Freescale Semiconductor
Application Note

Document Number: AN4152
Rev. 1 , 04/2012

Implementing an IR Blaster on HCS08-Based IEEE® 802.15.4 Wireless Nodes

1 Introduction

The availability of low-cost 2.4 GHz ISM band IEEE 802.15.4 wireless nodes has opened RF remote control applications in home TV and entertainment systems markets. The RF remote control application is most commonly built upon the ZigBee® BeeStack Consumer (RF4CE) communication standard or Freescale's proprietary SynkroRF software platform. Freescale's BeeStack Consumer (RF4CE) or SynkroRF software stacks are most commonly used on 9S08 MCU based platforms:

- MC1321x — HCS08A MCU plus transceiver in a single 71-pin LGA
- MC13202 transceiver used with MC9S08QE128 MCU
- MC13233 — HCS08 MCU plus a transceiver in a single 48-pin LGA

As manufacturers make the transition from conventional infrared (IR) based remote controls to RF communication, it may be advantageous to continue to provide IR blaster capability along with the RF radio. This application note provides some recommendations and guidelines for implementing an IR remote control blaster function on the above referenced HCS08-based MCUs.

Contents

freescale™ semiconductor

2 IR Remote Control Overview

The IR remote control sends a modulated IR light signal to a receiver (in a TV or other appliance) that encodes a control data stream. The appliance then demodulates and decodes the data stream and acts accordingly. Freescale application note AN3053, *Infrared Remote Control Techniques on MC9S08RC/RD/RE/RG Family*, provides an excellent description of IR theory including modulation and encoding schemes.

This application note focuses on implementing the IR transmitter or “blaster” function of the remote control specifically on the 9S08A and 9S08QE128 MCUs and MC13233 used with the referenced stacks. Although multiple consumer industry IR standards exist, the two specific examples of the Philips RC5 (Manchester encoding) and the NEC (pulsed distance encoding) standards are detailed in this note. With these standards as the background, other encoding methods can be derived.

3 Blaster Hardware

The hardware components for implementing the blaster includes the hardware external to the MCU and the Timer/Pulse-Width Modulator (TPM) MCU onboard peripheral function.

3.1 External Hardware

The external hardware associated with the IR blaster is typically a buffer transistor and one or two IR light-emitting diodes (LEDs). To provide the desired IR transmit power, the LED current can be in the 80 mA - 100 mA range or more. At times two LEDs are driven in parallel for very high power and greater range. Figure 1 shows a typical circuit where an MCU GPIO (in the figure, signal RC5_OUT) drives an NPN bipolar transistor that provides the high current drive to the IR LED.

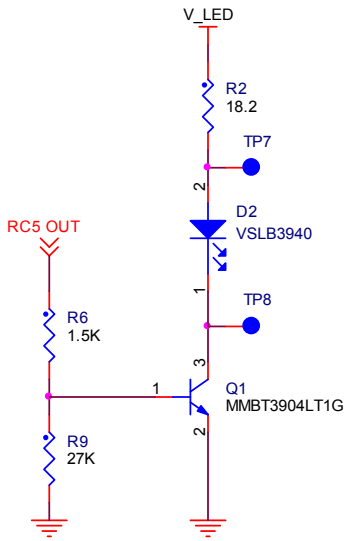

Figure 1. Typical IR LED drive circuit

Although bipolar transistors tend to be less expensive, logic-level gate MOSFETs are also used. The MOSFET has the advantage or requiring virtually no gate current from the GPIO pin, while the bipolar

transistor base drive current can be as high as 1/10th of the LED current. The MCU GPIO must be capable of supplying the required drive current.

3.2 Timer/Pulse-Width Modulator (TPM) Function

The timer/PWM TPM is a common function found in the HCS08 family of devices. One or more of the TPM functions are provided as part of the peripheral function set. The TPM module supports traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. Timing functions are based on a separate 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference. Figure 2 shows the typical TPM block diagram.

