SH7216

HW Data Collection and Filtering using the SH7216

R01AN0211EU0100
Rev.1.00
Jan 01,2011

Introduction

The purpose of this application note is to show how to connect multiple peripherals in an application to "off-load" the simple work of periodically triggering the ADC and collecting the data into a buffer.

In many applications it is necessary to collect Analog data from Sensors in real-time and process using complex filtering. Since many filters require multiple samples before any reasonable output can be expected, it is desirable to avoid all of the overhead of having interrupts and the CPU handling this data. It is preferred to handle one interrupt when the desired sample buffer size is ready. This Application note shows how to use 3 peripherals (MTU2, ADC and DMAC) in concert to collect two channels of analog data simultaneously and store in a memory buffer at rates up to ~300 kHz. In addition, it will show the filtering done by the Renesas Signal Processing Primitives Library (SPL) to reduce the work required for implementation of the filters.

Target Device

SH7216 (R5F72167FAV)

NOTE: This application note can be applied to other SH2A-FPU devices with minor changes. Basically, it will apply to any SH2A-FPU core device with the MTU2, 12 Bit ADC and DMAC, for example the SH7239.

Contents

1. Demo Requirements

The following items are required in order to build and run this demonstration.

1.1 Required Renesas Development Tools (software)

The following tools and their versions are required for building the demonstration and following the tutorial. Evaluation editions of these tools are all available for download from our website.

- High-performance Embedded Workshop (HEW) Version 4.08.xx.xx (or later)
  http://www.renesas.com/download

  Web Search Keyword: “High-performance Embedded Workshop”

- Renesas SuperH RISC Engine Standard Toolchain Version 9.3. Release 00 (or later)
  http://www.renesas.com/download

  Web Search Keyword: “SuperH Compiler”

- Renesas Signal Processing Primitive Library (SPL). Version 2.00 (or later).
  http://am.renesas.com/products/tools/middleware_and_drivers/c_splib/splib_standard/splib_standard_tools_product_landing.jsp

  Web Search Keyword: “Signal Processing Primitive”

IMPORTANT: This project contains the correct version of the SPL binary and .h file to run this demo. The user is advised to go to the WEB and download the latest version and agree to the End-User-Licensing (EULA). This is important to get the latest documentation (not included in this app note) and bug-fixes (if any).

1.2 Required Hardware for Demo

This application will run on either the RDK or the RSK platform.

The following items are needed for the demo below.

- Either RSK for SH7216 (includes E10A for Starter Kits) or RDK for SH7216.
  http://www.renesas.com/rsk

  http://am.renesas.com/products/tools/introductory_evaluation_tools/renesas_demo_kits/yrdksh7216/yrdksh7216.jsp

- 2 Analog Signal Sources (i.e. Signal Generators, etc…)

1.3 Demonstration Software

The Demonstration software is intended for use with the Renesas RSK/RDK board. You will also need two (2) analog signal sources to generate data to be sampled. More detailed documentation on the RSK/RDK hardware is included with the kit or can be downloaded from the Renesas Web site.

NOTE: The RDK has a 3 channel analog accelerometer which can be used as the signal source. If not used, it may need to be disconnected to attach alternate source.

2. Building the Code

The demonstration software can be built by opening the HEW Workspace (SH7216_Filters.hws) and building the code using the "**Build All**" icon . The user should select DebugRDK or DebugRSK in the build pull-down to select the target of the build, RDK or RSK respectively. The resulting 'SH7216_Filters.abs' file in the 'DebugRSK' or 'DebugRDK'directory, depending on the build configuration, can then be downloaded using the E10A and HEW.

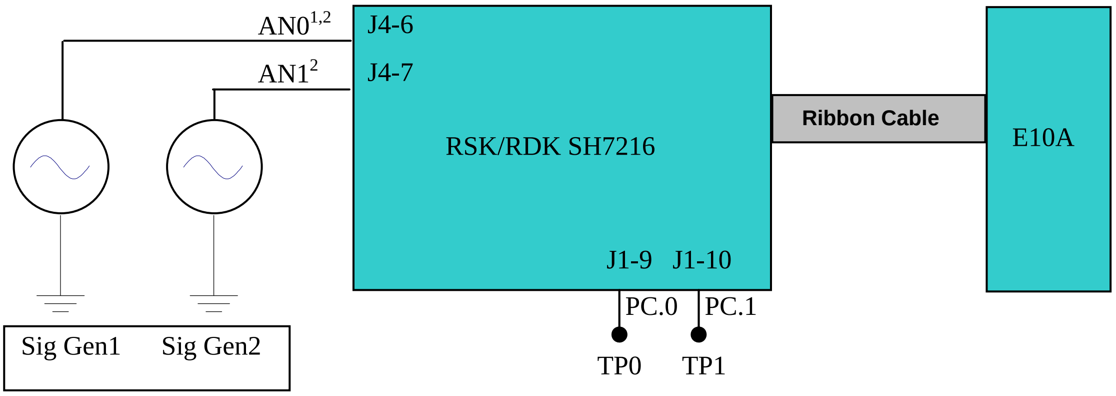
3. Hardware Set-up Description

The demonstration is setup to collect data on ADC Unit 0, channels 0 and 1. The Hardware should be setup as shown in Figure 1. There are some limitations to the RSK/RDK as delivered from the factory. The considerations for this demonstration are:

- AN0 is connected to the potentiometer on the RSK, AN0 and 1 are connected to Accelerometer on RDK.
- On the RSK AN channels have 3K/.01µF filters for protection.

The user should refer to the RSK or RDK schematic and change these according to the use of channel 0 and 1 and the frequencies of the signals you desire to sample.

