Application Report
SLAA599–May 2013

Low-Power Pedometer Using an MSP430™ MCU

Dennis Lehman *MCU Strategic Solutions*

ABSTRACT

This application report describes a low-power pedometer example application that uses an MSP430F5229 microcontroller and the TI Pedometer firmware algorithm. This application was developed and targeted for the health and fitness markets.

Fitness monitors typically measure both a person's amount and rate of exercise (traveled distanced and pace) as well as effort expended (calories burned in the process through the number of steps taken). Stored data such as steps and calories can be downloaded to a computer via USB or a wireless USB dongle. All parts of the system require ultra-low power embedded controllers and low-power RF for communications.

An MSP430™ microcontroller implementing the TI pedometer firmware combined with a low-power 3-axis MEMS accelerometer provides a low-power pedometer solution.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP430_Pedometer/latest/index_FDS.html.

Contents

1 Introduction 2
2 System Overview 2
3 Operation 2
4 Software 4
5 Pedometer Algorithm 5
6 Pedometer Firmware API 5
7 Technical 6
8 References 8

List of Figures

1 System Overview 2
2 Flow Chart of Operation 3
3 Software Components 4
4 Pedometer Algorithm 5
5 Sensor Sample and Pedometer I^2C Bus Activity 6
6 Pedometer Algorithm Execution Time 7

1 Introduction

The TI Pedometer algorithm is compact and efficient. It requires less than 1.2 Kbytes of code memory and approximately 640 bytes of data memory on the MSP430F5229. The algorithm uses efficient fixed-point computations and leverages the hardware multiplier for accelerated calculations that are performed independently from the CPU.

The algorithm uses sensor data sampled from an ADXL345 3-axis MEMS accelerometer to detect stepping motion in all axes. This feature allows multiple wearable configurations such as on the waist, in a shirt or pants pocket, or on the wrist. The sensor sampling rate can be from 50 Hz (20 milliseconds) to 62.5 Hz (16 milliseconds).

2 System Overview

A simple pedometer application can be implemented on any MSP430 microcontroller with a 32-bit hardware multiplier, at least 4 KB of flash program memory, and 1 to 2 KB of data RAM. One I^2C peripheral for sampling data from the accelerometer and one UART peripheral for sending step count data to a *Bluetooth*® radio module are also required.

The CPU can be clocked at a lower frequency during sensor and radio communications but requires at least a 4-MHz clock while processing the pedometer algorithm.

An example TI Pedometer platform features the MSP430F5229 MCU, an ADXL345 3-axis MEMS accelerometer, and a TI *Bluetooth* radio module. An Android mobile device with *Bluetooth* capability hosts the user interface application and displays the step count information sent from the pedometer platform. Figure 1 shows an overview of the pedometer application.

