if needed, the device can be reprogrammed for any other application.

No additional software is needed to run the demo in any mode; however a software terminal emulator (i.e. Hyper Terminal, Tera Term) can be used to write to a file the information dumped from the device.

In order to receive the remote alerts from the device, a Wireless extra node can be used. This node can be of any platform interfaced with a MC1319x (ZigBee transceiver). This node will be connected to a PC running a software terminal or a user's created GUI in order to respond to the events from the Human Fall Detection device. The firmware that will run into this device is included in the reference design code files.

2.3 Human Fall Detection using 3-axis Accelerometer Setup

Generally, the Human Fall Detection using 3-axis Accelerometer doesn't require any setup. The module is distributed with the application in Flash Memory and no additional configuration is needed.

Below is a picture of the Reference Design Board.

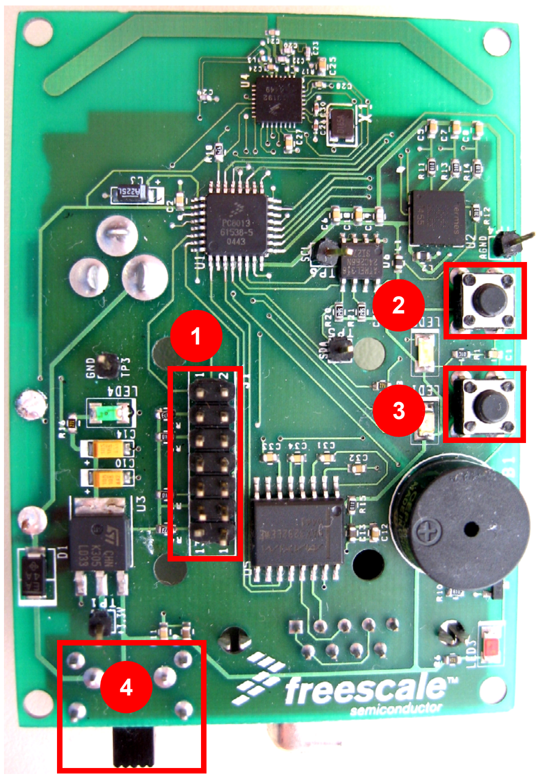


Figure 2-1. Human Fall Detection using 3-Axis Accelerometer Board (Front)

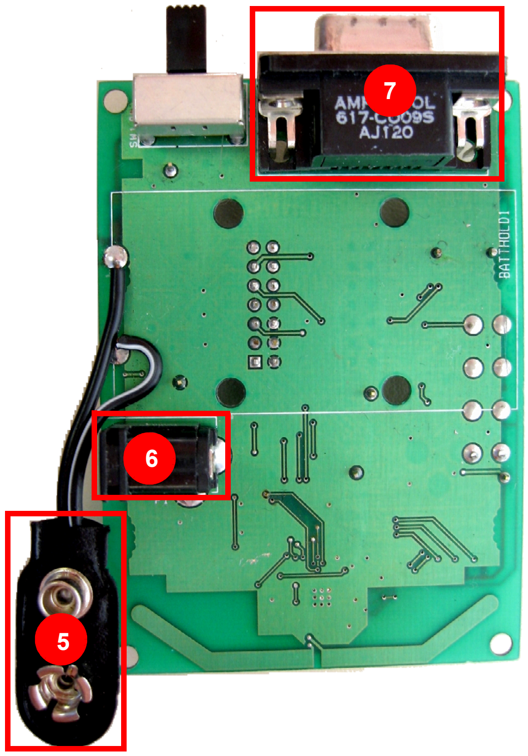


Figure 2-2. Human Fall Detection using 3-Axis Accelerometer Board (Back)

1. JTAG header
2. Reset push-button
3. Application push-button
4. Power-on Switch
5. 9 V Battery holder
6. DC-Power connector
7. Serial port

The following steps are required to run the demo application in the normal mode:

1. Connect Battery OR DC Power supply.
2. Turn Power-On Switch to its ON position.

The following steps are required to run the demo application in the dump mode:

1. Connect Battery OR DC Power supply.
2. Connect the device to a PC using an RS232
3. Open Hyperterminal (19200 bps, 8 bits, 1 stop bit, no parity, no flow control)
4. Go to "Transfer"/"Capture Text". And then select the file to save the data.
5. Select dump Mode by pressing the Application push-button.
6. Turn Power-On Switch to its ON position while the Application push-button is pressed.
7. After the dump is finished (no more movements in the Hyperterminal), turn the device off.

Chapter 3 Hardware Description

3.1 Introduction

This reference design is intended to be a hardware and software platform that enables evaluation of our ZigBee transceiver MC13192, 3-axis accelerometer MMA7260Q, and the MC56F8013 Digital Signal Controller.

This section describes in more detail the electrical design of the module, its features and the advantages of using a hardware architecture like the one proposed in this reference design.

Figure 3-1 presents the block diagram for the hardware module of the reference design. As can be seen, the design is centered around the processing unit (the MC56F8013 DSC). Some peripherals were added to enable user interaction, such as the buzzer, the push buttons, and the LEDs. A JTAG interface was added for programming and debugging. For additional debugging and to allow for serial communication, a serial interface was included. The board is powered either from a 9 V battery or from an external 9 V power supply. The voltage regulator provides 3.3 V. The RF Transceiver is controlled by the DSC, and accomplishes transmission and reception of data packets using the PCB dipole antennas. The antennas connect to the transceiver using matching networks.

The design is simple, yet it has the necessary elements to evaluate many applications, thus reducing development time and costs.

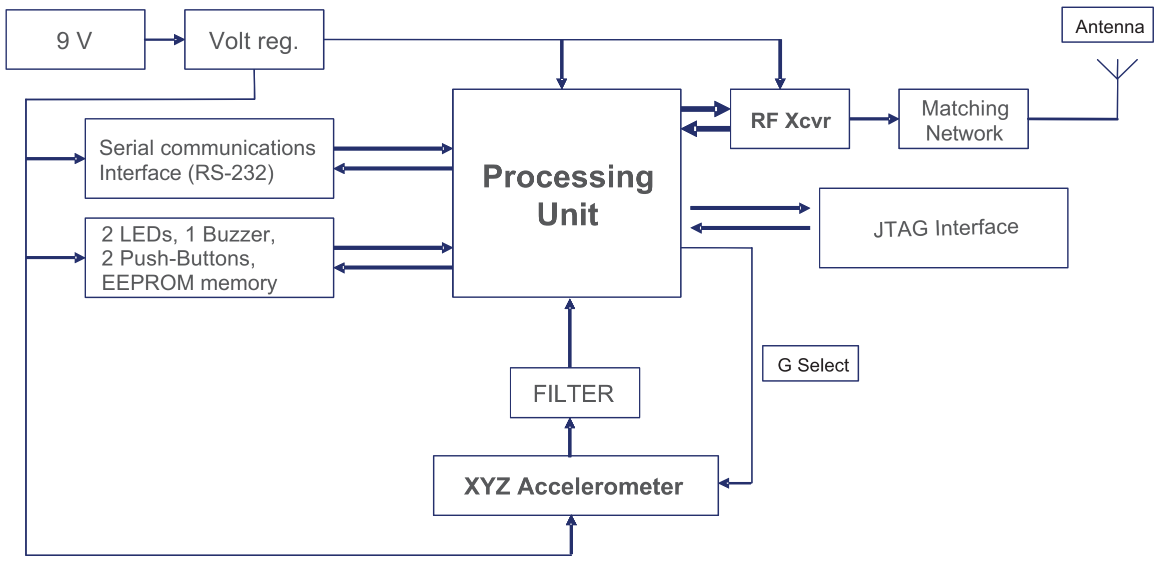


Figure 3-1. Human Fall Detection Building Block

3.2 Technical Data

This section provides technical detail of the Human Fall Detection using 3-axis Accelerometer as well of the components used in this reference design.

3.2.1 Operating Environment

Input Voltage Range: 5.0 to 12.0 Volts

Typical Input Voltage: 9.0 Volts

Typical transmission range (line of sight): 50m

3.2.2 MC56F8013 Digital Signal Controller

The control unit of the Human Fall Detection using 3-axis Accelerometer is a MC56F8013 Digital Signal Controller (DSC). The MC56F8013 is a member of the 56800E core-based family of DSCs. It combines, on a single chip, the processing power of a DSP and the functionality of a microcontroller with a flexible set of peripherals to create a wide range of cost-effective solutions.

3.2.2.1 56800E Core Features

- 56800E 32 MHz Harvard Architecture Core.
- Up to 32 MIPS at a guaranteed 32 MHz execution frequency.
- DSP and MCU functionality in a unified, C-efficient architecture.
- JTAG/Enhanced On-Chip Emulation (EOnCE) for unobtrusive, real-time debugging.
- Four 36-bit accumulators.
- 16- and 32-bit bidirectional barrel shifter.
- Parallel instruction set with unique addressing modes.
- Hardware DO and REP loops available.
- Three internal address buses.
- Four internal data buses.
- MCU-style software stack support.
- Controller-style addressing modes and instructions.
- Single-cycle 16 x 16-bit parallel multiplier-accumulator (MAC).

3.2.2.2 Memory Features

- Architecture permits as many as three simultaneous accesses to program and data memory.
- On-chip memory includes high-speed volatile and non-volatile components
  - 16KB of Program Flash
  - 4KB of Unified Data/Program RAM
- Flash security and Flash protection features prevent unauthorized accesses and accidental modifications.

3.2.3 3-Axis MMA7260Q Low g Acceleration Sensor

One of the typical applications of the MMA7260Q is fall detection since the device can measure low g forces on three axes. This device incorporates a new feature that allows the user to select between 1.5g, 2g, 4g and 6g levels of acceleration.

Additional features of the MMA7260Q:

- Low current consumption: 500 µA.
- Sleep mode: 3 µA
- Low voltage operation: 2.2 V—3.6 V
- Fast turn-on time: 1 ms
- Low package profile of 6 mm x 6 mm x 1.45 QFN
- Fast power-up response time of 1.0 ms
- High Accuracy, frequency and resolution for fall, tilt, motion, positioning, shock and vibration sensing

3.2.4 2.4 GHz Transceiver

The MC13192 is a short range, low power, 2.4 GHz Industrial, Scientific, and Medical (ISM) band transceiver. It contains a complete 802.15.4 physical layer (PHY) modem designed for the IEEE® 802.15.4 wireless standard which supports peer-to-peer, star, and mesh networking. When combined with an appropriate microcontroller (MCU), the MC13192 provides a cost-effective solution for short-range data links and networks. The software and processor can be scaled to fit applications ranging from simple point-to-point systems, through complete ZigBee networking solutions.

- Power supply range: 2.0 to 3.4 V
- 16 Channels
- 0 dBm nominal, programmable up to 4 dBm typical maximum output power
- Buffered transmit and receive data packets for simplified use with low cost MCUs
- Supports 250 kbps O-QPSK data in 5.0 MHz channels and full spread-spectrum encode and decode (compatible with IEEE® 802.15.4 Standard)
- Three power down modes for power conservation:
  - < 1 µA Off current
  - 2.3 µA Typical Hibernate current
  - 35 µA Typical Doze current

- RX sensitivity of -92 dBm (typical) at 1.0% packet error rate
- Four internal timer comparators available to reduce MCU resource requirements
- Programmable frequency clock output for use by MCU
- Onboard trim capability for 16 MHz crystal reference oscillator eliminates the need for external
- Variable capacitors and allows for automated production frequency calibration.
- Seven general purpose input/output (GPIO) signals
- Operating temperature range: -40°C to 85°C
- Small form factor QFN-32 Package
  - Meets moisture sensitivity level (MSL) 3
  - 260°C peak reflow temperature
  - Meets lead-free requirements

3.3 Human Fall Detection Reference Design Functionality

The Human Fall Detection using 3-axis Accelerometer reference design is intended for the medical and personal security market. The idea is to provide information that helps determine if a person has suffered an accident (if the person has fallen) and to provide information related to the fall to determine the magnitude and characteristics of the accident. This application could result extremely useful to the police, firemen, and elderly people.

3.4 Human Fall Detection Reference Design Architecture

Schematics for the Human Fall Detection using 3-axis Accelerometer are provided in BOM and Schematics. The Human Fall Detection building block diagram can be seen in Figure 3-1.

The Human Fall Detection using 3-axis Accelerometer reference design is a system designed to demonstrate the advantages of combining in the way proposed the functionality of the devices integrated.

By using a simple and effective design centered around the DSC (MC56F8013) the complexity of the design is considerably reduced, thus reducing developing time and costs. This architecture also allows to fully benefit from the functionality of the transceiver (MC13192) and the 3-axis accelerometer (MMA7260Q). Another advantage of this simple architecture is that it is easy to add more elements and increase the complexity of the design for more sophisticated applications.

The Human Fall Detection using 3-axis Accelerometer reference design module can be divided into the following basic blocks:

• Digital Signal Controller	• JTAG Interface
• 3-Axis Accelerometer	• RS-232 connection
• 2.4 GHz Transceiver	• Power Supply and Peripherals

3.4.1 Digital Signal Controller (MC56F8013)

The main function of this part of the Human Fall Detection using 3-axis Accelerometer reference design is to control the application. A Freescale 56800E DSC was selected to control the overall application. The DSC with its Hybrid architecture facilitates implementation of both control and signal processing functions in a single device and is the perfect fit to fulfill all the requirements of the Human Fall Detection reference design.

The application uses the following peripheral modules:

- Serial Peripheral Interface module (SPI)
- General-purpose Input/Output pins (I/O)
- 12-bit resolution Analog-to-Digital Converter (ADC)
- Serial Communication Interface module (SCI)
- 16-bit Timers

The DSC directs the MC13192, checks its status, and reads/writes data to the device through the 4-wire SPI port. The transceiver operates as a SPI slave device only. A transaction between the host and the MC13192 occurs as multiple 8-bit bursts on the SPI. The SPI signals are:

- SPI Clock (SCLK): The DSC drives the SPI clock into the MC13192.
- Master Out/Slave In (MOSI): Data from the DSC is presented to the MC13192 (slave) input on the MOSI pin.
- Master In/Slave Out (MISO): The MC13192 presents data to the DSC (master) on the MOSI.
- Chip Enable(CE)/Slave select (/SS): A transaction on the SPI port is framed by the active low CE input signal. A transaction is a minimum of 3 SPI bursts and can extend to a greater number of bursts (see NOTE).

NOTE

Since a SPI transaction must be framed within a Slave Select low-level, it has been implemented using GPIO control of the pin to allow this.

Data from the MMA7260Q accelerometer is sampled using the ADC module in the DSC. Each of the accelerometer's outputs (X-Axis, Y-Axis and Z-Axis acceleration data) is interfaced to three ADC input pins.

The SCI module is used for serial communications with the PC through a RS232 interface. The application transmits data serially at a periodic interval to provide current status of the person (i.e. lying, standing, walking, falling).

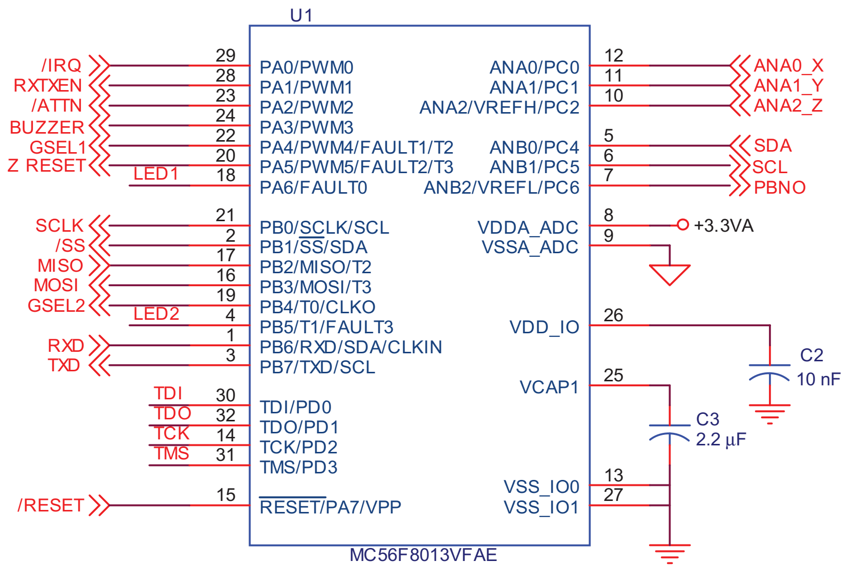


Figure 3-2. Digital Signal Controller Building Block

3.4.2 3-Axis Accelerometer

The 3-Axis accelerometer building block is composed of the MMA7260Q low g acceleration sensor, and some external passive components.

The g-Selection pins (Pins 1 and 2), are used to select the g-Range in order to choose the sensitivity between 1.5g, 2g, 4g and 6g (see the following table).

g-Select 2	g-Select1	g-Range	Sensitivity
0	0	1.5g	800 mV/g
0	1	2g	600 mV/g
1	0	4g	300 mV/g
1	1	6g	200 mV/g

The VDD pin is the power supply input and has a 0.1 µF capacitor to decouple the power source.

VSS is the power supply ground and is connected to the Analog Ground. As we can see in the schematic the ground was separated in order to isolate the Analog and Digital Ground.

N/C (pins 5, 6, 7, 8, 9, 10, 11 and 16) were left unconnected.

The /Sleep Mode PIN is set to 1 using pull up resistor in order to put the device in normal mode operation.

ZOUT is the output voltage of the accelerometer. Z Direction.

XOUT is the output voltage of the accelerometer. X Direction.

YOUT is the output voltage of the accelerometer. Y Direction.

An RC filter with 1.0 k and 0.1 µF was used on the outputs of the accelerometer to minimize clock noise (from the switched capacitor filter circuit).

