To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

H8S/2239

Direct Drive of 1/4 VGA LCD Via H8S Microcontroller

Introduction

This Application Note details how it is possible for a H8S microcontroller to drive a controller-less ¼ VGA LCM (Liquid Crystal Module).
The devices used in this application are a Renesas H8S/2239 and a Hitachi SP14Q006-ZTA.

The H8S/2239 is a high performance 16-bit embedded microcontroller with many integrated peripherals. By combining the functionality of the TPU (16-bit timer unit) and the DMAC (Direct Memory Access Controller), it is possible to drive the LCM.
It will be shown that despite the very intensive requirements of the application on CPU performance and Bus bandwidth, that only 30% of the H8S bus bandwidth is used, and when displaying a static image that < 1% of the available H8S CPU performance is used.

Contents

Overview of System

Figure 1 shows a simplified block diagram of the H8S/2239 & LCM used in this application note.

It shows that the H8S/2239 uses 3 of its 6 16-bit timers to generate the LCM control wavefroms and the DMAC is used to update the LCM 4-bit interface.

The CPU is only used when one complete screen of data has been transferred to the LCM. At this point the CPU reconfigures the TPU's, DMAC and starts everything once again.

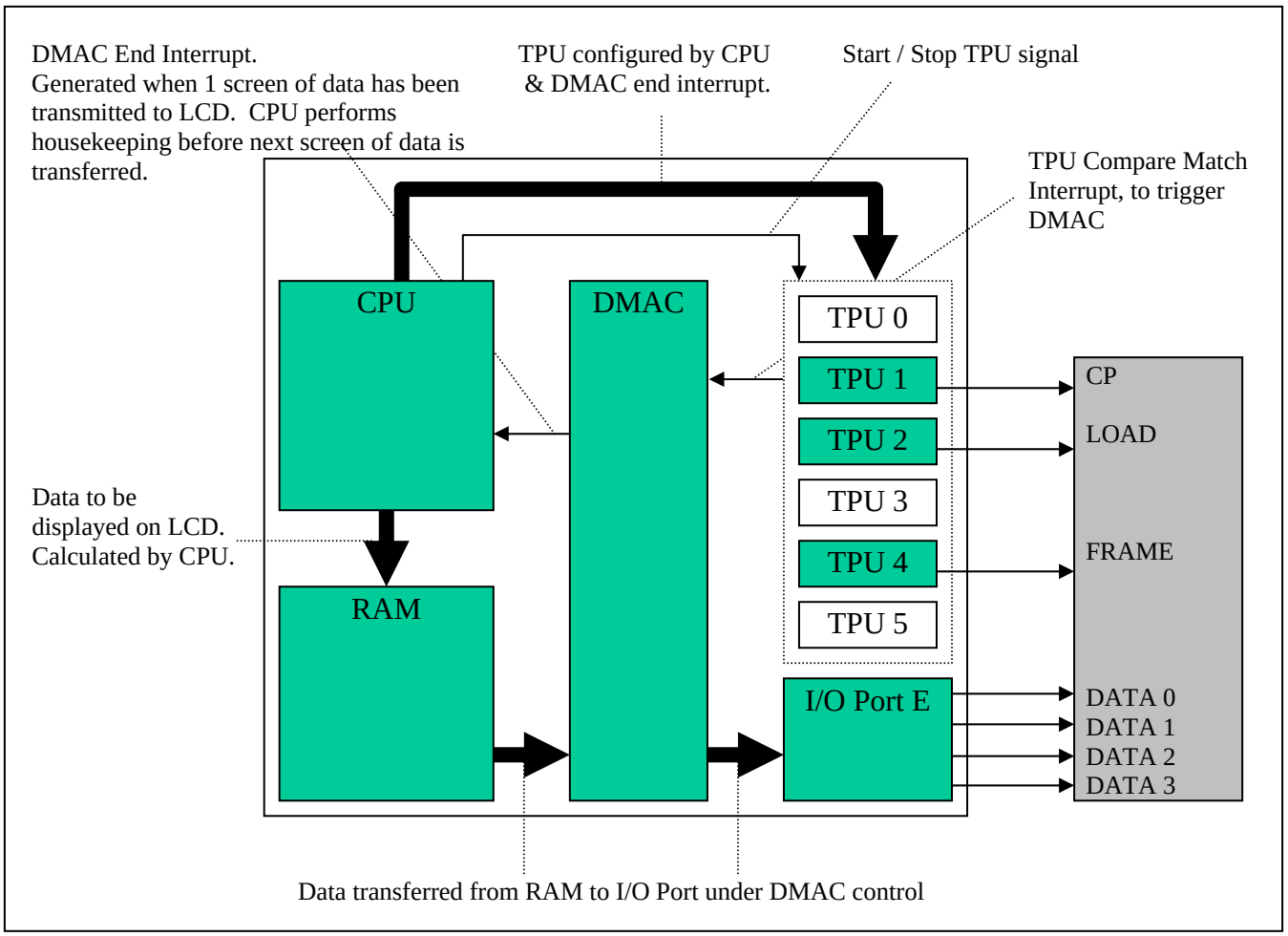

Figure 1.

Requirements for SP14Q006 LCM Direct Drive

Figures 2 & 3 show the basic interface timing and timing characteristics for the SP14Q006.

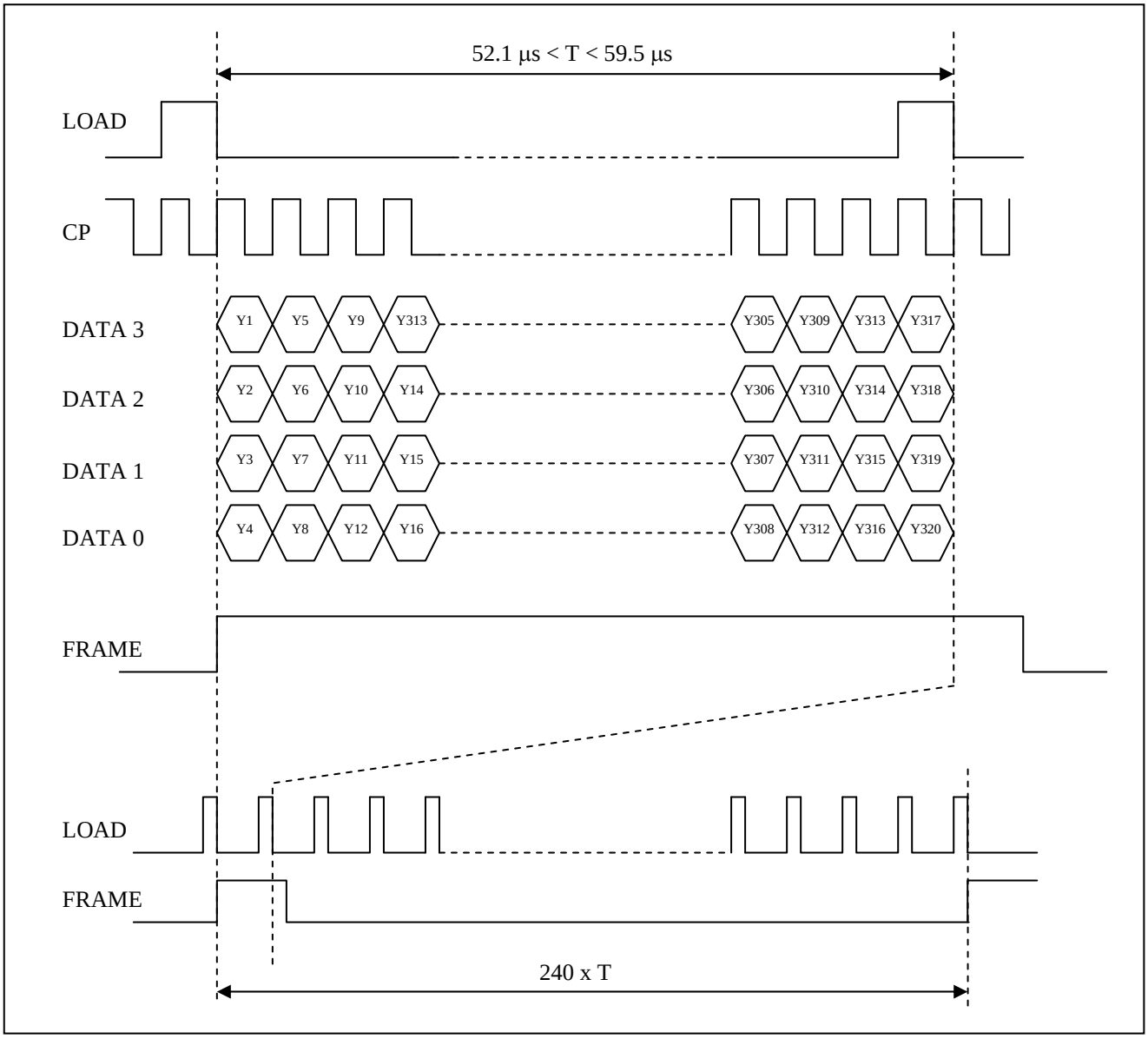

Figure 2.

