Freescale Semiconductor
Application Note

Document Number: AN2858
Rev. 0, 10/2007

Period and Pulse Accumulator (PPA) eTPU Function

by: Richard Soja
Austin, Texas

This eTPU application note describes the functionality of the Period and Pulse Accumulator (PPA) eTPU function and the usage of the C interface routines for initializing and retrieving results from the timing function. The interface routines are targeted for the MCF5235 and MPC5500 family of devices, but they should be portable to any device that has an eTPU. Example code is available for the MPC5554. This application note should be read with application note AN2864, “General C Functions for the eTPU.”

Contents

1 Function Overview

This eTPU input function measures the total accumulated high time, low time, or period of an input signal over a user-defined number of periods. The period may be measured from either rising or falling edges. The eTPU returns an accumulated result of measurements as a 24 bit unsigned integer value, representing clock counts, plus an 8 bit status value indicating overflow of the result. The C interface routines convert the 24 bit

result to a 32 bit unsigned integer data type. The host may be either the CPU or DMA controller integrated on the same device, or an external bus master if the microcontroller supports external bus arbitration. When the host is a CPU, the interface routines described in this document provide easy access to the PPA function. When the host is the DMA, the PPA programmers model, also described in this document, should be used as a reference to construct the DMA transfer control descriptors.

2 Function Description

The PPA function measures either period, high time, or low time of an input signal over a programmable number of periods. The number of periods over which a valid measurement can be accumulated is selectable between 1 and a maximum of approximately 200,000. The maximum depends on the frequency of the input signal being measured, because the PPA function stores the accumulated result in a 24 bit signed integer.

Four modes of measurement are available:

- Total high time over the selected number of periods.
- Total low time over the selected number of periods.
- Total period over the selected number of periods, starting on a rising edge.
- Total period over the selected number of periods, starting on a falling edge.

Figure 1 to Figure 4 show examples of the four measurement modes. All four examples are based on an accumulation of seven periods, which are numbered below each waveform. Dark-shaded areas in the bar below each waveform indicate what parts of the waveform are actually measured. Unshaded areas are not included in the measurement. Lightly shaded areas are part of the next measurement cycle.

The PPA measurement is expressed in counts of the selected eTPU timebase. The user can select either TCR1 or TCR2 as the timebase for the measurement. The number of TCR counts that have been accumulated multiplied by the period of one TCR count will give the total measurement expressed in seconds.

The PPA function operates continuously. After the specified number of periods has elapsed, the eTPU updates the accumulated result parameter, generates an optional interrupt request to the host CPU and optional DMA request to the DMA controller, and restarts the process for the next measurement cycle. The accumulated result is buffered to allow the CPU or DMA to delay reading the latest result until the next measurement is available.