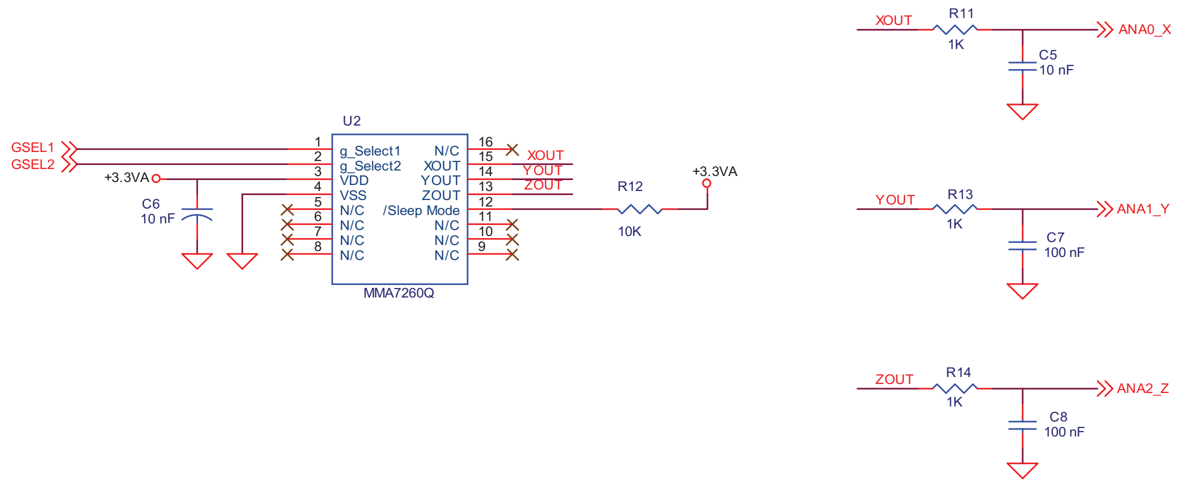


Figure 3-3. 3-Axis Accelerometer Building Block

3.4.3 2.4 GHz Transceiver (MC13192)

The MC13192 building block is composed of the MC13192 ZigBee transceiver, the TX and RX dipole antennas, and the matching networks that connect them to the transceiver. The circuit is shown in Figure 3-4.

Interface with the DSC is accomplished using a four wire serial peripheral interface (SPI) connection and an interrupt request output, as well as these other three connections:

/ATTN: Active Low Attention, Transitions IC from either Hibernate or Doze Modes to Idle.

RXTXEN: Active High. Low to high transition initiates RX or TX sequence depending on SPI setting. Should be taken high after SPI programming to start RX or TX sequence and should be held high through the sequence. After sequence is complete, return RXTXEN to low. When held low, forces Idle Mode.

RSTBi: Active Low Reset. While held low, the IC is in Off Mode and all internal information is lost from RAM and SPI registers. When high, IC goes to IDLE Mode, with SPI in its default state.

The oscillator in the MC13192 is a very low power type, this requires that the load and shunt capacitances of the crystal used be very low to ensure oscillation. For load capacitance 7-8 pF is recommended while the shunt capacitance should be around 2.5 pF and not much higher (below 5 pF). We can suggest two crystals: KDS DSX321G-16.000M-8pF-20-20 and TOYOCOM TSX-10A.

CLKO is an output that wasn’t used in this reference design, but that optionally could be used to provide a clock to the MCU. The CLKO output can be enabled/disabled and the frequency can be programmed via the SPI. The programmable frequencies are 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 62.5 kHz, 32.786+ kHz (default), and 16.393+ kHz. The CLKO output drive strength can also be adjusted.

Elements W101-W104 are transmission line stubs used for impedance matching the antennas to the RF input/output pins. At the frequency of operation, they are the equivalent of an inductor and a resistance.

The LC circuit in the transmit path made up by L3 and C24 is for filtering, it minimizes the spurious emissions.

This design is based on the Sensor Application Reference Design (SARD), therefore more information can be found in the SARD User’s Guide (MC13192SARDUG).

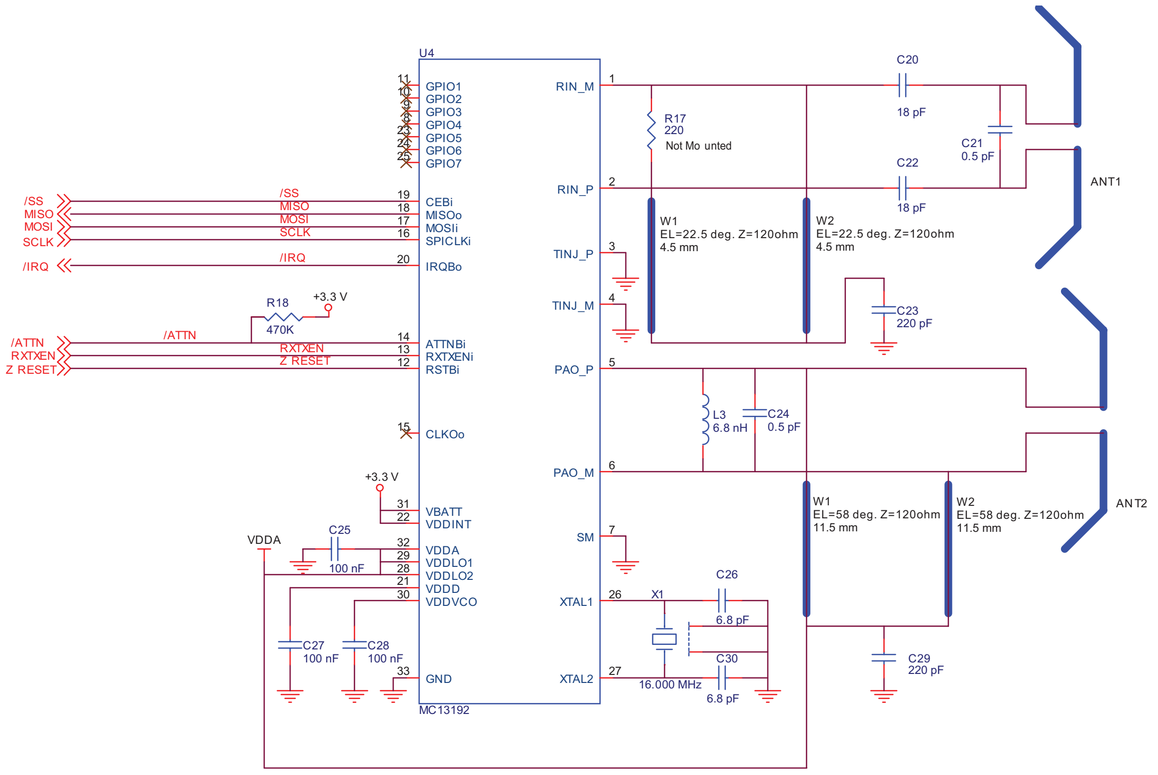


Figure 3-4. MC13192 Building Block

3.4.4 JTAG Interface

The JTAG interface is used for programming and unobtrusive, processor speed-independent, real-time debugging. The TCLK, TMS, TDO, and TDI pins form the JTAG interface onto which the Enhanced On-Chip Emulation port functionality is mapped. A more detailed description of this pins is included below:

Test Data Output (TDO) — This tri-stateable output pin provides a serial output data stream from the JTAG/EOnCE port. It is driven in the shift-IR and shift-DR controller states, and changes on the falling edge of TCK.

Test Data Input (TDI) — This input pin provides a serial input data stream to the JTAG/EOnCE port. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.

Test Mode Select Input (TMS) — This input pin is used to sequence the JTAG TAP controller's state machine. It is sampled on the rising edge of TCK and has an on-chip pull-up resistor.

Test Clock Input (TCK) — This input pin provides a gated clock to synchronize the test logic and shift serial data to the JTAG/EOnCE port. The pin is connected internally to a pull-up resistor. A Schmitt trigger input is used for noise immunity.

/DE — This pin is used for special debug functions and factory testing.

/RESET — This pin allows the JTAG interface to put the DSC in Debug mode, that way EOnCE module gains control of the DSC. When driven low, the JTAG interface generates a reset to the DSC.

NOTE

The TRST pin is not available and is tied to VDD in this DSC.

For more information on the JTAG interface please refer to Joint Test Action Group Port (JTAG) chapter in the 56F8000 Peripheral Reference Manual (MC56F8000RM).

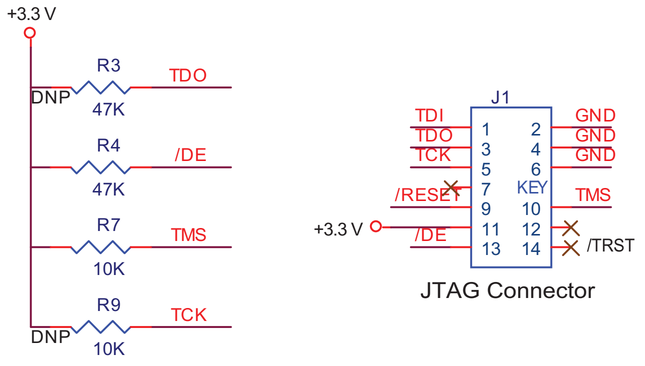


Figure 3-5. JTAG Interface Building Block

3.4.5 RS-232 Connection

By using a common RS-232 transceiver and a serial port from the DSC, the Digital Signal Controller is able to communicate relevant information to a computer or another device. This information exchange could be used for debugging purposes, or as part of an application.

For more information on the serial communication interface please refer to Serial Communications Interface (SCI) chapter in the 56F8000 Peripheral Reference Manual (MC56F8000RM).

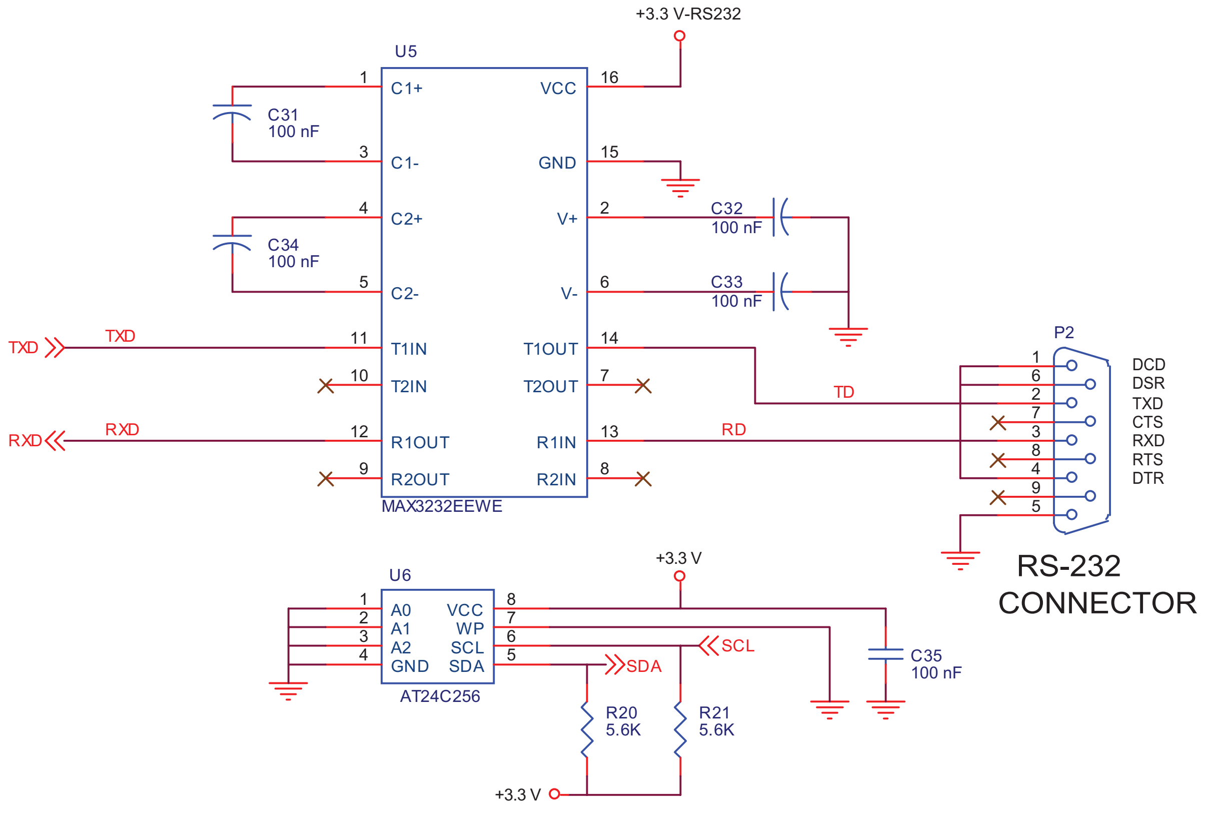


Figure 3-6. RS-232 Circuit

3.4.6 Power Supply and Peripherals

The supply circuit was designed so that the board could be powered either with an external 9 VDC connector, or with a 9 V battery. A reverse battery protection diode D1 was also included. Capacitor C10 should be a 10 µF or larger tantalum capacitor. Electrolytic and ceramic capacitors should be avoided at the output of the voltage regulator since they could cause instability. The capacitors at the input of the voltage regulators are not required for stability, however they improve noise immunity (especially when the external voltage connector is used).

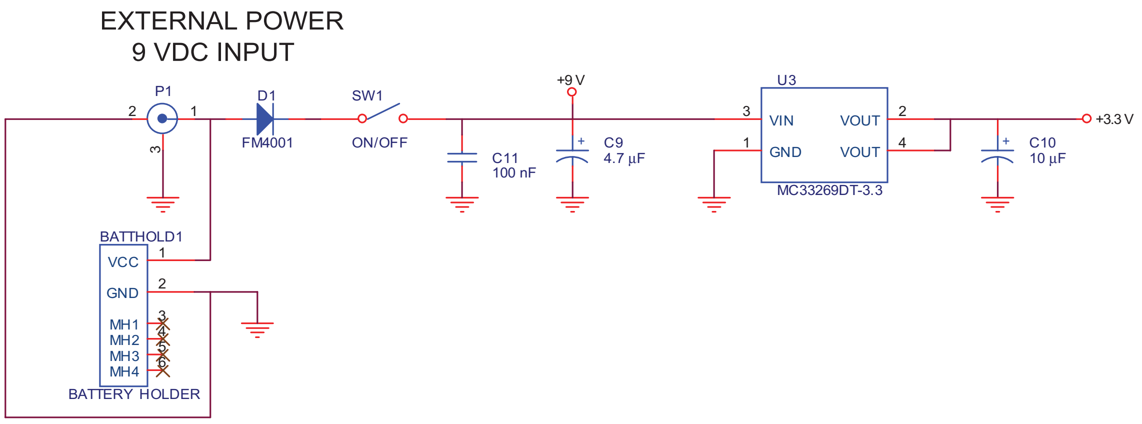


Figure 3-7. Power Supply Circuit

Since this reference design has several parts that are very sensitive to noise, it was decided to separate the power supply of the different sections using the scheme presented in Figure 3-8.

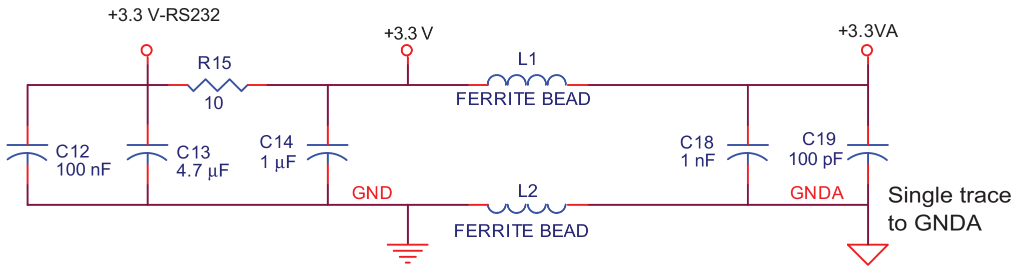


Figure 3-8. Power Supply Filters

Most components make use of the +3.3 V supply, while the +3.3 VA is only used by the 3-axis accelerometer (MMA7260Q). This part requires a very clean power supply to avoid having noise distort the results from the accelerometer (since the output from the accelerometer is an analogue signal, it is very sensitive to noise). +3.3V-RS232 is the supply used for the RS232 transceiver. The power supply for the RS232 was separated since its irregular current consumption produces considerable noise in the power supply. The use of a low pass filter structure made up by the 10 Ohm resistor R15 and the bypass capacitors prevents the noise from the RS232 from disturbing the sensitive analog and RF parts.

The EEPROM memory circuit is presented in Figure 3-9. We decided to include an external memory just to make the system more flexible since it provide more space to store interesting information, for instance to record the events. It is recommended that if a serial interface (RS232) is not used, pins SDA and SCL of the memory IC be connected to pins SDA and SCL of the Digital Signal Controller.

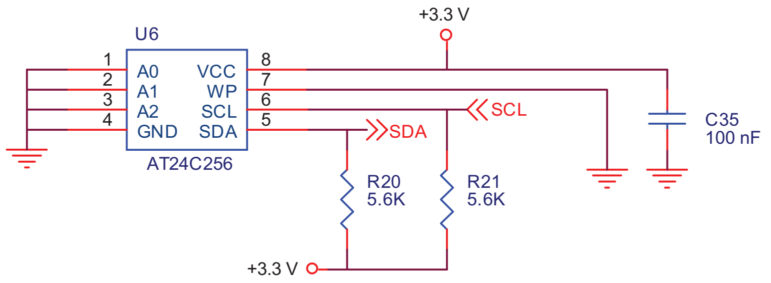


Figure 3-9. EEPROM Memory Circuit

The rest of the peripherals are presented in Figure 3-10. They are a power good LED, two general purpose LEDs, a reset push button, a general purpose push button, and a buzzer that has an LED on while it is active.