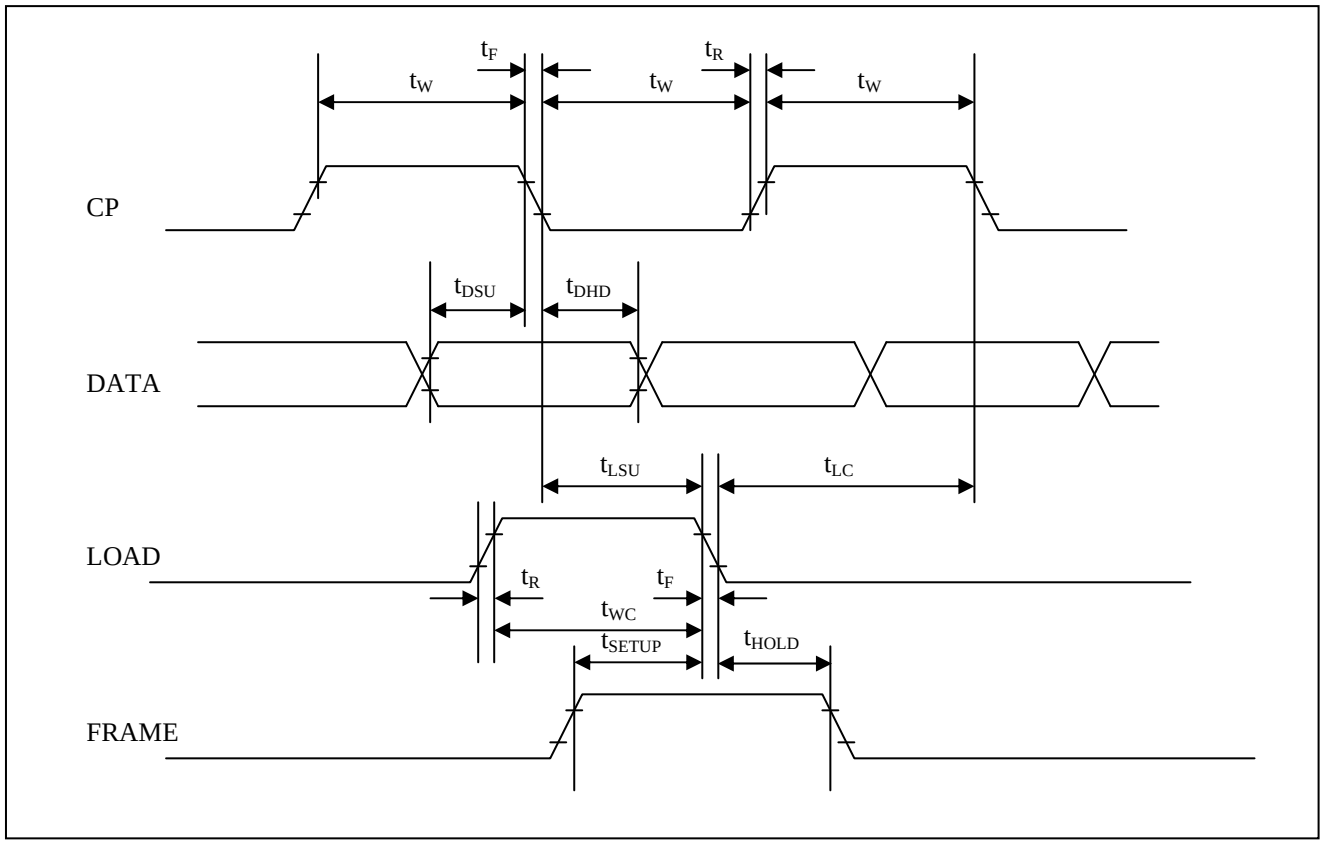

Figure 3.

Item	Symbol	Min	Max	Unit
Clock Frequency	f_{CP}	-	6.5	MHz
Clock Pulse Width	t_W	63	-	ns
Clock Rise / Fall Time	t_R / t_F	-	20	ns
Data Setup Time	t_{DSU}	50	-	ns
Data Hold Time	t_{DHD}	50	-	ns
Load Setup Time	t_{LSU}	80	-	ns
Load Clock Time	t_{LC}	100	-	ns
Frame Setup Time	t_{SETUP}	100	-	ns
Frame Hold Time	t_{HOLD}	100	-	ns
Load Pulse Width	t_{WC}	125	-	ns

Table 1.

To generate the required data and control signals, the H8S uses a combination of the TPU and DMAC.

The TPU is responsible for generating the 3 control signals, LOAD, CP and FRAME.

The DMAC is used to transfer the data from internal memory to an I/O port, which is connected to the DATA lines.

H8S Resources used

TPU: Channels 1, 2, 4.

TPU1: Generates CP signal. LCD data is clocked in on the falling edge of this signal DMAC is triggered on the rising edge and data is transferred to the I/O port.

TPU2: Generates LOAD signal. (1 Pulse per line of data)

TPU4: Generates FRAME signal. (1 Pulse after every screen)

SRAM: 19200 bytes to store LCD image

DMAC: Channel 0A
Configured for Short Address Mode, Dual Address Mode, Sequential Mode.[1]

TPU Settings

It can be seen from figures 2 & 3 that the LCM requires 80 (320 / 4) clock cycles to clock in a line of data. This data must be clocked in at such a rate so that time T is not violated.

From the data sheet:

$$52.1\ \mu s < T < 59.5\ \mu s$$

The maximum clock frequency of the H8S/2239 is 16 MHz @ 3V, but for this application the H8S/2239 was operating at a clock frequency of 14.7456 MHz[2].

Therefore, the H8S/2239 system clock, $f_{SYS} = 1 / 14.7456$ MHz

$\therefore \quad t_{SYS} = 67.82$ ns.

The quickest waveform to be generated is that of the CP signal. The relationship between this waveform and the H8S system clock is shown in figure 4.

[1] PLEASE REFER TO APPENDIX A FOR AN EXPLANATION ON DMAC MODES.

[2] THE APPLICATION CODE WAS PROGRAMMED INTO THE H8S/2239 FLASH MEMORY VIA SCI2. AN OPERATING FREQUENCY OF 14.7456MHZ ENABLES THE SCI TO COMMUNICATE WITH A HOST AT 115200 BAUD WITH 0% BIT ERROR.

