Freescale Semiconductor
Application Note

Document Number: AN3836
Rev. 2 8/2009

Advanced Headlights Control and Diagnostics

Using the Cross-Triggering Unit between PWM and ADC Channels on MPC560xB/C Microcontrollers

by: Petr Cholasta
Roznov CSC, Czech Republic, EMEA

This document describes how to benefit from the MPC560xB/C MCU family in the Lighting application. The information provided gives the user an example of how to utilize the MPC5604B eMIOS, CTU, and ADC modules in the Lighting application, supporting circuitry diagnostic, with no additional load on the MCU CPU. This example is based on xPC560BKIT144S EVB.

Part of this document is also the software AN3836SW.

The example given may be modified to suit the requirements of a specific application.

Contents

freescale™ semiconductor

1 Lighting Application Design Challenge

When controlling the lamp light bulb, a few specific points need to be considered.

Lamp power-supply voltage may differ comparing to nominal value. Figure displays lamp life time depending on power-supply voltage. The impact of 5% overvoltage (630 mV @ 12.6 V) on light bulb performance is significant. The light bulb luminous flux grows up to 120% and the light bulb life time is reduced to less than 60% comparing to nominal values. Therefore, the lamp power-supply voltage should be regulated to nominal value in order to keep the lamp life time. The most common control technique is the PWM, that can be used also for lamp light beam intensity control.

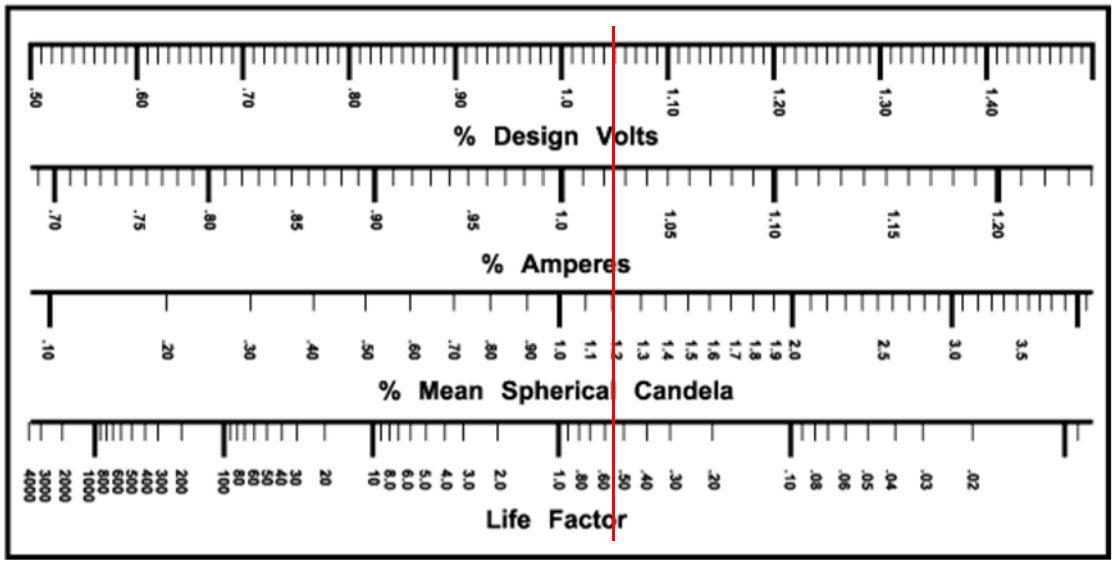

Figure 1. Lamp Light Bulb Characteristic

When turning lamp light bulb ON, the bulb current applied to cold filament can reach up to ten times of the rated current. The total current of several lamps causes significant power-supply voltage drop and the EMC emissions growth. Therefore, it is essential to not drive all lamps together.

The lamp life time is limited. In addition, the lamp life time can be even shorter if connected to faulty environment. If the design includes lamp diagnostic and protection, property can be saved.

All mentioned features need to be taken in account when designing the lighting module.

1.1 Lighting Module

Let's sum up the Lighting application design requirements:

- Battery-supply voltage equals 12.6 V nominal, but may vary depending on the actual battery voltage.
- Lamp bulb-to-LED exchange enablement.
- Lamp light beam intensity driven using the PWM technique, variable frequency.
- Lamp diagnostic and protection.
- EMC-compatible design (for example do not turn ON the lamps simultaneously).

Considering the facts mentioned, the Lighting module features should satisfy requirements as follows:

- Lamp light beam intensity independent on battery-supply voltage.
- Provide an independent PWM-driven control output for each lamp.
- Unified system for lamp as well as for LED control.
- Measure the lamp current for diagnostic purposes.
- Control PWM edge timing and slope.