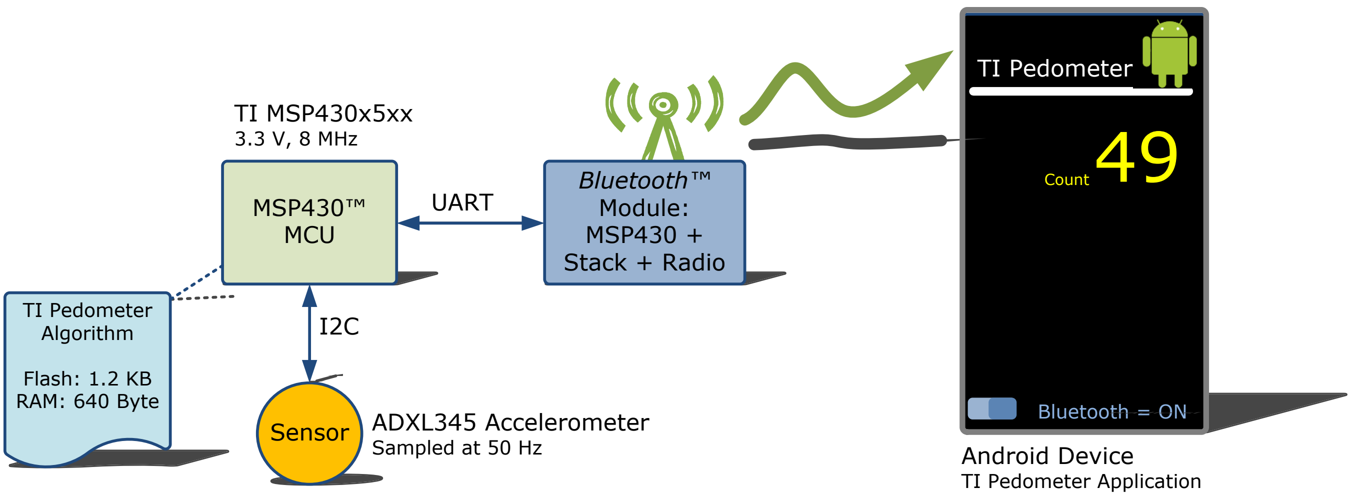

Figure 1. System Overview

3 Operation

See Figure 2 for an flow chart that describes the following modes of operation.

3.1 Initialization

At power on, the accelerometer is configured to generate an interrupt when new data is available every 20 milliseconds. To conserve power after initialization, the MCU and accelerometer stay in low-power modes until the Start/Stop button is pressed.

3.2 Run Mode

When the Start/Stop button is pressed while in Idle mode, the MCU exits low-power mode and enables the accelerometer. The accelerometer samples and generates interrupts every 20 milliseconds (50 Hz). The MCU reads the accelerometer and processes the data in the pedometer algorithm, returning to low-power mode between samples. When a step has been detected, the updated step count is sent to the radio module connected to the UART.

3.3 Idle Mode

When the Start/Stop button is pressed while in Run mode, data collections stops and the MCU and accelerometer enter low-power modes. Pressing the Start/Stop button toggles the application between Idle and Run modes.

