MSP-430 Host Software Example for the QF4A512

1) Introduction

The QF4A512 is controlled with a set of registers visible through a SPI interface. This Application Note, and the accompanying C source code, provides a complete working example of the QF4A512 when connected to a TI MSP-430 microcontroller.

The goals of this Note are –

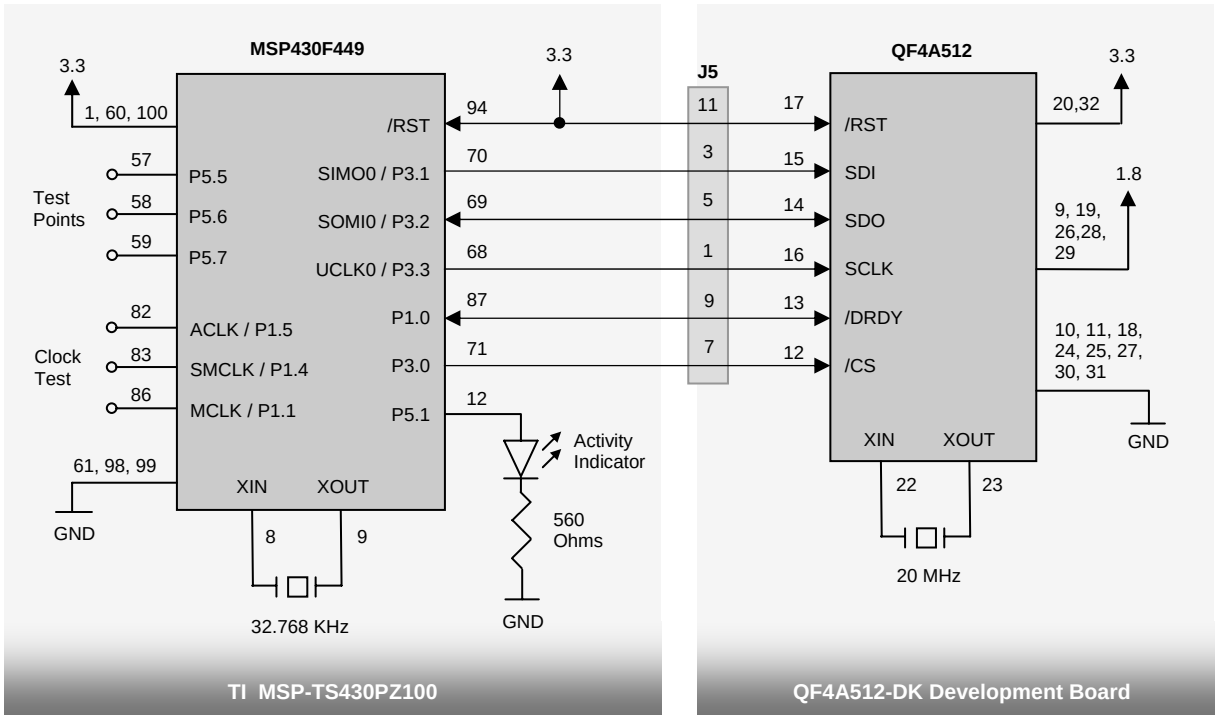
- Demonstrate QF4A512 operation in detail on a popular microcontroller
- Help the developer to get a target running quickly with high-level C code
- Provide reusable source code for customer projects

A .zip archive file, named "QFAN003–Project Content.zip", contains the technical content for this Application Note. Among that content is a Windows help file named "Source Code Documentation.chm" which contains complete documentation of the source code.

2) Demonstration Circuit

The platform for this demonstration is the TI MSP-TS430PZ100 Target Socket Module connected to a Quickfilter QF4A512-DK Development Board, via the USART0 SPI interface.