1.2 Microcontroller Unit

To satisfy the given requirements, the MCU modules should enable functionality as follows:

- ADC module
  - Senses battery-supply voltage to constantly control light beam intensity.
  - Senses light current for diagnostic and protection purposes.
  - Threshold interrupt for fast action in the case of a light bulb failure.
- PWM module
  - Has enough PWM channels to control lights independently.
  - PWM duty cycle reloads immediately after a request (synchronization).
  - PWM 0% and 100% duty cycle with no spikes (EMC-related).
  - PWM leading edge start delay is configurable for each channel independently (EMC-related).
  - Synchronization signal for an ADC channel conversion trigger at certain times to support PWM channel light diagnostics and protection.
- ADC trigger signal generator module
  - Using a Timer, trigger interrupts need to be handled by the MCU CPU.
  - MPC560xB/C family provides the CTU module, trigger interrupts do not disturb the MCU CPU.

2 MPC5604B Lighting Application Related Modules

The MPC5604B is an MPC560xB/C 32-bit microcontroller unit family member. It operates at speeds up to 64 MHz and provides a variety of peripherals (DSPI, IIC, ADC, CTU, eMIOS, FlexCAN, LINFlex) balanced with relatively large Flash, RAM, and an emulated EEPROM memory.

The section below covers the eMIOS, CTU, and ADC module description, which are essential parts of the Lighting application suitable for lumpily as well as for LED control. It is recommended to read the MPC560xB/C manual peripheral description available at www.freescale.com.

2.1 eMIOS — Enhanced Modular I/O Subsystem

The MPC5604B includes two identical eMIOS modules, that can together provide up to 56 channels with input and output capabilities. Each eMIOS module provides 25 channels with an OPWMT mode, especially suited for the Lighting application.

When configuring a channel in OPWMT mode, the channel counter can be clocked from three different sources:

- Global Prescaler 8-bit counter
- Counter Bus A driven by UC[23]
- Counter Bus B, C, D driven by UC[0], UC[8], UC[16]

For the proposed Lighting application, UC[23] is configured as a PWM 100 Hz enabled timebase, UC[0], UC[8], UC[16] are configured as PWM 200 Hz enabled timebases and UC[24] is configured as UC[25], UC[26], UC[27] timebase. This approach enables the user to select the PWM frequency for each channel independently. Using this scheme, 20 channels are configured with OPWMT capability on each eMIOS module, in other words 40 channels per device.

The OPWMT channel A1 register value defines the PWM leading edge start delay, the B1 register value trailing edge (duty cycle), A2 register is the sampling point for diagnostic purposes (the ADC channel trigger using the CTU), and in B2 the buffered value of the trailing edge, which is, on A1 match, loaded into the B1 register (see MPC5604B eMIOS module documentation available at www.freescale.com). When the B2 value equals the A1 value, the PWM duty cycle is 0%. Where a B2 value is higher than the channel period, the PWM duty cycle reaches 100%. The PWM output signal polarity is dependent on the user's choice.

When configuring channels UC[0], UC[8], UC[16], UC[23], and UC[24] in modulus counter mode (MC), the A1 register determines the counting period.

2.2 CTU — Cross-Triggering Unit

When the eMIOS OPWMT channel counter reaches the A2 value, the specific ADC channel conversion is initiated by the CTU module. The ADC channel number is specified by the user in the CTU event-configuration register. When configuring the event-configuration register, be aware, that:

- CTU channel 0-15 event can trigger any of ADC channels 0-15
- CTU channel 16-31 event can trigger any of ADC channels 32-47
- CTU channel 32-64 event can trigger any of ADC channels 64-95

2.3 ADC — Analog-to-Digital Converter

The ADC module converts analog signals into digital form with a 10-bit resolution offering normal, injected, and trigger-injected modes. The ADC module can be configured to provide either 36 channels with no additional external logic or 64 channels using an external multiplexer, all driven by the ADC module logic.

When using the MPC5604B:

- ADC channels 0-15 correspond to SIUL ANP pins 0-15.
- ADC channels 32-47 correspond to SIUL ANS pins 0-15.
- ADC channels 64-95 correspond to SIUL ANX pins 0-3 (external multiplexer driven by MA0-2 ADC pins).

Four analog watchdogs are present on the ADC peripheral to determine whether the converted channel value lies within a given guard area specified by threshold values given by the user. If the converted value lies outside the guarded area, an interrupt is generated. The analog watchdogs interrupts are well-suited for light bulb diagnostic purposes.

The ADC conversion cycle can be split into two main phases, a sampling phase, in other words charging an internal sampling capacitor, and an evaluation phase, in other words the 10-bit conversion itself. Both phases can be adjusted (see Section 4.1.4, “ADC Module“).

3 Lighting Application Example Demonstration

The Lighting application example code introduces essential MPC5604B initialization and configuration for demonstrating eMIOS and ADC synchronization using the CTU. An example is based on LA system study depicted on Figure , that it is suitable for lamp bulb as well as for LED control.

