

F²MC-16FX FAMILY
16-BIT MICROCONTROLLER
MB96340

**LAMP CONTROL AND MONITOR
WITH PPG AND ADC**

APPLICATION NOTE

Revision History

Date	Issue
2007-03-28	V1.0, First release, MPi
2007-07-24	V1.1, Updated with re-review findings from PHu, MPi

This document contains 19 pages.

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (e.g. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect

Contents

REVISION HISTORY	2
WARRANTY AND DISCLAIMER	3
CONTENTS	4
1 INTRODUCTION	5
1.1 Connection Diagram.....	5
2 OPERATION	6
2.1 Reload Timer.....	7
2.2 PPG	8
2.3 ADC and DMA.....	8
3 EXAMPLE CODE	10
3.1 Initialization Functions	10
3.1.1 Flowchart.....	10
3.1.2 C Code	11
3.2 Main Function.....	13
3.2.1 Flowchart.....	13
3.2.2 C Code	14
3.3 Interrupt Service Routines	15
3.3.1 Flowchart.....	15
3.3.2 C Code	17
3.4 Interrupt Vector	18
3.4.1 Code.....	18
4 ADDITIONAL INFORMATION	19

1 Introduction

This application note describes how to control and monitor the lights used in a dashboard or a HVAC system with the usage of PPG and ADC.

In such automotive applications the light intensity is controlled by the PPG outputs using the PWM technique. However, it is also essential to monitor whether a particular light is on and functioning properly. This is achieved by measuring the voltage across the lamp with an ADC.

1.1 Connection Diagram

The following figure shows the connection diagram. The Programmable Pulse Generator (PPG) output is connected to one terminal of the light/lamp and the other terminal is pulled down via a resistor and also connected to the Analog-to-Digital Converter (ADC) input.

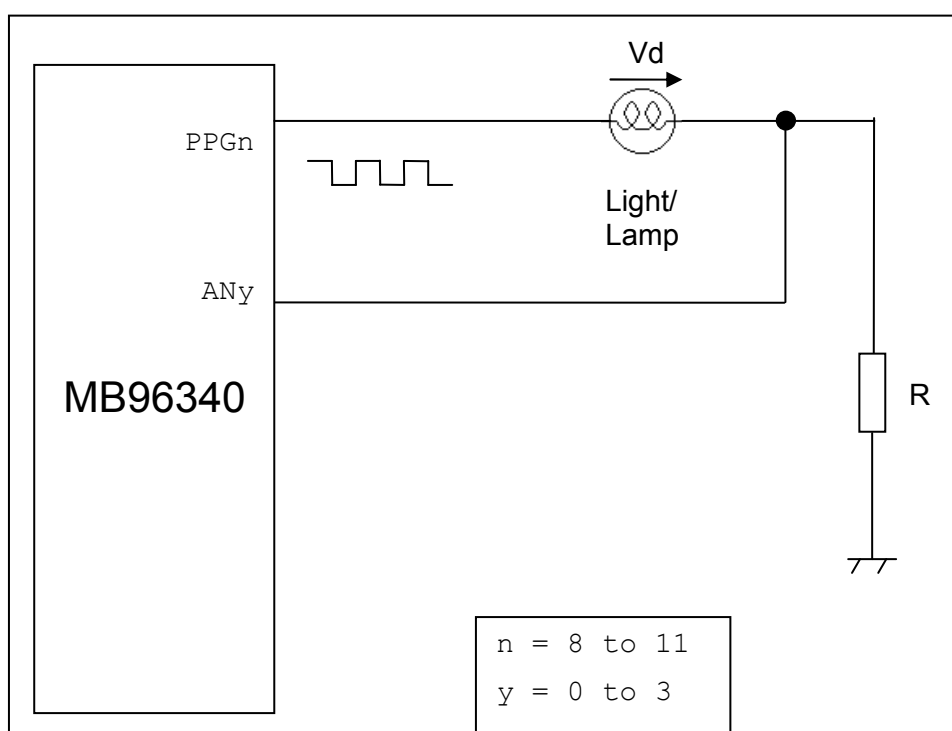


Figure 1-1: Connection Diagram - Lamp Control and Monitor

In the above diagram the PPG8 output corresponds to the AN0 input, PPG9 output corresponds to the AN1 input, so on and so forth. For the ease of understanding just one pair of PPG output and ADC input is shown.

For PPG outputs, PWM frequency of 250 Hz is used. This frequency is chosen so that it is high enough for the flicker in the lamp to be invisible and it's also low enough as not to introduce EMI noise.