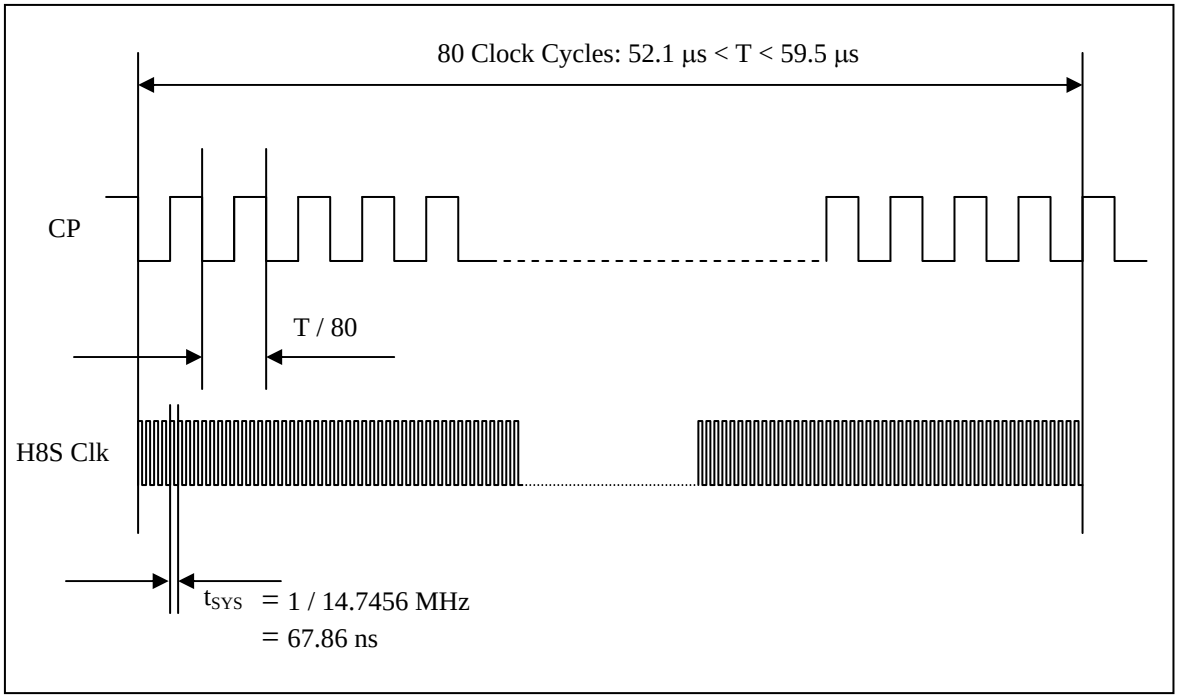

Figure 4.

It can be seen from figure 4 that it requires ‘X’ H8S clocks to generate a single CP clock.

The relationship is:

$$X = (T / 80) / (1 / 14.7456M)$$

Transposing for T:

$$T = (80 X) / 14.7456M$$

With this relationship, it is possible to try different values of X until the value of T is satisfied.

A value of X = 10 gives

$$T = 54.25\ \mu s$$

Knowing this value it is possible to calculate the TPU configuration settings.

TPU1

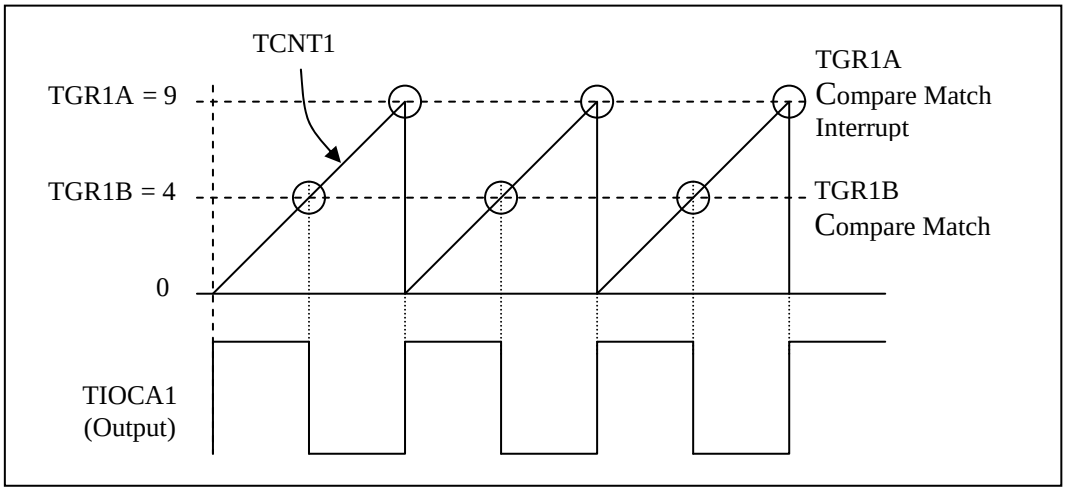

Figure 5.

TPU1 generates the CP Clock, which is used to clock the data into the LCM shift registers. To do this TPU1 is configured for PWM Mode 1 operation.

In this mode, the period is set by the TGR1A value and the duty cycle is set by the TGR1B value.

An interrupt request is made when TGR1A equals TCNT1. This interrupt request is used to trigger the DMAC, which in turn transfers data from the H8S memory to the LCM data lines.

At the same time as the Compare Match, the TPU is configured to clear the counter, TCNT1, down to 0. In addition, the TPU output pin TIOCA1 goes high.

When there is a compare match between TGR1B and TCNT, the TPU output pin goes low.

The TPU configuration code is shown in figure 6.

With the TPU configured to increment its counter (TCNT1) on the system clock, compare match values of 10 and 5 will generate a PWM waveform suitable for the CP clock.

Note that the TGR values are always 1 less than the required number of clock cycles, as the timer registers (TCNT) start counting from 0.

```
void ConfigTPU1(void)
{
    // Timer Control Register: Counter Clear
    //--------------------------------------
    TPU1.TCR.BIT.CCLR = 1;      // 0 - TCNT clearing disabled
                                // 1 - TCNT cleared by TGRA C/M, I/C
                                // 2 - TCNT cleared by TGRB C/M, I/C
                                // 3 - TCNT cleared by Synchronous Clearing

    // Timer Control Register: Clock Edge
    //-----------------------------------
    TPU1.TCR.BIT.CKEG = 0;      // 0 - Count on rising edge
                                // 1 - Count on falling edge
                                // 2 - Count on both edges
                                // NOTE - Setting ignored if Timer prescaler = clk / 1

    // Timer Control Register: Clock Prescaler
    //----------------------------------------
    TPU1.TCR.BIT.TPSC = 0;      // 0 - Timer prescaler = clk / 1
                                // 1 - Timer prescaler = clk / 4
                                // 2 - Timer prescaler = clk / 16
                                // 3 - Timer prescaler = clk / 64
                                // 4 - Timer prescaler = clk / TCLKA
                                // 5 - Timer prescaler = clk / TCLKB
                                // 6 - Timer prescaler = clk / 256
                                // 7 - Timer prescaler = TCNT2 overflow / underflow

    // Timer Mode Register: Mode Control
    //----------------------------------
    TPU1.TMDR.BIT.MD = 2;       // 0 - Normal Operation
                                // 2 - PWM Mode 1
                                // 3 - PWM Mode 2
                                // 4 - Phase Counting Mode 1
                                // 5 - Phase Counting Mode 2
                                // 6 - Phase Counting Mode 3
                                // 7 - Phase Counting Mode 4

    // Timer I/O Control Register: TGRA I/O Control
    //---------------------------------------------
    TPU1.TIOR.BIT.IOA = 6;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCA0 Rising Edge
                                // 9 - Capture TIOCA0 Falling Edge
                                // 10 - Capture TIOCA0 Both Edges

    // Timer I/O Control Register: TGRB I/O Control
    //---------------------------------------------
    TPU1.TIOR.BIT.IOB = 5;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCB0 Rising Edge
                                // 9 - Capture TIOCB0 Falling Edge
                                // 10 - Capture TIOCB0 Both Edges

    TPU1.TCNT = 0;

    TPU1.TGRA = 9;              // 0 -> 9 = 10!!
    TPU1.TGRB = 4;              // 0 -> 4 =  5!!

    TPU.TSYR.BIT.SYNC1 = 0;     // TPU 1 configured for NO Sync operation
```

```
    TPU1.TIER.BIT.TGIEA = 1;    // Enable TGR1A interrupt.  Used to trigger DMAC
    TPU1.TIER.BIT.TGIEB = 0;    // Disable other interrupts
    TPU1.TIER.BIT.TCIEV = 0;    // Disable other interrupts
    TPU1.TIER.BIT.TCIEU = 0;    // Disable other interrupts
}
```

Figure 6.