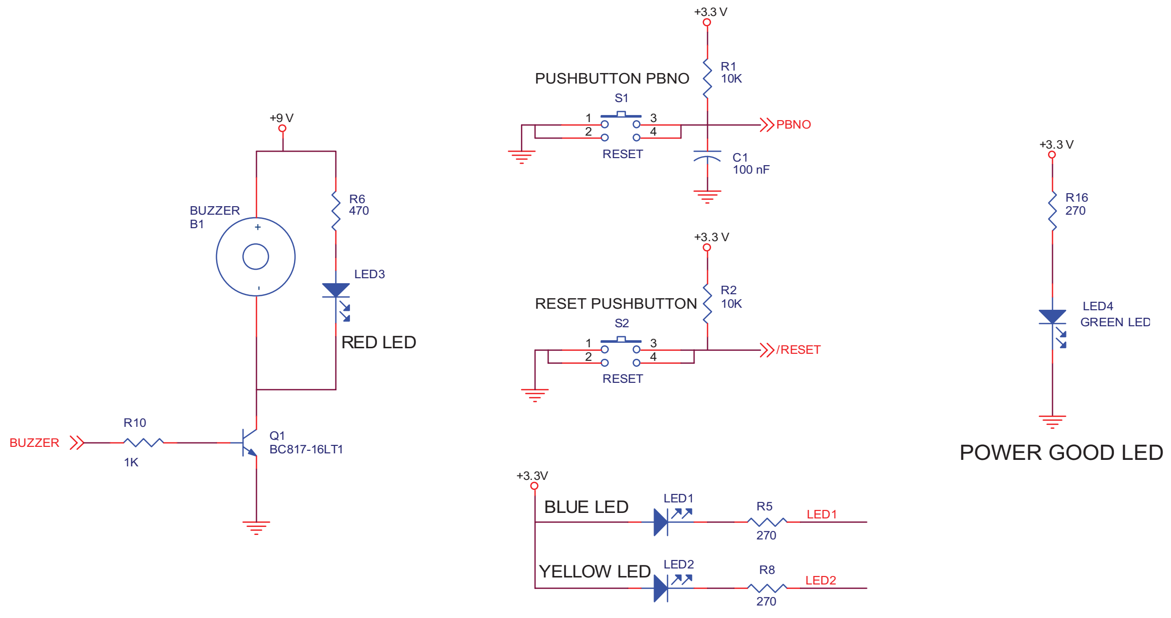


Figure 3-10. Buzzer, Push-Buttons, and LEDs

3.5 Board Layout

This section presents the board layout as well as some related considerations. The design of the board is based on the SARD, therefore they share many characteristics.

3.5.1 Human Fall Detection Board Layout

The component placement in the top layer is shown is Figure 3-11, and the component placement in the bottom layer is presented in Figure 3-12.

The most critical parts in the design of the layout are the RF section and the accelerometer section. The RF section was placed on one extreme of the board for two reasons. One reason was to have the antenna at the border, so the ground plane and the PCB would obstruct the least possible the RF radiation. The other was to try to avoid having currents from other sections go through the ground plane below the RF section and induce noise to this sensitive part of the board. This last reason was also the motivation for placing the accelerometer section in an extreme of the board. In general, placement was done trying to have together the components used for the same function (to minimize routing complexity and board size).

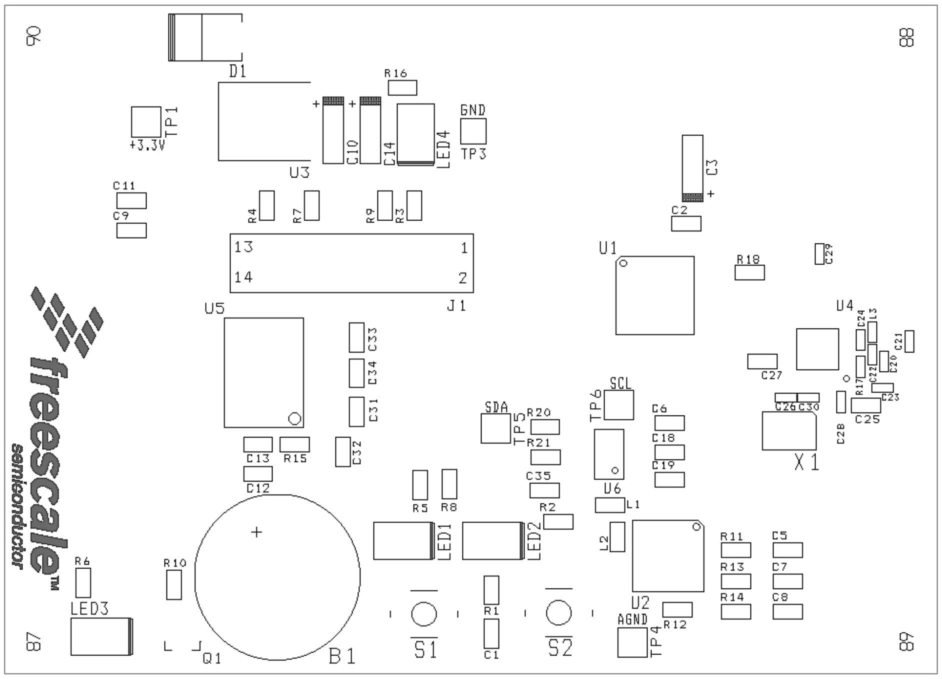


Figure 3-11. Component Placement (TOP)

The components around the transceiver (MC13192) were placed as close as possible to each other and to the transceiver to minimize unwanted transmission line effects (except for W101-W104 which were designed to be transmission lines).

As can be seen in Figure 3-12, very few components were placed in the bottom. Basically they were the on/off switch, the external voltage connector, the DB9 connector, and the battery. Since these have some metal parts, it was decided to place them as far away as possible to the antenna.

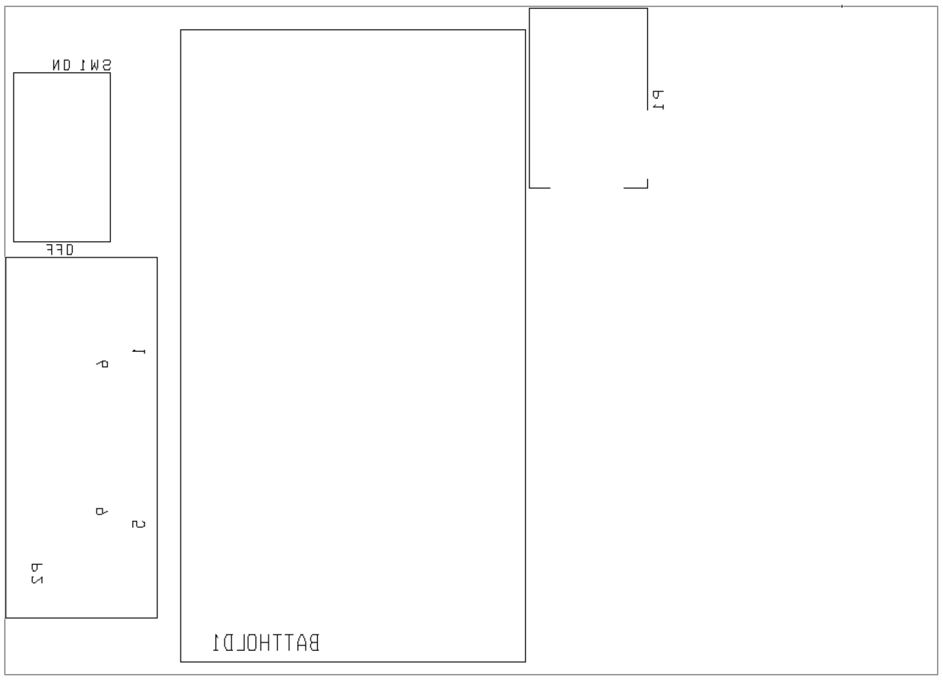


Figure 3-12. Component Placement (BOTTOM)

Copper in the bottom layer is shown in Figure 3-13. It can be observed that an effort was done to try to fracture the ground plane as little as possible (as little routing as possible on this layer, trying to do most of the routing on the top layer). A good ground plane is necessary for correct antenna operation.

The antennas are dipoles, the reason they are slightly bent is to improve the radiation pattern (to make it omnidirectional). The distance between the antennas and the ground plane affects the antenna impedance. Therefore, if a design is based on this design or another reference design (i.e. the SARD) the distances between the ground plane and the antennas should be left exactly the same.

Another important remark is that the ground area under the RF signal path should be unbroken (under components W101-W104, C24, L3, R17, C22, etc).

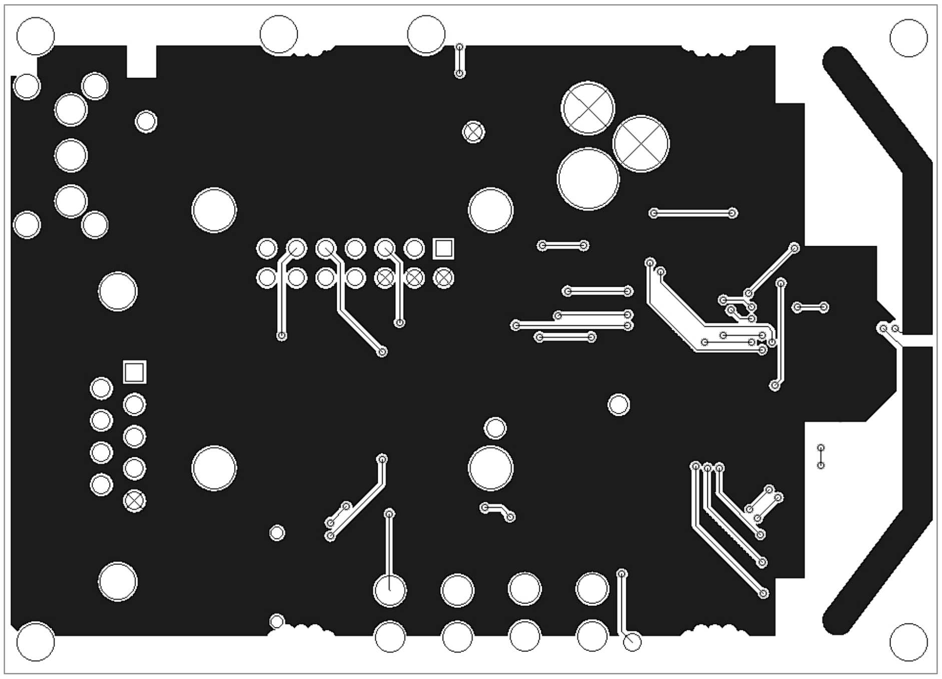


Figure 3-13. Bottom Layer

The top layer of the board is presented in Figure 3-14. Since the placement has already been discussed, now the comments will be on the routing.

In general, the length of the traces was tried to be kept at a minimum and the width of the traces was chosen in proportion to the current they were expected to carry (except for the RF traces).

Most of the routing was attempted to be done on the top layer to fracture the ground plane the least possible and to minimize the use of vias.

Enough space between traces was left (in most cases more than .010 inches) to try to minimize crosstalk.

In the case of the RF traces, their width (as well as their distance to the ground plane and the PCB material dielectric constant) determines the characteristic impedance of the traces (transmission lines). The length of the transmission lines determines their effect. The traces through which the RF signal passes and that were not intended to behave as transmission lines were made short and wide to minimize unwanted transmission line effects.

NOTE

This board was not tested for compliance with governmental regulations.

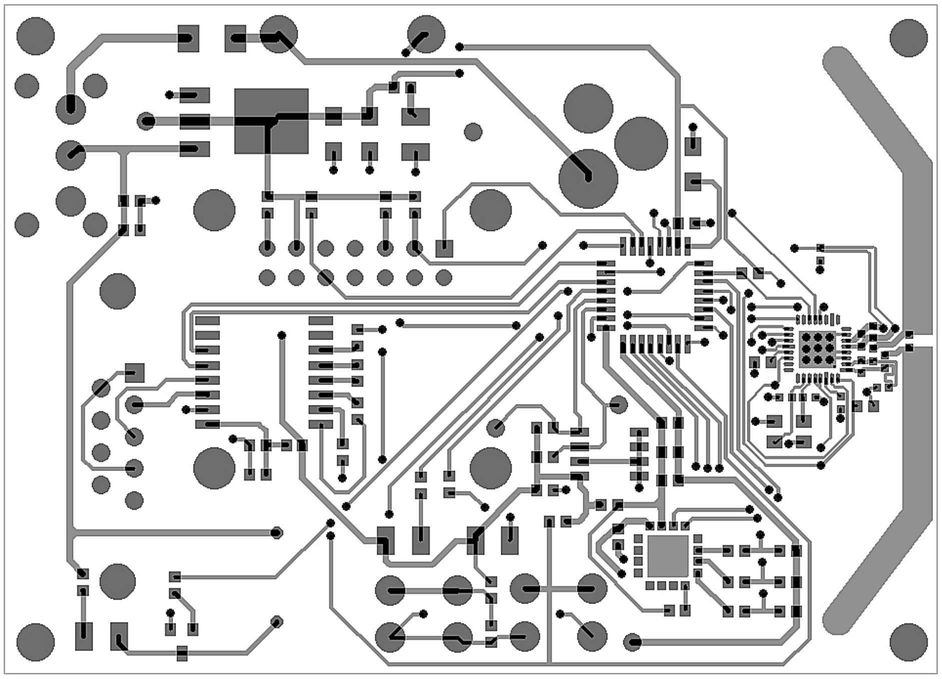


Figure 3-14. Top Layer

3.5.2 General Layout Considerations

General circuit board layout considerations were:

- Proper RF operation (i.e. minimum spurious emissions, etc.)
- Minimizing noise
- Minimizing size and routing complexity
- Easy to prototype an application (i.e. simulate a product)
- This board is not intended for mass production. It's intended to be used as a prototype

In other words the layout was designed in such a way that it enables proper RF operation, signal integrity, noise reduction and use as a prototype.

3.5.3 PCB Characteristics

The main PCB parameters are:

Board X:	3.100"
Board Y:	2.200"
Board Material:	FR-4
Layers:	2
Thickness:	0.031"
Finished Copper:	1.0 oz.
Solder Mask:	2 sided
Solder Mask Color:	Green
Silkscreen:	2 sided
Silkscreen Color:	White
Min Trace:	.005"
Min Space:	.004"
Surface Mount:	1 surface
Smallest Hole:	0.010-0.016
Number of Holes:	Under 500

The dielectric constant of FR4 is around Er=4.2 (it varies depending on the composite structure of the material). The effective dielectric constant seen by the transmission lines will depend on how the dielectric constant of FR4 and of the air average. Therefore the thickness of the board will have an effect on the effective dielectric constant Er. If the board material used or the thickness of the board are changed, the transmission lines and antennas would have to be recalculated.

3.5.4 16-LEAD QFN Footprint for MMA7260Q

Care must be taken when designing the footprint for the 16-LEAD QFN to guarantee a proper assembly of the device. The minimum recommended footprint provided in the Data Sheet Rev 1 on page 7 was taken as land pattern for this reference design. It is highly recommended to design the board with a solder mask layer to facilitate neat soldering. For further information regarding the suggested methods for soldering the Quad Flat No-Lead (QFN) package can be found in the reference section (Appendix C. References).

Chapter 4 Software Description

4.1 Introduction

This section of the reference design provides a complete documentation of the Human Fall Detection using 3-axis Accelerometer software.

4.1.1 Software Basics

All embedded software of this project was written using CodeWarrior for Freescale DSP56800/E 7.1.

The application was developed using Processor Expert, which offers the capability to easily select and configure each of the different peripherals in the Digital Signal Controller by clicking on different drivers (beans) and generating full-functioning code for each one of them.

4.1.2 Application Basics

The application was developed trying to satisfy the requirements that the Human Fall detection problem presents. Such requirements are:

- Detect the event of a human fall with high accuracy.
- Report the fall ASAP to the host. This requirement is met. But no application has been created, so the customer would need to create its own GUI and interface it with the services that the custom requirements need (i.e. Internet access for email reporting, Phone Network access for 911 reporting, etc.)
- Store in the external memory the after fall conditions. The application saves all the current “human state” and “energy expenditure” data before and after the fall. This data is downloadable through RS232, when starting the device in Dumping Mode.

NOTE

This reference design does not implement a Low-Power strategy. However, the software may be modified to implement Low Power consumption.

In order to meet these requirements, several approaches could have been taken, but based on the existing documentation (see bibliography for more information), the following software specific requirements have been declared(1):

- The software should sample all three axis accelerometer signal at 45 Hz.
- The software should implement a median filter of 13 samples in order to reduce the noise in the signal and provide the FIR with a cleaner signal.
- In order to obtain a dynamic acceleration needed in the following requirement, a High pass FIR filter of order 35, with stop frequency of 0.5 Hz should be applied to each axis’ clean signal.
- The Energy Expenditure is the sum of the integration of the square of the dynamic acceleration of the person for the specified time:

$$Energy\ Expenditure = \alpha IA = \alpha\left(\int\left|x^2\right| + \int\left|y^2\right| + \int\left|z^2\right|\right)$$

  In our case, α is 1.

  The calculation of the Energy Expenditure will yield one value out of a time window of 0.8 seconds.
- The detection of a Human Fall will be performed comparing the Energy Expenditure value of a 0.8 seconds window to a specific threshold obtained by experiments.

In the following pages you will see an example of the data after each part of the data processing.

1. These requirements were obtained from the article “Determining Activity Using Triaxial Accelerometer”, written by M.J. Mathie, N. H. Lovell, A. C. F. Coster and B.G. Celler. *Proceedings of the Second Joint EMBS/BMES Conference, Houston TX, USA; October 23-26, 2002.* IEEE.