CAUTION: Your analog sources should not violate the input specifications of the SH7216 ADC channels found in the Hardware Manual. For example maximum V_{AN} is listed as -0.3 to AVCC+0.3. See Absolute Maximum Ratings and ADC characteristics section of SH7216 Hardware manual.



Notes: 1. To use AN0 you should disconnect it from potentiometer by removing 0Ω resistor R87
2. Do not exceed the input specification of these ADC channels
To sample higher frequencies you may need to change input filters of the RSK

Figure 1: Hardware Setup Block Diagram

NOTE: The E10A is built into the RDK, so it is not a separate item as shown in the diagram.

4. Software Demo Description

The following sections describe the Software's operation. The demo does specific filtering on the collected data, but rather the user can easily modify the code in main() where the buffers are signaled as ready to do their own data processing.

Please refer to Figure 2 for details on the peripheral signaling flow.

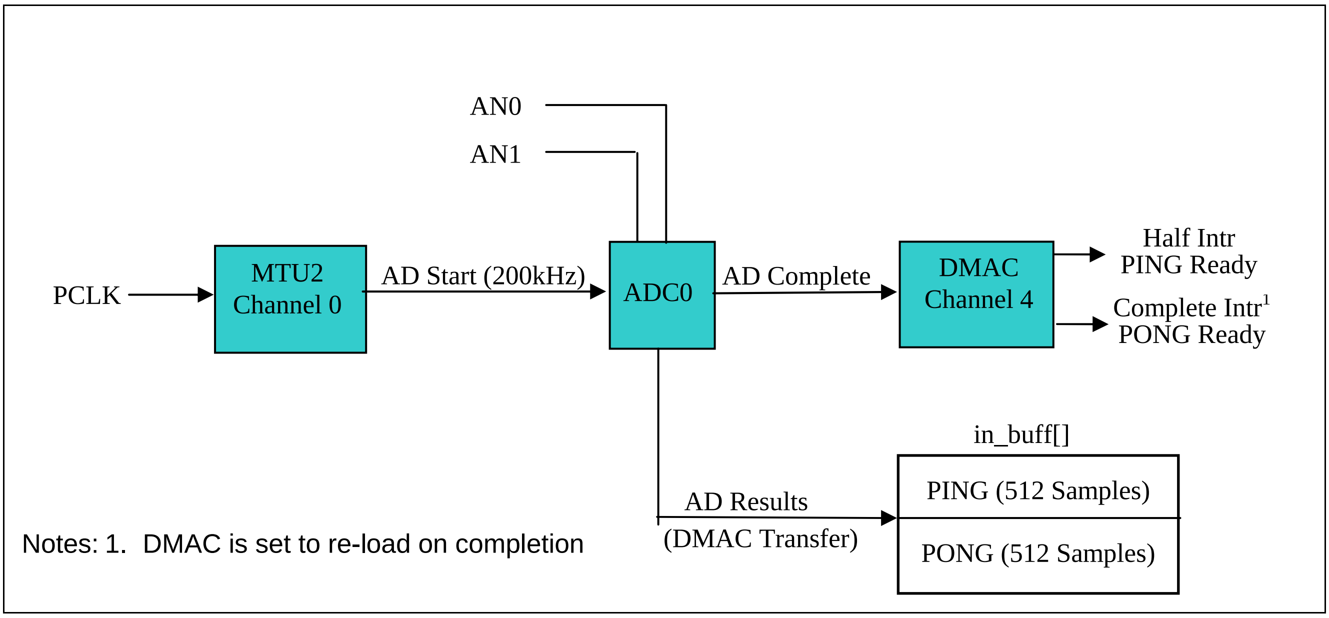


Figure 2: Peripheral Signaling Flow

4.1 PING/PONG Buffer Usage

Many applications utilize complex filters to extract the required information from the Analog signals. These filters typically take multiple input samples to get usable outputs. In addition, these filters may take some extensive compute time. In this demonstration, we utilize a PING/PONG buffer scheme. We collect data into the PING buffer until it reaches the desired sample size. The software signals that the PING buffer is ready and continues to collect the signals in the PONG buffer. This gives the software enough samples and processing time to operate (i.e. run the filters) on the PING buffer while the hardware continues to collect data in the PONG buffer. Compute time allowed is then:

Sample Rate Period * MIN_SAMPLE_SIZE before the next buffer is ready.

See section 0 Buffer Size Considerations for the calculations for this demo.

4.2 MTU2 Setup

The MTU2 is setup to create a periodic rate to the ADC Unit 0 to start a conversion. The sampling rate is set in MTU2.TGRA (compare A). This compare bit is used because it can trigger ADC start.

The rate is based off the definition for sampling frequency in the common_def.h file. These are based on MACROs which define the CPU Clock, I clock, B clock and P clock (see the SH7216 hardware manual for details on the clock structure). The maximum sample rate should not exceed the limits of the microcontroller. These limits are based on the various clock rates in the system, the number of channels converted, etc…. . For this demo, the SH7216 is running at maximum clock on ADC and converting two channels, so the limits are:

~312kHz with ADBYPSCR.SH=1 and ADBYPSCR.OFC=0 (S/H, offset correction ON)

```
#define     SAMPLING_FREQ 200000L   // 200.0 kHz
```

NOTE: If different crystal is installed the user will need to change the macros in common_def.h that sets those numbers. The PCLK driving the MTU2 basically is sourced by the CPU clock through some divider chains, so they are related. This is handled automatically by the build configurations for RSK or RDK.