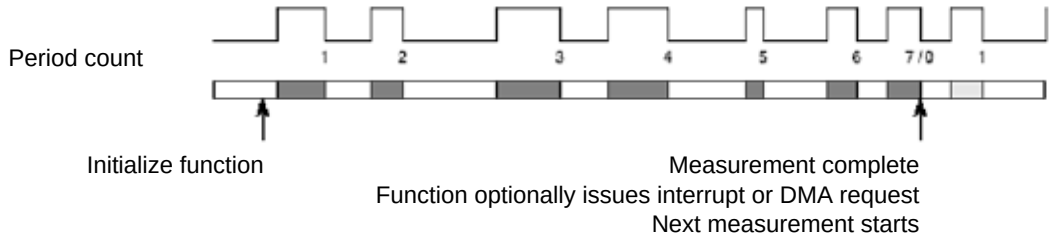

Figure 1. High-Time Pulse Measurement

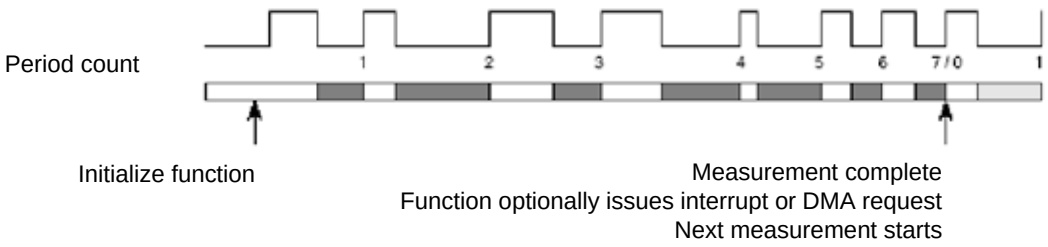

Figure 2. Low-Time Pulse Measurement

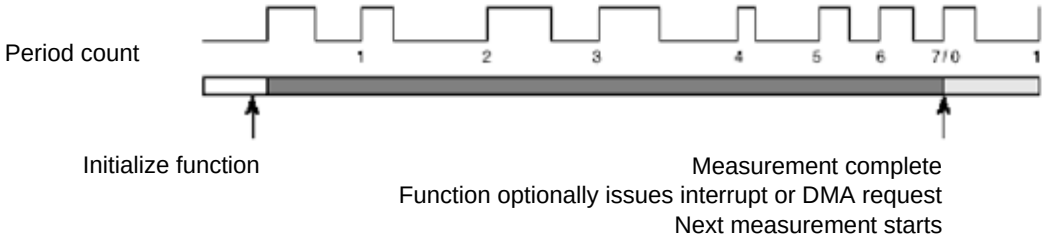

Figure 3. Rising- Edge Period Measurement

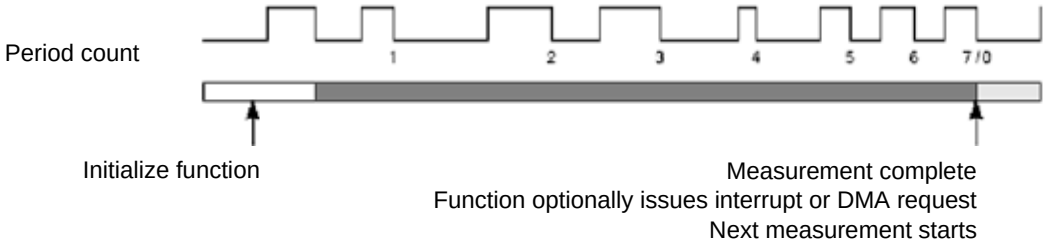

Figure 4. Falling-Edge Period Measurement

It is expected that an application will typically use the interrupt or DMA request mechanism to read an accumulated result. However, there are two other ways to read a result. These are discussed in sections 2.1 and 2.2 below.

2.1 Software Polling

Application software may poll the status of either the Channel Interrupt Status bit or the Data Transfer Request Status bit (found in the eTPU Channel Status Control register), to determine when a measurement is complete.

These bits are cleared by the following operations:

- By hardware after a power on or warm[1] reset (both bits)
- By software following an interrupt request (Channel Interrupt Status bit only)

1. A warm reset occurs when power is still applied to the device and software forces a system reset, usually by writing to a register or as a result of an unrecoverable program fault.

- By the DMA controller automatically when it reads the result after receiving a DMA request (Data Transfer Request Status bit only)

The flag is set after the PPA function writes the accumulated measurements to the accumulated result parameter, Result.

The initialization API defines a parameter to configure the PPA function to optionally enable an interrupt request and/or DMA request when a result is ready. If neither is enabled, software can poll the flag to determine when a result is complete.

In software-polled mode, it is the software designer's responsibility to ensure the flag is polled often enough to avoid losing a result.

2.2 Reading the Intermediate Value

An API is provided to allow software to read the currently accumulated measurement without waiting for the completion of the result. In this case, the status flag is not tested. As well as the currently accumulated measurement, the API returns the value of the period count associated with the currently accumulated period.

An important point to note is that the PPA function incorporates a parameter that allows all measurement modes to be accumulated at intervals shorter than the time between the edges required to complete the next measurement. For example, the uncompleted result of a pulse-width measurement can be updated while the pulse is still at its asserted level, without waiting for the pulse to terminate. Similarly, intermediate period measurements can also be updated at regular intervals shorter than the duration of the period of the edges. This may be useful to detect a stalled or excessively slow input signal. If the PPA is configured and accessed in this way, the intermediate result and period count may not be coherent. The granularity of the intermediate result depends on the update rate specified during initialization of the PPA function.