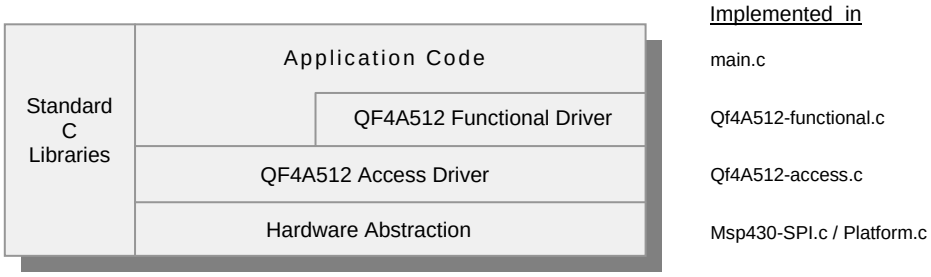
Figure 1 - Schematic Diagram for MSP-430 Example



3) Software Structure

The structure of the software in this example is shown in Figure 2 below. Two files, QF4A512-access.c and QF4A512-functional.c, are the focus of this project. QF4A512-access.c provides raw data transfer functions. QF4A512-functional.c provides a library of specific control features built on top of the API in QF4A512-access.c.

Figure 2 – Software Structure for MSP-430 Example



QF4A512-access.c and QF4A512-functional.c are written without any SPI hardware dependencies and are portable to any processor that uses little-endian byte order and provides an underlying SPI API appropriate to the target device. Msp430-SPI.c provides the SPI API in this case. Msp430-SPI.c and Platform.c are non-portable, and have to be modified for each CPU variant and hardware platform.

Table 1 - Primary Functions in the QF4A512 Library

Feature	Function
Load a configuration table from the Quickfilter Pro PC software	**qf4a512_LoadImageRegisterTable()**
Write a configuration table from the Quickfilter Pro PC software to EEPROM.	**qf4a512_WriteImageRegisterTableToEeprom()**
Read Run mode samples	**qf4a512_ReadSamples()**
Read/write Configuration Registers	**qf4a512_ReadConfigRegisters()** **qf4a512_WriteConfigRegisters()** **qf4a512_ReadConfigByte()** **qf4a512_WriteConfigByte()**
Read/write EEPROM	**qf4a512_ReadEeprom()** **qf4a512_WriteEeprom()** **qf4a512_ReadEepromByte()** **qf4a512_WriteEepromByte()**

Note that this code does not demonstrate device calibration. See "QFAN012 - Calibration of the QF4A512", at www.quickfiltertech.com for information on QF4A512 calibration.

4) Build Procedure

This project was compiled using the IAR Embedded Workbench (www.iar.de), connected to the Texas Instruments MSP-FET430UIF USB-to-JTAG debug interface (www.ti.com).

The Windows help file (HTML) documentation is automated using Doxygen, an open-source tool available at www.doxygen.org. Doxygen can (optionally) use a open-source tool called Dot to create call and inheritance graphs in the documentation. Dot is available at www.graphviz.org.

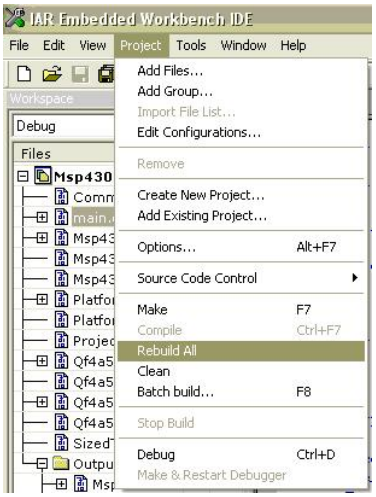
Build Procedure

To build this example, load the tools and unzip "QFAN003–Project Content.zip". Then double-click "IAR Workspace.eww" to open the workspace. To build, select **Project->Rebuild All** from the Main menu. This demonstration project should build with no warnings or errors. Note that the stack size is set to 100 (64h) bytes.

Other Tools

This source code can also been built with other compilers, after minor modification. The other tools are TI's Code Composer Essentials (www.ti.com) and Mspgcc (mspgcc.sourceforge.net).