4.3 ADC Setup

This demo uses ADC controller 0. This converter was chosen because it has the ability to trigger the DMAC. It is set for a single scan with Channels 0 and 1 selected to convert. The individual Sample and Holds are enabled so Channels 0 and 1 are sampled simultaneously (i.e. time coherent). The Offset correction is turned on, so there are extra clocks required for the Offset correction and thus limits the maximum rate (see MTU2 setup section above).

The ADC clock is programmed to maximum, 50MHz in the case of the default RSK for SH7216, it is 48MHz for the RDK.

4.4 DMAC Setup

We chose to use channel 4 of the DMAC controller. Channel 4 is the “subset” channel (i.e. channel 0 - 3 have some minor additional features that were not required when transferring data from ADC to internal memory).

NOTE: Although any DMAC channel can be used to do this ADC transfer, the designer must evaluate the system requirements when choosing channels to use. For example if you need to move data externally using single cycle transfers, these are only supported by channels 0-3. So it can be seen these should be allocated first.

The DMAC is setup to operate as follows:

- Triggered by ADC converter 0 complete
- One longword transfer per request
- total transfers = MIN_SAMPLE_SIZE*2
- Interrupt when Half complete
- Interrupt when complete
- Auto reload on completion of count

5. Filtering

The Demo code uses the Renesas Signal Processing Primitives Library (SPL) to filter and separate the two frequencies into two buffers after the data is collected. For this demo we will show you the results based on "simulated" (mathematically generated) data.

NOTE: This application note is not intended as a class in filtering. It is recommended the reader go to some of the references listed in the section 8 to review filter concepts.

5.1 Filter Construct and Flow

The filters for this demo are "constructed" using the User Process in the SPL to concatenate a Boxcar and a FIR filter with Decimation. This filter structure is shown in Figure 3 . When the minimum number of samples are collected, they converted to floats in a function call to prep_filter_in_buff(). This function converts the input to a from raw ADC data to float values. This float data is then passed to the filter Decimation process by a single call.

To find more detail on the Filter Decimation process the User can run the Demo projects that come with the installer of the SPL (see section 8).

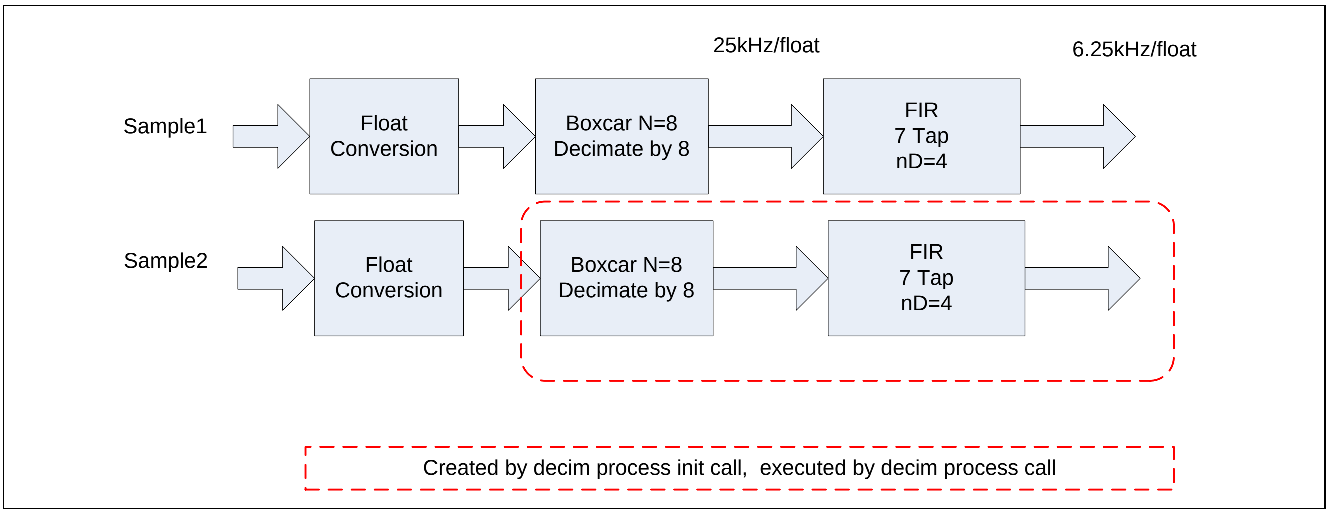


Figure 3: Filter Constructed by User Process Call

The boxcar filter coefficients are as expected, 1/8 or .125. The FIR filter coefficients are calculated using ScopeFIR. The LP filter, 7 tap FIR, is set using passband of 50Hz, stopband of 5000Hz, 2dB of ripple and 30 dB of attenuation in the stopband. The HP filter, 7 tap FIR, is set using stopband upper of 400Hz, passband of 5000Hz, 1dB of ripple and 40 dB of attenuation in the stopband.

NOTE: The user is referred to section 8.1, ScopeFIR for additional information.

5.2 Generated Sample Data

In the demonstration code, we use numerically generated sample data representing 400Hz and 5000Hz. The frequencies are defined at approximately line 364 of main.c

```
#define MY_HERTZ1 400.0f
#define MY_HERTZ2 5000.0f
```

The use of Generated data versus real ADC data is controlled by the define at line 45 of main.c

```
#define GEN_DATA
```

Please refer to section 6.2 to view the resulting sampled data.

The demo code has the coefficients set up top do a low pass and a high pass to separate these into two separate buffers. To break on a complete pass of the filter (i.e. on full output buffer), you can add a breakpoint at the nop at line 460 of main.c as shown below:

```
mtu2_ch0_ctl(OFF);      // stop timer, stops all since it is trigger
nop();
```

Please refer to section 6.3 to view the resulting filtered data

6. Demo Debug/Runtime Hints, Tips and Tricks

The following section gives some tips on using this demo application. .