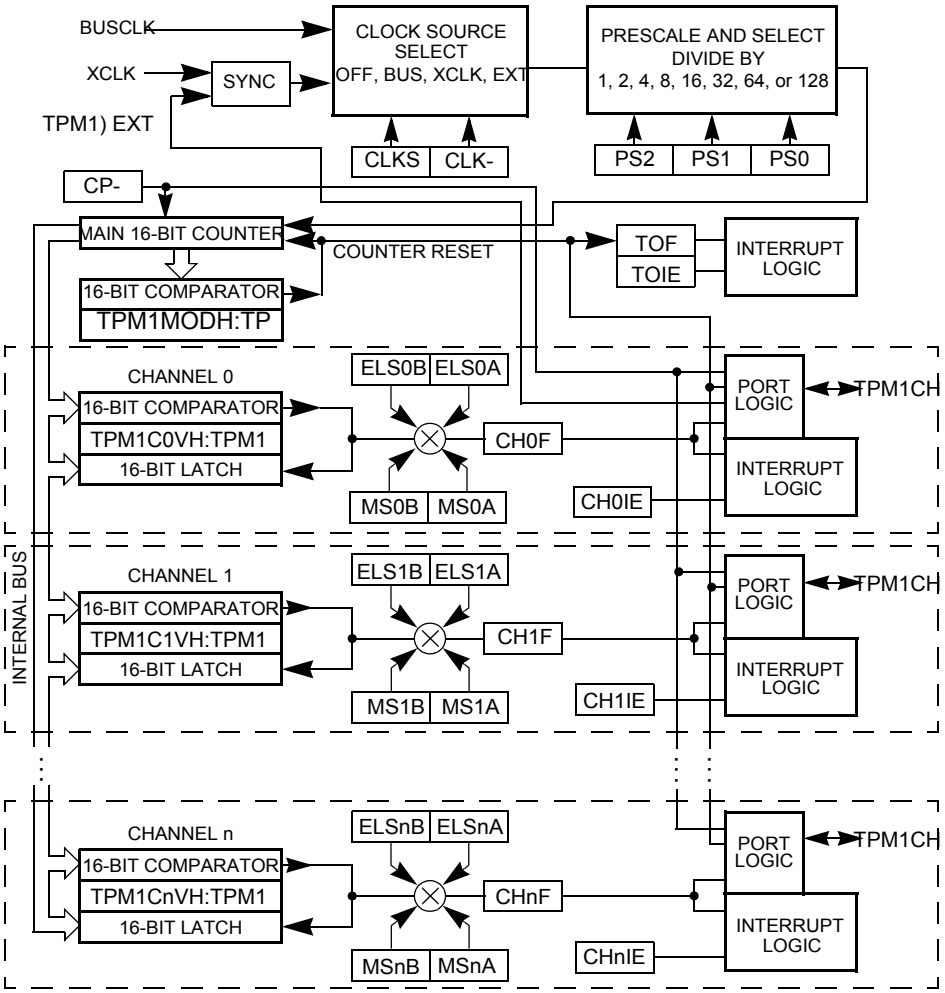

Figure 2. Typical TPM Block Diagram

The TPM is used to generate the 36–38 kHz pulse burst common to IR modulation schemes. The user is directed to the appropriate Freescale device reference manual for specific descriptions and details for the TPM as used with a specific MCU.

3.3 Carrier Modulator Timer (CMT) Function

The Carrier Modulator Timer (CMT) module is a IR LED driver. The CMT consists of a carrier generator, modulator, and transmitter that drives the infrared out (IRO) pin. The module can transmit data to the IRO output pin either in baseband or in FSK mode. The IRO pin has 20 mA current drive capability.

The features of the CMT module include:

- Four modes of operation
  - Time with independent control of high and low times
  - Baseband
  - Frequency shift key (FSK)
  - Direct software control of IRO pin
- Extended space operation in time, baseband, and FSK modes
- Selectable input clock divide: 1, 2, 4, or 8
- Interrupt on end of cycle
- Ability to disable IRO pin and use as timer interrupt
- CMT use as a timer resource

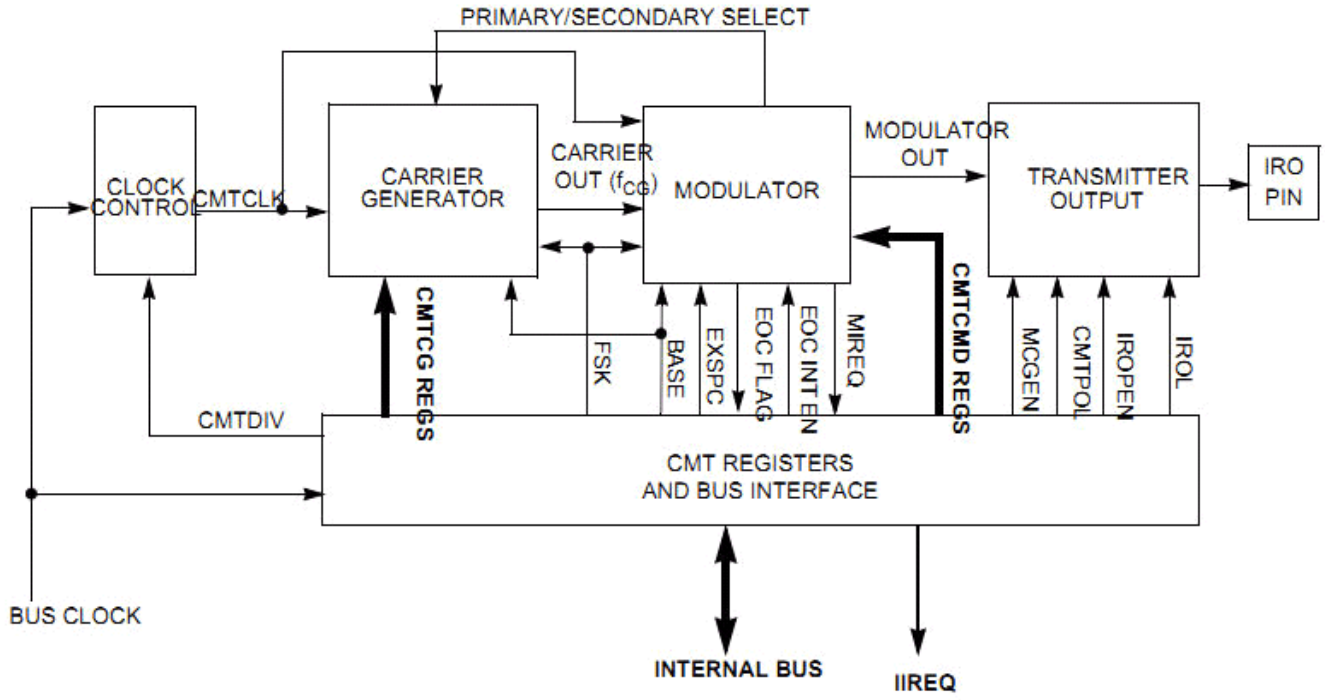

Figure 3. CMT Module Block Diagram

NOTE

For more information about the CMT module, refer to the MC1323x Software Reference Manual (MC1323xRM.pdf) Chapter 16 Carrier Modulator Timer (CMT) Module.

4 RC5 IR Protocol Implementation

The RC5 standard is based on a Manchester-coded modulation of frequency bursts.

4.1 RC5 Protocol Summary

The IR signaling for RC5 is based on bursts of pulses where the pulse frequency is nominally 36 kHz, which equates to a 27.8 µs period (receiver ICs are commonly used with band pass filter (BPF) center frequencies from 36.0 to 38.0 kHz). A single burst consists of 32 pulses (0.889 ms) and equates to ½ bit time, i.e., a bit time is 1.778 ms. The data are encoded as biphase or Manchester coding where a Data "0" is a frequency burst followed by ½ bit time of no signal and a Data "1" is ½ bit time of no signal followed by a frequency burst. See Figure 4.

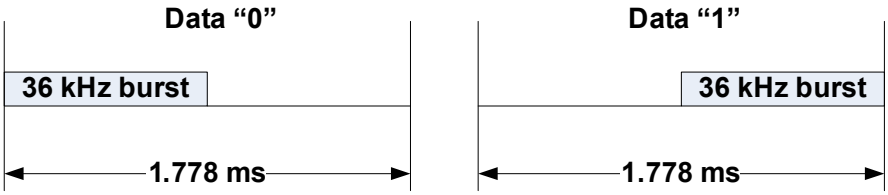

Figure 4. RC5 Manchester Data Encoding

The 36 kHz carrier frequency burst need not have a 50% duty cycle; variation from 25% to 50% is common and a lower duty cycle can save power at the sacrifice of range. Carrier frequency accuracy is suggested to be ± 1% or less. See Figure 5.