Figure 3 - Rebuild in IAR Embedded Workbench



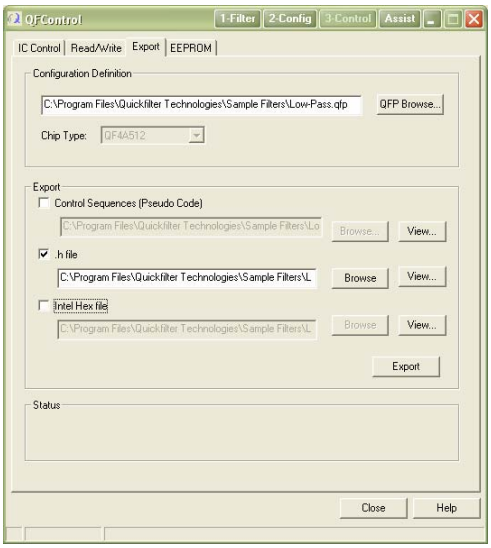
5) Design Flow with PC Software

This section describes how to integrate the output of the Quickfilter PRO PC software with this example code. It's important to point out that this flow is for designs that have changing filter requirements during runtime. Designs that need one configuration should consider programming the QF4A512 internal EEPROM once during production.

The first step is to get the PC software to produce a C header (.h) file that contains a table of configuration entries for the QF4A512. Be sure to use Version 3.1.1, or later, of the PC software. The Export tab on the QFControl dialog, shown below, produces that file.

The table of entries contains both the control register entries and the FIR filter coefficients. It is named "QFImageRegisterTable[]". To load the table at runtime, pass the array to the qf4a512_LoadImageRegisterTable() function.

Figure 4 – Export C header file (.h) from Quickfilter Pro Software



6) General Characteristics

The QF4A512 may be attached to devices of widely varying scale and integration. To support both ends of the spectrum, there are two target models for the QF4A512, Small and Large. This example code represents the **Small** model, which targets small microcontroller-based systems. Some features and assumptions of the Small Model library are –

- Assumes no operating system
  - No file system
  - No kernel logging
  - Unstructured interrupt handling
- Minimal memory footprint
  - No heap (statically allocated buffers)
  - Limited buffer size
- Minimal power consumption
  - CPU idle while not processing data
- Written in C (not C++)
  - Unstructured exception handling
  - Statically linked
  - No namespaces
- Minimal runtime overhead in Release build

Features and assumptions that are common to **all** models, both Small and Large are –

- Supports multiple logical devices
- Structured C
- QF4A512 driver decoupled from hardware by thin, simple hardware access API
- Defensive coding measures in Debug build
- Source code internally documented to produce an HTML reference.
- Interrupt and/or DMA driven

7) Other Notes

The assert() macro, and macros built upon it (AssertNonNull(), etc.), are used to instrument the code in the Debug build configuration. These statements are nulled in Release builds, and have no effect on code size or bandwidth in that case. However, they do have a noticeable impact on code space and runtime bandwidth in a Debug build. The tradeoff is an improvement in the ability to catch many types of bugs earlier in the debug cycle. See assert.h in the C library.

8) Conclusion

This Application note describes the coding example that demonstrates a Quickfilter QF4A512 connected to a Texas Instruments MSP-430 microcontroller. An approximate sample rate of 2.9 KHz for each channel, or 11.6 Ks/s total, was achieved. That rate will go down as the complexity of the algorithm goes up.

In addition to this documentation, a companion .zip archive file, named "QFAN003–Project Content.zip", contains the complete project source code.

Please send any comments, bug reports, etc. about this example to apps@quickfilter.com.

Contact Information:

Quickfilter Technologies, Inc.
1024 S. Greenville Avenue, Suite 100
Allen, TX 75002-3324

General: info@quickfilter.net
Applications: apps@quickfilter.net
Sales: sales@quickfilter.net
Phone: 214-547-0460
Fax: 214-547-0481
Web: www.quickfiltertech.com