6.1 STOPPING Data Collection in Break

As pointed out in earlier in this document, the data collection does not stop until told to do so by the processor. The easiest way to stop data collection is to just shutdown MTU2 Channel 0. The demo software provides a couple of methods to do this.

It can be stopped by a function call to stop_data_collection().

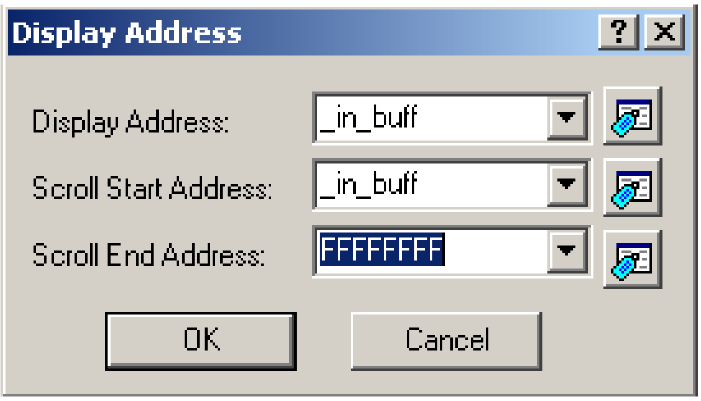
It can be stopped at buffer boundaries by placing an eventpoint or breakpoint at the nop() at line 626 in check_for_break(). The user adds stop_now to the watch window and sets it for automatic update (R). The user then sets the stop_now variable to a 1 and the program will break at the NOP and will be stopped on “buffer boundary” for viewing a full buffer.

6.2 Viewing ADC Sampled Data Excel

In this section we are showing you how to view sample data.   Since most sample data will not look so clean coming out of you sensors, we will use the generated data for this exercise, but it works regardless of what is in the sample buffer.

Using Menus: View ➔ CPU ➔ Memory

Set the Memory Dialog as Shown in Figure 4: Memory Window Dialog.



Display Address: _in_buff

Scroll Start Address: _in_buff

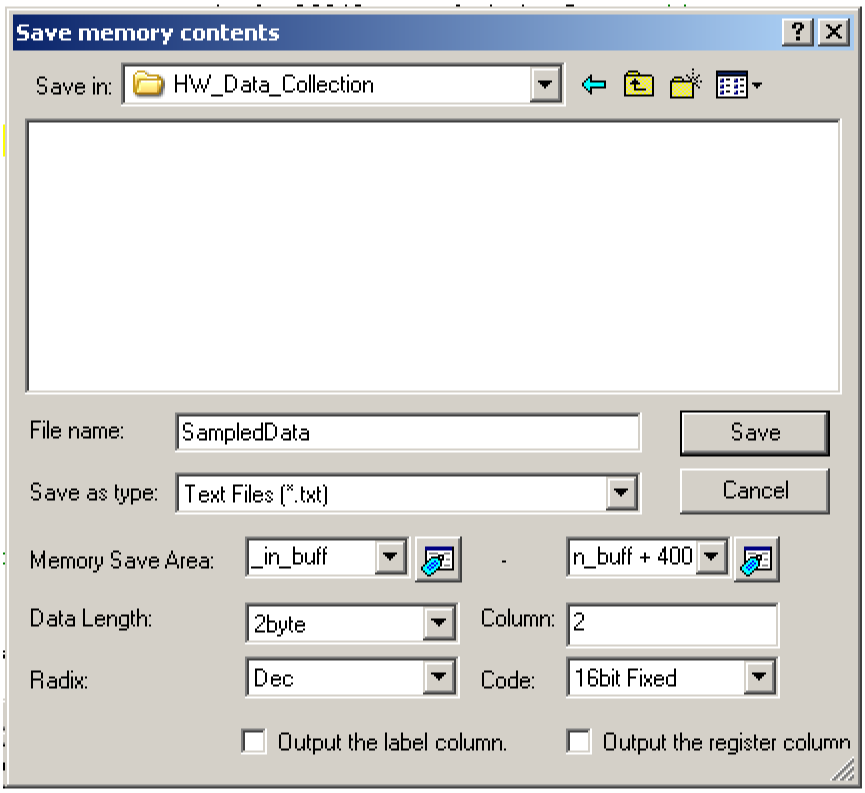
Scroll End Address: FFFFFFFF

Figure 4: Memory Window Dialog

Using right clicks in the memory window set it as follows:

- Data Length – 2 bytes
- Radix – Decimal
- Column – 2
- Code – 16 bit fixed

Right Click the memory window and “Save the Memory Contents” to a file setting the dialog box as shown in Figure 5: Save Memory Dialog.



Save memory contents

Save in: HW_Data_Collection

File name: SampledData

Save as type: Text Files (*.txt)

Memory Save Area: _in_buff - n_buff + 400

Data Length: 2byte

Column: 2

Radix: Dec

Code: 16bit Fixed

□ Output the label column.

□ Output the register column

Figure 5: Save Memory Dialog

NOTE: Memory range saved for the plot below is _big_sample_buff to_big_sample_buff + 1000. For plotting real data use _in_buff to in_buff + 1000.

You can now import the text file into Excel and plot either columns 3 and 4. The resulting plot will look something like Figure 6for the mixed sample data sine waves in this demo.