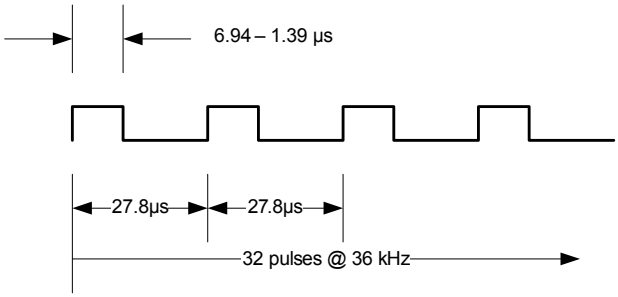

Figure 5. RC5 Carrier Burst Timing

With a bit time of 1.778 ms (~562 bps data rate), an RC5 transmission packet consists of 14 bits (24.89 ms transmission time). See Figure 6:

- Preamble – two start bits (each bit always a "1")
- Toggle bit – used to indicate the button is pushed or active (active is "0"); and is set to "1" for a final packet transmission when the button is released
- Address bits – five bit (A4–A0) receive device address

- Data bits – six bit (D5–D0) data field for the device command

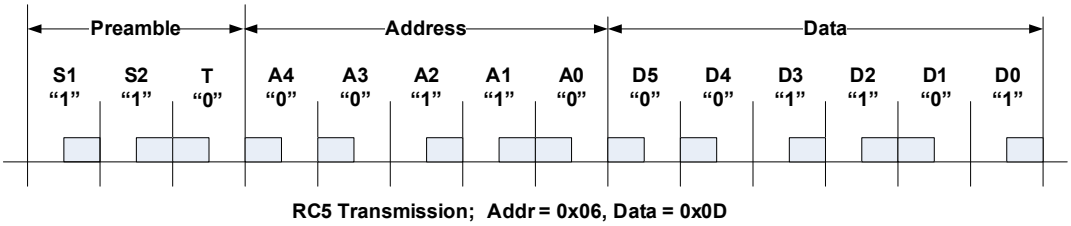

Figure 6. RC5 Initial or Repeat Transmission Packet

The RC5 transmission protocol causes the packet to be repeated every 114 ms while a the button is active. Once the button is released, the T bit is set to "1" for a last packet is sent.

4.2 Using the TPM Module for RC5 Transmission

The MC1321x 9S08A or MC9S08QE128 MCUs each have two or more available TPM modules. The Freescale (FSL) 802.15.4 based communication stacks reserve use of the first module, i.e.,TPM1; therefore TPM2 can be used for RC5 protocol generation.

For RC5 signal generation, the TPM mode called edge-aligned PWM mode is appropriate and requires the counter block and one channel. The edge-aligned PWM signal is used to generate the basic RC5 27.8 µs pulse (see Figure 5):

- The TPM2 clock source is selected as the bus rate clock (16 MHz for FSL 802.15.4 applications), and the Prescale Divisor is set to divide-by-1 – the resultant clock to the TPM2 counter is 16 MHz.
- The Timer Counter Modulo Registers (TPM2MODH:TPM2MODL) are programmed to a modulo value that sets the counter period to about 27.8 µs – as an example, TPM2MODH:TPM2MODL(15:0) = 0x01BC (444dec). This modulo divides the 16 MHz by 444 and the counter rolls-over to zero after reaching this count. This produces a nominal period of 27.8 µs and the overflow flag (TOF) can be used to count pulses.
- The Timer Channel Value Registers (TPM2CnVH:TPM2CnVL) set the pulse width "high" time (within the period) - with TPM2CnVH:TPM2CnVL (15:0) = 0x00DE (222 dec) for a 50% duty cycle. This value can be reduced to shorten the duty cycle, perhaps as low as 25%. The output compare flag (CHnF) can be used to detect the end of the pulse.
- The counter period and pulse width can both be trimmed to give desired results for active high pulse width and the total pulse width.

Figure 7 shows the counter mode to the pulse stream.

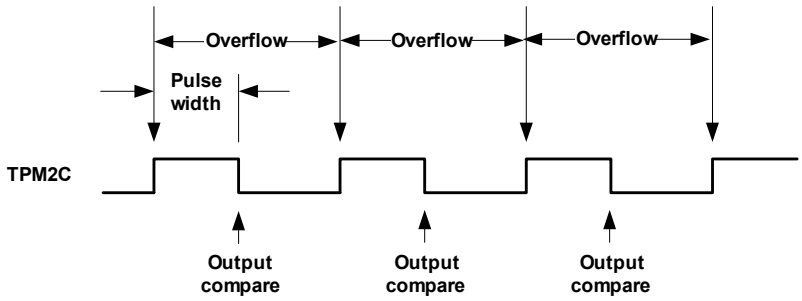

Figure 7. Edge-aligned PWM for RC5 Pulse Generation

TPM2 is thus setup to provide the basic RC5 pulse that, when repeated, produces the basic 36 kHz burst (32 cycles/burst). The general procedure starts the counter and counts cycles (using the overflow event) in groups of 32 (representing ½ bit time). The timer channel count is used to enable an IO output drive for each group of 32 cycles AS REQUIRED BY THE FRAME DATA MANCHESTER CODING; i.e., for each ½ bit time that a burst is desired (see See Figure 6), the output will be enabled and for each ½ bit time a burst is not desired, the output is disabled.

A basic encoding algorithm is then:

1. Initiate TPM2 –
   - Initiate edge-aligned PWM as stated.
   - The overflow flag is enabled to generate an interrupt request which is used to count pulses.
   - **Conditionally** - The TPM2 channel is setup to drive its associated GPIO pin as an output such that the output is active high during the pulse-width high time (driving the external circuit). This function is enabled/disabled on ½ bit time boundaries.
2. The counter is started and continues to run for an entire packet.
3. The main software function constructs the RC5 frame bit pattern and starts the counter. A soft counter loop is used to count 32 overflow events (this is driven by the overflow interrupt) to establish each ½ bit time delay.