It should be noted that the PPG output and the ADC input may not be able to connect directly to the lamp as shown in the above figure since the power requirements of the lamp may not be compatible with that of the micro. In such scenario, there might be the need of some driver or signal conditioning circuitry.

2 Operation

The below block diagram depicts the peripheral operation and dependency.

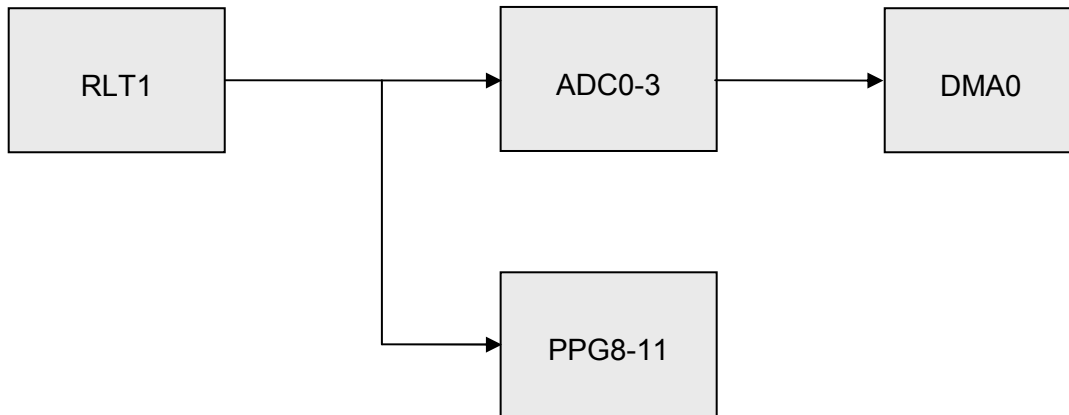


Figure 2-1: Block Diagram - Peripheral Operation and Dependency

2.1 Reload Timer

The Reload Timer 1 (RLT1) is used as the time base for triggering the ADC conversion as well as starting the PPGs. This means the ADC conversion on channels AN0-3 are triggered by RLT1 internal output.

RLT1 is required to issue an interrupt at an interval of 0.5 ms at 16 MHz CLKP1. Hence the RLT1 prescaler is configured to divide the CLKP1 by 16 and the reload register of RLT1 is loaded with 499. This is because the underflow would happen after value loaded with reload register + 1 cycle, i.e. after 500 clock cycles of RLT1. The following formula explains the same:

$$\text{Underflow Interrupt Interval} = \frac{1}{16\text{MHz}/16} \cdot (499 + 1) = 0.5\text{ms}$$

Hence the each ADC channel will start conversion after every 1 ms (after every complete RLT1 cycle that is after 2 underflows).

In the RLT1 interrupt service routine (ISR) the PPG8-11 are started after every 1 ms (i.e. after 2 interrupts). The PPG8 is started at the very first interrupt; PPG9 is started at the third interrupt, so on and so forth. This ensures the phase shift of $\frac{1}{4}$ (1 ms) of the total PPG pulse width (4 ms). It should be noted that the RLT1 ISR is only used during initial setup of PPGs and it is not required for normal operation.

The below figure describes the timing behaviour described above.