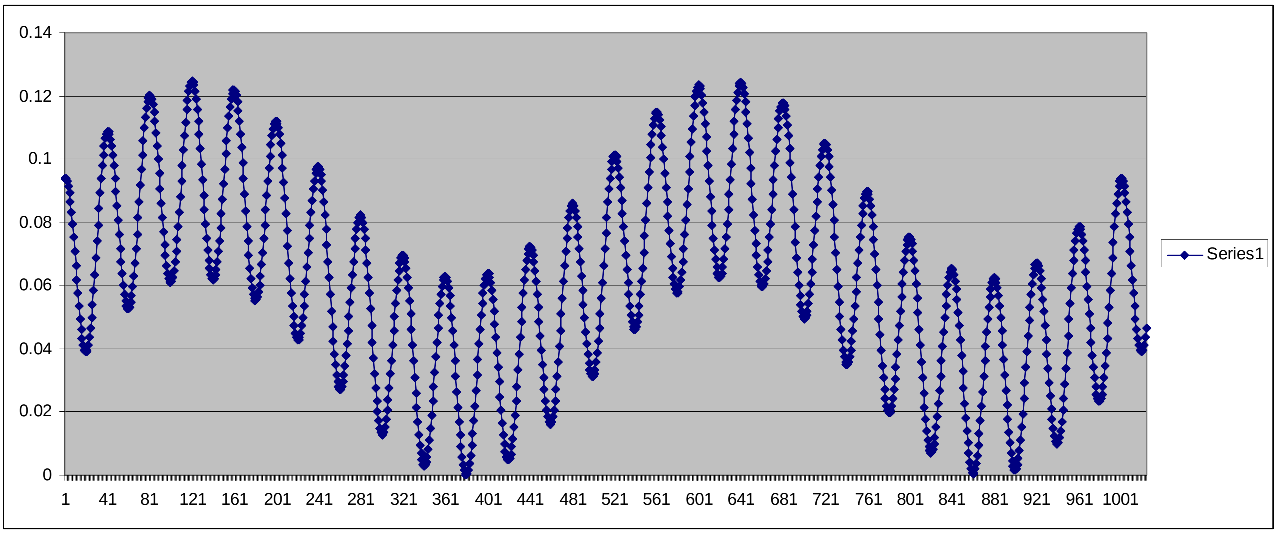


Figure 6: Sample Excel Chart

6.2.1 Viewing Generated Data

The generated data can be viewed in a number of ways, but the two methods that will give the user a feel for “what does the data look like?” are by Viewing Graphic or dumping the memory buffer into an Excel file and charting.

NOTE: The following data has been normalized from 12 Bit ADC to the range 0 to 32768 for easy viewing in the Waveform viewer. This is done when the data is generated by code in main() at line 376.

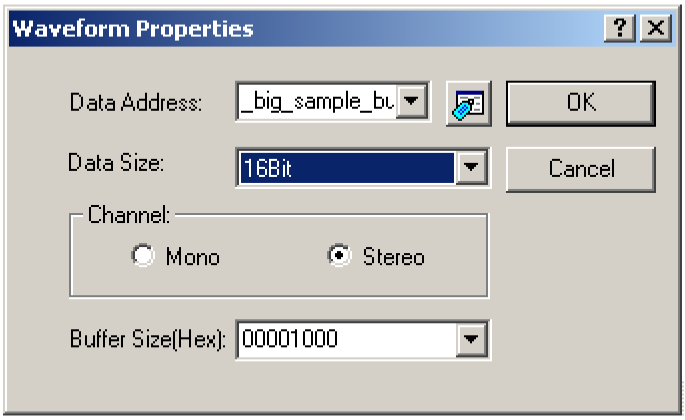
```
tmp_data = ((UI16) ((( sinf(angle1)*2048.0)+2047.0) + (UI16)(( cosf(angle2)*2048.0)+2047.0))/2) * 8;
```

NOTE: The sample data buffer is returned to the proper range by dividing by 8 in a subsequent loop before the code runs the filters.

6.2.2 Viewing Graphic

Using Menus: View ➔ Graphic ➔ Waveform

Fill in Dialog box as shown in Figure 7.



Waveform Properties

Data Address: _big_sample_bu

Data Size: 16Bit

Channel: ○ Mono ● Stereo

Buffer Size(Hex): 00001000

Figure 7: Waveform Dialog box

Figure 8 shows the data of two different frequency sine waves. In this case we generated a 400Hz sinewave mixed with a 5kHz sine wave 50/50.

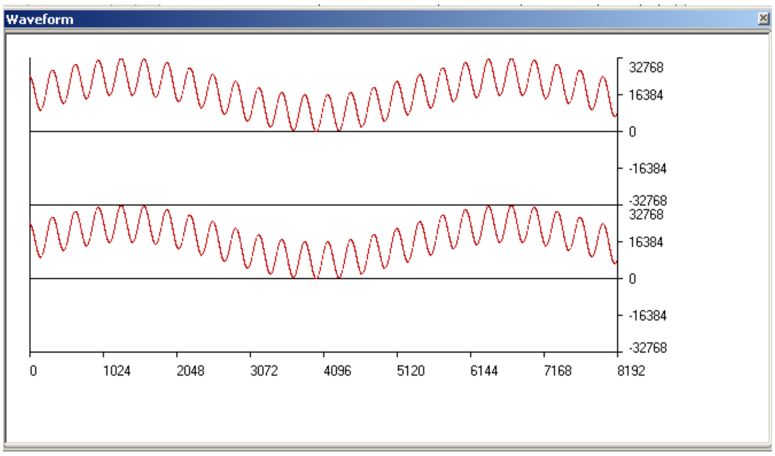


Figure 8: Waveform Window

6.3 Saving Filter Results for viewing in ScopeDSP

ScopeDSP is a low cost tool that can be used to quickly evaluate the results of your filter. In order to use it, you must first save the results into a file. We will save the memory contents and then with minor editing make it acceptable to ScopeDSP. This section shows you how to save the data for viewing in ScopeDSP; we will not give lessons in ScopeDSP.