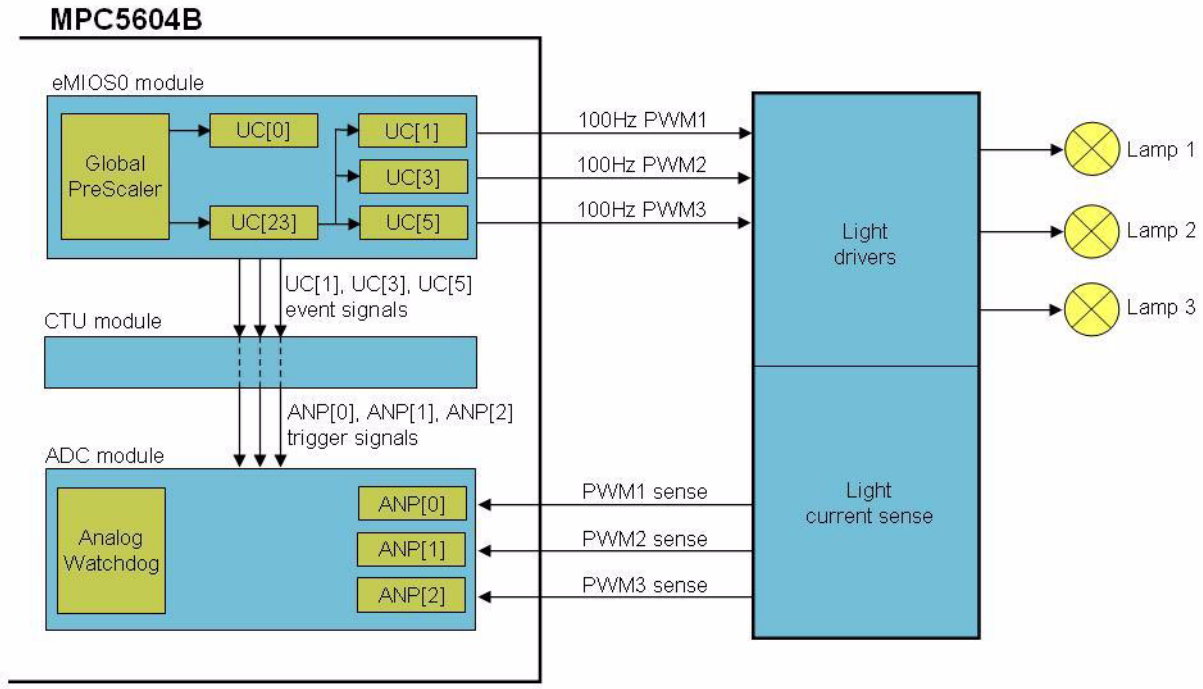

Figure 2. Lighting Application Example System Concept

The demonstration is based on the xPC560BKIT144S EVB and the application code is written in C language, on the Eclipse platform, using the Green Hills compiler (see AN3836SW).

Prepare the xPC560BKIT144S EVB with the MPC5604B MCU, turn ON the power supply, and download the *intflash.elf* file to the target MCU.

The *intflash.elf* file can be found in folder: *MPC5604B_BLA\MPC5604B_BLA\MPC5604B_BLA\bin*

Check if the on-board LED1 is turned ON.

3.1 eMIOS PWM Capability

Connect oscilloscope probes to the PJ7 connector:

- pin 2 (AN1), LA PWM1, MCU PA[1]: ADC[0] ANP[0] trigger demonstration
- pin 4 (AN3), LA PWM2, MCU PA[3]: start delay demonstration
- pin 6 (AN5), LA PWM3, MCU PA[5]: UC[3] configuration demonstration, when UC[23] rolls over and waveforms are seen as depicted in Figure . The yellow color belongs to the PWM1 channel signal running with a 50% duty cycle. The blue color represents the PWM2 signal with a 0.5 ms start delay compared to PWM1. The PWM3 signal is colored pink, showing a 1 ms start delay compared to PWM1.