TPU2

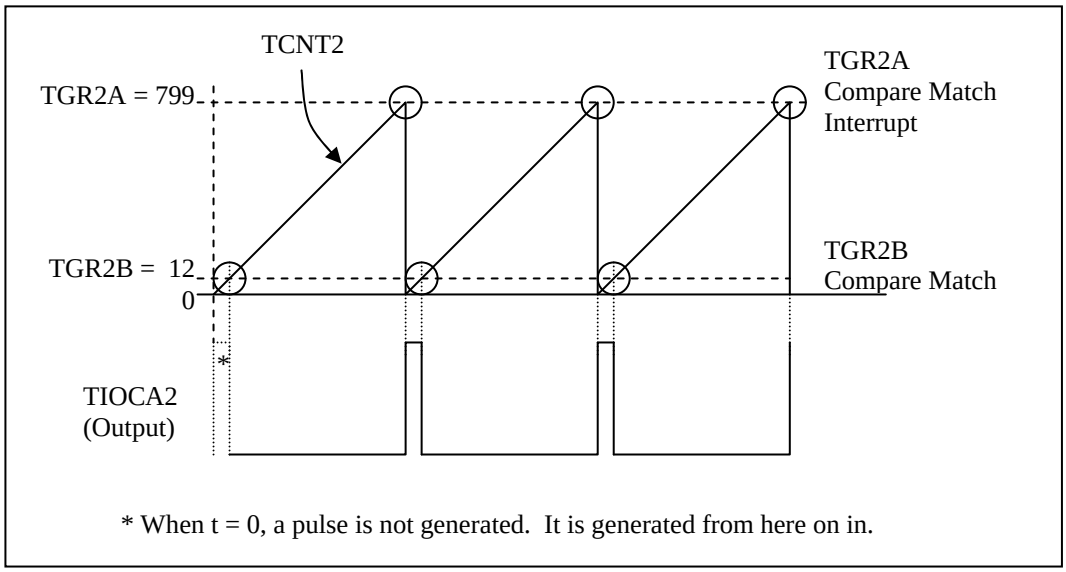

Figure 7.

TPU2 is responsible for generating the LOAD signal, which is responsible for ‘Loading’ the 80 packets of data clocked into the LCM by the CP signal, onto the LCD cell.

Internally to the LCM, the LOAD signal increments the Row driver, ready for the next line of data.

TPU2 is configured for PWM Mode 1 operation.

The TPU configuration code is shown in figure 8.

With the TPU configured to increment its counter (TCNT2) on the system clock, compare match values of 800 and 13 will generate a PWM waveform suitable for the CP clock.

Note that the TGR values for are always 1 less than the required number of clock cycles, as the timer registers (TCNT) start counting from 0.

```
void ConfigTPU2(void)
{
    // Timer Control Register: Counter Clear
    //---------------------------------------
    TPU2.TCR.BIT.CCLR = 1;      // 0 - TCNT clearing disabled
                                // 1 - TCNT cleared by TGRA C/M, I/C
                                // 2 - TCNT cleared by TGRB C/M, I/C
                                // 3 - TCNT cleared by Synchronous Clearing

    // Timer Control Register: Clock Edge
    //-----------------------------------
    TPU2.TCR.BIT.CKEG = 0;      // 0 - Count on rising edge
                                // 1 - Count on falling edge
                                // 2 - Count on both edges

    // Timer Control Register: Clock Prescaler
    //----------------------------------------
    TPU2.TCR.BIT.TPSC = 0;      // 0 - Timer pre-scaler = clk / 1
                                // 1 - Timer pre-scaler = clk / 4
                                // 2 - Timer pre-scaler = clk / 16
                                // 3 - Timer pre-scaler = clk / 64
                                // 4 - Timer pre-scaler = clk / TCLKA
                                // 5 - Timer pre-scaler = clk / TCLKB
                                // 6 - Timer pre-scaler = clk / TCLKC
                                // 7 - Timer pre-scaler = clk / 1024

    // Timer Mode Register: Mode Control
    //----------------------------------
    TPU2.TMDR.BIT.MD = 2;       // 0 - Normal Operation
                                // 2 - PWM Mode 1
                                // 3 - PWM Mode 2
                                // 4 - Phase Counting Mode 1
                                // 5 - Phase Counting Mode 2
                                // 6 - Phase Counting Mode 3
                                // 7 - Phase Counting Mode 4

    // Timer I/O Control Register: TGRA I/O Control
    //---------------------------------------------
    TPU2.TIOR.BIT.IOA = 2;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCA0 Rising Edge
                                // 9 - Capture TIOCA0 Falling Edge
                                // 10 - Capture TIOCA0 Both Edges

    // Timer I/O Control Register: TGRB I/O Control
    //---------------------------------------------
    TPU2.TIOR.BIT.IOB = 1;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCB0 Rising Edge
                                // 9 - Capture TIOCB0 Falling Edge
                                // 10 - Capture TIOCB0 Both Edges



    TPU2.TCNT = 0;

    TPU2.TGRA = 799;            // 0 -> 799 = 800!!
    TPU2TGRB = 12;
```

```
    TPU.TSYR.BIT.SYNC2 = 0;     // TPU 2 configured for Sync operation

    TPU2.TIER.BIT.TGIEA = 0;
    TPU2.TIER.BIT.TGIEB = 0;
    TPU2.TIER.BIT.TCIEV = 0;
    TPU2.TIER.BIT.TCIEU = 0;
}
```

Figure 8.

TPU4

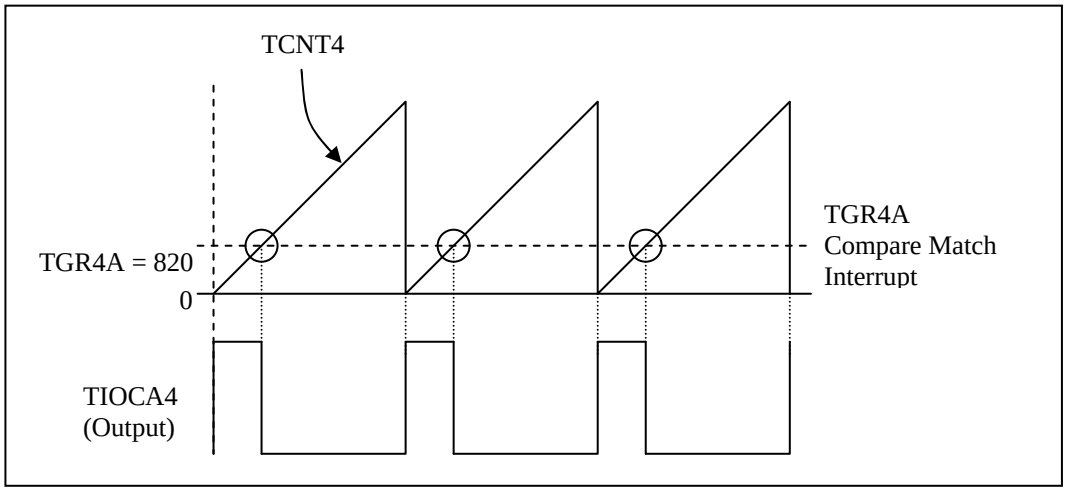

Figure 9.

TPU4 is responsible for generating the FRAME (First Line Marker) signal. This signal is responsible for 'Resetting' the Line Drivers back to the First Line so that a new image can be clocked in.

TPU4 is configured for PWM Mode 1 operation.