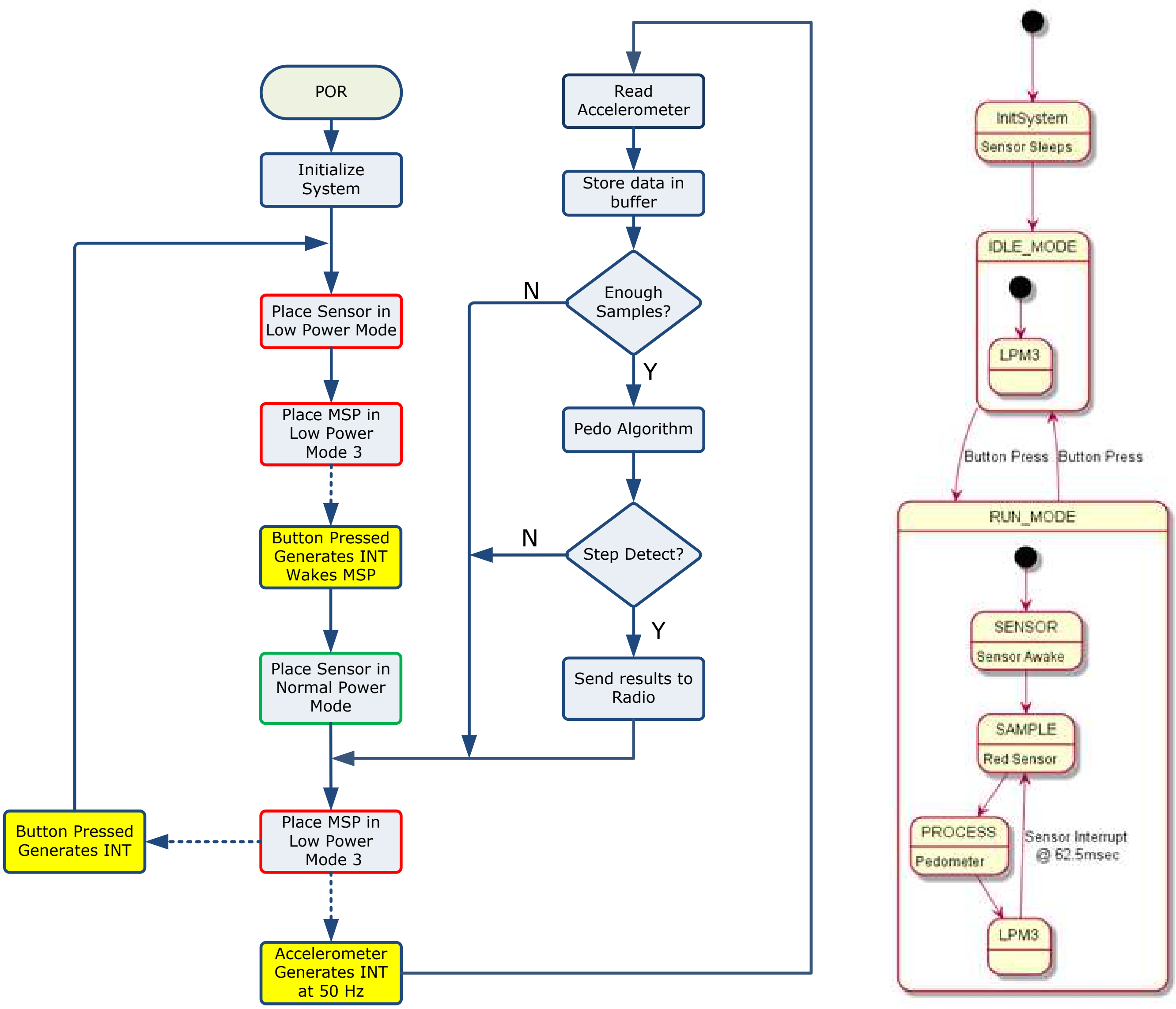

Figure 2. Flow Chart of Operation

4 Software

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP430_Pedometer/latest/index_FDS.html. The source code can be compiled using Code Composer Studio™ IDE version 5.3 or IAR Embedded Workbench™ IDE version 5.1.4. Note that the TI pedometer algorithm is provided as a library only (no source code) and is linked in during the build process.

Figure 3 shows the relationship between the software components implemented in this application.

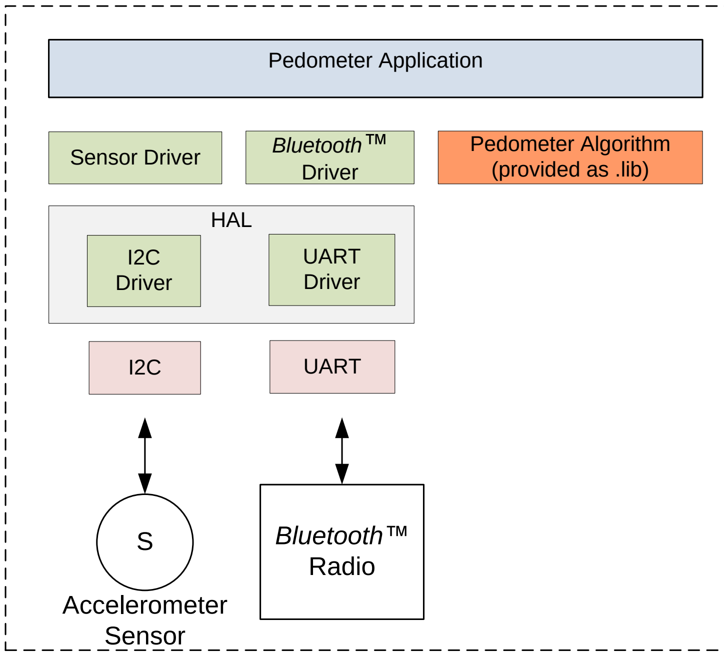

Figure 3. Software Components

The application code is located in main.c. After initialization, it implements a simple loop that checks the condition of system flags and otherwise remains in low-power mode 3. A simple state machine is controlled by the user pressing the Start/Stop button. Depending on the state, the system is either collecting sensor data or sleeping. Interrupts generated by either the Start/Stop button or the sensor are handled by their respective interrupt handlers near the end of main.c

The platform I/O, system clock, timer, I^{2}C, and UART files are located in the HAL (hardware abstraction layer) directory. There are several _def.h "definition" files in the HAL directory that provide a simple method to control the compile-time configuration of the timers, I^{2}C, UART, and system clock. Change the system configuration by modifying these definition files or the platform.h file and compiling the project.