NOTE

The Freescale MAC-based stacks on the HCS08 require that interrupt service routines be short. Within a period of 64 µs, the application may disable the MC1319x/MC1320x/MC1321x interrupts for a maximum duration of 10 µs, when running at 16 MHz bus clock. As a result, Freescale recommends that the overflow interrupt handler increment only the software loop count and no other function within the handler.

4. The main software function enables the counter IO pin for each ½ bit time as determined by the Manchester coding for each bit within the constructed frame.
5. After the complete frame is sent, another timer or TPM2 can be re-configured and used to set the delay between needed, repeat transmissions.

4.3 Using the MC1323x CMT Module for RC5 Transmission

The MC1323x devices have a Carrier Modulator Timer (CMT) module that can be used to drive an IR LED. For RC5 signal generation, the CMT Time mode is appropriate and the IRO pin could be optionally enabled to drive the IR LED transmissions.

The CMT Time mode is used to generate the logic 0 and 1 values:

- The CMT clock source is selected as the bus rate clock (16 MHz for MC1323x applications) and the prescaler is set to provide the desired resolution by writing to CMTDIV[2:0] field, which is located across two registers.
- Set the carrier frequency to 36 kHz by setting the primary high and low values of the Carrier Generator Data Registers (CMTCGH1, CMTCGL1). With a prescaler value of 2, a high time value of 0x49 (73 dec) and a low time value of 0x95 (149 dec) will generate a 36 kHz frequency with a 30% duty cycle.

$$FCG = 8\ MHz / (73\ dec + 149\ dec) = 36\ kHz$$

$$\text{Duty Cycle} = \frac{73}{73 + 149} = 0.33$$

NOTE

With 36 kHz carrier frequency and a 30% duty cycle, a prescaler value of 1 will not work because it would require the Primary Carrier Low Time Data value to be greater than 0xFF (maximum value allowed).

- Select the timer mode operation by setting the CMTMSC_BASE and CMTMSC_FSK bits to zero (default value after reset).
- Set the polarity type of the IRO pin to zero (IRO pin is active low) in CMTOC_ CMTPOL register.
- Optionally, enable the IRO output pin by setting the CMTOC_IROPEN bit.

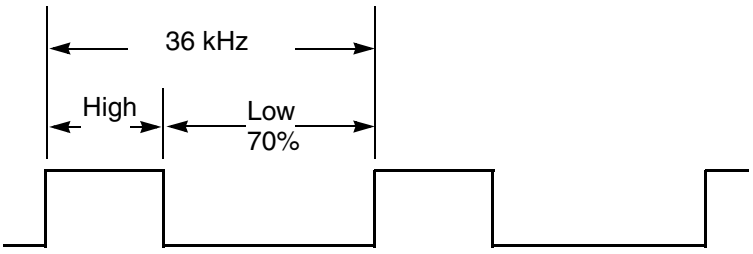

Figure 8. CMT carrier waveform for RC5 pulse generation

A basic encoding algorithm using the drivers is then:

1. Initiate the CMT:
   a) Set a 36 kHz carrier frequency with a 30% duty cycle as stated.
   b) Select time mode operation.
   c) The IRO pin is enabled to drive the out the CMT transmitter output with the proper polarity.
2. The mark and space times for both logic 0 and 1 values are initialized as follows:

Table 1. Mark and space times for logic 0 and 1 values

Bit value		Time in μs	Time in clock ticks Prescaler = 2
Logic 0	Mark	889	888
	Space	889	889
Logic 1	Mark	889	888
	Space	0	0

Then write the CMTCMD1:CMTCMD2 registers with a value of 888 (dec) for both logic 0 and 1 values and CMTCMD3:CMTCMD4 registers with a value of 889 for logic 0 and a value of 0 for logic 1 (change the value when required).

Given the time mode operation of the CMT module, the mark duration cannot be 0 and thus the logic 1 could not be generated straight away. This is why the total time of the logic 1 seems to be 889 μs. Therefore, the extended space operation is needed to generate the first half bit of the logic 1 value in RC5.

NOTE

Please refer to the MC1323x Reference Manual — Chapter 16.5.5 Extended Space Operation, for further details about extended space functionality.

Setting the EXSPC bit in the CMTMSC register will force the modulator to treat the next modulation period (beginning with the next load of the counter and space period register) as a space period equal in length to the mark and space counts combined. This will result in a space value of 889 μs (mark + space duration).

After the modulation period is over, the extended space bit is cleared and the normal operation with the logic 1 values (mark = 889 μs and space = 0 μs) could be executed to generate the proper waveform.

3. Enable the CMT module and the CMT interrupt by setting the MCGEN and EOCIE bits in the CMT Modulator Status and Control Register (CMTMSC) respectively. The interrupt will be set after a modulator cycle is completed. The application should use this interrupt to set the proper mark and space times when the next bit is to be sent.

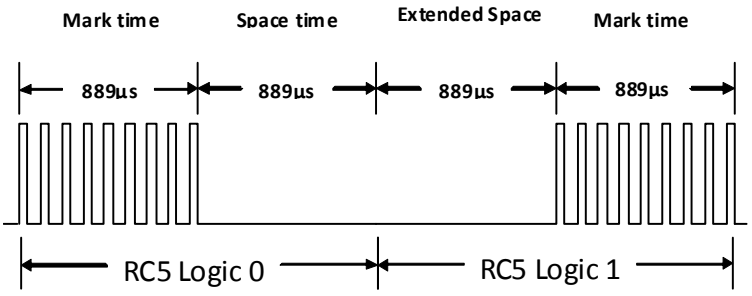

Figure 9. CMT mark and space times for RC5 logic values

NOTE

To ease the user's development process, Freescale provides a CMT driver that can be used/modified to meet the applications requirements. Please refer to Appendix A for a code example using this driver to implement the RC5 protocol.

5 NEC IR Protocol Implementation

The NEC standard is based on pulse distance data encoding.