Figure 5 shows an example of an intermediate result fetched during the fourth period of measurement.

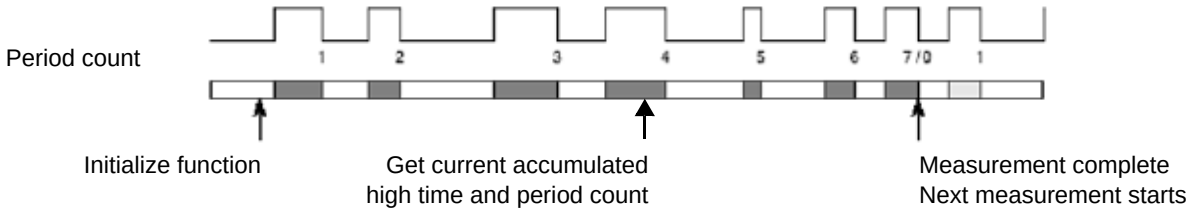

Figure 5. Read Intermediate Result and Period Count

The returned result will be the sum of the first three high pulses, or the sum of the first three high pulses plus the currently measured time of the fourth period, depending on the update rate. In either case, the period count returned will be 3.

2.3 Performance

Like all eTPU functions, the PPA function performance on a channel is to some extent dependent upon the service time (latency) of other active eTPU channels. This is due to the operational nature of the scheduler. However, the PPA function makes use of the dual action capture hardware to measure high or low pulses as short as one TCR clock period, provided the repetition rate of the pulse is not less than the worst case service time for the channel. The high and low pulse measurements modes use a state machine based on previous pin level to validate the measurement phase. A short pulse that occurs during a measurement, but has ended before the function has been able to respond is ignored. Because period measurements do not take advantage of the dual-action hardware, their performance is limited to the service time of the PPA channel for every active edge. For example, if configured to measure periods on rising edges, the eTPU hardware cannot detect falling edges. The presence of noise on the inactive edge could be construed as a valid-active edge, and cause incorrect results and reduction in performance.

Worst-case latency in any eTPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the eTPU, use the guidelines given in the eTPU reference manual and the PPA Worst Case Timing (WCT) for each measurement mode shown in Table 1. The numbers in the table are in system clock counts, and should be used to calculate the maximum rate at which any of the measurement modes can run continuously. For example, if the system clock was 128 MHz, and only the PPA function was running on only one channel, the maximum frequency that could be measured is (system clock / WCT) = 128 MHz / 80 = 1.6 MHz.

Table 1. PPA WCT for Each Measurement Mode (System Clocks)

Mode	Period on Rising Edges	Period on Falling Edges	High Pulse	Low Pulse
WCT[1]	80[2]	80[2]	100[2]	100[2]

1 The WCT values given here were obtained from code compiled with revision 1.0.7.42 of the Bytecraft Compiler.

2 In the event that a measurement overflows, the PPA function takes an additional 8 system clocks to update a status parameter.

Provided the continuous rate is not exceeded, the minimum width of detectable high or low pulses is limited only by the pin filter logic at the physical pad and the resolution of the timer selected by the programmer. This typically allows pulse measurements as low as 1 timer count. An input pulse width that is shorter than this will not be detected by the PPA function.

Exceeding the measurement rates of the PPA function (as calculated for a complete eTPU application) could result in aliasing effects that will be manifested as measurement errors.

3 PPA C Level API

The APIs described here provide a simple C programming interface to initialize and provide run-time access to one or more PPA functions. There are three APIs for controlling a PPA function:

- Initialization Function:
  - void etpu_ppa_init (uint8_t channel,
    uint8_t priority,
    uint8_t timebase,

uint8_8 mode,
uint8_t max_count,
uint32_t sample_count)

- Value Return Functions:
  - uint8_t etpu_ppa_get_accumulation (uint8_t channel,
    uint32_t * result)
  - uint8_t etpu_ppa_get_immediate (uint8_t channel,
    uint32_t * result,
    uint32_t * current_count)

The API C source code is contained in the file etpu_ppa.c.