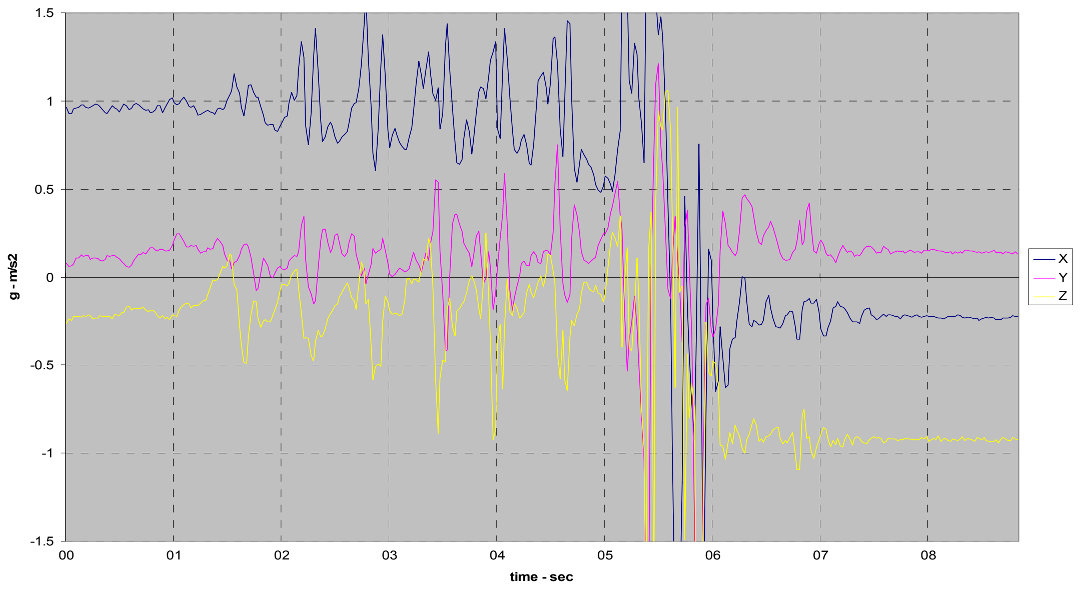


Figure 4-1. Input Data From the Triaxial Accelerometer

Figure shows the data that is coming from the Triaxial accelerometer, sampled at 45 Hz. In the first part (0-1 sec) we see the signals stable. This is because the person with the device attached was standing.

Then, we can see a periodic movement from the information of the accelerometer when the person was walking.

From 5.5 sec to 6.1 sec, the person has fallen, and it is very clear.

After the fall, we see that the offset in the signals varied, representing a change in the inclination of the device (the person is fallen) with some movement at the beginning, and totally quiet after.

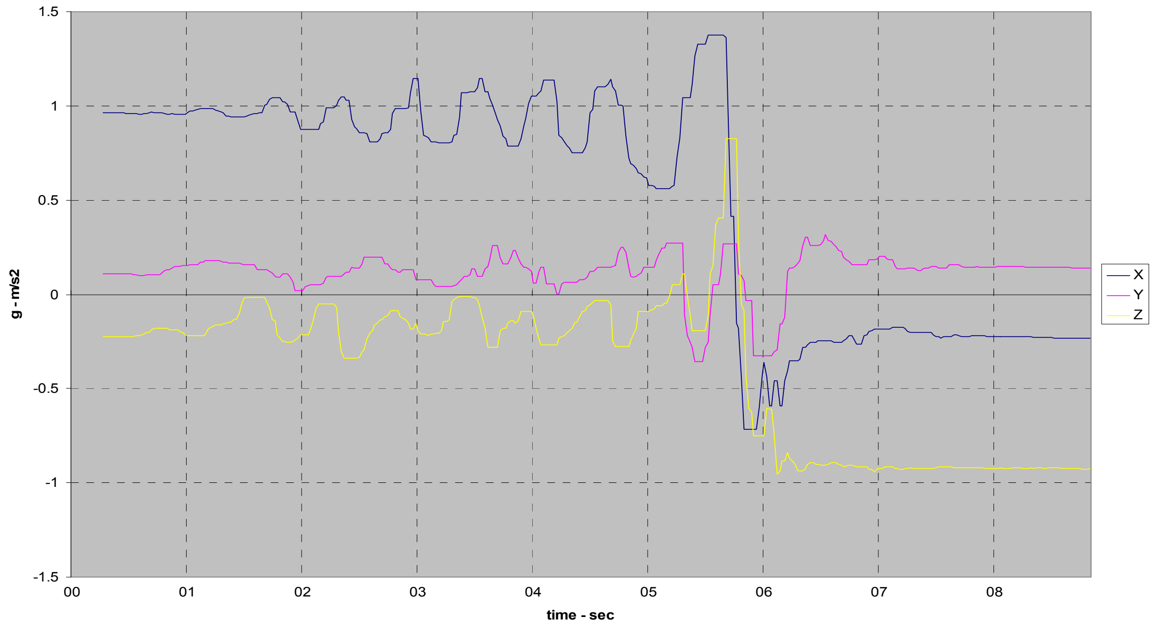


Figure 4-2. Signals After the Median Filter

Figure shows the status of the signals after the median filter. The median filter is used to take out the high frequency noise from the signal.

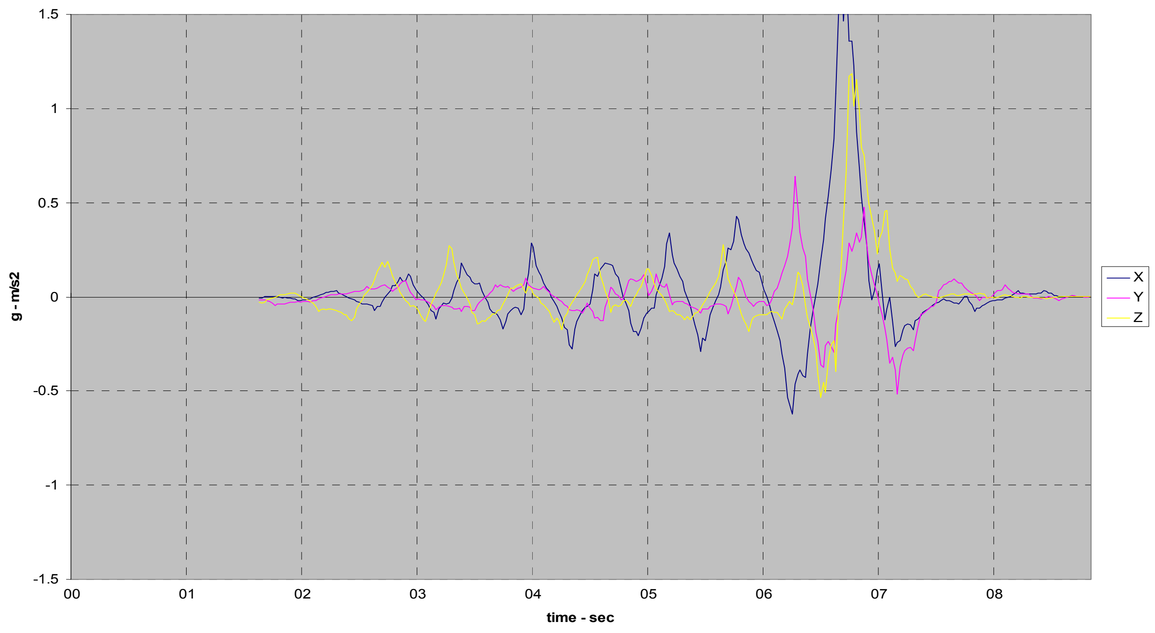


Figure 4-3. Signals After the High Pass FIR Filter

Figure shows the result of the high pass FIR filter. This filter is used to remove the offset from the signals (take out the gravity component). Now we have only “dynamic” acceleration.

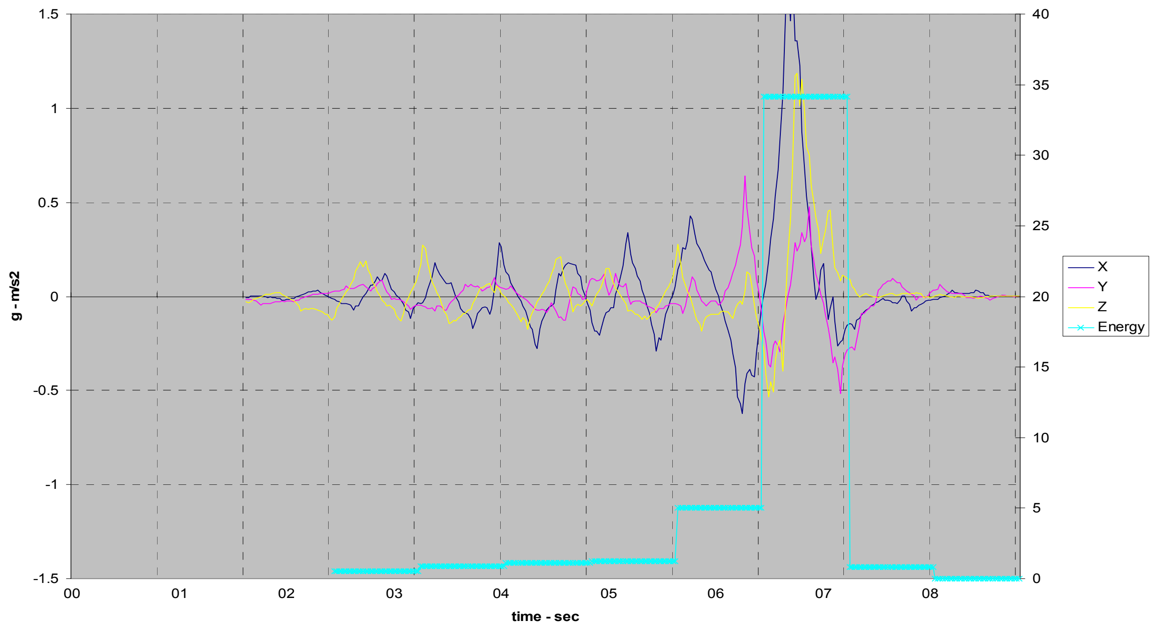


Figure 4-4. Graphs Showing the High Pass Filter Response and the Energy Expenditure

The Energy Expenditure is calculated only once every 0.8 seconds (36 samples) and it yields only one value per 0.8 second window.

As you can see, there is a huge difference between the window in the second 7 and second 3 or 4. This is why we used this method to classify windows. Many tests were performed and a threshold value was calculated to differentiate when an energy level is a fall and when it is not. The thresholds can be found in the source code.

4.2 Project Introduction

This section gives an introduction and description of the software implementation of the Human Fall Detection using 3-axis Accelerometer.

4.2.1 Coding Convention

All source codes were written using several rules and guidelines which make the final product more readable, reusable and portable.

Here is a list of the most important ones:

- Variables:
  - — The variable types used in the project are the following:

Variable Type	Explanation
INT8	Integer, signed, 8 bits
INT16	Integer, signed, 16 bits
INT32	Integer, signed, 32 bits
UINT8	Integer, unsigned, 8 bits
UINT16	Integer, unsigned, 16 bits
UINT32	Integer, unsigned, 32 bits

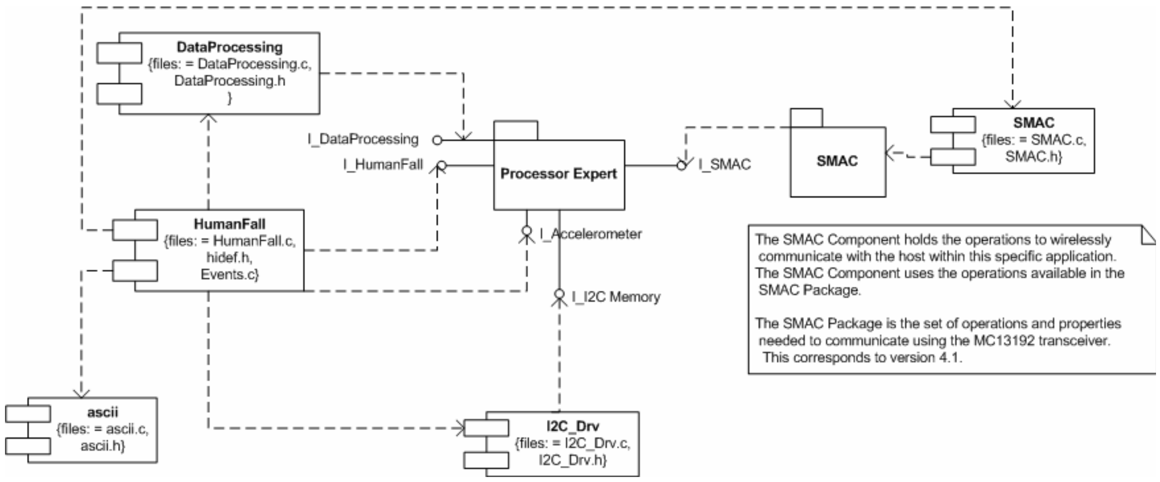
  - — All variables are declared with a prefix defining its type according to the following table:

Prefix	Description
g	Global
i8	signed integer 8 bits
i16	signed integer 16 bits
i32	signed integer 32 bits
u8	unsigned integer 8 bits
u16	unsigned integer 16 bits
u32	unsigned integer 32 bits
b	Boolean
s	Structure
p	Pointer
a	Array

- Functions:
  - — All function names will have uppercase only at the beginning and to distinguish between words. The use of underscore is not allowed.
- Macros:
  - — All macro names will be in uppercase, with underscores to distinguish between words.

4.2.2 List of Project Files

This project was written using the Metrowerks CodeWarrior for DSP56800/E V7.1. There are many files involved within the project, but it is recommended to modify only the following files, which are required for the project:



The SMAC Component holds the operations to wirelessly communicate with the host within this specific application. The SMAC Component uses the operations available in the SMAC Package.

The SMAC Package is the set of operations and properties needed to communicate using the MC13192 transceiver. This corresponds to version 4.1.

Figure 4-5. Software File Mapping

As shown in Figure 4-5, we use 2 packages (which include a lot of files that are of no interest for this document) which are the SMAC (Simple MAC – Used for wireless communication with the host) and the Processor Expert packages (which includes all the files supporting the application relation with the hardware).

Besides the two packages, you can see 5 **components**; each one is a set of files that holds specific functionality of the application. For instance, the DataProcessing component is conformed by two files: DataProcessing.c and DataProcessing.h and those files hold and implement the functionality of the data processing within the application. This component depends on the interface I_DataProcessing of Processor expert, and the main application component (the **HumanFall component**) depends on the DataProcessing component.

An interface attached to the Processor Expert package is a group of “beans” of processor expert (a **bean** is a software object that encapsulates the functionality of basic elements of embedded systems like CPU core, CPU on-chip peripherals, standalone peripherals, virtual devices and pure software algorithms) that provides the functionality and methods needed by some components. Most of the “beans” used in this project are for communication with the hardware (ADC, GPIO…), but there are others that are also software-only beans (like the DFR1 bean, which implements a library for calculating FIRs).

For instance, the SMAC package depends on the interface I_SMAC implemented by the Processor Expert package. This interface implements the beans for SPI, IRQ, some GPIOs (ATTN, RTXEnable, CE…), and those beans are needed to communicate the SMAC with the MC13192 transceiver. Below is a list of the beans (and its types) implemented by each interface of Processor Expert:

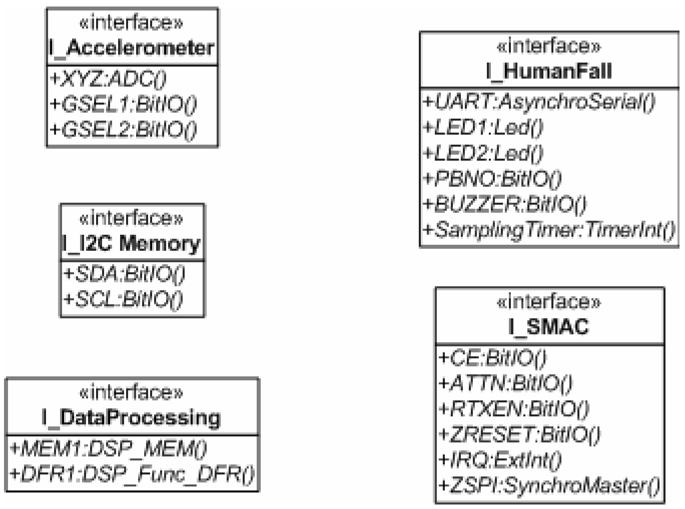


Figure 4-6. “Interfaces” (Groups of Beans) used by Processor Expert

I_Accelerometer: The interface I_Accelerometer implements three beans: 1 ADC bean (for reading data from Triaxial accelerometer) and 2 gpio (BitIO) beans for selecting the current range of measurements.

I_I2C Memory: The Interface I_I2C Memory implements only 2 BitIO beans: one for the SDA signal and the other for the SCL signal (it interfaces with an I2C EEPROM, the I2C protocol is bit banged: implemented by software).

I_Data Processing: I_DataProcessing interface implements the software beans MEM1 (of type DSP_MEM: dynamic memory for digital signal controllers) and DFR1 (of type DSP_Func_DFR, which implements some fixed point DSP operations, like FIR operations).

I_SMAC: The I_SMAC interface implements a SynchroMaster bean for SPI communications with the MC13192 (ZSPI), one ExtInt bean for implementing an external Interrupt (IRQ) and 4 BitIO beans for interfacing with the MC13192 (CE, ATTN, RTXEN, ZRESET). For more information on how to interface the MC13192 with a MCU, please refer to document MC13192RM.

I_HumanFall: The interface I_HumanFall holds the rest of the beans needed for the application itself. The bean UART (AsynchroSerial) permits the application to send data through the RS232 port on the board; the LED1 and LED2 beans (of type Led) permits to turn on and of each of the LEDs within the board; the PBNO bean (of type BitIO) permits to read the value of the pushbutton of the board; the BUZZER bean (BitIO) permits the application to interface with the buzzer located in the board and the SamplingTimer (TimerInt) gives the 45 Hz interruption that is used as the sampling frequency.