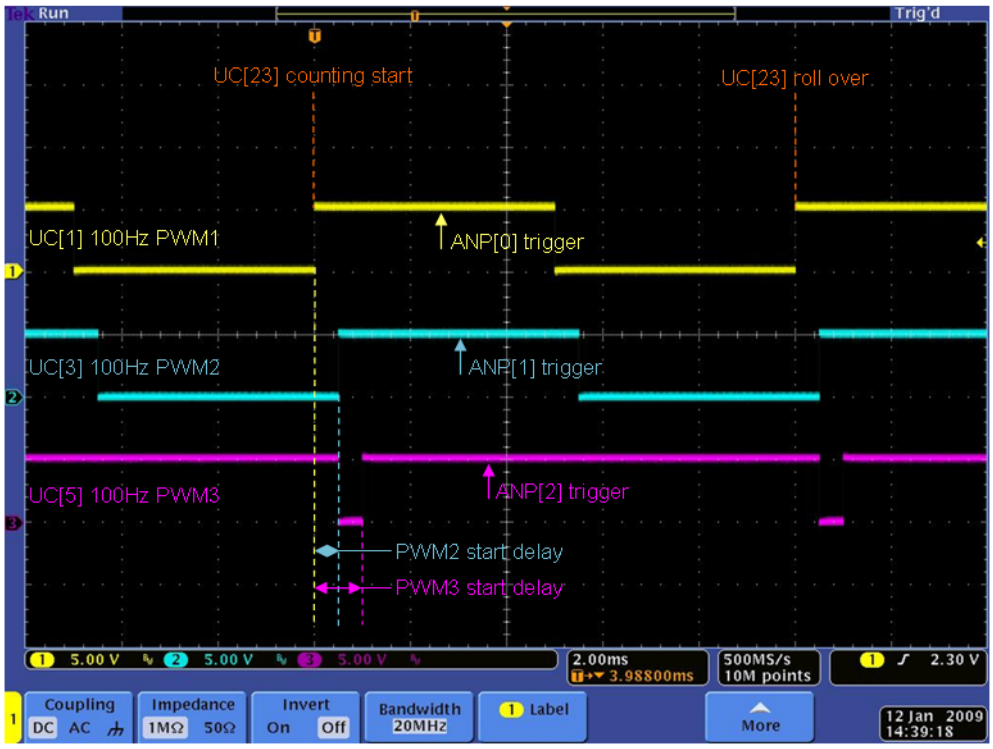

Figure 3. PWM Channels Output Signals

The PWM channels example initially completes a configuration as follows:

- PWM1 — 100 Hz frequency (10 ms period), 50% duty cycle (5 ms ON, 5 ms OFF), 0% start delay (0 ms), 25% (2.5 ms from PWM1 leading edge) ADC ANP [0] trigger point
- PWM2 — 100 Hz frequency (10 ms period), 50% duty cycle (5 ms ON, 5 ms OFF), 5% start delay (0.5 ms), 25% (2.5 ms from PWM2 leading edge) ADC ANP [1] trigger point
- PWM3 — 100 Hz frequency (10 ms period), 95% duty cycle (9.5 ms ON, 0.5 ms OFF), 10% start delay (1 ms), 25% (2.5 ms from PWM3 leading edge) ADC ANP [2] trigger point.

The application parameters can be changed in the editor (Eclipse). Go to folder *MPC5604B_BLA* and double click on *open_workspace.bat*. The Eclipse BLA workspace is ready to use.

Open the *main.c* file in the Eclipse workspace folder *MPC5604B_BLA* (see Figure), and work with the parameters as follows:

- ch[PWM1].perSel — select the PWM1 channel period as either 100 Hz or 200 Hz
- ch[PWM1].startDelayTime — the PWM1 leading edge start delay count, when the UC[23] timebase rolls over. The value can be selected in 0.5% steps, in the range of 0 (0 ms) to 200 (100 Hz ~10 ms, 200 Hz ~ 5 ms).
- ch[PWM1].dutyCycle — the PWM1 duty cycle. The value can be selected in 0.5% steps, in the range of 0 (PWM1 output OFF) to 200 (PWM1 output ON). Value 100 corresponds to a 50% duty cycle.
- ch[PWM1].triggerAdc — the PWM1 ANP[0] sample point. The value can be selected in 0.5% steps, in the range of 0 (0 ms) to 200 (100 Hz ~10 ms, 200 Hz ~ 5 ms). Value 50 corresponds to a 2.5 ms (25%) start delay on the PWM1 leading edge.

The user can select another channel than the PWM1 to control. Simply exchange the ch[PWM1] statement with the requested one, for example ch[PWM2], ch[PWM3].

Proceed code compile and download the *intflash.elf* file to the MCU.

When the demostration is finished, change the parameters back to:

- ch[PWM1].perSel = 100;
- ch[PWM1].startDelayTime = 0;
- ch[PWM1].dutyCycle = 100;
- ch[PWM1].triggerAdc = 50;

This approach enables the user to change the OPWMT channel parameters run-time.

3.2 ADC Conversion Start Using the CTU

Connect the trimmer W1 output J18 to connector PJ5 pin 5 (AN23, LA ADC PWM1 trigger, MCU PB4, ANP[0] pin). Now change the trimmer W1 value and see how the on-board LEDs are turned ON/OFF based on the ADC ANP[0] pin value, which is periodically converted by the ADC module, starting when there is a match on the eMIOS PWM1 channel compare module with the A2 register.

It is possible to carry out the same demonstration for channels PWM2 and PWM3.

In the case of PWM2:

- connect the trimmer W1 output to connector PJ5 pin 6 (AN24, LA ADC PWM2 trigger, MCU PB5, ANP[1] pin)
- open the *main.c* file and change the ADC channel to drive LED control:
  data0 = Mcu_Adc_0_Data_Prec(1)
- compile and download the code to the MCU

In the case of PWM3:

- connect the trimmer W1 output to connector PJ5 pin 7 (AN25, LA ADC PWM3 trigger, MCU PB6, ANP[2] pin)

- open the *main.c* file and change the ADC channel for LED control: *data0 = Mcu_Adc_0_Data_Prec(2)*
- compile and download the code to the MCU.

When the demonstration has finished, change the statement back to ANP[0]: *data0 = Mcu_Adc_0_Data_Prec(0)*

3.3 PWM Channel ADC Conversion Trigger Point

The ADC trigger point is given by the *ch[PWM1].triggerAdc* variable value. This value can be changed in 0.5% steps, in the range of 0 (0 ms) to 200 (100 Hz ~10 ms, 200 Hz ~ 5 ms).