5.1 NEC Protocol Summary

The IR signaling for the NEC format is based on bursts of 21 pulses where the pulse frequency is nominally 37.5 kHz, which equates to a 560 µs burst period (receiver ICs are commonly used with band pass filter (BPF) center frequencies from 36.0 to 38.0 kHz). The data are encoded as "pulse distance" coding (see Figure 10) where:

- Data "0" is a standard 0.56 ms frequency burst followed by an equal time of no signal (total of 1.12 ms period)
- Data "1" is a standard 0.56 ms frequency burst followed by a 3x burst time of no signal (total of 2.24 ms period)

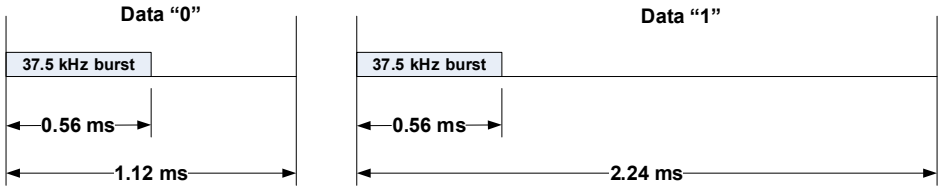

Figure 10. NEC Pulse Distance Data Encoding

The 37 kHz carrier frequency burst typically does not have a 50% duty cycle; variation from 25% to 33% is common and lower duty cycle can save power at the sacrifice of range. Carrier frequency accuracy is suggested to be 37.5–38.0 kHz. See Figure 11.

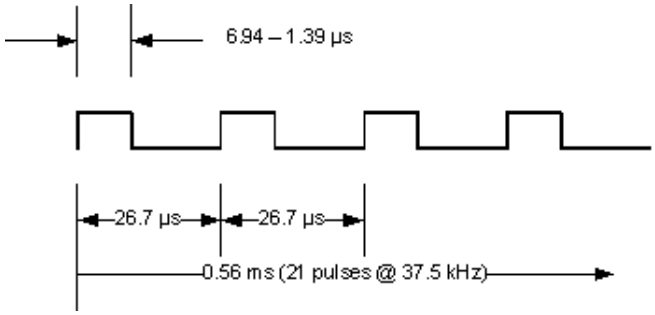

Figure 11. NEC Carrier Burst Timing

A key press starts a series of transmission events:

1. The NEC transmission starts with an initial Command packet (Figure 13 shows the entire packet structure), which consists of:

— Lead Code – provides the equivalent of a preamble that consists of a 9 ms frequency burst of the base 37.5 kHz carrier followed by a space of 4.5 ms. The frequency burst is used as a start condition and can be used for receiver AGC adjust if required. See Figure 12.

NOTE

9 ms is approximately 336 pulses @ 37.5 MHz, which equates to 16 times the length of a standard data pulse. This is useful information when implementing the encoding algorithm.

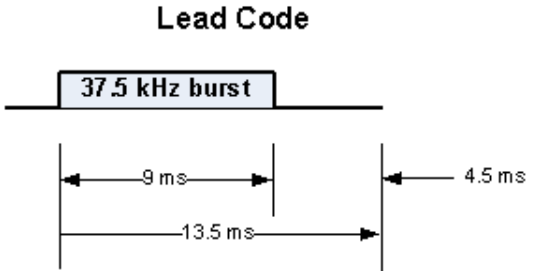

Figure 12. NEC Transmission Lead Code

— Address Code – 16-bit address field consisting of first an 8-bit Low Address (sent LSB first) and second an 8-bit High Address (sent LSB first)

— Data Code – 16-bit data field consisting of first an 8-bit Low Data (sent LSB first) and second an 8-bit High Data (sent LSB first)

NOTE

Variants of the address and data formats include:

- Upper 8-bit address is the inversion of the lower 8-bit address
- Upper 8-bit data is the inversion of the lower 8-bit data
- Address is a unique full 16-bit code
- Data is a unique full 16-bit code

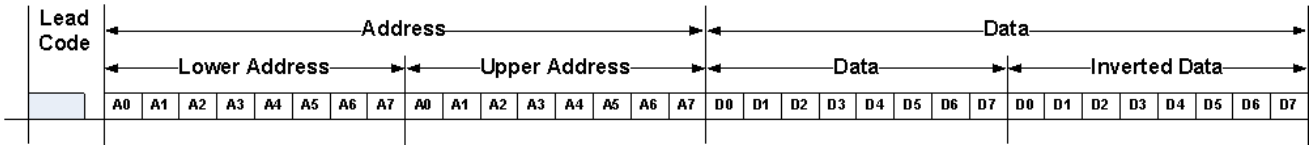

Figure 13. NEC Transmission Frame

2. As long as the key remains pressed, a Repeat transmission is resent every ~108 ms -
   — The initial command packet is NOT resent
   — The first Repeat transmission is sent 108 ms after the beginning of the Command packet
   — The Repeat transmission format consists of;
      – a 9 ms frequency burst (similar to the Lead Code)
      – followed by a space of 2.25 ms
      – followed by a standard 0.56 ms data frequency burst

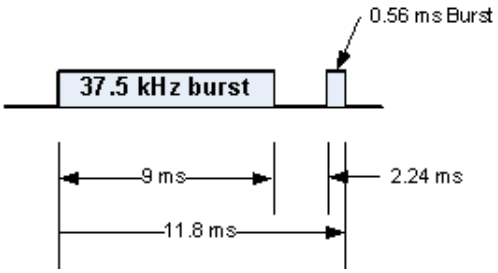

Figure 14. NEC Repeat Transmission Format

— Repeat transmissions continue at 108 ms intervals as long as the key remains pressed. See Figure 15.

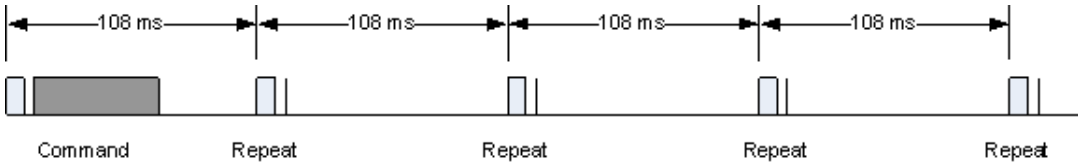

Figure 15. NEC Repeat Transmission Sequence

NOTE

108 ms is 4032 pulses @ 37.5 MHz, which equates to 192 times the length of a standard data pulse. This is useful information when implementing the encoding algorithm.