If you need further details on the description of beans, please refer to the Processor Expert documentation.

4.2.3 Used MCU Peripherals

This section briefly describes all the MC56F8013 peripherals used in the project.

It gives an overall summary picture of the necessary DSC resources.

Module	DSC Resource	Purpose
HumanFall component	SCI (UART) GPIO (LED1:portA6) GPIO (LED2:portB5) GPIO (PushB:portC6) GPIO (BUZZER:portA3) Timer (SamplingTimer: timer2 Free)	To communicate with the computer To interface the LED1 of the board To interface the LED2 of the board To interface the PushButton To interface with the Buzzer Generates an interrupt at the sampling frequency
Accelerometer component	ADC (XYZ:ana0-2) GPIO (GSEL1:portA4) GPIO (GSEL2:portB4)	To measure the accelerometer data Line for selecting the Acc range Line for selecting the Acc range
I2C Memory Component	GPIO (SCL:portC5) GPIO (SDA:portC4)	Occurs when a byte is received from the UART bean.
SMAC package	SPI (ZSPI) GPIO (IRQ:portA0) GPIO (ZRESET:portA5) GPIO (RTXEN:portA1) GPIO (ATTN:portA2) GPIO (CE:portB1)	Occurs when the MC13192 tries to contact the MCU due to an event in it. It calls the IRQIsr function located within the SMAC package.

4.2.4 Used Interrupts

All interrupts used within the Human Fall Detection using 3-axis Accelerometer are listed in Table 4-1.

Table 4-1. Interrupts

Module	Event function	Purpose
HumanFall component	SamplingTimer_OnInterrupt	It is the sampling frequency. It starts the ADC conversion
HumanFall component	XYZ_OnEnd	Occurs when the ADC conversion of the three axis is done. It formats the ADC readings for the data processing stage. It also computes the median filter and triggers the DataProcessing when the buffer is ready.
HumanFall component	UART_OnRxChar	Occurs when a byte is received from the UART bean.
SMAC package	IRQ_OnInterrupt	Occurs when the MC13192 tries to contact the MCU due to an event in it. It calls the IRQIsr function located within the SMAC package

4.2.5 Main Variables of the project

In this section a brief description of the most important variables and flags is given.

The following variables are declared globally and used to control the functionality of the module:

Variable Name	Type	Description
gu8InputReadySignal	UINT8	This is Triggered when FIR Buffer full to signal Data Processing
gu8DataProcessingSignal	UINT8	This variable is “1” when in the data processing stage
gu8ResetSignal	UINT8	The reset signal used to reset the system
gu8DP_HPFilterSignal	UINT8	This variable is “1” when in High Pass FIR (within the DataProcessing)
gu8WCOM_TxRequest	UINT8	Variable used to signal the transmission of the current state through the air.
gsInputData	tInputData	Holds the last values read by ADC

4.2.6 Memory usage

The following table shows the Human Fall Detection using 3-axis Accelerometer software memory usage.

Table 4-2. Memory Usage

Type of Memory	Total Size (Bytes)	Used Memory (Bytes)	Free Memory (%)
FLASH	10000h	3184h	Approx. 80%
EEPROM	400h	1F4h	Approx. 51%
RAM	1000h	3B0h*	Approx. 77%
RAM for STACK	100h	-	-

* Includes RAM for Stack

4.3 Software Implementation

In this section the complete description of the key software modules of the Human Fall Detection using 3-axis Accelerometer is given.

4.3.1 Software Architecture

The software architecture for this reference design was designed as simple foreground-background architecture. In the background, the reading of the ADC, the sampling frequency and the Interrupts from the RF Transceiver are processed.

The Foreground holds the System initialization, Data Processing, Human State computation and Wireless communications with the host.

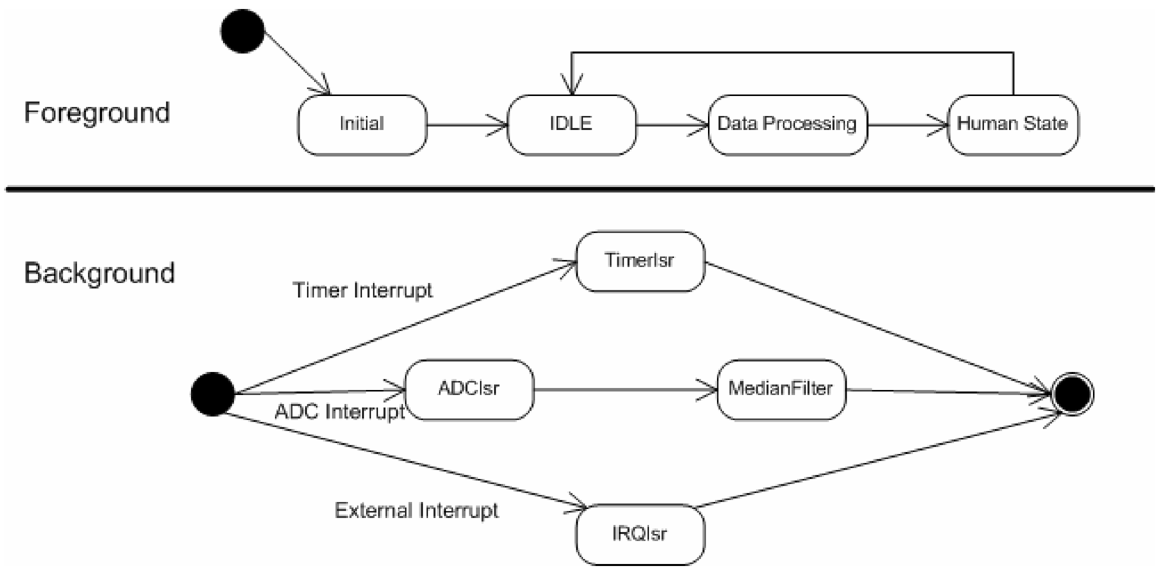


Figure 4-7. Software Architecture

Details on the Foreground can be seen in Figure 4-8.

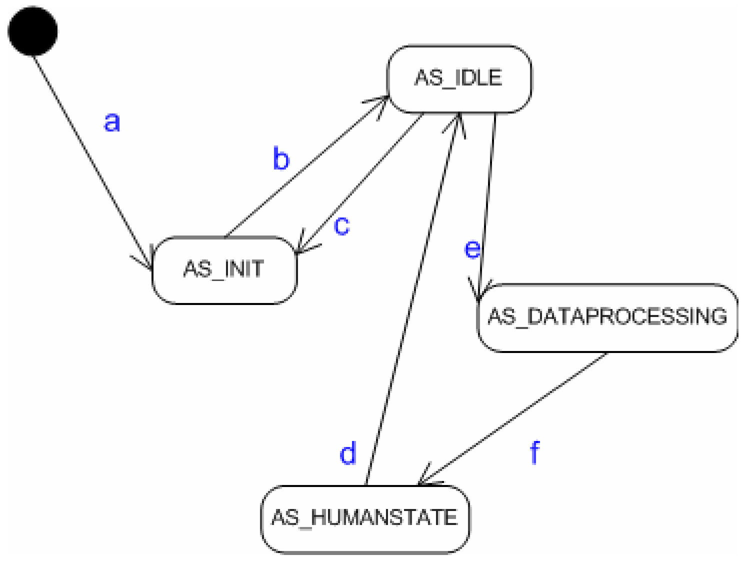


Figure 4-8. Main Application State Machine

States Definition:

AS_INIT: In this state, the system is initialized: Buffers are reset and the system is configured to acquire the data from the ADC.

AS_IDLE: In this state, all the service for user-level hardware is done: Pushbutton, Leds, and Buzzer. The signal gu8InputReadySignal is also checked. If it is true, it will transition to state AS_DATAPROCESSING.

If the signal gu8ResetSignal is true, it will transition to AS_INIT.

The application is most of the time in this state.

AS_DATAPROCESSING: In this state, the DataProcessing function (PDProcessData) is called, and thus, its state machine starts to run.

When it returns from the Data processing, it transitions to AS_HUMANSTATE.

AS_HUMANSTATE: Computes the Human State and responds to it.

User defines what behavior should be taken when a human fall condition is met.

It returns to AS_IDLE state.

Transitions Definition:

a: Default start transition

b: After initializing system, always transition to AS_IDLE state

c: Is gu8ResetSignal is asserted, then transition to AS_INIT state

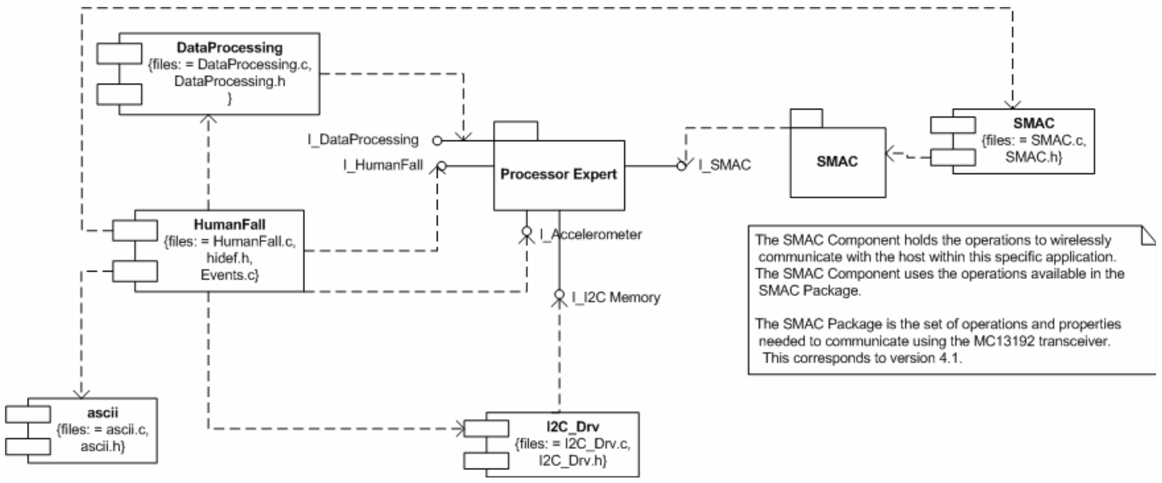
d: Always go to AS_IDLE state

e: If signal gu8InputDataReadySignal is asserted, then transition to Process data. This signal is asserted once the High pass filter buffer is full. This buffer fills after having received data from the ADC and filtered through the median filter.

f: Always go to AS_HUMANSTATE.

4.3.2 Details of the Application's Components

This is the description of each component used in the Human Fall Detection System.



The SMAC Component holds the operations to wirelessly communicate with the host within this specific application. The SMAC Component uses the operations available in the SMAC Package.

The SMAC Package is the set of operations and properties needed to communicate using the MC13192 transceiver. This corresponds to version 4.1.

Figure 4-9. Important Files used in the Human Fall Detection System

4.3.2.1 Human Fall Component

This component holds the behavior of the entire application. Please refer to the "4.3.1 Software Architecture" section for further details.

4.3.2.2 Data Processing Component

This component implements the data processing needed for the application, as defined in section Software Basics. This component is supposed to implement everything needed for processing the data, which in this case we refer to: FIR computation, EnergyExpenditure calculation, median filter and other calculus.

This component is located physically in the files "DataProcessing.c" and "DataProcessing.h".

This component can be divided into 4 different logical modules:

Generic Data processing: This module has all the functions and variables required to direct the different steps in the data processing. One of its functions implements the Data Processing state machine (explained later).

Median Buffer Framework: This module defines and implements the structures, functions, variables and defines needed to perform the median filter calculus.

FIR Framework: This module defines and implements all the structures, functions, variables and defines required to execute a FIR.

Energy Expenditure Framework: Everything needed to perform the calculation of the Energy Expenditure is located here.

Please refer to the following figure for more details on the Data Processing modules:

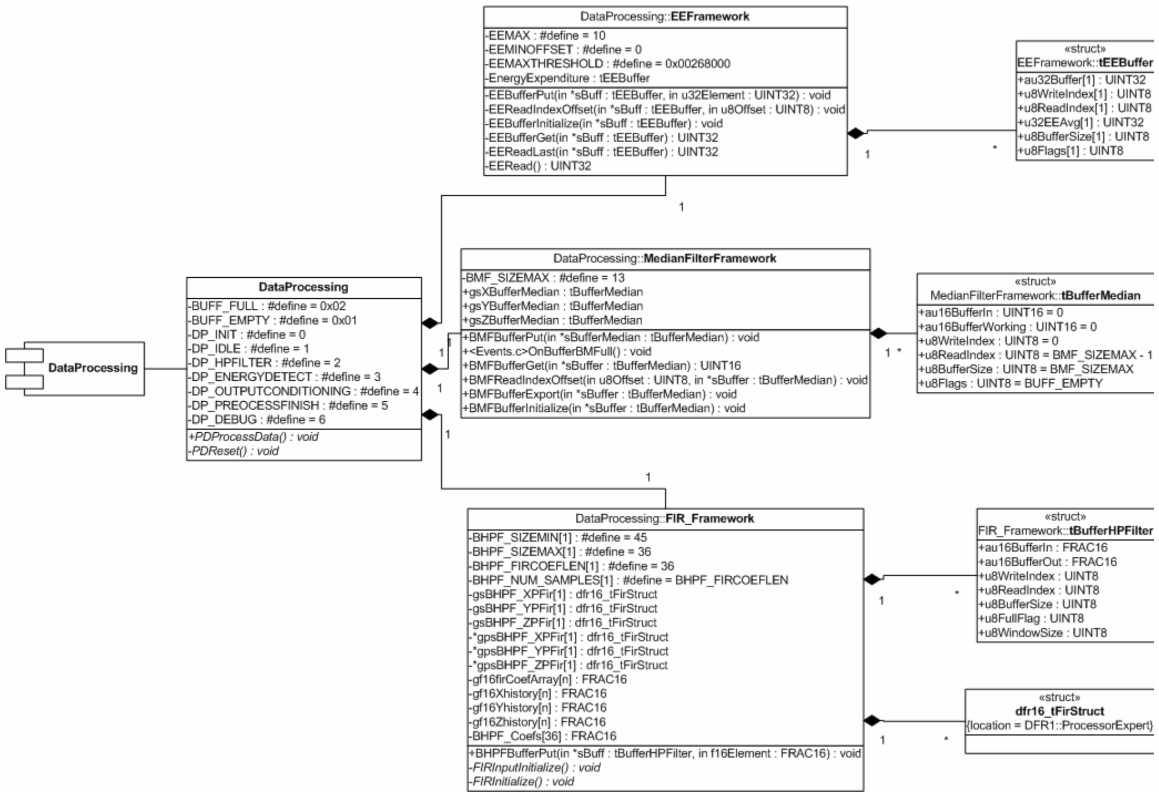

DataProcessing::EEFramework
-EEMAX : #define = 10
-EEMINOFFSET : #define = 0
-EEMAXTHRESHOLD : #define = 0x00268000
-EnergyExpenditure : tEEBuffer
-EEBufferPut(in *sBuff : tEEBuffer, in u32Element : UINT32) : void
-EEReadIndexOffset(in *sBuff : tEEBuffer, in u8Offset : UINT8) : void
-EEBufferInitialize(in *sBuff : tEEBuffer) : void
-EEBufferGet(in *sBuff : tEEBuffer) : UINT32
-EEReadLast(in *sBuff : tEEBuffer) : UINT32
-EERead() : UINT32
«struct»
EEFramework::tEEBuffer
+au32Buffer[1] : UINT32
+u8WriteIndex[1] : UINT8
+u8ReadIndex[1] : UINT8
+u32EEAvg[1] : UINT32
+u8BufferSize[1] : UINT8
+u8Flags[1] : UINT8
1
*
1
DataProcessing::MedianFilterFramework
-BMF_SIZEMAX : #define = 13
+gsXBufferMedian : tBufferMedian
+gsYBufferMedian : tBufferMedian
+gsZBufferMedian : tBufferMedian
+BMFBufferPut(in *sBufferMedian : tBufferMedian) : void
+<Events.c>OnBufferBMFull() : void
+BMFBufferGet(in *sBuffer : tBufferMedian) : UINT16
+BMFReadIndexOffset(in u8Offset : UINT8, in *sBuffer : tBufferMedian) : void
+BMFBufferExport(in *sBuffer : tBufferMedian) : void
+BMFBufferInitialize(in *sBuffer : tBufferMedian) : void
«struct»
MedianFilterFramework::tBufferMedian
+au16BufferIn : UINT16 = 0
+au16BufferWorking : UINT16 = 0
+u8WriteIndex : UINT8 = 0
+u8ReadIndex : UINT8 = BMF_SIZEMAX - 1
+u8BufferSize : UINT8 = BMF_SIZEMAX
+u8Flags : UINT8 = BUFF_EMPTY
1 *
DataProcessing
DataProcessing
-BUFF_FULL : #define = 0x02
-BUFF_EMPTY : #define = 0x01
-DP_INIT : #define = 0
-DP_IDLE : #define = 1
-DP_HPFILTER : #define = 2
-DP_ENERGYDETECT : #define = 3
-DP_OUTPUTCONDITIONING : #define = 4
-DP_PREOCESSFINISH : #define = 5
-DP_DEBUG : #define = 6
+PDProcessData() : void
-PDReset() : void
1
1
1
1
DataProcessing::FIR_Framework
-BHPF_SIZEMIN[1] : #define = 45
-BHPF_SIZEMAX[1] : #define = 36
-BHPF_FIRCOEFLEN[1] : #define = 36
-BHPF_NUM_SAMPLES[1] : #define = BHPF_FIRCOEFLEN
-gsBHPF_XPFir[1] : dfr16_tFirStruct
-gsBHPF_YPFir[1] : dfr16_tFirStruct
-gsBHPF_ZPFir[1] : dfr16_tFirStruct
-*gpsBHPF_XPFir[1] : dfr16_tFirStruct
-*gpsBHPF_YPFir[1] : dfr16_tFirStruct
-*gpsBHPF_ZPFir[1] : dfr16_tFirStruct
-gf16firCoefArray[n] : FRAC16
-gf16Xhistory[n] : FRAC16
-gf16Yhistory[n] : FRAC16
-gf16Zhistory[n] : FRAC16
-BHPF_Coefs[36] : FRAC16
+BHPFBufferPut(in *sBuff : tBufferHPFilter, in f16Element : FRAC16) : void
-FIRInputInitialize() : void
-FIRInitialize() : void
«struct»
FIR_Framework::tBufferHPFilter
+au16BufferIn : FRAC16
+au16BufferOut : FRAC16
+u8WriteIndex : UINT8
+u8ReadIndex : UINT8
+u8BufferSize : UINT8
+u8FullFlag : UINT8
+u8WindowSize : UINT8
1
*
«struct»
dfr16_tFirStruct
{location = DFR1::ProcessorExpert}
1
*

Figure 4-10. Internal Structure of the DataProcessing Component

The previous diagram shows the internal elements of the Data Processing component. The component “DataProcessing”, which is physically located in the files “DataProcessing.c” and “DataProcessing.h” implements the functionality described by the rest of the diagram. As you can see, the four modules are shown graphically as a class diagram:

- The generic module’s elements (variables, defines and functions) is shown within the *DataProcessing* class. Its properties are only defines for the state of the buffers used by any of the frameworks (BUFF_FULL and BUFF_EMPTY) and the states of the DataProcessing state machine (which is implemented in the PDProcessData() function). The internal function PDReset() is used to initialize all the frameworks’ buffers as well as the state machine.
- The EEFramework’s properties are three internally used defines and an EnergyExpenditure buffer (which will hold the value of the energy expenditure of the last 10 windows of 0.8 seconds). Then we found the functions used internally by the DataProcessing component.