3.1 PPA Function Header Files

The API includes several header files that contain the function prototypes and define symbolic values for the initialization and return functions.

3.1.1 etpu_ppa_auto.h

This header file is automatically generated by the eTPU compiler and defines symbols and their associated values needed to initialize the PPA function and the offset addresses (in bytes) for each PPA parameter. It is recommended that the content of this header file should not be modified, because some of the symbol values depend on other functions integrated into the function set, and these may change depending on the function set used.

3.1.2 etpu_ppa.h

This header file contains the function prototypes of the PPA API C source code contained in etpu_ppa.c.

3.1.3 etpu_util.h

This header file contains the function prototypes to initialize and configure the behavior of the eTPU engine. This header file also contains symbols used by the PPA function API. The C source code for configuring and loading the eTPU engine is contained in etpu_util.c.

3.2 Initialization Function

The initialization function should be executed after the appropriate eTPU engine has first been configured and loaded with the relevant microcode. A number of the parameters passed to the initialization function are defined symbolically in etpu_ppa_auto.h and etpu_util.h.

3.2.1 void etpu_ppa_init(uint8_t channel, uint8_t priority, uint8_t timebase, uint8_8 mode, uint8_t max_count, uint32_t sample_time)

This call associates a PPA timing function with an eTPU channel and initializes the channel to run in the desired mode. The call has no return value, and has the following five pass parameters:

- **channel**—The OC channel number. For devices with two eTPUs, this parameter should be assigned a value of 0-31 for eTPU_A and 64-95 for eTPU_B. For products with a single eTPU, this parameter should be assigned a value of 0-31.
- **priority**—This is the eTPU scheduler priority assigned to the channel. The value should be one of
  - FS_ETPU_PRIORITY_LOW
  - FS_ETPU_PRIORITY_MIDDLE
  - FS_ETPU_PRIORITY_HIGH

  These symbols are defined in the etpu_util.h file. A value of FS_ETPU_PRIORITY_DISABLE may be used to disable the channel.
- **timebase**—This defines which eTPU timebase is used for PPA measurements. The value should be either FS_ETPU_PPA_INIT_TCR1 or FS_ETPU_PPA_INIT_TCR2.

  These symbols are defined in the etpu_ppa_auto.h file.
- **mode**—This defines the PPA measurement mode. The value should be one of
  - FS_ETPU_PPA_LOW_PULSE
  - FS_ETPU_PPA_HIGH_PULSE
  - FS_ETPU_PPA_FALLING_EDGE
  - FS_ETPU_PPA_RISING_EDGE

  These symbols are defined in the etpu_ppa_auto.h file.
- **max_count**—This is the number of periods or pulses accumulated before the measurement restarts. This parameter should be assigned a value in the range 0 to 0x7FFFFF. A value of zero or one results in the accumulation of one period or pulse width.
- **sample_time**—This value defines the interval at which to update the intermediate measurement. If the sample_time interval is greater than the width or period being measured, the PPA function will not make an intermediate measurement. This parameter should be assigned a value in the range 0 to 0x7FFFFF.

When initializing eTPU channels the eTPU's behavior may be unpredictable if a channel is reconfigured while it is running. A channel must be stopped before it is configured. This is done by setting the channel's priority to disabled. If the channel is currently being serviced when the priority is set to disable it will continue to service the channel until the state ends. To make sure the channel is not being serviced, software should wait for the longest state execution time after disabling the channel. All channels are disabled out of reset so the channels can be configured immediately from reset.

The *etpu_ppa_init* function disables the channel before configuring it; however if the channel is already enabled before being disabled, the actual execution speed of this function call will depend on the type of

timing function being disabled. eTPU channels can be disabled separately using the e*tpu_disable* function in the mpc5500_utils.c file.