NOTE: For a step by step lab in generating filter coefficients and using ScopeDSP for analysis the user can refer to the DevCon lab listed in section 8 of this document.

Open a memory window as follows:

Menu: View ➔ CPU ➔ Memory

Display Address: big_out_buff1

Scroll Start Address: big_out_buff1

Scroll End Address: big_out_buff1 + H'7ff

Right click in the memory window and setup window to display an integer in Hex format. This is the data that has been output by the High Pass filter.

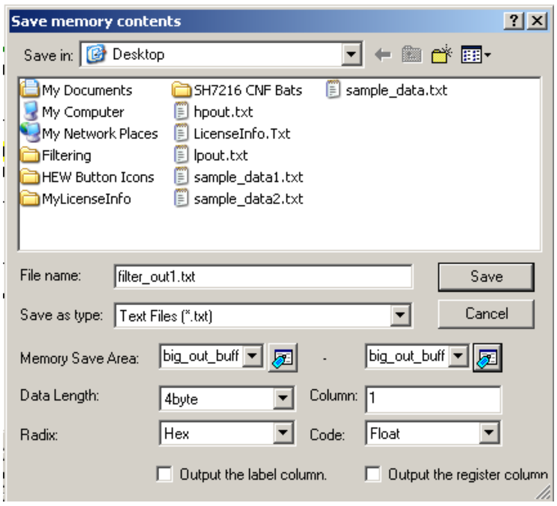
Column – 1

Data Length – 4 byte

Radix – Hex

Code – Float

Right click to Save the memory contents as shown in the dialog box in Figure 9, it should "inherit the setting from the window, but if necessary, fill in dialog with same information as you used in setting up the Memory Window, saving big_out_buff1 to big_out-buff1 + h'7ff.



Save memory contents

Save in: Desktop

File name: filter_out1.txt

Save as type: Text Files (*.txt)

Memory Save Area: big_out_buff - big_out_buff

Data Length: 4byte

Column: 1

Radix: Hex

Code: Float

□ Output the label column.

□ Output the register column

Figure 9: Output Buffer Save Dialog

Open the text file in HEW or your favorite Text Editor. Remove the first line of text (which should be "ADDRESS +0 FLOAT"). Remove the first column of data leaving only the list of floating point results.

You can now read this "Time Data" into Scope DSP. Your resulting FFT should look like Figure 10. The sample data used to generate this plot is available in file *HP_Sample_Data.txt* in the DOCS directory of the project.

If you repeat the steps above but save data from big_out_buff2 to big_out-buff2 + h'7ff, you can view the results of the low-pass filter running in this demo.

You can now read this "Time Data" into Scope DSP. Your resulting FFT should look like Figure 11. The sample data used to generate this plot is available in file *LP_Sample_Data.txt* in the DOCS directory of the project.

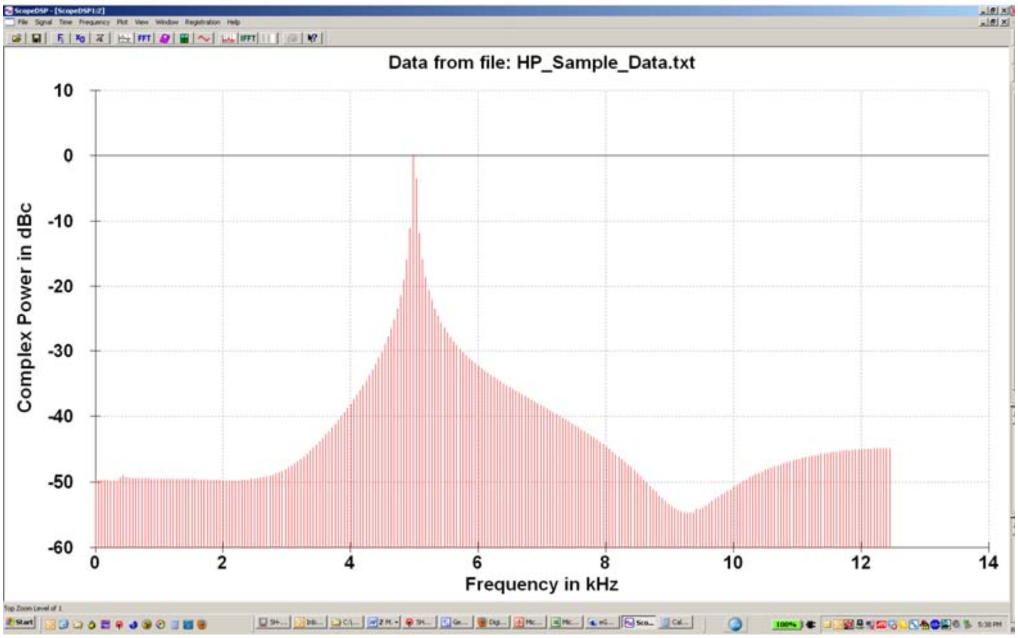

ScopeDSP - [ScopeDSP1:2]
File Signal Time Frequency Plot View Window Registration Help
Data from file: HP_Sample_Data.txt
Complex Power in dBc
10
0
-10
-20
-30
-40
-50
-60
0
2
4
6
8
10
12
14
Frequency in kHz