The TPU configuration code is shown in figure 10.

```
void ConfigTPU4(void)
{
    // Timer Control Register: Counter Clear
    //---------------------------------------
    TPU4.TCR.BIT.CCLR = 1;      // 0 - TCNT clearing disabled
                                // 1 - TCNT cleared by TGRA C/M, I/C
                                // 2 - TCNT cleared by TGRB C/M, I/C
                                // 3 - TCNT cleared by Synchronous Clearing

    // Timer Control Register: Clock Edge
    //-----------------------------------
    TPU4.TCR.BIT.CKEG = 0;      // 0 - Count on rising edge
                                // 1 - Count on falling edge
                                // 2 - Count on both edges

    // Timer Control Register: Clock Prescaler
    //----------------------------------------
    TPU4.TCR.BIT.TPSC = 0;      // 0 - Timer pre-scaler = clk / 1
                                // 1 - Timer pre-scaler = clk / 4
                                // 2 - Timer pre-scaler = clk / 16
                                // 3 - Timer pre-scaler = clk / 64
                                // 4 - Timer pre-scaler = clk / TCLKA
                                // 5 - Timer pre-scaler = clk / TCLKB
                                // 6 - Timer pre-scaler = clk / 1024
                                // 7 - Timer pre-scaler = TCNT overflow / underflow

    // Timer Mode Register: Mode Control
    //----------------------------------
    TPU4.TMDR.BIT.MD = 0;       // 0 - Normal Operation
                                // 2 - PWM Mode 1
                                // 3 - PWM Mode 2
                                // 4 - Phase Counting Mode 1
                                // 5 - Phase Counting Mode 2
                                // 6 - Phase Counting Mode 3
                                // 7 - Phase Counting Mode 4

    // Timer I/O Control Register: TGRA I/O Control
    //---------------------------------------------
    TPU4.TIOR.BIT.IOA = 5;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCA0 Rising Edge
                                // 9 - Capture TIOCA0 Falling Edge
                                // 10 - Capture TIOCA0 Both Edges

    // Timer I/O Control Register: TGRB I/O Control
    //---------------------------------------------
    TPU4.TIOR.BIT.IOB = 5;      // 0 - Output Disabled
                                // 1 - Initial Output is 0.  0 on Compare Match
                                // 2 - Initial Output is 0.  1 on Compare Match
                                // 3 - Initial Output is 0.  Toggle on Compare Match
                                // 5 - Initial Output is 1.  0 on Compare Match
                                // 6 - Initial Output is 1.  1 on Compare Match
                                // 7 - Initial Output is 1.  Toggle on Compare Match
                                // 8 - Capture TIOCB0 Rising Edge
                                // 9 - Capture TIOCB0 Falling Edge
                                // 10 - Capture TIOCB0 Both Edges


    TPU4.TCNT = 0;

    TPU4.TGRA = 820;
    TPU4.TGRB = 820;

    TPU.TSYR.BIT.SYNC4 = 0;
```

```
    TPU4.TIER.BIT.TGIEA = 0;
    TPU4.TIER.BIT.TGIEB = 0;
    TPU4.TIER.BIT.TCIEV = 0;
    TPU4.TIER.BIT.TCIEU = 0;
}
```

Figure 10.

RAM Usage

The Hitachi SP14Q006-ZTA has a resolution of 320 x 240 pixels. To facilitate data transfer, the LCD is fitted with a 4-bit interface. Therefore, (320 / 4) x 240 = 19200 nibbles (4 bit) of data are required to be transferred to display a LCD image.

Therefore, the LCD image could be stored in 9600 bytes of RAM.

However, to transfer this data the DMAC is used. Unfortunately, the DMAC can only transfer byte (8 bit) or word (16 bit) data.

Therefore, to transfer a byte of data effectively, this data has to be split across 2 bytes, so that the data can be transferred to the 4-bit interface.

The concept is shown in figure 11.

As a result, 19200 bytes of SRAM are required to store a complete LCD image.

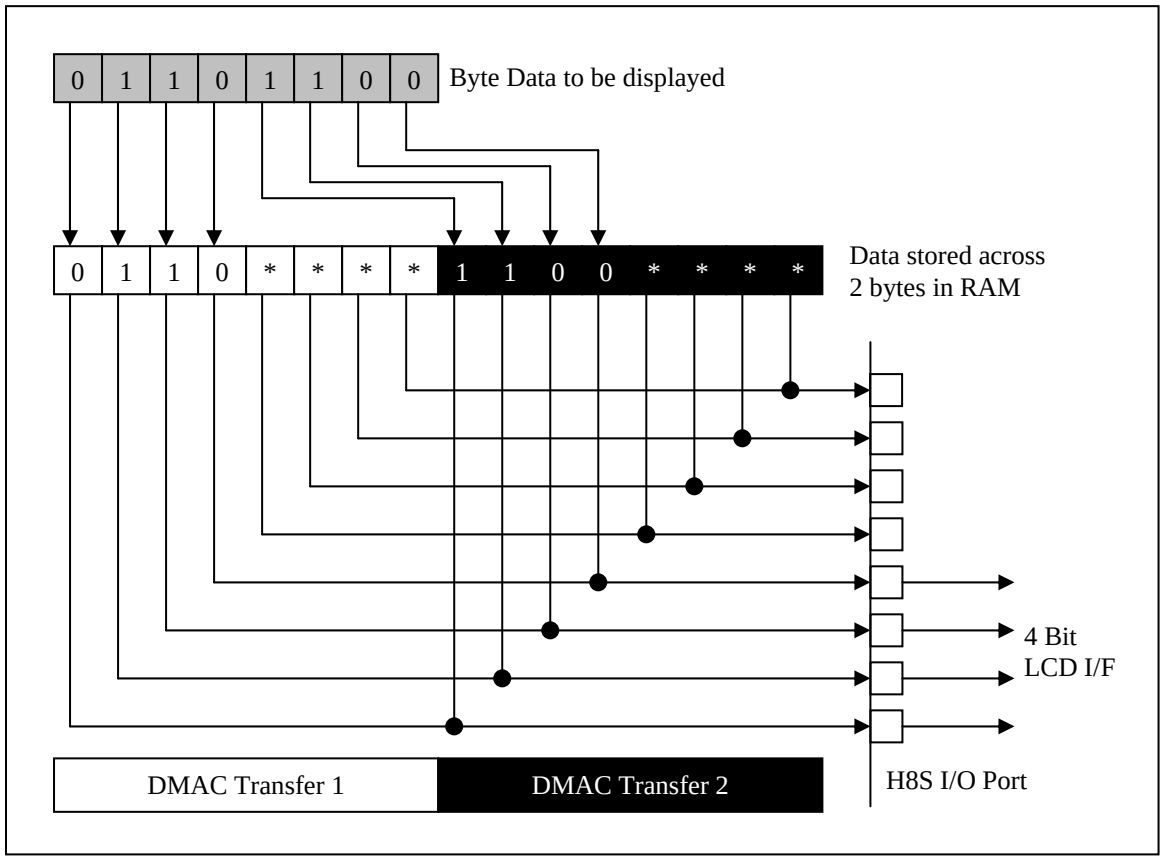

Figure 11.

DMAC Settings

The DMAC is configured so that the TPU TGRA1 Compare Match Interrupt triggers it. When the DMAC is triggered, it transfers a byte between 2 addresses, that of the RAM and the I/O Port.

The DMAC is configured to transmit a byte of data 19200 times before it generates a DEND interrupt. The DEND interrupt is responsible for stopping, reconfiguring and starting the TPU's and reconfiguring the DMAC, ready for another screen transfer.

The DMAC configuration code and DMAC ISR (Interrupt Service Routine) are shown in figures 12 and 13.

```
void InitDmac0A_SAM(void)
{
     DMAC0A.DMABCR.BIT.DTIE = 0; // Disable Interrupt Request
     DMAC0A.DMABCR.BIT.DTE = 0;  // Disable transfers while initialising

     DMAC0A.DMABCR.BIT.FAE = 0;  // 0 - Short Address Mode
                                 // 1 - Full Address Mode

     DMAC0A.DMABCR.BIT.SAE = 0;  // 0 - Transfer in Dual Address Mode
                                 // 1 - Transfer in Single Address Mode

                                 // DTE  DTA
     DMAC0A.DMABCR.BIT.DTA = 1;  // 0    x       Data transfer disabled.
                                 //              The interrupt will be handled by
                                 //              CPU or DTC regardless of DTA setting
                                 // 1    0       Data transfer enabled
                                 //              Interrupt issued to CPU or DTC
                                 // 1    1       Data transfer enabled
                                 //              DMAC clears interrupt flag.
                                 //              CPU or DTC does not receive interrupt

     // Specify source or destination address
     // Source or destination is selected by DTDIR bit in DMACR
     DMAC0A.MAR = (char*)&LcdRam[0][0];

     // Specify source or destination address
     // Source or destination is selected by DTDIR bit in DMACR
     DMAC0A.IOAR = 0xff0d;       // The bottom two bytes of address
                                 // (FF)FF0D : PE data register
                                          // for POD0, PDO1, PDO2, PDO3

     DMAC0A.ETCR = 0x4b00 - 1;   // (320/4) x 240 = 19200 (0x4b00)

    DMAC0A.DMACR.BIT.DTSZ = 0;   // 0 - Transfer bytes
                                 // 1 - Transfer word

    DMAC0A.DMACR.BIT.DTID = 0;   // 0 - MAR is incremented after transfer
                                 // 1 - MAR is decremented after transfer
```