3.3 Value Return Functions

3.3.1 uint8_t etpu_ppa_get_accumulation(uint8_t channel, uint32_t * result)

This function call returns the status of the last measurement, and passes back the last accumulated pulse widths or accumulated periods, in counts of the selected timebase used to initialize the function: either TCR1 or TCR2

This function is normally called when the eTPU signals the host CPU via an interrupt request that the accumulation is complete, or after the CPU has read a set status flag. To ensure the returned result is coherent with the interrupt or status flag, the CPU function must read the result parameter before the completion of the next measurement. However, if the CPU is late in executing the call, the most recent measurement will be returned instead.

- uint8_t channel —This is the same channel number assigned by the etpu_ppa_init() function.
- uint32_t * result—This is a pointer to where the PPA result will be stored.

This function call can also be executed without waiting for the status flag to be set. A valid result will be returned provided the PPA function has completed at least one accumulation of measurements.

3.3.2 uint8_t etpu_ppa_get_immediate(uint8_t channel, uint32_t * result, uint32_t * current_count)

This function call returns the status of the current measurement in progress, and passes back the currently accumulated pulse widths or accumulated periods, in counts of the selected timebase used to initialize the function: either TCR1 or TCR2. It also passes back the current number of periods or pulses that have been accumulated.

This function may be called at any time, but note there is no guarantee that the result and current_count are coherent.

- uint8_t channel - This is the channel number assigned to the PPA function.
- uint32_t * result - This is a pointer to where the PPA result will be stored.
- uint32_t * current_count - This is a pointer to where the current elapsed number of periods or pulse widths will be stored.

3.4 Change Operation Functions

No API is defined to specifically change any PPA parameters while the function is running. To change the operating mode of an enabled PPA function you must first stop and then re-initialize the function. One way to stop the PPA function is to make an etpu_ppa_int() call with the priority parameter set to 0.

4 Example of Function Use

C source code that shows an example of how to use the eTPU PPA timing function is available from Freescale for the MPC5500 device family.

The following sections describe the steps needed to initialize the PPA timing function and read the result. It is assumed the eTPU engine has already been configured and loaded with binary code that includes the PPA timing function microcode.

The example code is contained in the C source files ppa_test.c and ppa_test.h.

4.1 Initializing the PPA

The e*tpu_ppa_init()* initialization routine initializes the channel to run the PPA function.

The initialization routine performs these operations:

1. Disables the channel by clearing the appropriate channel priority bits.
2. Assigns the PPA function's parameter RAM location for the desired channel.
3. Writes the PPA function number to the channel function select bits.
4. Writes the channel parameter RAM.
5. Writes the function mode bits to select the desired action edge and mode of operation.
6. Issues an HSR to initialize the function.
7. Writes a non-zero value to the channel priority bits to enable the function and assign the channel priority.

The example C code in file ppa_test.c assigns the PPA function to eTPU channel 10, and configures it to measure high pulse widths, continuously accumulating the measurements every two periods, with a resolution defined by the TCR1 timebase. The SampleTime is set to 0x100000, which means intermediate accumulations will only occur if the width of the high pulse exceeds 0x100000 TCR1 counts.

4.2 Getting measurements from the PPA

Generally the most effective method of getting a measurement is to implement an interrupt handler that responds to the measurement complete interrupt generated by the PPA or to enable a DMA channel to transfer the result plus status to system memory. Alternatively, software may poll the channel interrupt status flag without enabling interrupts to determine when a measurement is complete, though this may cause the result to be obtained late in the measurement cycle. A final method is to read the result at any time without checking the status flag. Because the PPA function runs continuously, and the result is buffered, this method is guaranteed to always return a valid result except during the very first measurement period, when no accumulated result is yet available. In this case, it is possible to obtain an intermediate, less accurate measurement by executing the *etpu_ppa_get_immediate()* call, provided a properly sized SampleTime has been set up during initialization.

In the example code, the interrupt status flag bit for eTPU channel 10 is polled to determine when a new result is available.

Prior to polling the flag, the interrupt status flag, DMA request flag, and overrun flags are all cleared.