The accelerometer driver files are located in the sensors directory. The driver is written to support the basic features of an ADXL345 3-axis MEMS accelerometer.

The TI Pedometer algorithm is provided as a .lib only and is located in the pedometer directory. A header file for the pedometer provides the API information.

5 Pedometer Algorithm

The algorithm is compact and efficient, requiring less than 1.2 Kbytes of code memory and approximately 640 bytes of data memory (see Figure 4). The algorithm uses efficient fixed-point computations and leverages the MCU's hardware multiplier for accelerated calculations that are performed independently from the CPU.

The algorithm uses sensor data sampled from the MEMS 3-axis accelerometer at a rate of 50 Hz to detect stepping motion in any axis. This feature allows multiple wearable configurations such as on the waist, in a shirt or pants pocket, or on the wrist.

As motion is detected, the pedometer algorithm starts calculating and accumulating step counts. After the first ten (approximately) valid steps have been detected, the step count is updated with the latest step count. As motion continues, the algorithm produces an updated step count as each step is taken. If the motion stops, the algorithm resets and wait for the next ten valid steps to be detected.

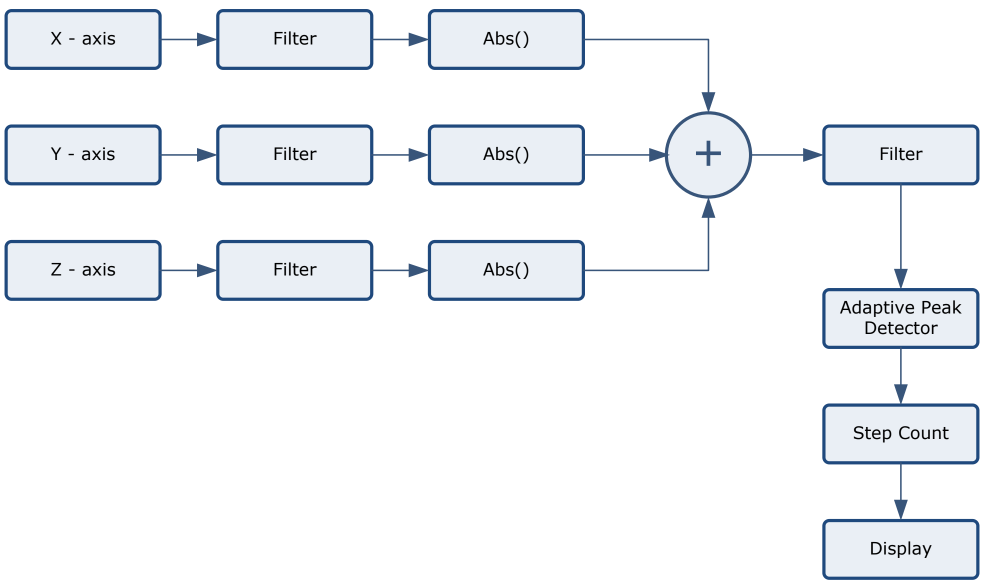

Figure 4. Pedometer Algorithm

6 Pedometer Firmware API

The TI pedometer is provided as a .lib library file. The API interface is provided in an accompanying header file and describes three functions: ped_step_detect_init, ped_step_detect, and ped_update_sample.

char ped_update_sample(short* p_data)

Description: Updates sampling buffer
Input: p_data = pointer to x, y and z axis sensor data
Returns: (0) if buffer not full, (1) buffer is full

void ped_step_detect_init(void)

Description: Initializes the pedometer algorithm data structures
Input: none
Returns: none

unsigned short ped_step_detect(void)

Description: Detect and update step count
Input: none
Returns: accumulated step count

unsigned short ped_get_version(void)

Description: Gets pedometer version
Input: none
Returns: 16-bit Pedometer algorithm version (upper 8 bit = major level, lower 8 bit = minor level)

7 Technical

7.1 Sensor Sampling and Pedometer Timing

The TI pedometer algorithm requires data samples from the 3-axis MEMS accelerometer at a rate of approximately 50 Hz (20 milliseconds) and calculates a user step count every 540 milliseconds (see Figure 5). During periods of inactivity, the MSP430 remains in low-power mode 3, approximately 80% of the time.