Figure 12.

```
    DMAC0A.DMACR.BIT.RPE = 0;   // RPE  DTIE
                                // 0    0              Transfer in sequential mode
                                //                     (no transfer end interrupt)
                                // 0    1              Transfer in sequential mode
                                //                     (with transfer end interrupt)
                                // 1    0              Transfer in repeat mode
                                //                     (no transfer interrupt)
                                // 1    1              Transfer in idle mode
                                //                     (with transfer end interrupt)

    DMAC0A.DMACR.BIT.DTDIR = 0; // DTDIR  SAE
                                // 0      0            Transfer with MAR as source &
                                //                     IOAR as destination
                                // 0      1            Transfer with MAR as destination &
                                //                     IOAR as source
                                // 1      0            Transfer with MAR as source &
                                //                     /DACK as write strobe
                                // 1      1            Transfer with MAR as destination &
                                //                     /DACK as read strobe

    DMAC0A.DMACR.BIT.DTF = 9;   // Activation Source
                                // 0 - No activation
                                // 1 - ADC end interrupt
                                // 2 - No activation
                                // 3 - No activation
                                // 4 - SCI0 transmission complete interrupt
                                // 5 - SCI0 reception complete interrupt
                                // 6 - SCI1 transmission complete interrupt
                                // 7 - SCI1 reception complete interrupt
                                // 8 - TPU0 CM / IC A interrupt
                                // 9 - TPU1 CM / IC A interrupt
                                // 10 - TPU2 CM / IC A interrupt
                                // 11 - TPU3 CM / IC A interrupt
                                // 12 - TPU4 CM / IC A interrupt
                                // 13 - TPU5 CM / IC A interrupt
                                // 14 - No activation
                                // 15 - No activation

    DMAC0A.DMABCR.BIT.DTE = 1;  // Enable transfers

    DMAC0A.DMABCR.BIT.DTIE = 1; // 0 - Transfer End Interrupt disabled
                                // 1 - Transfer End Interrupt enabled
                                // Note - If this bit is set to 1 while
                                // DTE = 0, the DMAC will interpret this as
                                // an End Of Transfer Interrupt
}
```

Figure 12 Continued.

```
#pragma interrupt(_INT_DEND0A)
void _INT_DEND0A (void)
{
    // DMAC interrupt is generated after 1 screen of data has been transferred
    // Stop the timers & Reconfigure if necessary
    // Reconfigure the DMAC
    // Generate a FRAME signal
    DMAC0A.DMABCR.BIT.DTIE = 0; // Disable the Transfer End Interrupt
    DMAC0A.DMABCR.BIT.DTE = 0;  // Disable all Transfers

    TPU.TSTR.BYTE = 0x00;       // Stop all TPU's (except TPU5)

    // Reset the TPU counter registers so that they all start from the same point
    TPU1.TCNT = 0;

    TPU2.TIOR.BIT.IOA = 2;      // Set start conditions for LOAD signal
    TPU2.TIOR.BIT.IOB = 1;      // For first cycle, LOAD is 0
    TPU2.TCNT = 0;

    // To remove the problem of excessive clocks, generate an extra LOAD pulse
    // This will reset the shift registers
    TPU2.TIOR.BIT.IOA = 5;
    nop();
    TPU2.TIOR.BIT.IOA = 2;

    TPU4.TIOR.BIT.IOA = 5;      // FRAME signal goes High (goes Low on Compare Match)
    TPU4.TCNT = 0;

    DMAC0A.MAR = (char*)&LcdRam[0][0];   // Reset the Source Address
    DMAC0A.ETCR = 0x4b00 - 1;   // (320/4) x 240 = 19200 (0x4b00)
                                // Reset the Transfer Count Register
    DMAC0A.DMABCR.BIT.DTE = 1;  // Enable transfers
    DMAC0A.DMABCR.BIT.DTIE = 1; // Enable Transfer End Interrupt

    TPU.TSTR.BYTE = 0x16;       // Start all TPU's at once
}
```

Figure 13.

How it all works!

The required peripherals are enabled in the module stop register and are then initialised.

Prior to the TPU's being started, the SRAM which reflects the image to be displayed on the LCD is populated with the required data.

The three TPU's (TPU1, TPU2 & TPU3) are started at the same time by setting the corresponding bits in the Timer Start Register.

```
TPU.TSTR.BYTE = 0x16;           // Start all 3 TPU's at once: TPU --54 3210
                                //                            0001 0110
```

This ensures that all of the required clock edges are synchronised with one another.

The waveforms generated are shown in figure 14.

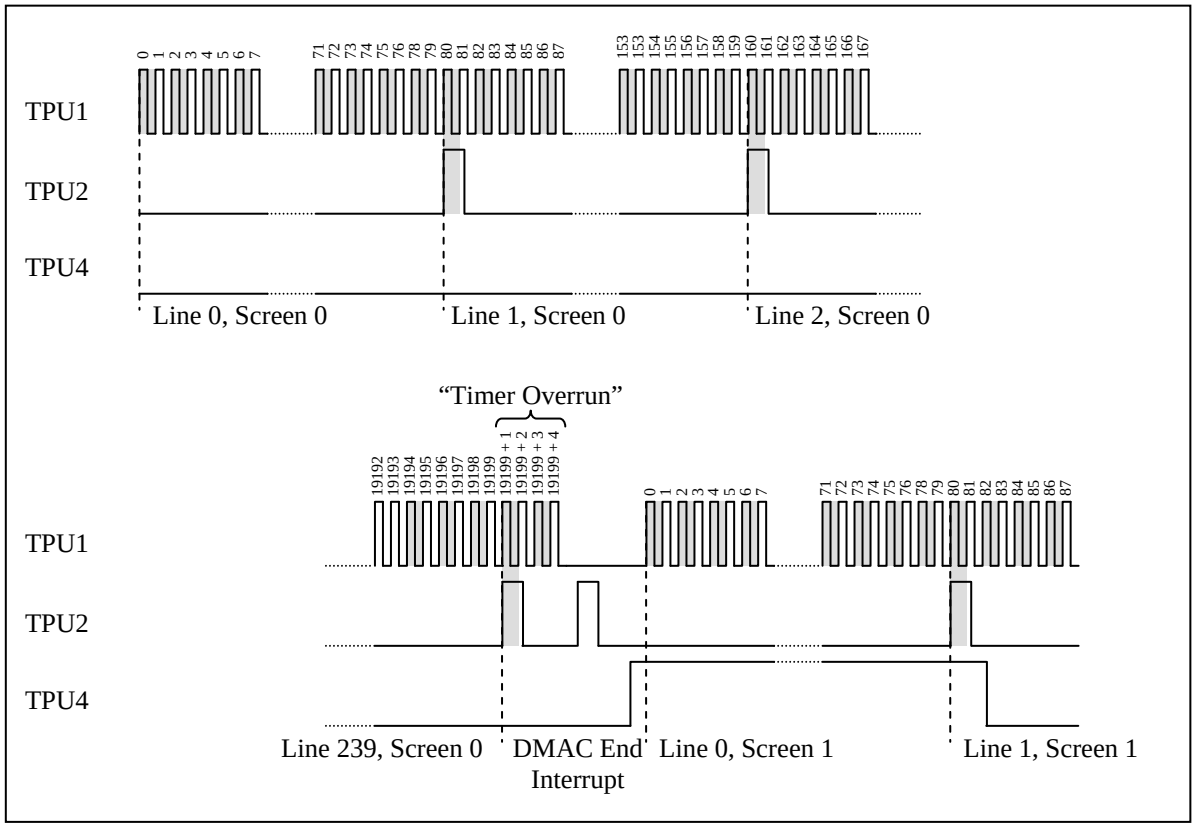

Figure 14.