To verify the trigger point position, proceed with the following steps.

1. Connect the PWM1 output (AN1) to PJ5 pin 5 (AN23 ~ MCU ANP[0] pin).
2. Modify the *main.c* file: *ch[PWM1].triggerAdc = 100*, compile and download the code to the MCU. LED2, LED3, LED4 are turned OFF, because the ADC samples the PWM1 channel when output is OFF.
3. Modify the *main.c* file: *ch[PWM1].triggerAdc = 99*, compile and download the code to the MCU. LED2, LED3, LED4 are turned ON, because the ADC samples the PWM1 channel when output is ON.

4 Lighting Application Example Software Introduction

Let's introduce the workspace folders and files' structures (see Figure):

- Folder *bin* includes, among the others, the *intflash.elf* file, the final MCU code ready to download.
- Folder *drv* includes drivers for the LA MCU peripherals.
- Folder *system* includes the MCU startup code and an interrupt vector table.
- File *jdp.h* includes the MCU register definitions.
- File *main.c* is the application main file.
- File *typedefs.h* defines the types.
- *Makefile* includes the rules used during the link process.

The application code can be separated into two main segments (see *main.c* file). The first one is processed once after the MCU reset and can be treated as the MCU initialization. The second part is processed continuously and offers the user a space for application control.

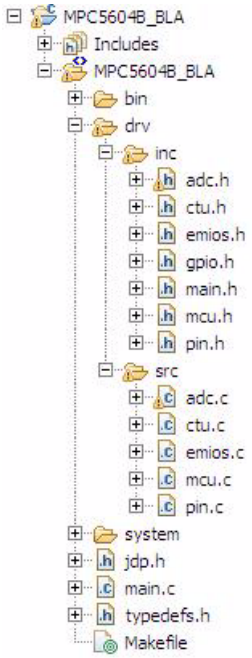

Figure 4. Lighting Application Folder Structure

4.1 MCU Initialization

4.1.1 Setup Clocks

Code is present in the *mcu.c* and *mcu.h* files, which can be found in the workspace folder *drv*. This module enables clocks to all peripherals and sets the 64 MHz bus clock. For a detailed description, see file *mcu.c*, function *void Mcu_Per_Enb_Osc_Init(void).*

4.1.2 Configure Pins

Code is present in the *pin.c* and *pin.h* files, which can be found in the workspace folder *drv*. The *pin.h* file includes a set of macros covering all possible pin configurations:

```
#define RST   0x0000   /* Reset state */
#define GPI   0x0100   /* GP Input mode */
#define GPO   0x0200   /* GP Output mode */
#define F1O   0x0400   /* Alternative Function 1 */
#define F2O   0x0800   /* Alternative Function 2 */
#define F3O   0x0C00   /* Alternative Function 3 */
#define FXI   0x0100   /* Alternative Function 1, 2, 3 Input mode */
#define FAI   0x2000   /* Alternative Function ADC input pin */
```

Each SIUL module Port Configuration Register (PCR) used in the LA is configured as follows:

```
SIU.PCR[0].R    = F2O;    /* PA0 - MCU CLKOUT */
SIU.PCR[1].R    = F1O;    /* PA1 - eMIOS 0, channel 1 */
SIU.PCR[3].R    = F1O;    /* PA3 - eMIOS 0, channel 3 */
SIU.PCR[5].R    = F1O;    /* PA5 - eMIOS 0, channel 5 */
SIU.PCR[20].R   = FAI;    /* PB4 - ANP[0] */
SIU.PCR[21].R   = FAI;    /* PB5 - ANP[1] */
SIU.PCR[22].R   = FAI;    /* PB6 - ANP[2] */
SIU.PCR[68].R   = GPO;    /* PE4 - LED1 */
SIU.PCR[69].R   = GPO;    /* PE5 - LED2 */
SIU.PCR[70].R   = GPO;    /* PE6 - LED3 */
SIU.PCR[71].R   = GPO;    /* PE7 - LED4 */
```

The rest of the MCU pins are configured in RST mode.

4.1.3 eMIOS 0 Module

Code is present in the *emios.c* and *emios.h* files, which can be found in the workspace folder *drv*. The *emios.h* file includes a set of macros as follows:

- channel assignment (PWM, counter A and B bus)

```
#define PWM1  1
#define PWM2  3
#define PWM3  5
#define CTRA  23
#define CTRB  0
```

- PWM channel control structure

```
typedef struct
{
        vuint8_t perSel;          // PWM period (100Hz, 200Hz)
        vuint8_t dutyCycle;       // PWM Duty cycle (from 0 to 200 - 0.5% step)
        vuint8_t startDelayTime;  // PWM shift to base (from 0 to 200 - 0.5% step)
        vuint8_t triggerAdc;      // ADC trigger point (from 0 to 200 - 0.5% step)
}chPar;
```

- scaling constants

```
// PWM 100Hz period
// PWM Duty cycle 0.5% step
#define DUTY_CYCLE_STEP_100   200
// PWM Start delay 0.5% step
#define START_DELAY_TIME_STEP_100   200
// PWM ADC trigger point 0.5% step
#define TRIGGER_ADC_STEP_100  200
// PWM period counter roll over value
#define COUNTER_PERIOD_100    40000


// PWM 200Hz period
// PWM Duty cycle 0.5% step
#define DUTY_CYCLE_STEP_200   100
// PWM Start delay 0.5% step
#define START_DELAY_TIME_STEP_200   100
// PWM ADC trigger point 0.5% step
#define TRIGGER_ADC_STEP_200  100
// PWM period counter roll over value
#define COUNTER_PERIOD_200    20000
```

4.1.3.1 eMIOS0 Module Initialization

The eMIOS0 module configuration is processed, when the function *void Mcu_Emios_0_Init(void),* present in the *emios.c* file, is called.