5.2 Using the TPM Module for NEC Transmission.

As stated in Section 4.2, "Using the TPM Module for RC5 Transmission, the Freescale 802.15.4 based communication stacks reserve use of the first module, i.e.,TPM1; therefore TPM2 can be used for NEC protocol generation.

For NEC signal generation, the TPM mode called edge-aligned PWM mode is appropriate and requires the counter block and one channel. The edge-aligned PWM signal is used to generate the basic NEC 26.7 µs pulse (see Figure 11):

- The TPM2 clock source is selected as the bus rate clock (16 MHz for FSL 802.15.4 applications), and the Prescale Divisor is set to divide-by-1 – the resultant clock to the TPM2 counter is 16 MHz.
- The Timer Counter Modulo Registers (TPM2MODH:TPM2MODL) are programmed to a modulo value that sets the counter period to about 26.7 µs – as an example, TPM2MODH:TPM2MODL(15:0) = 0x01AA (426dec). This modulo divides the 16 MHz by 426 and the counter rolls-over to zero after reaching this count. This produces a nominal period of 26.7 µs and the overflow flag (TOF) can be used to count pulses.
- The Timer Channel Value Registers (TPM2CnVH:TPM2CnVL) set the pulse width "high" time (within the period) – with TPM2CnVH:TPM2CnVL (15:0) = 0x008E (142 dec) for a 33% duty cycle. This value can be varied as required for power consumption. The output compare flag (CHnF) can be used to detect the end of the pulse.
- The counter period and pulse width can both be trimmed if needed to give desired results for active high pulse width and the total pulse width.

Figure 16 relates the counter mode to the pulse stream.

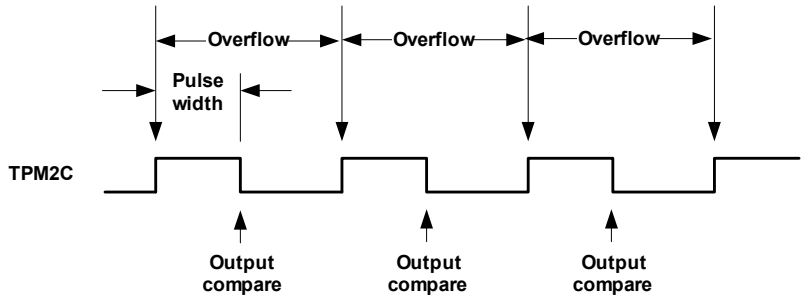

Figure 16. Edge-aligned PWM for NEC Pulse Generation

TPM2 is thus setup to provide the basic NEC pulse that, when repeated, produces the basic 37.5 kHz burst (21 cycles/burst). The general procedure starts the counter and counts cycles (using the overflow event) in groups of 21 (representing the standard burst time).

- The count of 21 cycles/burst is used a time base or "time tick" to establish all other protocol timing (reference the previous notes)
  - A Data "0" period equates to 2 burst periods
  - A Data "1" period equates to 4 burst periods
  - The Lead Code burst period is 16 burst periods
  - The Repeat transmission period is 192 burst periods
- The timer channel count is used to enable an IO output drive for group(s) of 21 cycles as required by the different burst widths used in the protocol.

A basic algorithm is then:

1. Initiate TPM2 –
   - Initiate edge-aligned PWM as stated.
   - The overflow flag is enabled to generate an interrupt request which is used to count pulses.
   - **Conditionally** – The TPM2 channel is setup to drive its associated GPIO pin as an output such that the output is active high during the pulse-width high time (driving the external circuit). This function is enabled/disabled based on the different required burst widths.
2. The counter is started and continues to run for an entire transmission sequence.
3. The main software function constructs the NEC frame bit pattern and starts the counter. A soft counter loop is used to count 21 overflow events (this is driven by the overflow interrupt) to establish each burst time boundary.

NOTE

The Freescale MAC-based stacks on the HCS08 require that interrupt service routines be short. Within a period of 64µs, the application may disable the MC1319x/MC1320x/MC1321x interrupts for a maximum duration of 10µs, when running at 16 MHz bus clock. As a result, Freescale recommends that the overflow interrupt handler increment only the software loop count and no other function within the handler.

4. The main software function enables the counter IO pin as required for the initial Command format -
   - Generate the Lead Code - enable the IO for the initial 9 ms burst and then disable for the 4.5 ms space
   - Transmit data pattern as required -
     - For a Data “0” enable the IO for a single burst and disable for a burst period
     - For a Data “1” enable the IO for a single burst and disable for 3 burst periods
5. The main software function must also maintain a running count of the cumulative burst periods to generate the 108 ms periods between required Repeat transmissions.
   - After 192 burst periods send the Repeat transmission pattern. Again, use burst periods to establish timing
   - Continue Repeat transmissions as long as key remains pressed.

5.3 Using the MC1323x CMT module for NEC Transmission

As stated in Section 4.3, “Using the MC1323x CMT Module for RC5 Transmission,” the MC1323x devices have a Carrier Modulator Timer (CMT) module that can be used to drive an IR LED. For NEC signal generation, the CMT Time mode is appropriate and the IRO pin could be optionally enabled to drive the IR LED transmissions.

The CMT Time mode is used to generate the logic 0 and 1 values:

- The CMT clock source is selected as the bus rate clock (16 MHz for MC1323x applications) and the prescaler is set to provide the desired resolution by writing to CMTDIV[2:0] field, which is located across two registers.
- Set the carrier frequency to 38 kHz by setting the primary high and low values of the Carrier Generator Data Registers (CMTCGH1, CMTCGL1). With a prescaler value of 2, a high time value of 0x45 (69 dec) and a low time value of 0x8D (141 dec) will generate a 38 kHz frequency with a 30% duty cycle.

$$FCG = 8\ MHz / (69\ dec + 141\ dec) = 38\ kHz$$

$$\text{Duty Cycle} = \frac{69}{69 + 141} = 0.33$$

NOTE

With 38 kHz carrier frequency and a 30% duty cycle, a prescaler value of 1 will not work because it would require the Primary Carrier Low Time Data value to be greater than 0xFF (which is the maximum value allowed).

- Select the timer mode operation by setting the CMTMSC_BASE and CMTMSC_FSK bits to zero (default value after reset).
- Set the polarity type of the IRO pin to zero (IRO pin is active low) in CMTOC_ CMTPOL register.
- Optionally, enable the IRO output pin by setting the CMTOC_IROPEN bit.