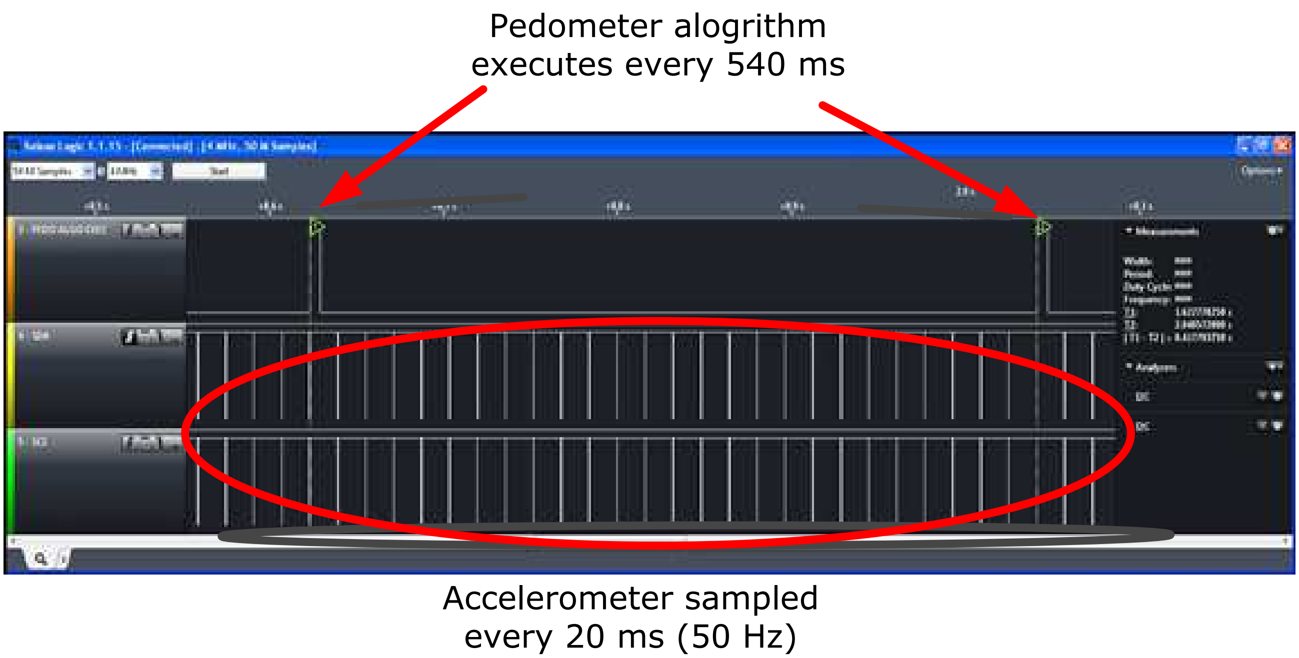

Figure 5. Sensor Sample and Pedometer I²C Bus Activity

7.2 Pedometer Execution Time

The TI pedometer algorithm execution time is approximately 9 milliseconds running at 4 MHz on an MSP430F5229 (see Figure 6).