The most important thing to note from figure 14 is what happens when 240 lines of data have been transferred to the LCD. At this point, the DMAC has transferred its allotted number of data bytes and is therefore no longer servicing the TPU 1 Compare Match interrupt. However, the TPU channels are still running and as a result TPU1 is still clocking data into the LCD.

As the DMAC is no longer taking the TPU1 Compare Match interrupt, the interrupt request is handled by the CPU and the TPU1A ISR, Interrupt Service Routine.

This stops the TPU channels. The reason for the "Timer Overrun", is due to the time it takes for the ISR to be serviced and the 'Stop TPU' instruction to be executed.

```
#pragma interrupt (_INT_TGI1A)
void _INT_TGI1A( void )
{
    TPU1.TSR.BIT.TGFA = 0;      // Clear flag
    TPU.TSTR.BYTE = 0x00;       // After 1 screen of data has been transferred to the LCD via
                                // DMAC, the DEND interrupt is called.  At this point, the TPU1A
                                // interrupt will now be sent to the CPU.
                                // The TPU1A interrupt is going every 10 clock cycles
                                // Consequently the CPU spends all its time processing
                                // the TPU interrupt and the system 'locks up'
                                // Therefore, stop all TPU's in the interrupt service routine
                                // Timers will be restarted in the DEND interrupt
}
```

Figure 15.

If the “Timer Overrun” did not occur and all timers stopped when the DMAC had finished transferring its data, the final LOAD signal generated would not be generated. Therefore, the “Timer Overrun” is required. The problem that arises from “Timer Overrun” is that four nibbles of data are clocked into the LCD and appear at on the LCD at addresses 0, 1, 2 & 4.

To remove this, an extra LOAD pulse is generated as part of the DMAC ISR. This resets the column drivers back to 0 so that when the next line of data is clocked in, it is clocked to the correct location. However, when the LOAD pulse is generated it increments the line counter within the LCD logic and it results in only being able to write to 239 of the 240 lines of the LCD. It is hoped that version 2 of this application note will address this very slight problem.

CPU Overhead and Operating Bandwidth

The direct drive of the ¼ VGA display is only possible on the H8S series of microcontroller due the DMAC peripheral. The DMAC, which is triggered via the TPU, is responsible for transferring a complete screen worth of data without CPU intervention.

When the DMAC is triggered by the TPU, 3 clock cycles are required to transfer the data form the SRAM to the I/O Data register. When transferring data, the DMAC will take the address and data bus from the CPU and perform its transfer.

With the H8S/2239 running at 14.7456 MHz, a DMAC transfer request is generated every 10 clock cycles. Therefore, 30% of the available bus bandwidth is taken by the DMAC. This could have a detrimental affect on the CPU by starving it of data. This would mean that it would not be able to perform any meaningful processing. However, many of CPU operations are internal to the CPU general registers and as such CPU performance is not affected.

This means that while the DMAC is transferring data to the LCD, the CPU can be populating the SRAM with new data, ready to be displayed. It follows that if a static image is being displayed, the only CPU interaction required is that of the DMAC ISR.

The CPU has to service the DMAC end interrupt when a screen of data has been transferred to the LCM. This occurs every 13 ms. The DMAC ISR is shown in figure 13. This code takes approximately 8.6 μs to execute.

It can be seen that the impact of the LCD drive software on the CPU is very minimal. For every screen of data displayed, the CPU spends 0.07% of that time executing associated code.

Application Note Software

All of the H8S software is available for download as an accompaniment to this Application Note. All of the software was written in the 'C' programming language using the development environment Renesas HEW (High Performance Workbench) using the Renesas H8S Toolchain.

The HEW workspace may be imported into HEW, Version 1.3, 2.x and 3.x.

If the reader of this application note does not wish to use HEW to view the source files, then the source files can be viewed in the editor of their choice.

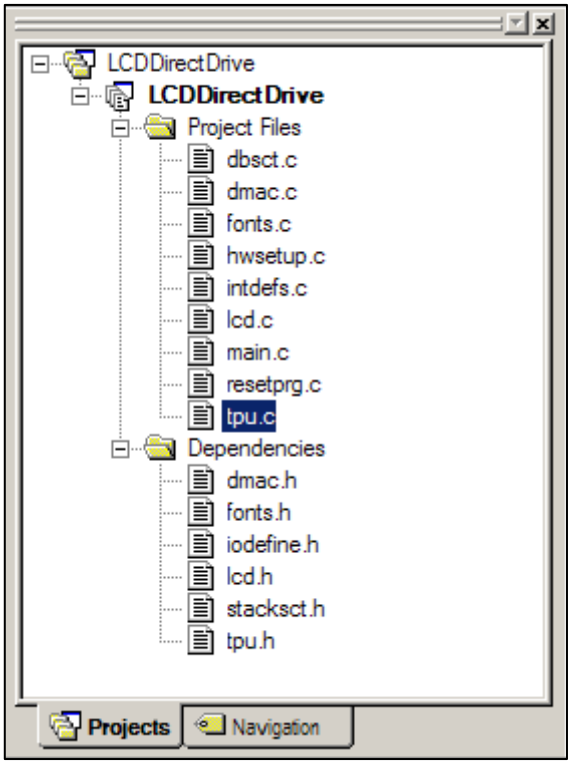

Figure 16.

Figure 16 shows the 'C' Source and Header files associated with this application note.

Table 2 details the main function of each file.

The source files have extensive comments so the reader is advised to read the source files for a greater understanding of how the software and H8S operate.

File	Description
dbsct.c	Section initialisation code. Initialises initialised and non-initialised variables. Called as part of power-on-reset code.
dmac.c	Code for initialising DMAC and DMAC ISR.
fonts.c	User defined fonts for displaying Alpha Numeric characters on the LCD.
hwsetup.c	Hardware initialisation of the H8S. Called as part of power-on-reset code.
intdefs.c	Instantiation of Interrupt Vector Table. Default ISR's for all interrupts.
lcd.c	Functions for writing text and drawing lines on LCD.
main.c	Called after power-on-reset and hardware setup. Enables and configures peripherals required for LCD direct drive.
resetprg.c	Called after a power-on-reset. Initialises Stack Pointer, calls function that performs low level initialisation of hardware, section initialisation and main()
tpu.c	Code for configuring required TPU channels and TPU ISR's
dmac.h	Associated Header files
fonts.h	
iodefine.f	
lcd.h	
stacksct.h	Stack size definition.
tpu.h	Associated Header files

Table 2.

Application Note Hardware

The H8S/2239 is a low power microcontroller which operates in the voltage range 2.7 to 3.3 V. The SP14Q006 requires 2 voltages, one to drive the LCM logic at 5V and the other is required to drive the LC. This voltage is in the range –23.1 V to –20.9 V.

Therefore, this application required 3 voltage supplies and the H8S required level shifters to interface to the LCM.

Using a LT1587 Voltage regulator and an NMA0512D DC-DC Converter, only one 5V supply was required to power the entire application

Figures 17, 18, & 19 show how was achieved.

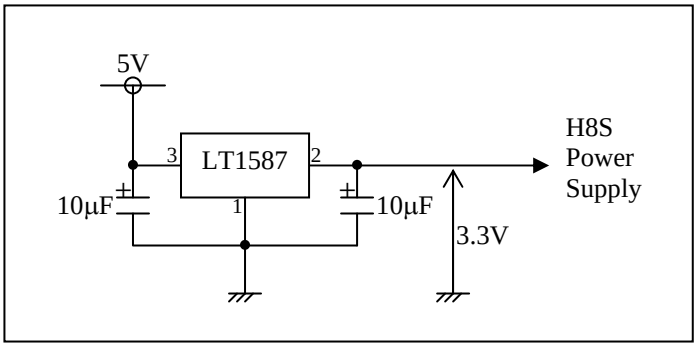

Figure 17.