  As you can see, there is a struct attached to the EEFramework class. This struct defines the elements of tEEBuffer, which is implemented as the property “EnergyExpenditure” in the EEFramework class.
- The MedianFilterFramework has 4 properties (1 define and three structs as defined in the struct **tBufferMedian**) and 2 functions, one of which is implemented in the file Events.c.
- The FIRFramework class has several properties (defines, Frac16 and structs as defined by the structs **tBufferHPFilter** and **dfr16_tFirStruct**. The DataProcess component depends on the “Processor Expert” package. The exact dependency is within this class, because it needs the beans for calculating the FIRs (although it is not shown in the diagram).

We have been discussing the internal structure of the DataProcessing component. But how does this component interacts with the rest of the application?

This component can be seen as a subsystem, thus, requiring some input data, processing it and generating output data. The input data is given by other components (suppliers) of the application and the generated output data will feed other components within the system.

The components that supply the input data to the DataProcessing component don't need to know what and how the output data is sent. And the components requiring the output data don't need to know how the input data was supplied.

This gives us a chance to define different component *interfaces* that will define the relationship between DataProcessing and its related components. An interface is only a set of *public* functions that are executed by external components. For instance, the Main state machine will call the Data Processing state machine, located in the function PDProcessData(). So the interface for executing the DataProcessing state machine will include only this function and will be called by the file main.c.

The following figure shows the different interfaces of the DataProcess component:

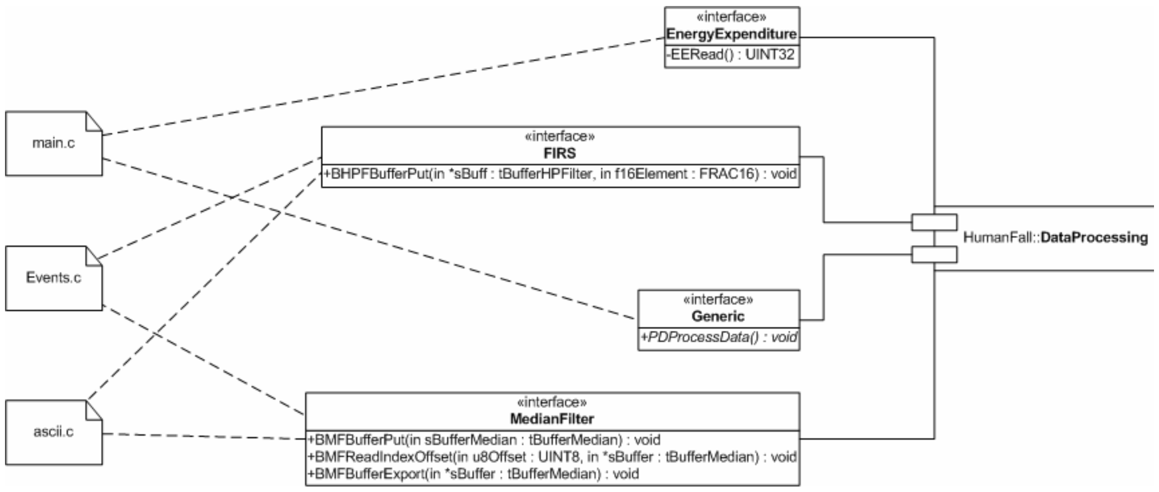

«interface»
EnergyExpenditure
-EERead() : UINT32
main.c
«interface»
FIRS
+BHPFBufferPut(in *sBuff : tBufferHPFilter, in f16Element : FRAC16) : void
HumanFall::DataProcessing
Events.c
«interface»
Generic
+PDProcessData() : void
ascii.c
«interface»
MedianFilter
+BMFBufferPut(in sBufferMedian : tBufferMedian) : void
+BMFReadIndexOffset(in u8Offset : UINT8, in *sBuffer : tBufferMedian) : void
+BMFBufferExport(in *sBuffer : tBufferMedian) : void

Figure 4-11. Interfaces of the DataProcessing Component

In Figure 4-11 you can see the four interfaces that the DataProcessing component implements. You can see as well the list of public functions that are executed outside of the component. The files from which the public functions are called are shown also.

4.3.2.2.1 DataProcessing state Machine

The DataProcessing component should execute the steps defined in 4.1.2 Application Basics. Everything but the median Filter is computed in this state machine:

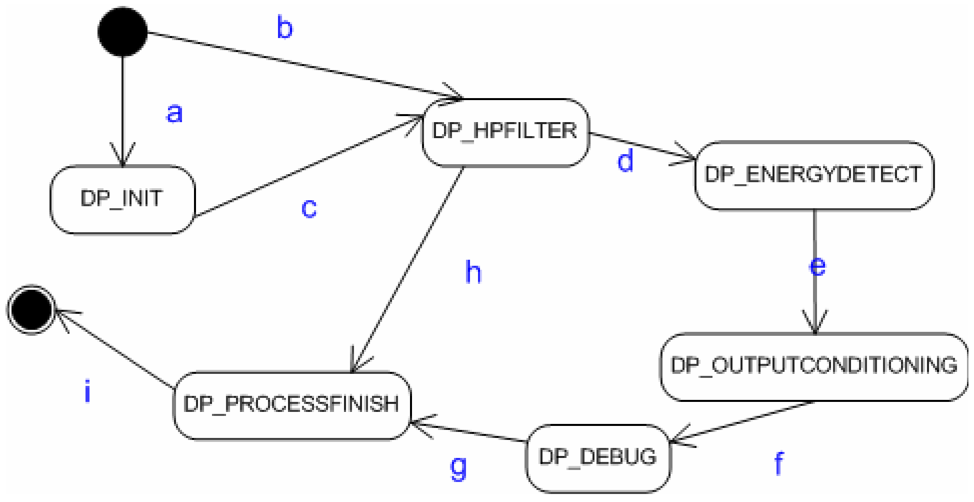


Figure 4-12. DataProcessing Component's State Machine

States Definition:

DP_INIT: This is the initial state, where all the variables and buffers needed in the Data Processing stage are initialized.

DP_HPFILTER: In this state, the HighPassFilter is performed on the corresponding buffers. In this implementation, the filter used has a window of 36.

DP_ENERGYDETECT: This stage calculates the Energy Expenditure of the Filtered signal for the length of a full window (since the HighPass Filter window is 36, the energy expenditure is calculated also with windows of 36) which is at this moment 0.8 seconds.

DP_OUTPUTCONDITIONING: In this state, the outputs of the Data Processing module are prepared to be read by the HumanState calculus (within the main state machine).

DP_DEBUG: This state permits the developer debug the DataProcessing state. It allows the debugging via RS232 and/or SMAC. There are two levels of debugging at this moment:

- Output of the product of the HighPass filter.
- Output of the product of the EnergyExpenditure.

DP_PROCESSFINISH: Prepares the DataProcessing variables for the next Data Processing stage.

Transitions Definition:

a: This Transition occurs the first time that the DataProcesing stage is executed.

b: This is the normal transition during the operation of the system.

c: Always go to DP_HPFILTER state

d: If not in the first execution of this stage, go to DP_ENERGYDETECT

e: Always go to DP_HUMANSTATE

f: As it is implemented in this moment, always go to DP_DEBUG.

g: Always go to DP_PROCESSFINISH state.

h: This transition occurs the first time that the DataProcessing stage is executed, because the high pass filter is not stable.

i: Exit the DataProcessing stage.

4.3.2.3 External Memory (I2C_Drv) Component

The external memory component interfaces the DSC to an external I2C memory for data log. This memory should be used for logging the information on the current EnergyExpenditure as well as the current human state, for each 0.8 second window.

The user interacts with this module in two situations:

- First, saving the data into the memory just by using the Human Fall Detection board.
- Second, retrieving the data from the memory by resetting the board while pressing down the push button (A memory dump).

In the following figure you can see how the memory space is organized:

Next Location Index 2bytes	Available space 32K - 2 bytes

Figure 4-13. I2C Memory Structure

The first two bytes points to the address where the new data will be written. If the memory is full, then it will begin to overwrite the oldest data.

Each human state computation will be saved in the memory using the format shown in Figure 4-14.

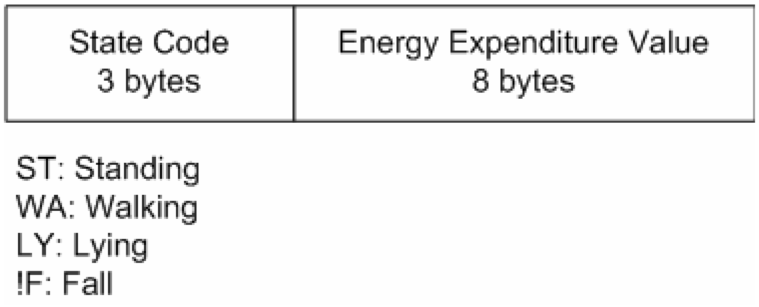


Figure 4-14. Event Information within the I2C

The following is the list of public functions of this component that are used by the file main.c:

RandomRead: Used to read a byte from any location from the external memory.

I2CWriteChar: Used to write a byte to the external memory, updating the next address index.

If the user needs to write specific information on the I2C, it is recommended to use the functions SendStr (to transmit strings) and SendChar (to transmit 1 byte), using as the destination parameter the DBG_PORT_MEM.

Please refer to source code (main.c file) for more details in such functions.

4.3.2.4 Wireless Communications (SMAC) Component

This component uses the SMAC Package to communicate with the Host through the air, by means of MC13192 transceiver.

4.3.2.4.1 SMAC (Simple Medium Access Control) Package

The SMAC package is a simple MAC that provides basic functionality to communicate with remote devices within the range of influence. The functionality that it provides can be summarized in the following groups of primitives:

- Data transmission functionality (for transmitting and receiving)
- Energy detection (for multiple access to the channel and link quality).
- Transceiver low power modes (for low power techniques in the Xcvr).
- Access to other internal functions of the Transceiver.

This package uses some of the MCU resources, such as 5 GPIOs and 1 SPI.

In order to communicate with no error with the host, a protocol must be defined. Such protocol should use the capability of the SMAC package in order to increase the reliability of communications.

4.3.2.4.2 Wireless Communications Protocol

This is a simple protocol designed to transmit information from a device (a Human Fall Detection system) to a Host (which would be the terminal that will report the emergency).

In this implementation, the protocol is really simple and provides capability for a single way communication (from device to host). However, the component's state machine was designed for enabling two way communications (the developer would need to modify this component in order to have this functionality).

- The characteristics of the protocol are:
- The device ALWAYS starts the communication.
- Therefore, the Host should ALWAYS be listening for packets from the devices (except when responding to a device's packet).
- All the packets have the same structure.
- One transaction is started with a packet from a device and closed with an Acknowledge packet from the host.
- One transaction may hold many transfers of data/commands to/from devices and host. This is accomplished by
  — Device sends a packet
  — Host responds with an ACK-CMD packet
  — Thus, the device executes the command and sends the status back to the host
  — The host may send other ACK-CMD packets, which will be responded by the device
  — Finally, when the host doesn't have any command to send to the device, it will respond with a simple ACK packet in order to close the transaction.

This is the packet structure of the simple protocol:

Application Identifier 4bytes	Instance Identifier 2bytes	Packet Descriptor 2bytes	Payload 100bytes max

Figure 4-15. Simple Protocol Packet Structure

The first field, the **Application Identifier** will always be one number, in order to differentiate this application (Human Fall Detection) from other traffic within the channel. This application identifier is defined in the file SMAC.h.

The second field, the **Instance Identifier** represents the unique number of the device. This is a “Device ID”, that will differentiate others within the same area.

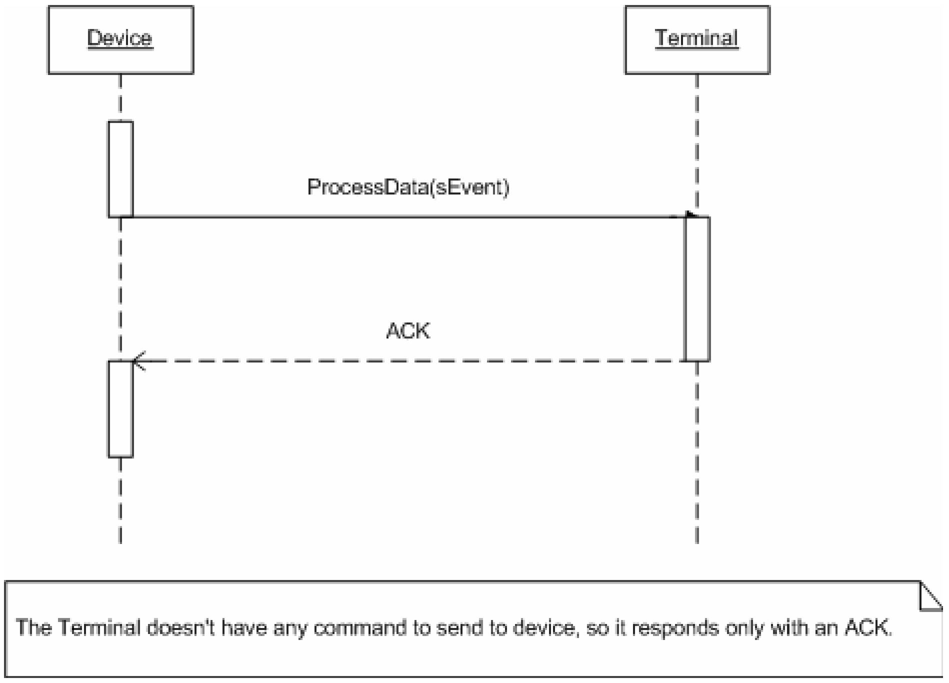
The third field, the **Packet Descriptor** defines what kind of packet it is. The possibilities for this field are:

- WCOM_PD_EVENT: Is typically the type of packet sent initially by the device (the transaction starter).
- WCOM_PD_ACK: Is the type of packet sent by the Host, for finishing the transactions.
- WCOM_PD_ACK_CMD: Is the type of packet sent by the Host, after a WCOM_PD_EVENT or A WCOM_PD_CMD_RESPONSE type to request the device to execute a command defined in the payload.
- WCOM_PD_CMD_RESPONSE: After executing a required command by the Host, the devices send this type of packet to the Host in order to show the result of the execution of the command.