After testing the interrupt status flag, the new result is read into an array using the *etpu_ppa_get_accumulation()* call and all status and overrun flags are cleared again.

This process is repeated twelve times in the example, and then hangs in a dummy-while loop.

The results stored in the twelve elements of the array may be inspected using a debugger.

4.3 Measurement status value

The functions *etpu_ppa_get_immediate()* and *etpu_ppa_get_accumulation()* return an 8 bit unsigned value to indicate whether an overflow has occurred on the intermediate value or final accumulated value respectively.

An overflow is indicated when the most significant bit is set in the returned value.

A return value of 0 indicates that no overflow has occurred.

A return value that has bit 0 and/or bit 1 set indicates an attempt was made to reconfigure the mode of the function while it was making a measurement, and an invalid state was entered. To ensure correct operation, the measurement mode should be changed only after first halting the PPA function as described in Section 3.2.1, "void etpu_ppa_init(uint8_t channel, uint8_t priority, uint8_t timebase, uint8_8 mode, uint8_t max_count, uint32_t sample_time)," and re-initializing the PPA function in the new mode.

5 PPA Function Programmer's Model

This section provides detailed descriptions of the PPA timing function parameters stored in channel parameter RAM. Each parameter name is followed by the symbolic name used by the API to locate the offset of the parameter in memory.

The eTPU compiler exports these symbols and their values to the file etpu_ppa_auto.h.

5.1 Accum

FS_ETPU_PPA_ACCUM_OFFSET

This 24 bit value should be initialized to 0 by the host and updated by the eTPU only. The parameter contains the current uncompleted period or pulse measurement while measurement is in progress. The parameter is updated at a rate defined by SampleTime.

5.2 StatAcc

FS_ETPU_PPA_STAT_ACC_OFFSET

This 8 bit value contains a single bit flag to indicate that the currently active accumulation has overflowed a 24 bit value. The parameter is initialized to 0 by the CPU and updated by the eTPU. The eTPU copies this value to the Status parameter when the measurement completes, then resets StatAcc to 0. An overflow is indicated when StatAcc is 0x80.

5.3 SampleTime

FS_ETPU_PPA_SAMPLE_TIME_OFFSET

This 24 bit value defines the rate at which the measurement is updated in the absence of an edge. This allows the host to make an interim measurement without waiting for the current pulse or period to complete. The CPU should initialize this value to the number of timebase counts desired between each update; the eTPU schedules the first update event when a measurement is activated by a valid edge. If the update period is longer than the maximum width between active pulse or period edges, then no measurement update will occur. Note that excessively high update rates (very small values of SampleTime) may result in performance degradation of the eTPU.

5.4 LastTime[1]

This 24 bit value is used by eTPU only. It contains the time of the last transition during active measurement of a pulse or period.

5.5 PeriodCount

FS_ETPU_PPA_PERIOD_COUNT_OFFSET

This 24 bit value should be initialized to 0 by the host and is updated by the eTPU after the PPA function is started. It contains the current number of accumulated pulses or periods since the PPA function started or since the last measurement was returned in the parameter Result.

5.6 MaxCount

FS_ETPU_PPA_MAX_COUNT_OFFSET

This 24 bit value specifies the number of pulses or periods that should be measured. MaxCount is initialized by the host and tested by the eTPU.

5.7 Result

FS_ETPU_PPA_RESULT_OFFSET

This 24 bit value is updated by the eTPU only. It contains the final total accumulation when the measurement completes. The parameter Status indicates whether the result has overflowed. The host may read this value at any time.

5.8 Status

FS_ETPU_PPA_STATUS_OFFSET

This 8 bit value indicates whether the completed measurement that is available in parameter Result has overflowed. At the end of the measurement, the eTPU copies StatAcc to Status after it has copied Accum

1.Because it is exclusive to the PPA function, no symbol is exported for the parameter LastTime.

to Result, and then the eTPU sets the channel interrupt status flag and DMA request flag. This ensures the host may coherently read the measurement result and its status.

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
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN2858
Rev. 0
10/2007

freescale™
semiconductor