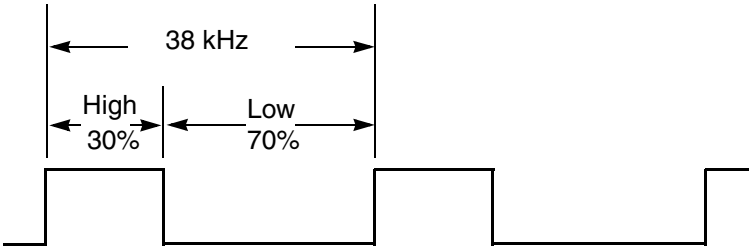

Figure 17. CMT carrier waveform for NEC pulse generation

A basic encoding algorithm using the drivers is then:

1. Initiate the CMT-
   a) Set a 38 kHz carrier frequency with a 30% duty cycle as stated.
   b) Select time mode operation.
   c) The IRO pin is enabled to drive the out the CMT transmitter output with the proper polarity.
2. The mark and space times for both logic 0 and 1 values are initialized as follows:

Table 2. Mark and Space times for logic 0 and 1 values

Bit value		Time in μs	Time in clock ticks Prescaler = 2
Logic 0	Mark	560	559
	Space	560	560
Logic 1	Mark	560	559
	Space	1680	1680

   Then write the CMTCMD1:CMTCMD2 registers with a value of 559 (dec) for both logic 0 and 1 values and CMTCMD3:CMTCMD4 registers with a value of 560 for logic 0 and a value of 1680 for logic 1 (change the value when required). If the NEC lead code is to be sent first, the mark and space times vary from the ones stated in Table 2. In this case, the mark and space times should be 9 ms and 4.5 ms, respectively.
3. Enable the CMT module and the CMT interrupt by setting the MCGEN and EOCIE bits in the CMT Modulator Status and Control Register (CMTMSC) respectively. The interrupt will be set after a modulator cycle is completed. The application should use this interrupt to set the proper mark and space times when the next bit is to be sent.

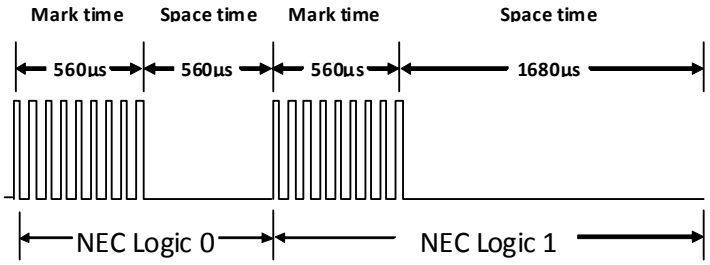

Figure 18. CMT mark and space times for NEC logic values

NOTE

To ease the user's development process, Freescale provides a CMT driver that can be used/modified to meet the applications requirements. Please refer to Appendix A for a code example using this driver to implement the NEC protocol.

6 Summary

This application note provides specific recommendations for implementing a remote control IR blaster function on HCS08-based MCUs using the TPM timer module and the MC13233 CMT module. The RC5 and NEC IR codes have been examined as typical examples. Key factors to consider:

- The Freescale IEEE 802.15.4 MAC and MAC-based stacks use a standard 16 MHz bus clock on the MCU platforms. The should be the basis of all timing considerations.
- The first or primary TPM function is reserved for use by the stack application.
- Typically, the TPM is used in edge-aligned PWM mode to generate the required modulation signal bursts.
- Be aware of limiting interrupt handler runtime to approximately 10 µs or less when the MAC is active.
- Use the provided examples as guidelines for other protocols.

Appendix A

To ease the user's development process, Freescale provides a CMT driver that can be used/modified to meet the applications requirements. The corresponding files are located in the PLM Interface (Cmt_Interface.h) and source (CMT.C and CMT.h) folders. To use them in your application, include the Cmt_Interface.h header file.

A.1 Implementing the RC5 transmission using the MC1323x CMT drivers

First of all, the clock frequency should be set to 36 kHz with a duty cycle of 30%. This is done in Cmt_Interface.h file by defining the following values:

```
#define gCmtDefaultCarrierFrequency_c   gCmtCarrierFrequency36kHz_c
#define gCmtDefaultCarrierDutyCycle_c   gCmtCarrierDutyCycle33_c
```

Then, the mark and space times are set as stated in Table 2. Mark and space times for logic 0 and 1 values among other configurations:

```
/* Logic 0 */
#define gCmtDefaultLog0MarkInMicros_c     889
#define gCmtDefaultLog0SpaceInMicros_c    889
/* Logic 1 */
#define gCmtDefaultLog1MarkInMicros_c     889
#define gCmtDefaultLog1SpaceInMicros_c    0

/* Default bits shifting: LSB first */
#define gCmtLsbFirstDefault_c             FALSE

/* Default mode of operation: Time */
#define gCmtTimeOperModeDefault_c         TRUE

/* Define the IRO output pin polarity */
#define gCmtOutputPolarityDefault_c       FALSE
```

After these values have been set, it is necessary to call CMT_Initialize() function in your main application. This will initialize the module with the previous values so it will be ready to transmit any data by calling CMT_TxBits() function .

Nevertheless, some modifications are necessary to implement the RC5 protocol properly. As stated in Section 4.3, “Using the MC1323x CMT Module for RC5 Transmission,” the extended space operation is needed to generate the logic 1 signal. Because the CMT driver does not consider using this feature, the CMT_TxBits()and CMT_InterruptHandler()functions must be modified.

In CMT_TxBits(), the extended space should be enabled in the case the first bit is a logic 1. This is done by setting the CMTMSC_EXSPC bit in the code section where the logic 1's mark and space values are set.