Figure 10: FFT of HP_Sample_Data.txt

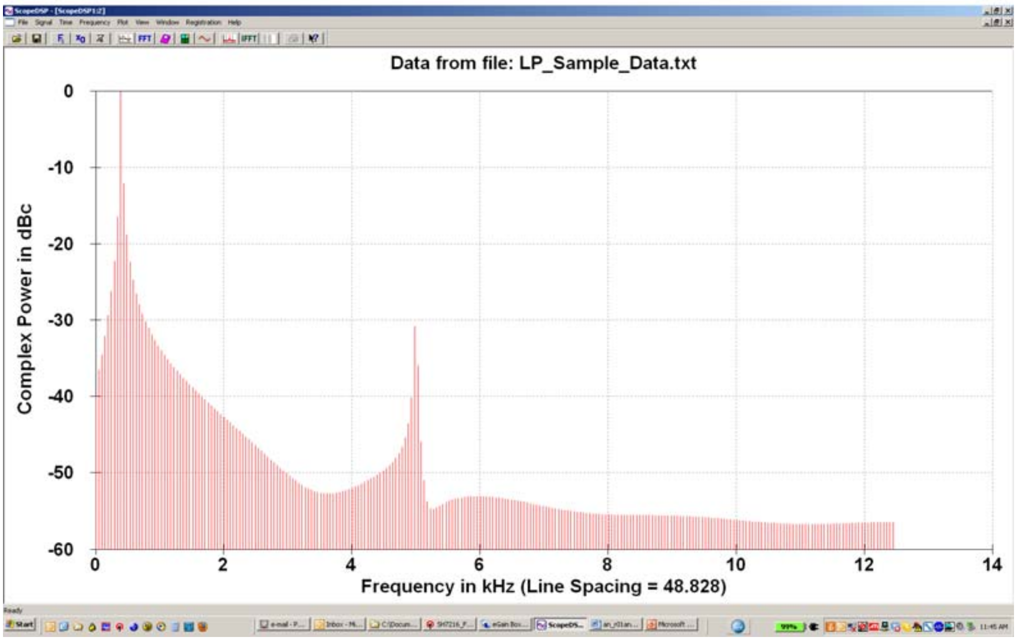

ScopeDSP - [ScopeDSP1:2]
File Signal Time Frequency Plot View Window Registration Help
Data from file: LP_Sample_Data.txt
Complex Power in dBc
0
-10
-20
-30
-40
-50
-60
0
2
4
6
8
10
12
14
Frequency in kHz (Line Spacing = 48.828)

Figure 11: FFT of LP_Sample_Data.txt

6.4 Buffer Size Considerations

The data rate at which you collect data along with the size of the buffer will affect the processing time you have in between the signaling of a "buffer ready" status. An example of this is the rate and buffer size used in this demo software.

Example: MIN_SAMPLE_SIZE= 512 samples collected at a rate of 200kHz. Therefore you will get a BUFFER Ready status (PING or PONG) every (1/200kHz)*512 = 5µS*512 = 2.56mS. So you have 2.56mS to act on the 512 samples of data before the next buffer is ready.

So it is important to validate that whatever you need to do with the data buffer (Filters, normalization, etc…) you have sufficient compute time to avoid the data collection buffer being overwritten by the incoming data (i.e. overwriting of the collected data by the hardware before the software has had time to "digest" it).

6.5 Buffer Usage Considerations

The data that has been collected obviously must be processed. This data processing may be as simple as converting from raw ADC values to scaled values representing motor current, voltage, etc… to complex filtering. Since the hardware data collection does not stop, care should be taken that the data processing does not exceed the time calculated in section 6.3

In the demo code, the incoming data is "processed" in line 439 through 443. The code processes the data as follows:

Line 439 – scales the incoming data and converts to floats in preparation for the floating point filter.

Lines 440 and 441 – Sets pointers used by the filter process function call.

Line 443 – Filter Process function call

Lines 448 and 445 – put the output from the filter call into the correct buffer for later viewing.

The code is written for clarity and may be optimized in your application.

6.6 Test Point Usage

The Demo software provides two GPIO which can be used as test points for triggering or measurement points. These are declared (along with macros for setting and clearing) in the file custom_bsp.h as follows:

```
#define TRIGGER_PORT          PC.DR.BIT.B0         // PC0 data bit
#define TRIGGER_PORT_PFC      PFC.PCCRL1.BIT.PC0MD  // Pin Function
#define TRIGGER_PORT_DIR      PFC.PCIORL.BIT.B0    // Pin direction (IN/OUT)

#define TRIGGER_PORT1         PC.DR.BIT.B1         // PC0 data bit
#define TRIGGER_PORT1_PFC     PFC.PCCRL1.BIT.PC1MD  // Pin Function
#define TRIGGER_PORT1_DIR     PFC.PCIORL.BIT.B1    // Pin direction (IN/OUT)

#define SET_TRIGGER_PORT(ON_OR_OFF)   (TRIGGER_PORT = ON_OR_OFF)
#define SET_TRIGGER_PORT1(ON_OR_OFF)  (TRIGGER_PORT1 = ON_OR_OFF)
```

The demo code as shipped is using the first test point, TRIGGER_PORT1 to indicate when data is being processed by the filter. The trigger port is set and cleared at lines 435 and 453 respectively. Scope shot shown in Figure 12.

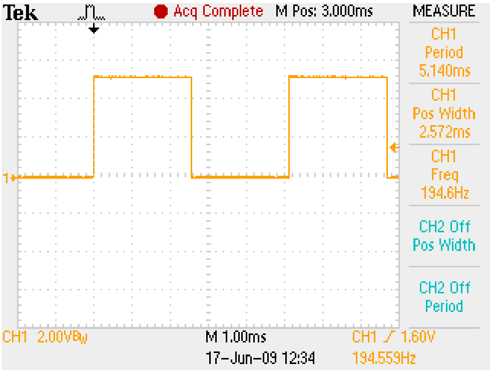


Figure 12: Trigger port 1

7. Limitations of Testing

This demo software combines the use of multiple peripherals and thus contains example code for these peripherals.