There are two defined (but not implemented) types of commands that the host can send to the device:

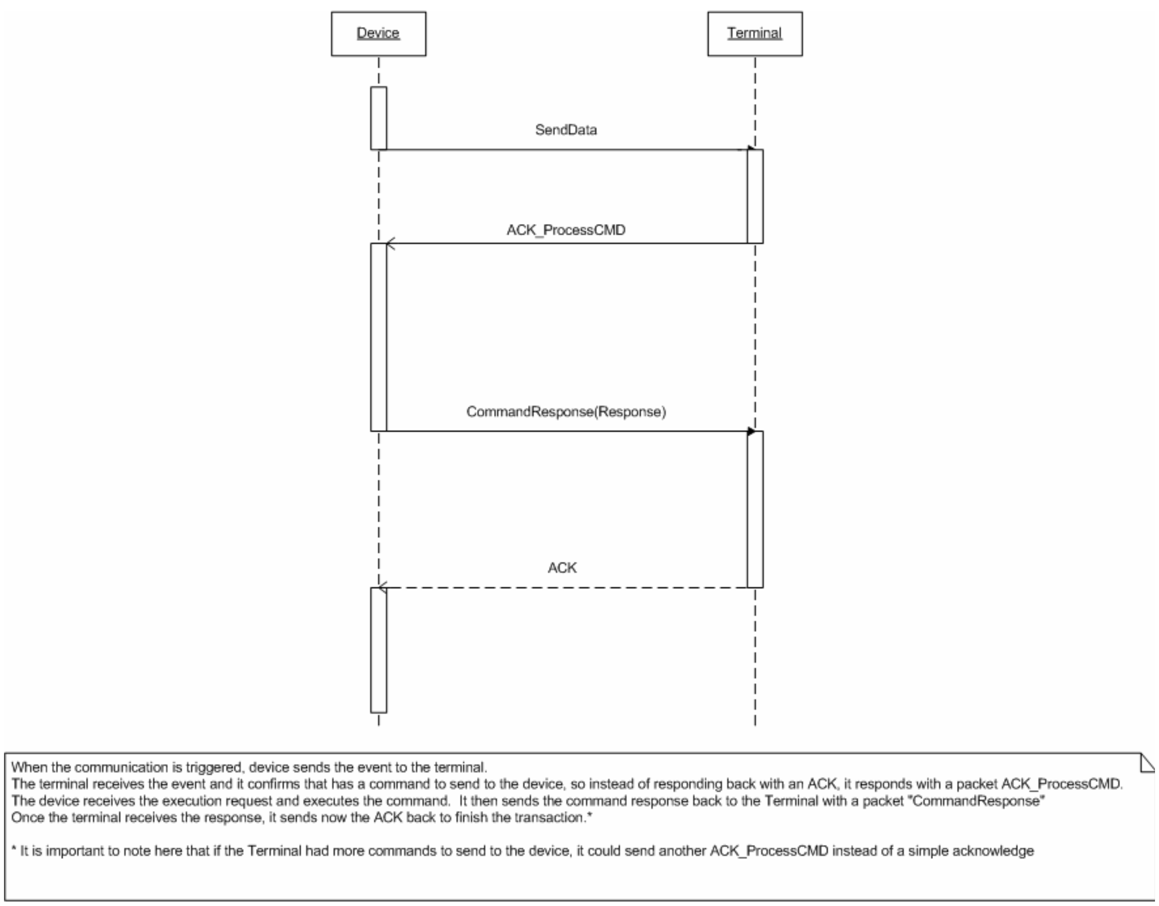
- Device commands, which are commands that ask the device to change its configuration, or requests a specific state on that.
- Memory commands, which are commands that will interface the host with the external memory (for instance, make a remote memory dump of the external memory).

The following diagrams are the sequence diagrams for the different cases of communication between the devices and the host.



The Terminal doesn't have any command to send to device, so it responds only with an ACK.

Figure 4-16. Sequence Diagram for a Simple Report Packet from the Device



When the communication is triggered, device sends the event to the terminal.
The terminal receives the event and it confirms that has a command to send to the device, so instead of responding back with an ACK, it responds with a packet ACK_ProcessCMD.
The device receives the execution request and executes the command. It then sends the command response back to the Terminal with a packet "CommandResponse"
Once the terminal receives the response, it sends now the ACK back to finish the transaction.*

* It is important to note here that if the Terminal had more commands to send to the device, it could send another ACK_ProcessCMD instead of a simple acknowledge

Figure 4-17. Sequence Diagram for a Terminal Requiring the Device to Execute a Command

For more information in regards of the SMAC, please refer to document SMAC Users Guide.

If you need further details on the capabilities of the RF Transceiver, refer to MC13192 Reference Manual.

4.3.2.5 Debugging the Application

This application was created with a limited embedded debug system. It uses the RS232 connection to dump information from the DSC.

There are 5 debug modes which can be turned on/off in the file hidef.h. Only one can be active at a time:

Debug Mode	Description
DBGINPUTDATA	This debug mode shows every data read from the accelerometer.
DBGMEDIANTEST	In this mode, the data after the median filter is thrown out to the RS232.
DBGHPTEST	This mode is useful because you can check the output data from the FIR stage of dataprocessing component.
DBGEETEST	Here you can debug the EnergyExpenditure.
DP_TEST	In this mode you can verify that your FIR works fine: You specify what data will be the input of the FIR and then executes it and the result is give to you by RS232. This way you can use this to evaluate if the FIR is executing ok.

The modes can be turned on by defining its name as a macro in the file hidef.h. For instance, the sentence

```
#define DBGINPUTDATA 0
```

will enable the debug mode DBGINPUTDATA.

Please make sure to enable only one debug mode at a time.

There is a global debug mode enabler, in the macro defined as DEBUG_MODE. If this macro is not defined, then no debug mode will work.

Appendix A BOM and Schematics

A.1 Bill of Materials

Table A-1. Human Fall Detection Reference Design Bill of Materials

Item	Qty	Reference Designator	Description	Manufacturer	Part Number	Value	Tolerance	Rating	Footprint
1	1	B1	BUZZER	CUI INC	CEP-2242	4.1 kHz		3-16 V	
2	15	C1, C2, C5, C6, C7, C8, C11, C12, C25, C27, C31, C32, C33, C34, C35	CAPACITORS	ANY		100 nF	10%	X7R	0603
3	1	C3	CAPACITOR	PANASONIC	ECST1AY225R	2.2 µF	10%	TANTALUM	A-EIA
4	2	C9, C13	CAPACITORS	Any		4.7 µF	20%	X5R	0603
5	1	C10	CAPACITOR	PANASONIC	ECST1AY106R	10 µF	10%	TANTALUM	A-EIA
6	1	C14	CAPACITOR	PANASONIC	ECS-T1CY105R	1 µF	10%	TANTALUM	A-EIA
7	1	C28	CAPACITOR	ANY		100 nF	10%	NPO	0402
8	1	C18	CAPACITOR	ANY		1 nF	10%	X7R	0603
9	1	C19	CAPACITOR	ANY		100 pF	5%	NPO	0603
10	2	C20, C22	CAPACITORS	ANY		18 pF	5%	NPO	0402
11	2	C21, C24	CAPACITORS	ANY		0.5 pF	±.25 pF	NPO	0402
12	2	C29, C23	CAPACITORS	ANY		220 pF	10%	X7R	0402
13	2	C30, C26	CAPACITORS	ANY		6.8 pF	±.25 pF	NPO	0402
14	1	D1	DIODE RECTIFIER	VISHAY SEMICON.	ES1A				DO-214AC
15	1	J1	HEADER 7X2	SAMTEC	TSW-107-23-S-D				
16	1	LED1	SMD BLUE LED	AGILENT TECH.	HSMS-C650				1206
17	1	LED2	SMD YELLOW LED	AGILENT TECH.	HSMY-C650				1206
18	1	LED3	SMD RED LED	AGILENT TECH.	HSMD-C650				1206
19	1	LED4	GREEN LED	AGILENT TECH.	HSMG-C650				1206
20	2	L2, L1	FERRITE BEAD	SMEC	FCB0603-1000HNT				0603
21	1	L3	INDUCTOR	TOKO	LL1005-FH6N8K	6.8 nH			0402
22	1	P1	SWITCHCRAFT, RAPC-722	SWITCH CRAFT	2.1mm RAPC-722				
23	1	P2	DB9 FEMALE	AMPHENOL	617-C009S-AJ120				
24	1	Q1	TRANSISTOR NPN	ON SEMI.	BC817-16LT1			500 mA	SOT-23
25	5	R1, R2, R7, R9, R12	RESISTORS	ANY		10k			0603
26	2	R3,R4	RESISTORS	ANY		47K			0603
27	3	R5,R8,R16	RESISTORS	ANY		270			0603
28	1	R6	RESISTOR	ANY		470			0603
29	4	R10, R11, R13, R14	RESISTORS	ANY		1k			0603
30	1	R15	RESISTOR	ANY		10			0603

Table A-1. Human Fall Detection Reference Design Bill of Materials (continued)

Item	Qty	Reference Designator	Description	Manufacturer	Part Number	Value	Tolerance	Rating	Footprint
31	1	R17	RESISTOR	ANY		NP			0402
32	1	R18	RESISTOR	ANY		470k			0603
33	1	R19	RESISTOR	ANY		360			0603
34	2	R20,R21	RESISTORS	ANY		5.6K			0603
35	1	SW1	ON/OFF		EG1224				
36	2	S2, S1	PUSHBUTTONS	PANASONIC	EVQ-PAD05R				
37	3	TP1, TP5, TP6	RED TEST POINT						
38	3	TP2, TP3, TP4	BLACK TEST POINT						
39	1	U1	MC56F8013 DIGITAL SIGNAL CONTROLLER	FREESCALE	MC56F8013VFAE				LQFP-32
40	1	U2	3-AXIS ACCEL	FREESCALE	MMA7260Q				QFN-16
41	1	U3	VOLTAGE REGULATOR	ON SEMI.	MC33269DT-3.3			3.3 V	DPAK
42	1	U4	2.4GHz TRANSCEIVER	FREESCALE	MC13192FC				
43	1	U5	MAX3232EEWE		MAX3232EEWE				16 WIDE .300" SOIC
44	1	U6	AT24C256 EEPROM	ATMEL	AT24C256N-10SI-2.7				JEDEC SOIC
45	1	X1	XTAL 16.000MHz		DSX321G-16.000M-8pF-20-20				
46	1	BATTHOLD1	BATTERY HOLDER	KEYSTONE	1294				

A.2 Schematics

See Figure A-1 through Figure A-5 for the Human Fall Detection using 3-axis Accelerometer schematics.