The bus clock is divided by one in the System Clock Prescaler. Then the eMIOS0 module Global Prescaler divides the clock by four. In the case of a 64 MHz bus clock, the eMIOS0 universal counters from UC[0] to UC[27] can be clocked with a frequency of 16 MHz (64 MHz / 4). The eMIOS module Global Prescaler functionality is controlled by the eMIOS0 Module Configuration Register (MCR).

```
// Module Configuration register
// Global prescaler enable, divide ratio 4
EMIOS_0.MCR.R = 0x04000300;
```

It is important to enable transfers between the A and B registers (see Section 2.1, "eMIOS — Enhanced Modular I/O Subsystem").

```
// Module Output Update Disable register
// Enable transfers (A2 -> A1, B2 -> B1) for all 28 channel registers
EMIOS_0.OUDR.R = 0x00000000;
```

The 16 MHz clock is used to clock the PWM timebases UC[0] and UC[23]. The UC[0] counter generates a clock for counter bus B, channels UC[1] through to UC[7]. The same approach can be used for counter bus C using UC[8], counter bus D using UC[16], and counter bus E using UC[24]. UC[23] generates the counter bus A timebase, common for all OPWMT mode UC channels.

This example covers the initialization of UC[0] and UC[23] as bus counters and UC[1], UC[3], UC[5] as OPWMT channels. That is why the clocks are enabled only for the channels as follows:

```
// Enable channels: 0, 1, 3, 5, 23 (0 -> enable)
EMIOS_0.UCDIS.R = 0xFFFFFFFF;
EMIOS_0.UCDIS.B.CHDIS0 = 0;
EMIOS_0.UCDIS.B.CHDIS1 = 0;
EMIOS_0.UCDIS.B.CHDIS3 = 0;
EMIOS_0.UCDIS.B.CHDIS5 = 0;
EMIOS_0.UCDIS.B.CHDIS23 = 0;
```

UC[23] is configured as the timebase for 100 Hz OPWMT channels. The UC[23] input clock equals 16 MHz. It is required to generate an OPWMT with a reasonable resolution. The maximum value of an OPWMT 16-bit channel comparator A value equals 65535, corresponding to the comparator A clock 65536 steps.

Let's consider a UC[23] value of four. Then the counter bus A is clocked at a speed of 4 MHz (16 MHz / 4). The OPWMT channel PWM frequency equals 100 Hz. Then, the OPWMT comparator A match value equals 40000 (4 MHz/100 Hz), which is appropriately below the comparator A maximum value. Using this configuration it is possible to achieve a PWM channel of 0.0025% resolution.

UC[23] is configured in MC mode using the Channel Control Register (CCR). The counter period is configured by writing the value 39999 (40000 clock steps) into the A1 register (CADR).

Function *Mcu_Emios_0_Channel_Timebase_Init(CTRA, COUNTER_PERIOD_100)* proceeds:

```
// Channel 23 configuration - MC mode - counter bus A 100Hz
// Enable prescaler (divide ration 4), global clock, MC mode
EMIOS_0.CH[23].CCR.R = 0x0E000310;
// Config A1: 4MHz/100Hz = 40000, counter bus A period = A1 + 1 => A1 = 39999
EMIOS_0.CH[23].CADR.R = COUNTER_PERIOD_100 - 1;
```

The same calculation can be processed for UC[0] acting as the timebase for the 200 Hz OPWMT channels.

Function *Mcu_Emios_0_Channel_Timebase_Init(CTRB, COUNTER_PERIOD_200)* proceeds:

```
// Channel 0 configuration - MC mode - counter bus B 200Hz
// Enable prescaler (divide ration 4), global clock, MC mode
EMIOS_0.CH[0].CCR.R = 0x0E000310;
// Config A1: 4MHz/200Hz = 20000, counter bus B period = A1 + 1 => A1 = 19999
EMIOS_0.CH[0].CADR.R = COUNTER_PERIOD_200 - 1;
```