Due to the mark and space times configured for the logic 1 (889 µs and 0 µs respectively), a modulation cycle with the extended space operation enabled will generate only the first half of the bit (889 µs of space time) and the mark space still needs to be sent (889 µs). Therefore a sort of flag should also be set to indicate the application that the logic 1 bit is to be sent.

```
…
…
else
{
 /* Bit value: Logic 1 */
 mCMT_Modulation_MarkRegs  = mCmt_Log1MarkWidth;
 mCMT_Modulation_SpaceRegs = mCmt_Log1SpaceWidth;
 CMTMSC_EXSPC = 1; //Enables extended space operation
 repeat=1; //The logic 1 bit is to be sent

}
…
…
```

A similar modification is needed in the CMT_InterruptHandler() function. The number of bits left to be transmitted should not be decreased if the logic 1's mark time has not been sent (repeat flag is 1) to complete the logic 1 transmission. The complete modified function is as follows:

```
INTERRUPT_KEYWORD void CMT_InterruptHandler(void)
{
#if(gCmtSupported_d == 1)
  volatile uint8_t dummyRead;

  /* Clear the EOCF bit by reading the CMTMSC and accessing the CMTCMD2 register */
  dummyRead = CMTMSC;
  /* Disable the extended space operation */
  CMTMSC_EXSPC = 0;
  /* Touch the CMTCMD2 register */
  dummyRead = CMTCMD2;

  /* Check the number of bits left for Tx */
 if(repeat)mCmt_BitsLeft++;
  if((--mCmt_BitsLeft) != 0)
  {
    /* There still available bits for transmission */
    /* Depending on the bits shifting, different bit is loaded to be send */
         if(!repeat)
          {
                if(TRUE == mCmt_LsbFirst)
                {
                  /* LSB first */
                  /* Shift the data byte by one bit to the right */
                  mCmt_DataByte >>= 1;
                  /* Get the current bit value */
                  mCmt_CurrentBit = (mCmt_DataByte & 0x01);
                }
                else
                {
                  /* MSB first */
                  /* Shift the data byte by one bit to the left */
                  mCmt_DataByte <<= 1;
                  /* Get the current bit value */
                  mCmt_CurrentBit = (mCmt_DataByte & 0x80);
          }

  }
```

```
      else{
           if(TRUE == mCmt_LsbFirst) mCmt_CurrentBit = (mCmt_DataByte & 0x01);
           else mCmt_CurrentBit = (mCmt_DataByte & 0x80);
       }
 /* Determine the current bit value; depending on the logical bit value different mark/space
 /* is reloaded into modulation mark/space registers */
 if(!mCmt_CurrentBit)
  {
    /* Bit value: Logic 0 */
   mCMT_Modulation_MarkRegs  = mCmt_Log0MarkWidth;
    mCMT_Modulation_SpaceRegs = mCmt_Log0SpaceWidth;

  }
  else
  {
      if(repeat==0)
      {
                /* Bit value: Logic 1 */
               mCMT_Modulation_MarkRegs  = mCmt_Log1MarkWidth;
               mCMT_Modulation_SpaceRegs = mCmt_Log1SpaceWidth;
                CMTMSC_EXSPC = 1;
                repeat=1;

      }
       else {
               repeat=0;

               mCMT_Modulation_MarkRegs  = mCmt_Log1MarkWidth;
             mCMT_Modulation_SpaceRegs = mCmt_Log1SpaceWidth;
               CMTMSC_EXSPC = 0;
         }
      }
  }
  else
  {  /* The number of bits to be sent has been finished */
    /* Disable the module and the CMT interrupt */
    CMTMSC &= ~(mCMT_CMTMSC_MCGEN_c | mCMT_CMTMSC_EOCIE_c);

     /* Deasserts the Tx active flag */
      mCmt_TxActiveFlag = FALSE;

     /* Call the function callback; if it is a valid pointer to it */
     if(pfCmtTxCallBack)
      {
       pfCmtTxCallBack();
       }

    }
 #endif
 }
```

Finally, the RC5 transmission would consist of sending the start and toggle bits, the 5-bit address, and the 6-bit command:

```
toggle^= 0x01;
CMT_Status=CMT_TxBits(0x06 | toggle,3); //start + toggle
while(CMT_IsTxActive());
CMT_Status=CMT_TxBits(u8Address,5); // 5-bit address
while(CMT_IsTxActive());
CMT_Status=CMT_TxBits(u8Command,6); // 6-bit Command
while(CMT_IsTxActive());
```

A.2 Implementing the NEC transmission using the MC1323x CMT drivers

First of all, the clock frequency should be set to 38 kHz with a duty cycle of 30%. This is done in Cmt_Interface.h file by defining the following values:

```
#define gCmtDefaultCarrierFrequency_c   gCmtCarrierFrequency38kHz_c
#define gCmtDefaultCarrierDutyCycle_c   gCmtCarrierDutyCycle33_c
```

Then, the mark and space times are set as stated in Table 2. Mark and space times for logic 0 and 1 values among other configurations:

```
/* Logic 0 */
#define gCmtDefaultLog0MarkInMicros_c     560
#define gCmtDefaultLog0SpaceInMicros_c    560
/* Logic 1 */
#define gCmtDefaultLog1MarkInMicros_c     560
#define gCmtDefaultLog1SpaceInMicros_c    1690

/* Default bits shifting: LSB first */
#define gCmtLsbFirstDefault_c             TRUE

/* Default mode of operation: Time */
#define gCmtTimeOperModeDefault_c         TRUE

/* Define the IRO output pin polarity */
#define gCmtOutputPolarityDefault_c       FALSE
```

After these values have been set, it is necessary to call CMT_Initialize() function in your main application. This will initialize the module with the previous values so it will be ready to transmit any data by calling CMT_TxBits() function .

```
void NEC_protocol(uint8_t u8command, uint8_t u8address)
{
    (void)CMT_TxModCycle(u16NECStartmark,u16NECStartspace); // Send start of frame
    while(CMT_IsTxActive()); // Wait until ready for next transmission
    (void)CMT_TxBits(u8command,8); // Send command - 7 bits always
    while(CMT_IsTxActive());

// Wait until ready for next transmission
    (void)CMT_TxBits(u8address,8); // Send address - 5 bits in 12-bit SIRC version
    while(CMT_IsTxActive());

}
```

Prior to sending the frame, you could use the available CMT functions to set the start of frame's mark and space values as follows:

```
u16NECStartmark = CmtModMarkTicksInMicroSeconds(9000, mCmtClockInput8MHz_c); // Send start
u16NECStartspace = CmtModSpaceTicksInMicroSeconds(4500, mCmtClockInput8MHz_c);
```


How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4152
Rev. 1
04/2012