7.1.1 DMAC Driver

The demo code for the application note contains a mini-DMA Driver. It supports the programming of multiple channels on the DMAC, but all permutations have not been tested. It has only been used to the extent required by this application note, which is DMAC Channel 4 triggered by ADC0 Complete, Half-interrupt enable, Transfer Complete Interrupt enable and Auto-reload.

7.1.2 ADC Driver

The demo code for the application note contains a mini-ADC Driver. It supports the programming of multiple converters and channels on the ADC, but all permutations have not been tested. It has only been used to the extent required by this application note, that is ADC0, channels 0 and 1 enabled for conversion, single sweep, and AD Start triggered by MTU2 channel 0.

7.1.3 MTU2 Channel 0 Usage

MTU2 channel 0 was chosen to trigger the ADC, but any of the MTU2 or MTU2S channels that can create an AD_START can be used. In addition, you can save the timer and use /ADTRG (ADC trigger input) to start the process if you have some fixed rate hardware signal related to the analog sampling.

7.1.4 RSK/RDK Considerations

Much of the demo is based on RSK code and as such should be very familiar to those who have built tutorials on the RSK or RDK. We have not endeavored to strip any of this so some functions are available to enhance this demo code (i.e. switch and LCD functions), but are not extensively used.

CMT0/LED: CMT0 is programmed for 500mS rate and its interrupt service routine toggles LED0 on the RSK, LED1 on the RDK. This is used to indicate that the program is running.

LCD: The LCD is initialized as in RSK tutorials and a SPLASH screen is displayed before the Data collection is actually running. Beyond that, the LCD may be used to post other messages if desired.

CONSOLE: We have included code to enable stdin, stdout and stderr console operation. The Serial port may be connected to a terminal emulator set as follows: 115200 baud, 8 data, 1 stop, no Parity.

CRYSTAL Selection: The demo program runs on both the RDK and the RSK. When connecting to the target using E10A debugger, the user should select 12.00 MHz crystal for the RDK and 12.50 MHz for the RSK. This is necessary because we only setup one Session and may not be necessary if the user generates two separate sessions.

7.1.5 Reducing / Expanding the concept

This demo is written to fulfill a very specific data collection concept, two analog channels at a high rate and then post processed by some numerical function. To reduce this should be a very simple process (not undertaken here). Simply reduce the sample buffer to one entry (sample1) of MIN_SAMPLE_SIZE, set the DMAC to transfer one word instead of long word.

To expand the concept is just a little more difficult. This dual channel data collection works because we use some tricks in the process. We set the DMAC to do one transfer in response to a request from the ADC, we set the transfer size to a longword. When the DMAC does the single transfer it is broken into two words internally and since ADC result register 0 is adjacent to ADC result register 1, we get both channel 0 and channel 1 results transferred to memory (in_buff.sample1 and in_buff.sample2 respectively).

The Concept can be extended by using the DTC to perform a more complex transfer from the ADC result registers to the memory buffer. In addition, the DTC can be triggered by ADC1 so this concept could be extended to the full number of ADC channels in the microcontroller. Of course the total data collection rate will be based on the ADC Clock, the number of channels and the size of the data buffer. This exercise is left to the reader.

8. References

SH7216 Group Hardware Manual, REJ09B0543-0100

Renesas Start Kit 2 for SH7216 Documents

- User’s Manual, REG10J0147-0100
- Schematic, RJJ99J0066_rsk+sh7216
- Quick Start Guide D009374_11_S01_V01

Renesas Demo Kit for SH7216 Documents

- User’s Manual, REU10B0007-0100
- RDK Schematic (Available on Installer CD)
- RDK Quick Start Guide

SH7216 Group Application Notes

- SH7216 Group A/D Activation at MTU2 Channel 0, REJ06B1013-0100
- DMAC Dual Address Mode, REJ06B0767-0100 Rev 1.00

SH7137 Group Application Notes

- 12-Bit A/D Converter: Example of Settings for Conversion in Single-Scan Mode, REJ06B0746-0100 Rev 1.00

Renesas Signal Processing Primitive Library (SPL), Version 2.00

- User’s Manual, REU11B0002-0200 Rev 1.06
- HEW Sample Project Instrctions, SH2A, REU12B0001-0105

DevCon 2010 Lab A30L FPU Filter Lab (available on DevCon website)

8.1 Filter References

Iowegian ScopeFIR, ScopeDSP and Scope IIR tools available at http://www.iowegian.com/

DSP Guru http://www.dspguru.com/

WinFilter http://www.winfilter.20m.com

The Scientist and Engineer's Guide to Digital Signal Processing, copyright ©1997-1998 by Steven W. Smith. For more information visit the book's website at: www.DSPguide.com

C. E. Shannon, "Communication in the presence of noise", Proc. Institute of Radio Engineers, vol. 37, no. 1, pp. 10–21, Jan. 1949. Reprint as classic paper in: *Proc. IEEE*, vol. 86, no. 2, (Feb. 1998)

9. Glossary

ADC – Analog to Digital Converter

CMT – Compare Match Timer

CPU – Central Processing Unit

DMAC – Direct Memory Access Controller

DTC – Data Transfer Controller

FIR – Finite Impulse Response

GPIO – General Purpose Input/Output

IIR – Infinite Impulse Response

LCD – Liquid Crystal Display

MTU2 – Multifunction Timer Unit 2

PCLK – Peripheral Clock

RSK - Renesas Starter Kit

SPL – Signal Processing Primitive Library

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.01.11	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

     In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

     In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0