Then PWM1 (UC[1]), PWM2 (UC[3]), PWM3 (UC[5]) initial configuration (OPWMT mode, select counter bus A, output signal polarity and channel CTU enablement) proceeds. An example follows:

```
// PWM1 channel configuration - OPWMT mode
// Enable prescaler (divide ration 1), counter bus A clock, OPWMT mode
EMIOS_0.CH[PWM1].CCR.R = 0x02000026;
// A match comparator A sets output, while match B clears it.
EMIOS_0.CH[PWM1].CCR.B.EDPOL = 1;
// Flag generate DMA request (CTU trigger)
EMIOS_0.CH[PWM1].CCR.B.DMA   = 1;
// Enable the flag generate DMA request (CTU trigger)
EMIOS_0.CH[PWM1].CCR.B.FEN   = 1;
```

When initialization proceeds, global timebase is enabled:

```
// Global time base enable
EMIOS_0.MCR.B.GTBE = 1;
```

4.1.3.2 eMIOS0 OPWMT Channel Initial Parameters Storage

During the function *void Mcu_Emios_0_Channel_Param_Load(void)* processing, the PWM channels parameters are loaded into the channel structures stored in RAM memory. An example follows.

```
// Channel PWM1
// Period 100Hz
ch[PWM1].perSel = 100;
// Leading edge at the beginning of a counting cycle
ch[PWM1].startDelayTime = 0;
// Duty cycle 50%
ch[PWM1].dutyCycle = 100;
// ADC trigger point 25%
ch[PWM1].triggerAdc = 50;
```

This approach enables the user in run-time to collect channel parameters whenever needed, and to change all OPWMT channels parameters synchronously at a given moment.

Once data for each PWM channel is loaded into the channels structures, the OPWMT channel registers configuration proceeds.

4.1.3.3 eMIOS0 OPWMT Channel Initialization

During the function *void Mcu_Emios_0_Channel_OPWMT_Init(vuint8_t i)* call, the following parameters are loaded:

- clock select into the Channel Control Register (CCR), bit BSL, and scaling constants initiation:

```
// Change counter clock to 100Hz (Counter Bus A)
EMIOS_0.CH[i].CCR.B.BSL = 0;

// Load recalculation constants
chDutyCycleStep = START_DELAY_TIME_STEP_100;
chStartDelayStep = DUTY_CYCLE_STEP_100;
chTriggerAdcStep = TRIGGER_ADC_STEP_100;
counterPeriod = COUNTER_PERIOD_100;
```

- channel leading edge calculation and load into the A1 register (CADR):

```
// Channel Start delay time
// Recalculate PWM leading edge Start delay time
chStartDelay = (vuint32_t)(ch[i].startDelayTime * chStartDelayStep);
// Load PWM leading edge start delay time into A1
EMIOS_0.CH[i].CADR.R = chStartDelay;
```

- channel duty cycle recalculation and load into the B2 register (CBDR). Recalculation, when timebase counter UC[23] rolls over, included.

```
// Channel PWM Duty Cycle
// Recalculate Duty cycle
chDutyCycle = (vuint32_t)(ch[i].dutyCycle * chDutyCycleStep);
// Load duty cycle into channel B2 register (transferred to B1 on a counter A1 match)
if (chDutyCycle == counterPeriod)
{
        // PWM Duty cycle 100%
        EMIOS_0.CH[i].CBDR.R = chDutyCycle;
}
else if ((chDutyCycle + chStartDelay) > counterPeriod)
{

        // PWM trailing edge after counter roll over
        EMIOS_0.CH[i].CBDR.R = chDutyCycle + chStartDelay - counterPeriod;
}
else
{
        // PWM trailing edge before counter roll over
        EMIOS_0.CH[i].CBDR.R = chDutyCycle + chStartDelay;
}
```

- ADC channel trigger point calculation and load into the A2 register (ALTCADR). Recalculation, when the timebase counter UC[23] rolls over, included.

```
// Channel ADC trigger point
// Recalculate Trigger ADC
chTriggerAdc = (vuint32_t)(ch[i].triggerAdc * chTriggerAdcStep);
// Load channel trigger ADC into the A2 register
if ((chTriggerAdc + chStartDelay) > counterPeriod)
{
        // Channel ADC trigger occurs after counter roll over
        EMIOS_0.CH[i].ALTCADR.R = chTriggerAdc + chStartDelay - counterPeriod;
}
else
{
        // Channel ADC trigger occurs before counter roll over
        EMIOS_0.CH[i].ALTCADR.R = chTriggerAdc + chStartDelay;
}
```

4.1.4 ADC Module

Code is present in the *adc.c* and *adc.h* files, which can be found in the workspace folder *drv*. The ADC module is running on a 32 MHz clock (64 MHz / 2). Selected ADC channels ANP[0], ANP[1], ANP[2] are running in normal conversion mode started on a CTU trigger signal.

```
// Init configuration registers, ADCclk = 64MHz/2 = 32MHz
// Disable power down
ADC_0.CLR2.R = 0;
// Disable external trigger, enable CTU
ADC_0.CLR0.R = 0x00000008;
// Right aligned data, One shot conversion mode
ADC_0.CLR3.R = 0;
// Init CLR1 and CLR4
ADC_0.CLR1.R = 0;
ADC_0.CLR4.R = 0;

// Reset NCMR registers
ADC_0.NCMR[0].R = 0x00000000;
ADC_0.NCMR[2].R = 0x00000000;
ADC_0.NCMR[4].R = 0x00000000;
ADC_0.NCMR[5].R = 0x00000000;
// Configure ANP[0] to normal mode
ADC_0.NCMR[0].B.CH0 = 1;
// Configure ANP[1] to normal mode
ADC_0.NCMR[0].B.CH1 = 1;
// Configure ANP[2] to normal mode
ADC_0.NCMR[0].B.CH2 = 1;
```