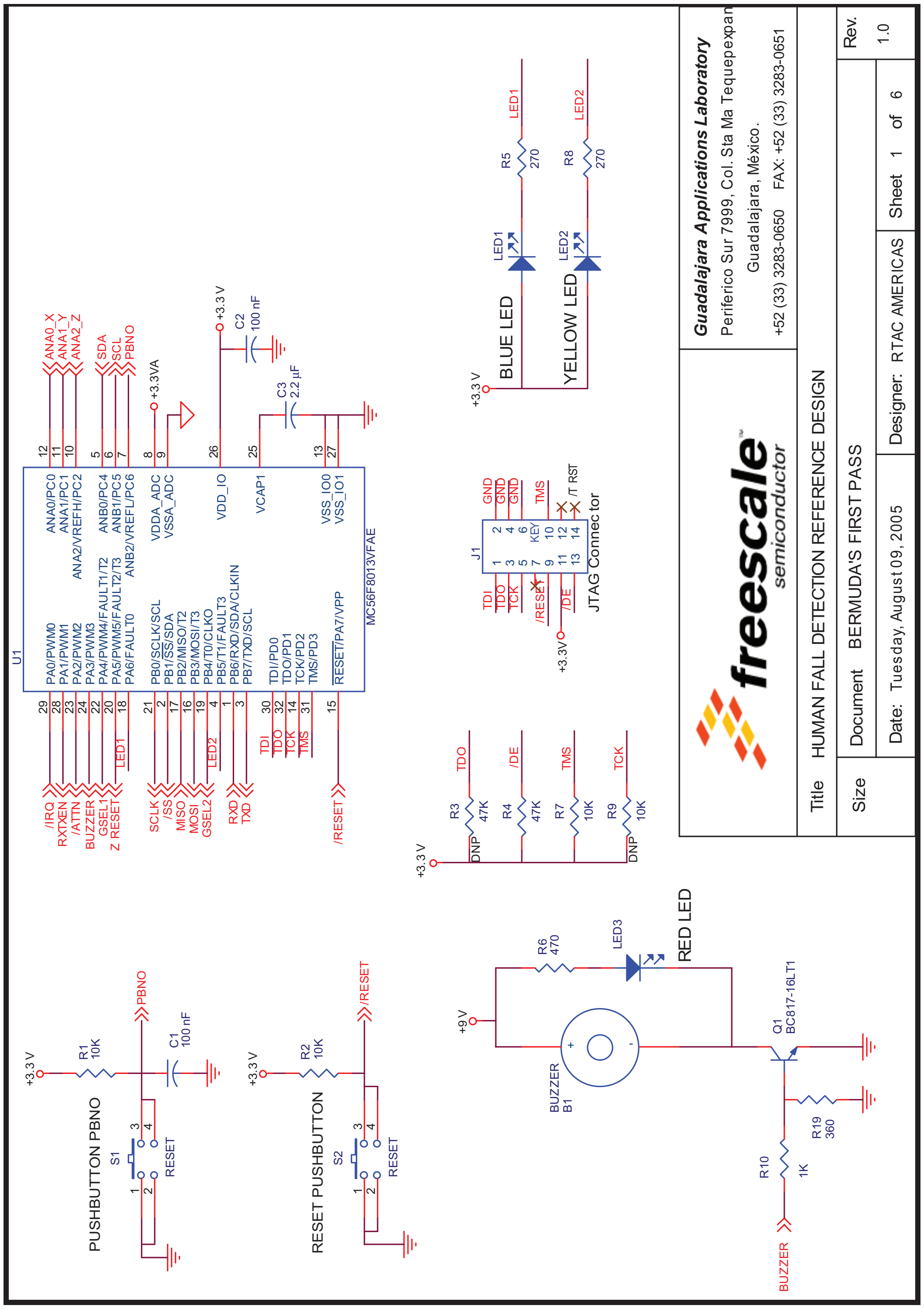


Figure A-1. DSC Schematic

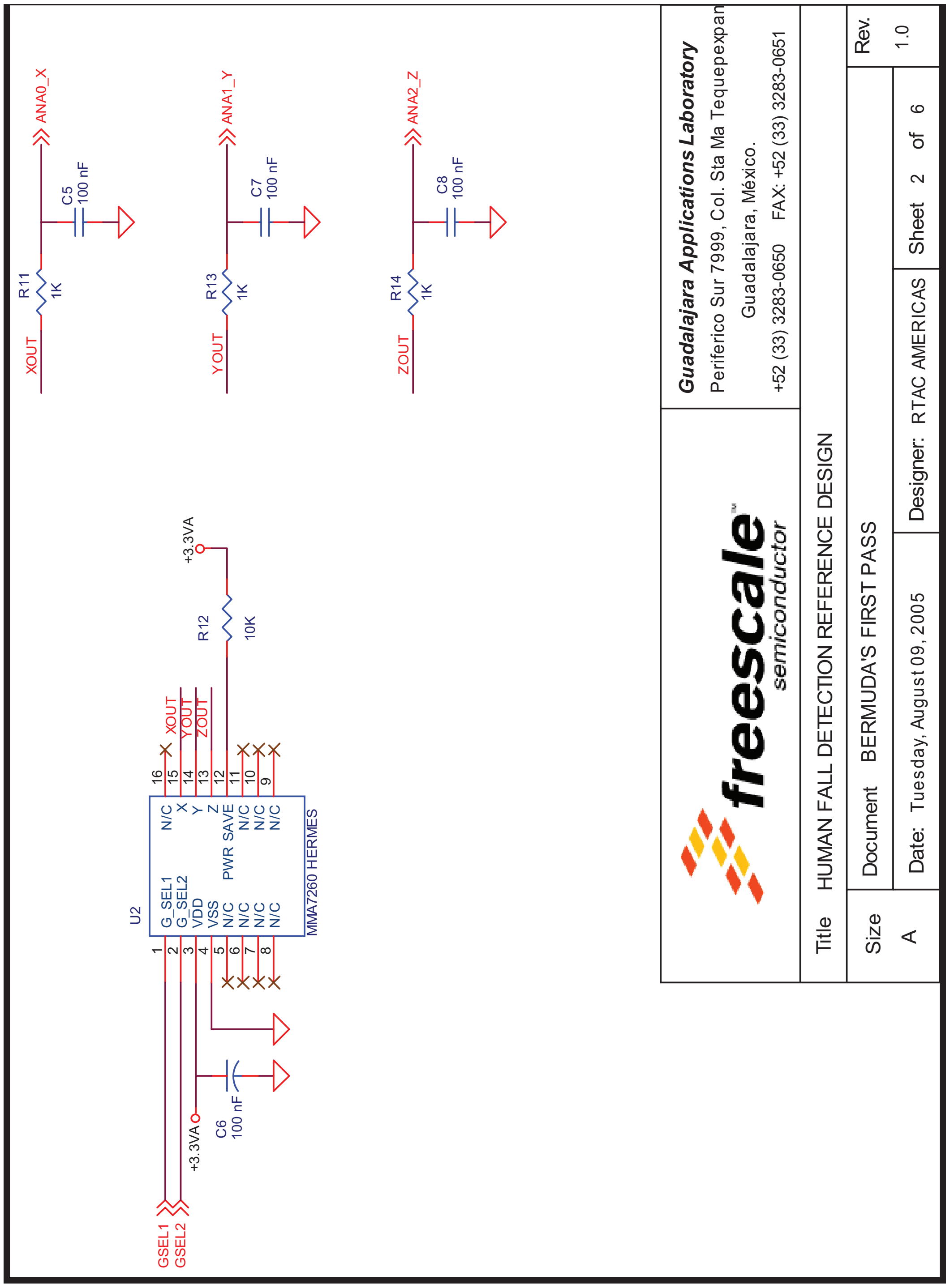


Figure A-2. 3-Axis Accelerometer Schematic

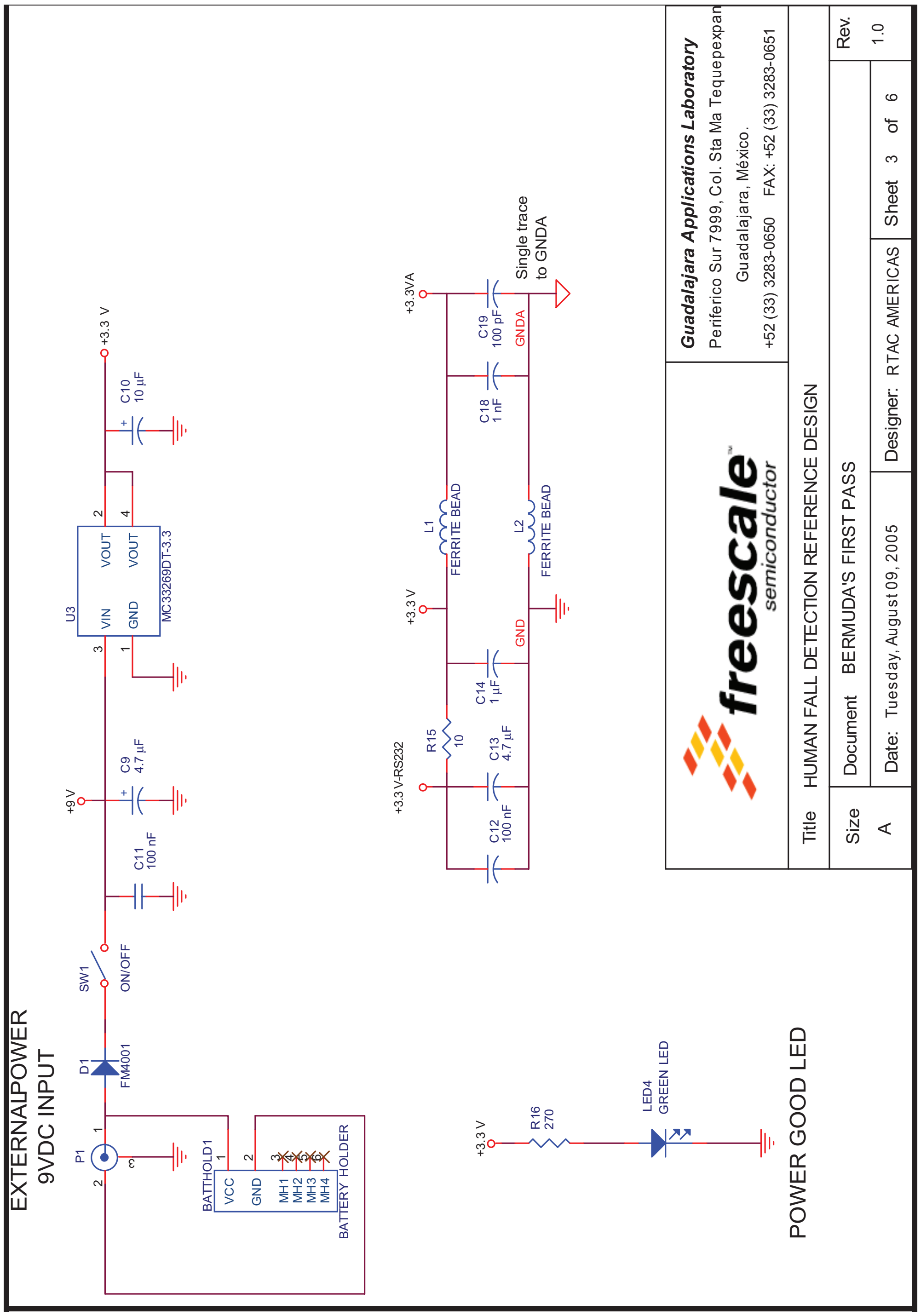


Figure A-3. Power Supply

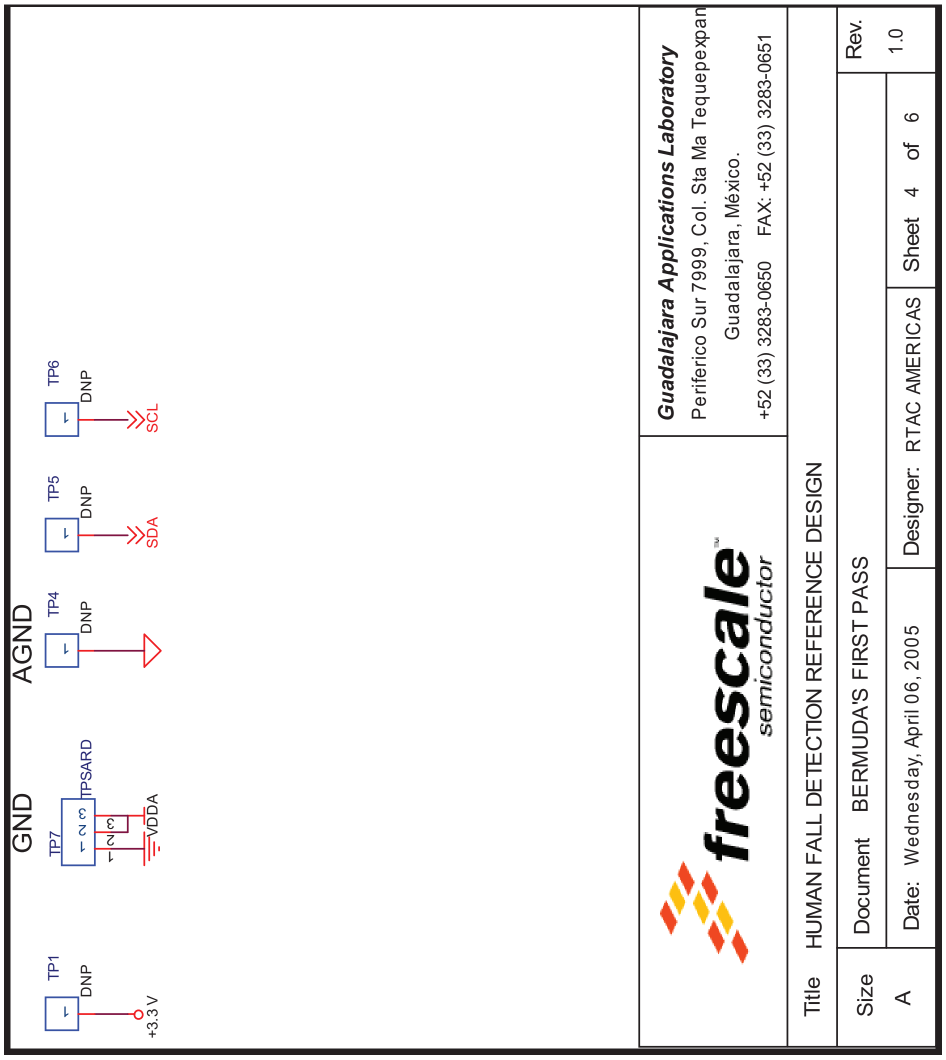

TP1
DNP
+3.3 V
GND
TP7
TPSARD
VDDA
AGND
TP4
DNP
TP5
DNP
SDA
TP6
DNP
SCL
freescale semiconductor
Guadalajara Applications Laboratory
Periferico Sur 7999, Col. Sta Ma Tequepexpan
Guadalajara, México.
+52 (33) 3283-0650 FAX: +52 (33) 3283-0651
Title HUMAN FALL DETECTION REFERENCE DESIGN
Size A
Document BERMUDA'S FIRST PASS
Rev. 1.0
Date: Wednesday, April 06, 2005
Designer: RTAC AMERICAS
Sheet 4 of 6

Figure A-4. Test Points

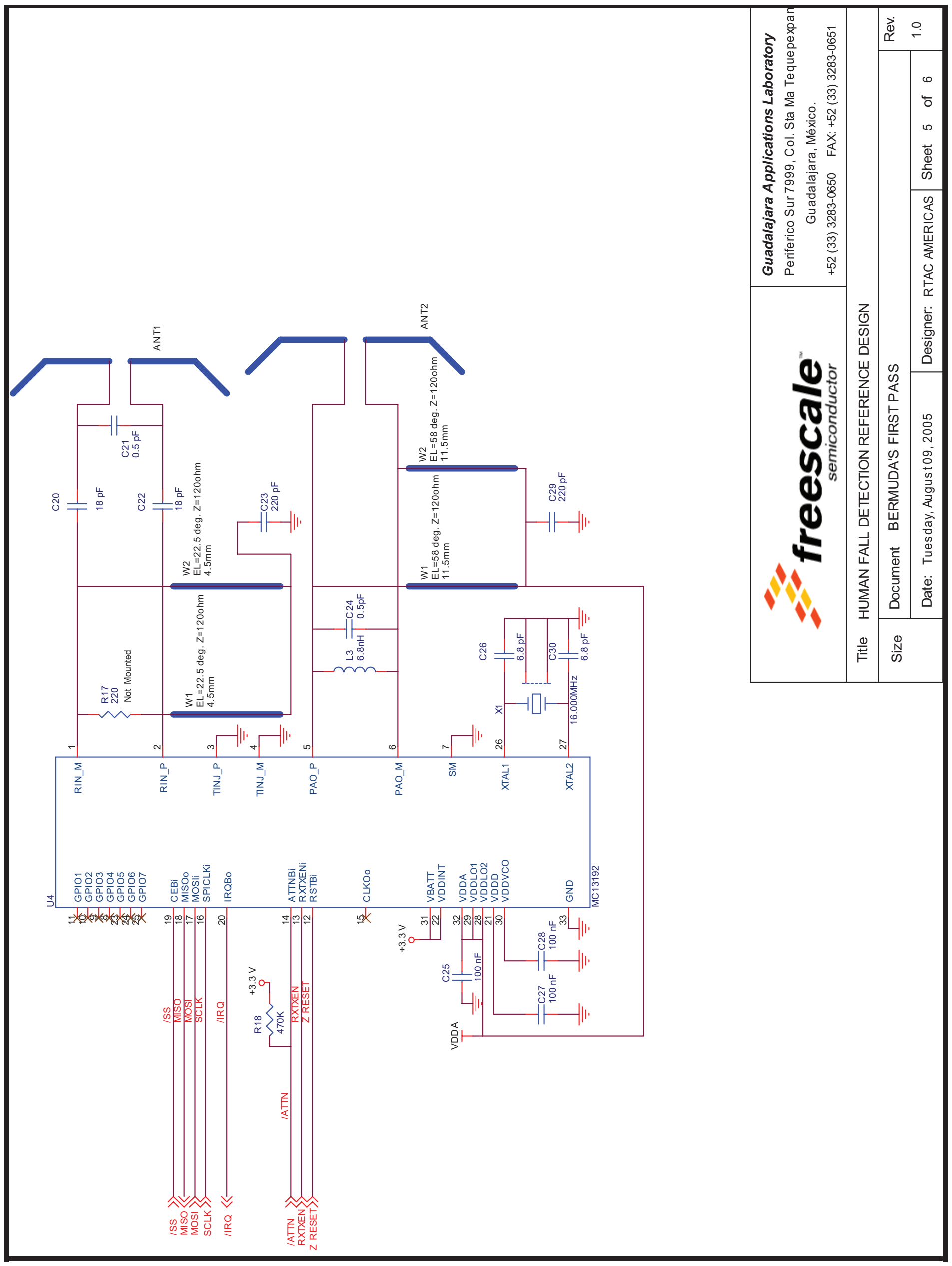


Figure A-5. MC13192 ZigBee Transceiver

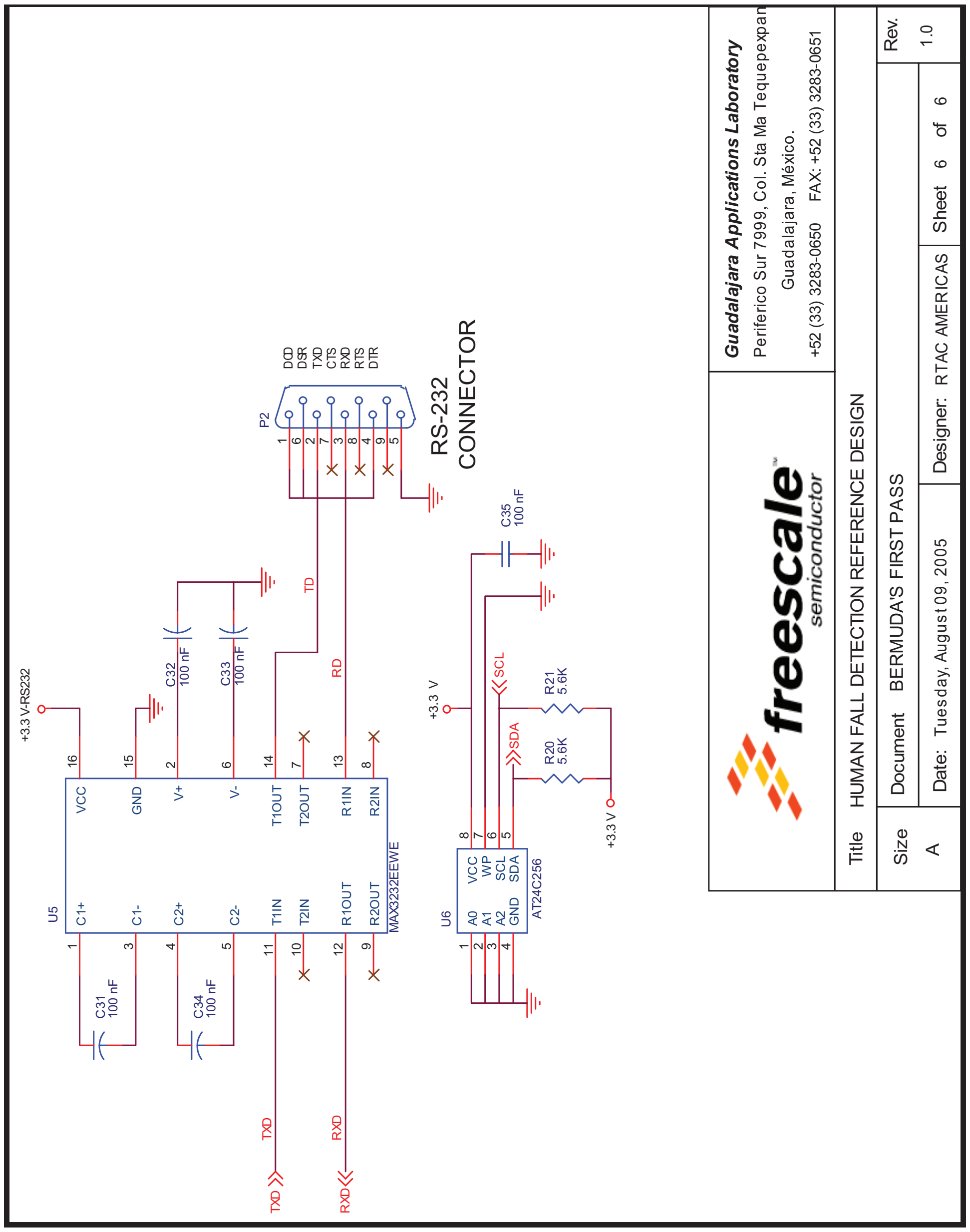


Figure A-6. RS-232 and I^{2}C Interface

Appendix B Glossary

56800/E — Freescale's family of Digital Signal Controllers.

Accelerometer — Sensor that measure acceleration. The transducer converts mechanical motion into an electrical signal that is proportional to the acceleration value of the motion. Accelearation can be due to gravity or changing motion (changes in velocity).

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

baudrate — The total number of bits transferred per unit of time.

binary — Relating to the base 2 number system.

bit — A binary digit. A bit has a value of either logic 0 or logic 1.

bus — A set of wires that transfers logic signals.

byte — A set of 8 bits.

DSC — Digital Signal Controller.

DSP — Digital Signal Processor.

EEPROM — Electrically Erasable and Programmable, non-volatile Memory.

EOnCE: Enhanced On-Chip Emulation.

g: Unit of acceleration defined as $9.8m/s^2$.

I^{2}C — Inter-Integrated Circuit Bus invented by Philips. A protocol for providing serial communication between integrated circuits.

Impedance — Opposition offered by an electric circuit to the flow of an alternating current.

IRQ — HCS12 maskable Interrupt Request Input.

JTAG — Joint Test Action Group.

Layout — The arrangement of the different elements that make up the design of the PCB.

LIN — Local Interconnect Network.

mask — A logic circuit that forces a bit or group of bits to a desired state.

MCU — Microcontroller unit. A complete computer system, including a CPU, memory, a clock oscillator, and Input/Output (I/O) on a single integrated circuit.

Nibble — A set of four bits (half byte).

PC — Personal Computer.

PCB — Printed Circuit Board.

Polarity — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, VDD and VSS.

Polling — Periodically reading a status bit to monitor the condition of a peripheral device.

Port — A set of wires for communicating with off-chip devices.

Prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

Program — A set of computer instructions that cause a computer to perform a desired operation or operations.

QFN — Quad Flat Pack No-lead package.

RF — Radio Frequency (used for wireless transmission of data).

SCI — Asynchronous Serial Communication Interface.

software — Instructions and data that control the operation of a microcontroller.

SPI — Synchronous Peripheral Interface.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

toggle — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

Transceiver: A communications device capable of both transmitting and receiving.

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

word — A set of two bytes (16 bits).

Appendix C References

56F8000 Peripheral Reference Manual (MC56F8000RM), Freescale Semiconductor, Inc..

MC56F8013 Data Sheet (MC56F8013), Freescale Semiconductor, Inc..

MMA7260Q Data Sheet (MMA7260Q), Freescale Semiconductor, Inc..

MMA7260Q Fact Sheet (MMA7260QFS), Freescale Semiconductor, Inc..

Sensor Application Reference Design User's Guide (MC13192SARDUG), Freescale Semiconductor, Inc.

SMAC User's Guide (SMACRM), Freescale Semiconductor, Inc.

MC13192 Data Sheet (MC13192DS), Freescale Semiconductor Inc.

MC13192 Fact Sheet (MC13192FS), Freescale Semiconductor Inc.

Application Note (AN1902), Quad Flat Pack No-lead (QFN), Freescale Semiconductor, Inc.

Application Note (AN3111), Soldering the QFN Stacked Die Sensors to PC Board, Freescale Semiconductor, Inc.

Application Note, (AN3003), PCB Layout Guidelines for the MC1319x , Freescale Semiconductor, Inc.

REVISION HISTORY

Revision	Date	Description of Changes
1.0	7/2005	Initial Release
1.0.1	8/2005	Software Section Added
2.0	12/2005	Converted to Freescale Format

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MC33794RM
Rev 2.0
12/2005

freescale™
semiconductor