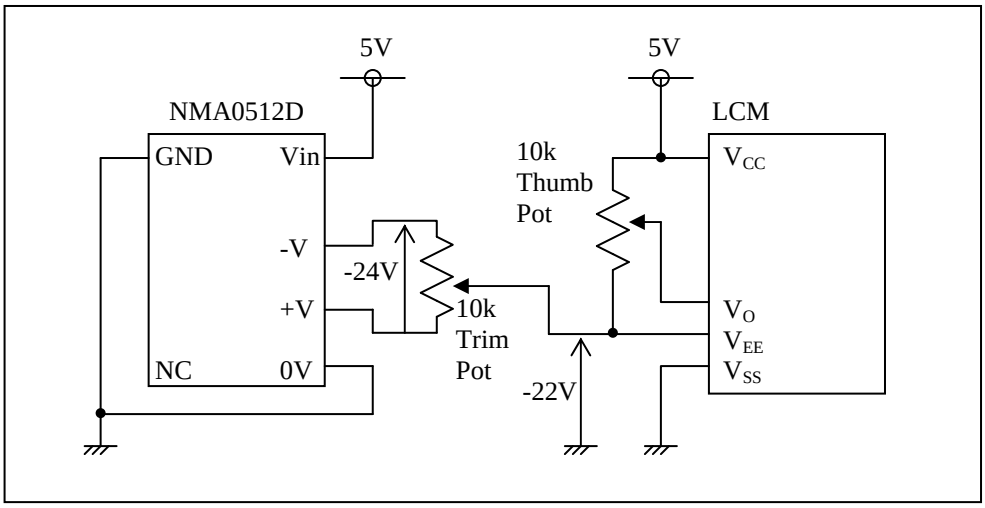

Figure 18.

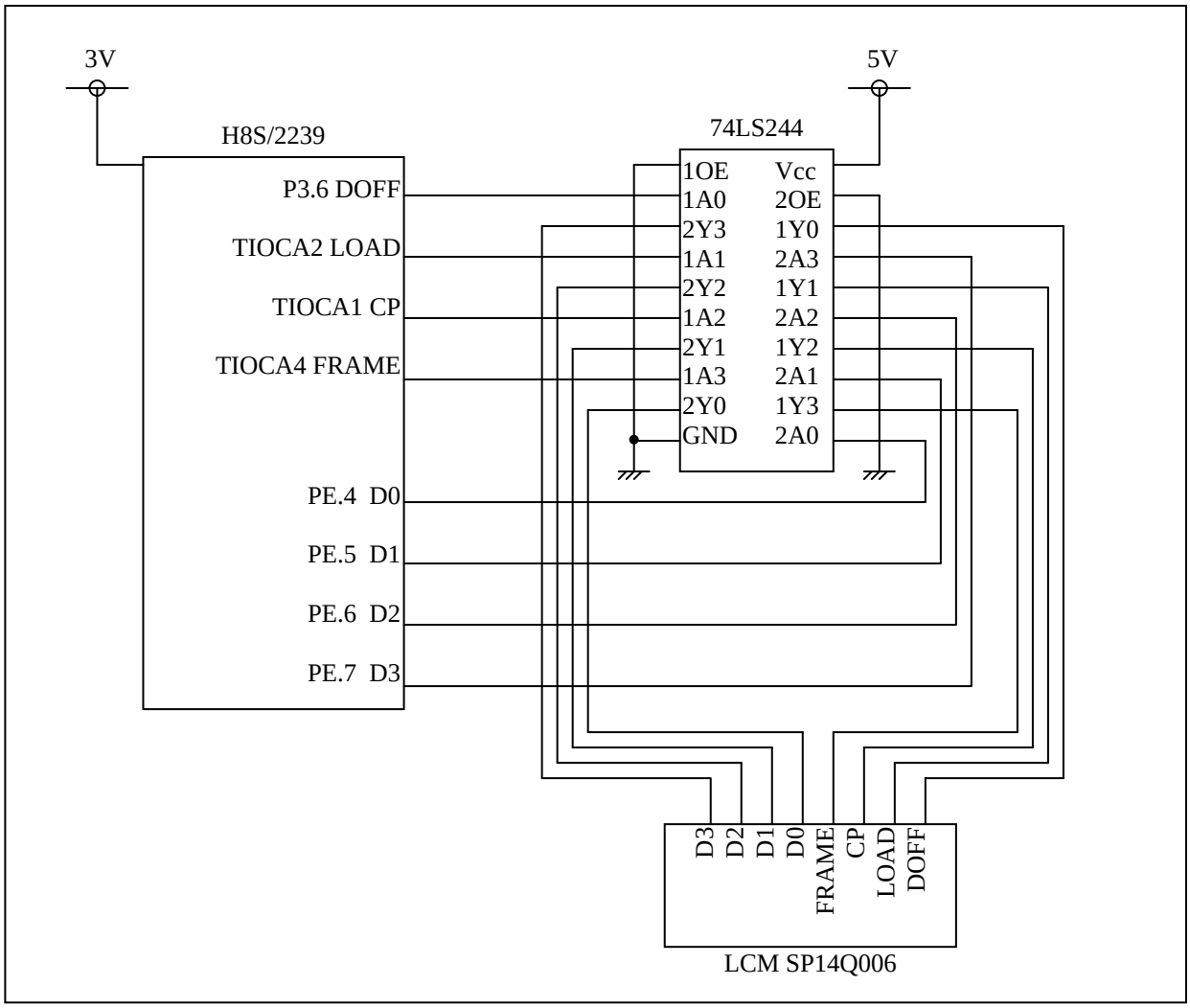
3V
5V
H8S/2239
74LS244
P3.6 DOFF
TIOCA2 LOAD
TIOCA1 CP
TIOCA4 FRAME
PE.4 D0
PE.5 D1
PE.6 D2
PE.7 D3
1OE
1A0
2Y3
1A1
2Y2
1A2
2Y1
1A3
2Y0
GND
Vcc
2OE
1Y0
2A3
1Y1
2A2
1Y2
2A1
1Y3
2A0
D3
D2
D1
D0
FRAME
CP
LOAD
DOFF
LCM SP14Q006

Figure 19.

APPENDIX 1. DMAC MODES

The H8S/2239 features a DMAC, Direct Memory Access Controller. This peripheral is found on many of the devices in the H8S family.

The DMAC is a peripheral for performing high-speed data transfers between memory addresses without using the CPU. This leaves the CPU free to do what it does best, computational operations on data; not moving data from location to location.

The DMAC can operate in two main modes, Short Address Mode and Full Address Mode. Of these two main modes, there are several sub modes.

Tables A1 and A2 detail the available DMAC modes of operation.

SHORT ADDRESS MODE

Transfer Mode	Transfer Source	Address Register Bit Length Source	Destination
Dual Address Mode • Sequential Mode 1 Byte or 1 Word transfer executed for one transfer request. Memory address incremented / decremented by 1 or 2. 1 to 65536 transfer. • Idle Mode 1 Byte or 1 Word transfer executed for one transfer request. Memory address fixed. 1 to 65536 transfer. • Repeat Mode 1 Byte or 1 Word transfer executed for one transfer request. Memory address incremented / decremented by 1 or 2. After specified number of transfers (1 to 256), initial state is restored and operation continues.	TPU Channel 0 to 5 Compare Match A Interrupt SCI Transmit Data Empty Interrupt SCI Receive Complete Interrupt ADC Conversion End Interrupt External Request	24 / 16	16 / 24
Single Address Mode 1 Byte or 1 Word transfer executed for one transfer request. Transfer in 1 bus cycle using /DACK pin in place of address specifying I/O. Specifiable for sequential, idle and repeat modes.			

Table A1. Short Address Mode

FULL ADDRESS MODE

Transfer Mode	Transfer Source	Address Register Bit Length Source	Destination
Normal Mode • Auto Request Mode Transfer request retained internally. Transfers continue for the specified number of times (1 to 65536). Choice of burst or cycle steal transfer. • External Request Mode 1 Byte or 1 Word transfer executed for one transfer request. 1 to 65536 transfer	Auto Request External Request	24	24
Block Transfer Mode Specified block size transfer executed for one transfer request. 1 to 65536 transfers. Either source or destination specifiable as block area. Block size: 1 to 156 bytes or words.	TPU Channel 0 to 5 Compare Match A Interrupt SCI Transmit Data Empty Interrupt SCI Receive Complete Interrupt ADC Conversion End Interrupt External Request	24	24

Table A2. Full Address Mode

Website and Support

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

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

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.