The conversion time can be calculated based on the application requirements.

```
// Conversion timing registers
// ANP[x], 1.126us @ 32MHz (Teval ~ 938ns, Tsample ~ 172ns)
ADC_0.CT[0].R = 0x00008C06;
// ANS[x], 1.126us @ 32MHz (Teval ~ 938ns, Tsample ~ 172ns)
ADC_0.CT[1].R = 0x00008C06;
// ANX[x], 1.126us @ 32MHz (Teval ~ 938ns, Tsample ~ 172ns)
ADC_0.CT[2].R = 0x00008C06;
```

4.1.5 CTU Module

Code is present in the *ctu.c* and *ctu.h* files, which can be found in the workspace folder *drv*. The *void Mcu_Ctu_Init(void)* function matches the eMIOS0 and ADC modules trigger channels.

```
// EMIOS[1] triggers ADC ANP[0]
CTUL.EVTCFGR[1].R = 0x00008000;
CTUL.EVTCFGR[1].B.CHANNELVALUE = 0;

// EMIOS[3] triggers ADC ANP[1]
CTUL.EVTCFGR[3].R = 0x00008000;
CTUL.EVTCFGR[3].B.CHANNELVALUE = 1;

// EMIOS[5] triggers ADC ANP[2]
CTUL.EVTCFGR[5].R = 0x00008000;
CTUL.EVTCFGR[5].B.CHANNELVALUE = 2;
```

Each trigger event has an assigned channel. This channel can generate a trigger signal to start the ADC conversion. For proper configuration it is requested to follow the rules mentioned in Section 2.2, “CTU — Cross-Triggering Unit“.

The ADC trigger proceeds when the eMIOS0 A2 register (ALTCADR) value matches the OPWMT channel module clock value.

4.2 Application Software

The Lighting Application configuration is finished when the on-board LED1 is turned ON and the MCU program counter reaches the while loop. Then it is periodically processed:

- Reading selected analog inputs, function *data0 = Mcu_Adc_0_Data_Prec(0)*, where the function argument number equals the ADC channel number (see Section 3.2, “ADC Conversion Start Using the CTU“).
- On-board LED2, LED3, LED4 control based on the *data0* variable value.
- Run-time PWM channel parameters control.
- Run-time PWM channel parameters configuration.

An application example demonstrates how to change the PWM channel parameters run-time:

- Change PWM parameters in structure.
- Call OPWMT init function: *Mcu_Emios_0_Channel_OPWMT_Init(vuint8_t i)*.

In addition, PWM channel timebase parameters can be changed run-time, by calling function *void Mcu_Emios_0_Channel_Timebase_Init (vuint8_t ch, vuint16_t ctr),* where:

- ch: 8-bit channel number
- ctr:16-bit roll-over value

5 Conclusion

The MPC5604B Lighting application example introduces the MCU eMIOS, CTU, and ADC modules usage within Lighting application, which is in this case well-suited for the lamp light bulbs control as well as for the lamp LEDs control with no major change required in peripheral's software.

The benefit of using the CTU module is significant comparing to solution when using the timer to trigger ADC at certain PWM channel cycle. The timer produces frequent interrupts which need to be processed by the MCU CPU, and therefore increase the CPU load and suspend run-time processes. Using the CTU gives the application designer a powerful tool to avoid diagnostic timer configuration challenges and makes the Lighting application simpler to design and test.

The ADC module gives the designer the opportunity of using watchdog threshold interrupts to save the MCU performance and enables fast responses in case of lamp light bulb failures. If required, the number of the ADC channels can be increased from 36 up to 64 using simple external logic circuitry fully controlled by the ADC module.

The eMIOS module offers up to 50 PWM channels suitable for Lighting application as well as for other PWM-based applications. The eMIOS module presents PWM channels with high resolution and flexibility.

The Lighting application example consists of an example description and complete software demo based on the xPC560BKIT144S EVB.

Acronyms

ADC	Analog-to-Digital Converter
ALTCADR	eMIOS UC A2 Register
CADR	eMIOS UC A1 Register
CBDR	eMIOS UC B2 Register
CTU	Cross-Triggering Unit
eMIOS	Configurable Enhanced Modular I/O Subsystem
EVB	Evaluation Board
LA	Lighting Application
MC	eMIOS Channel Modulus Counter Mode
MCU	Microcontroller Unit (MPC5604B)
OPWMT	eMIOS Channel Output PWM Mode with Trigger
PCR	Pad-Configuration Register
PWM	Pulse Width Modulation
SIUL	System Integration Unit Lite
UC	eMIOS Universal Counter Channel

References

1. www.freescale.com

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3836
Rev. 2
8/2009

freescale™
semiconductor