Micromega Corporation

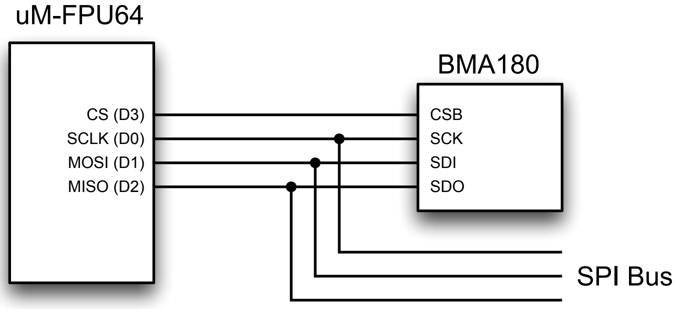
Code Example
uM-FPU64

BMA180 3-axis Accelerometer

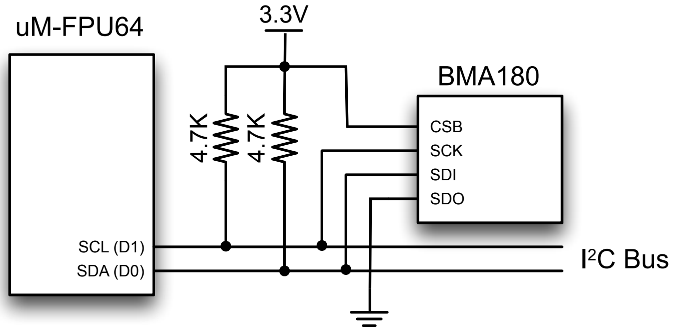
Introduction

This code example provides FPU functions for interfacing the BMA180 3-axis accelerometer to the uM-FPU64 using an SPI or I²C interface.

Connecting the BMA180 to the FPU using SPI



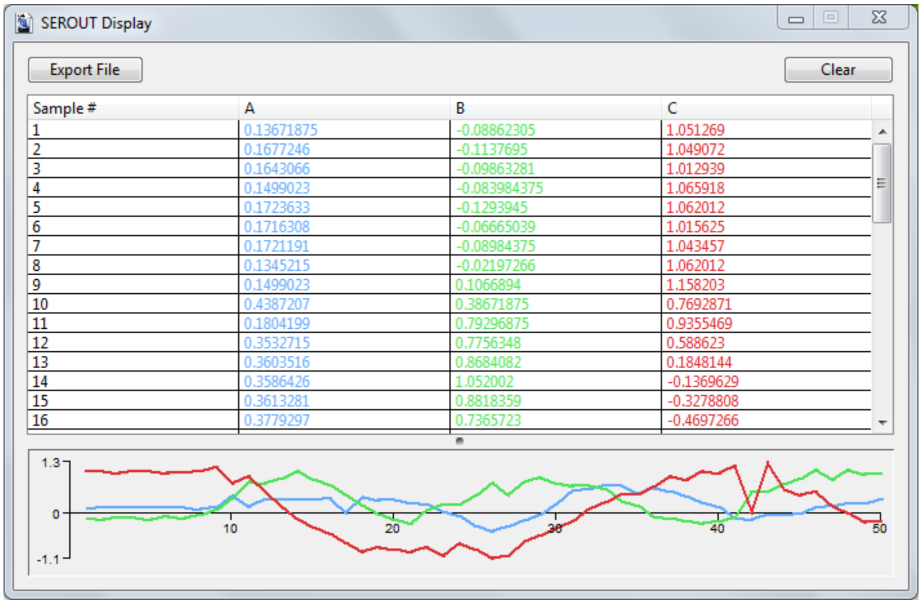
Connecting the BMA180 to the FPU using I²C



Using the SEROUT Display in the uM-FPU64 IDE

When the IDE is connected to the FPU chip and debugging is enabled, all output sent to `SEROUT` is captured by the IDE and can be displayed in one of the ***SEROUT Display*** windows. Tab-delimited data can be displayed in a table and as a graph. The following diagram shows how the output from the *test-bma180.ino* program.

Select the *Window>Serial Setup Options…* menu item, select the ***SEROUT*** tab, and select the option to display ***Table and Graph***.



Sample #	A	B	C
1	0.13671875	-0.08862305	1.051269
2	0.1677246	-0.1137695	1.049072
3	0.1643066	-0.09863281	1.012939
4	0.1499023	-0.083984375	1.065918
5	0.1723633	-0.1293945	1.062012
6	0.1716308	-0.06665039	1.015625
7	0.1721191	-0.08984375	1.043457
8	0.1345215	-0.02197266	1.062012
9	0.1499023	0.1066894	1.158203
10	0.4387207	0.38671875	0.7692871
11	0.1804199	0.79296875	0.9355469
12	0.3532715	0.7756348	0.588623
13	0.3603516	0.8684082	0.1848144
14	0.3586426	1.052002	-0.1369629
15	0.3613281	0.8818359	-0.3278808
16	0.3779297	0.7365723	-0.4697266

The graph shows the data from the 3-axis BMA180 accelerometer as the accelerometer is being moved in different orientations. The graph is auto-scaling and is intended to give a quick visual representation of the data. If further analysis is required, the data can exported to a tab-delimited file using the ***Export File*** button.

FPU Functions

Arduino test file:	*test_bma180.ino*
FPU functions:	bma180-spi.fp4 bma180-i2c.fp4

```
bma_init
```

Initializes the SPI or I^{2}C interface and resets the BMA180.

```
bma_write_enable(long)
```

If *arg1* is zero, writing to the image registers is disabled. If *arg1* is non-zero, writing to the image registers is enabled.

```
bma_write(long, long)
```

Writes a data byte to BMA180 image register. The register address is passed as *arg1*, and the data byte is passed as *arg2*.

```
bma_read(long) long
```

Reads a data byte from a BMA180 image register. The register address is passed as *arg1*, and the data byte returned in register 0.

```
bma_readacc
```

Read the X, Y, and Z acceleration registers, and converts the values to floating point gravitational units (g). The values are returned in registers *xacc*, *yacc*, *zacc*.

```
bma_read_temp() long
```

Reads the temperature register, and converts the value to floating point degrees Celsius. The value is returned in register 0.

uM-FPU64 features shown in Code Example

Compiler Functions and Procedures	uM-FPU64 Instructions
`devio(SPI,…)`	`DEVIO`
`devio(I2C,…)`	`DEVIO`

Shows a demonstration of the uM-FPU64 IDE ***SEROUT Display*** window.

Further Information

See the Micromega website (http://www.micromegacorp.com) for additional information regarding the uM-FPU64 floating point coprocessor, including:

uM-FPU64 Datasheet
uM-FPU64 Instruction Set
uM-FPU64 IDE User Manual
uM-FPU64 IDE Compiler Manual