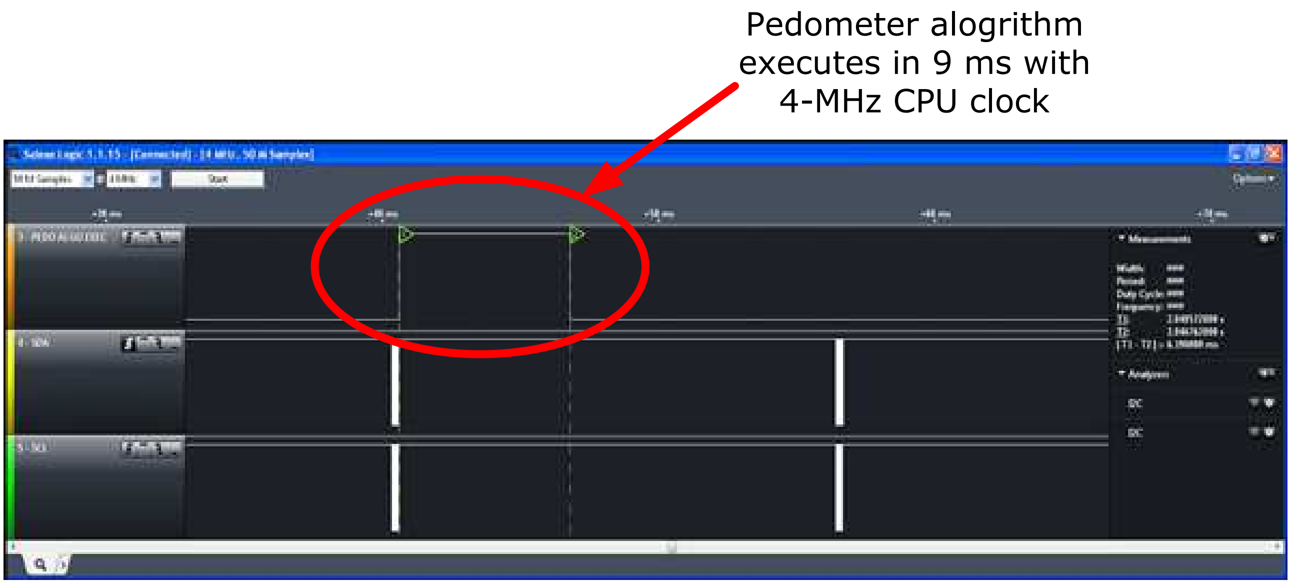

Figure 6. Pedometer Algorithm Execution Time

7.3 Power Measurements

Low-power operation is one of the MSP430 MCU's strengths and is demonstrated in the following power measurements. For this example, the MSP430F5229 I_{CC} operating current data was collected under the following conditions:

- V_{CC} = 3.3 V, MCLK = 4 MHz, SMCLK = 4 MHz (DCO with REFO clock source for reference)
- All peripherals disabled, except USCI_B0 (I^2C), USCI_A0 (UART), and 32-bit hardware multiplier. I^2C clock is 400 kHz, and UART baud rate is 9600 bps.
- All unused I/O pins configured as output and driven low.

State 1 – Application in low-power mode 3

MSP430F5229 I_{CC} = 5 µA

ADXL345 I_{CC} = 1 µA

State 2 – Application running

MSP430F5229 I_{CC} = 40 µA (avg), 18 µA (min), 86 µA (max)

ADXL345 I_{CC} = 100 µA

7.4 Build Statistics

7.4.1 CCS 5.2.1, Compiler 4.14

Pedometer Application Demo + TI Pedometer Library

Optimization settings: -O3

- Total Flash = 4104 bytes (code + const)
- Total RAM = 933 bytes

Optimization settings: -O0 (default)

- Total Flash = 4218 bytes (code + const)
- Total RAM = 933 bytes

TI Pedometer Library

Optimization settings: -O3

- Pedometer Flash = 1188 bytes (code + const)
- Pedometer RAM = 640 bytes

Optimization settings: -O0 (default)

- Pedometer Flash = 1310 bytes (code + const)
- Pedometer RAM = 640 bytes

7.4.2 IAR 5.51.6

Pedometer Application Demo + TI Pedometer Library

Optimization settings: none

- Total Flash = 4470 bytes (code + const)
- Total RAM = 933 bytes

Optimization settings: medium

- Total Flash = 4266 bytes (code + const)
- Total RAM = 933 bytes

TI Pedometer Library

Optimization settings: none

- Pedometer Flash = 1386 bytes (code + const)
- Pedometer RAM = 640 bytes

Optimization settings: medium

- Pedometer Flash = 1190 bytes (code + const)
- Pedometer RAM = 640 bytes

8 References

1. *MSP430F522x, MSP430F521x Mixed Signal Microcontroller* data sheet (SLAS718)
2. ADXL345 data sheet (http://www.analog.com/ADXL345)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265