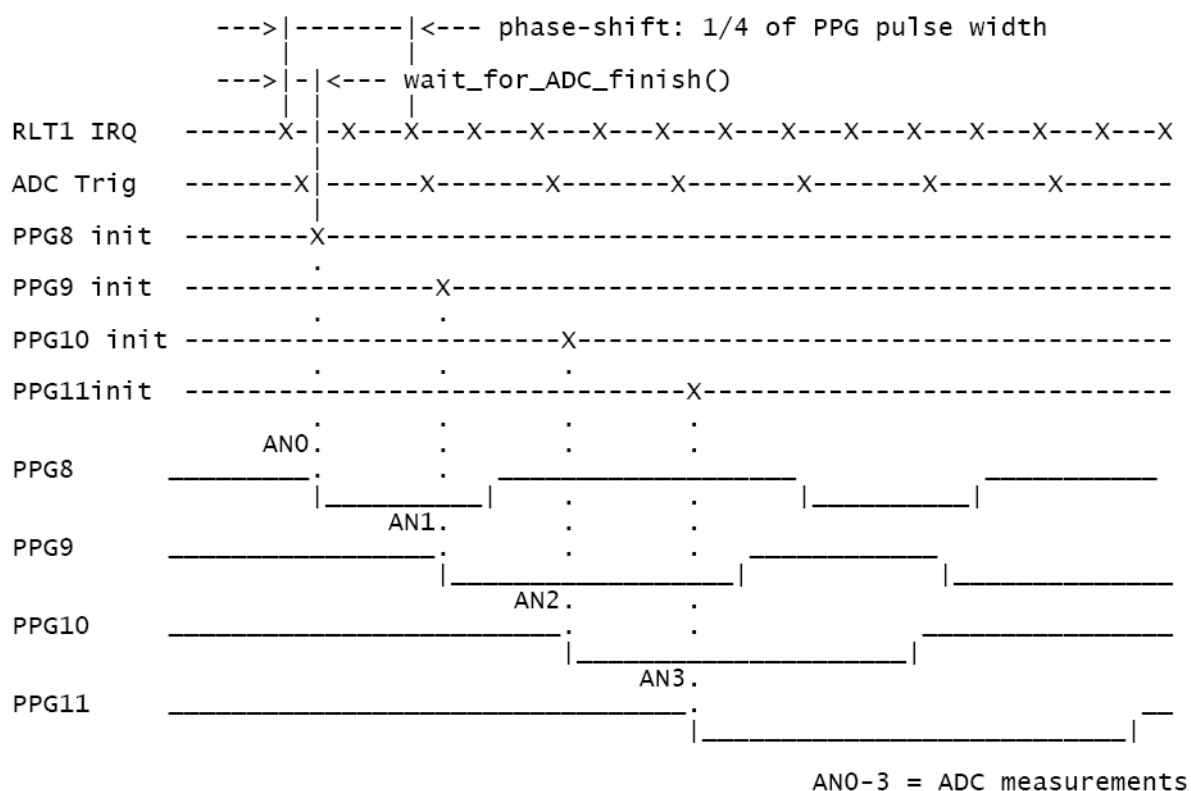


Figure 2-2: Timing Diagram

2.2 PPG

All the PPGs (PPG8-11) are fed by 16 MHz of CLKP1 clock with prescaler of 4, hence the clock of 1 MHz. In the RLT1 ISR the PPGs are started as below:

- The PPG8 is started at the first interrupt with period of 4 ms and a duty of 75%.
- The PPG9 is started at the third interrupt with period of 4 ms and a duty of 62.5%.
- The PPG10 is started at the fifth interrupt with period of 4 ms and a duty of 50%.
- The PPG11 is started at the seventh interrupt with period of 4 ms and a duty of 37.5%.

The actual period is the value loaded in period register (PCSR) +1 and the actual duty is the value loaded in the duty register (PDUT) + 1. Hence for the period of 4 ms and a duty of 75% the value required to be loaded in PCSR8 is 15999 and the value required to be loaded in PDUT8 is 11999. Period and duty register of the remaining PPGs are configured considering the same.

2.3 ADC and DMA

The ADC is used in STOP mode with 10 bit conversion resolution. Channel 0 to 3 will be converted one after another at every trigger from RLT1 i.e. after every 1 ms. The sampling time is configured as 8 cycles¹ (this is only valid if the supply voltage is between range - 4.5V ≤ AV_{CC} ≤ 5.5V) where as the compare time is configured as 22 cycles². Hence at 16 MHz of CLKP1, the total time taken for each channel conversion would be 1.875 μs. The conversion will go on in cyclic manner (i.e. channel 0-1-2-3-0-1-2-3).

In the RLT1 ISR, there is a wait loop of about 6 μs before starting each of the PPGs. This wait time should be at least equal to the conversion time ADC (1.875 μs, in this case). This makes sure that the PPG output is ALWAYS HIGH when the ADC conversion happens.

The following logic can be applied to determine the status of the lamp:

- If the lamp is operational, then the ADC would also see the voltage equivalent to AV_{CC} – V_d (voltage drop across the lamp).
- If the lamp is non-operational (open), then the ADC would see voltage equivalent to V_{SS} (ground potential).
- If the lamp is non-operational (short), then the ADC would see voltage equivalent to V_{CC} (no voltage drop across the lamp).

In order to accommodate the tolerance in the supply voltage and the voltage drop across the lamp, the ADC equivalent of lamp voltage is not compared with the above mentioned fixed voltages but range of voltages as below:

Sr. No	Condition	Voltage Across the Lamp in Ideal Condition	ADC Equivalent of Compared Voltage Range	
			Minimum	Maximum
1	Lamp Operational	AV _{CC} – V _d	AV _{CC} V _d MIN	AV _{CC} V _d MAX
2	Lamp Non-operational : Open	V _{SS}	AV _{SS} CNT	AV _{SS} MAX
3	Lamp Non-operational : Short	V _{CC}	AV _{CC} MIN	AV _{CC} CNT

¹ The cycle setting for the Sample Time depends on the driving impedance and the supply voltage AV_{CC}.

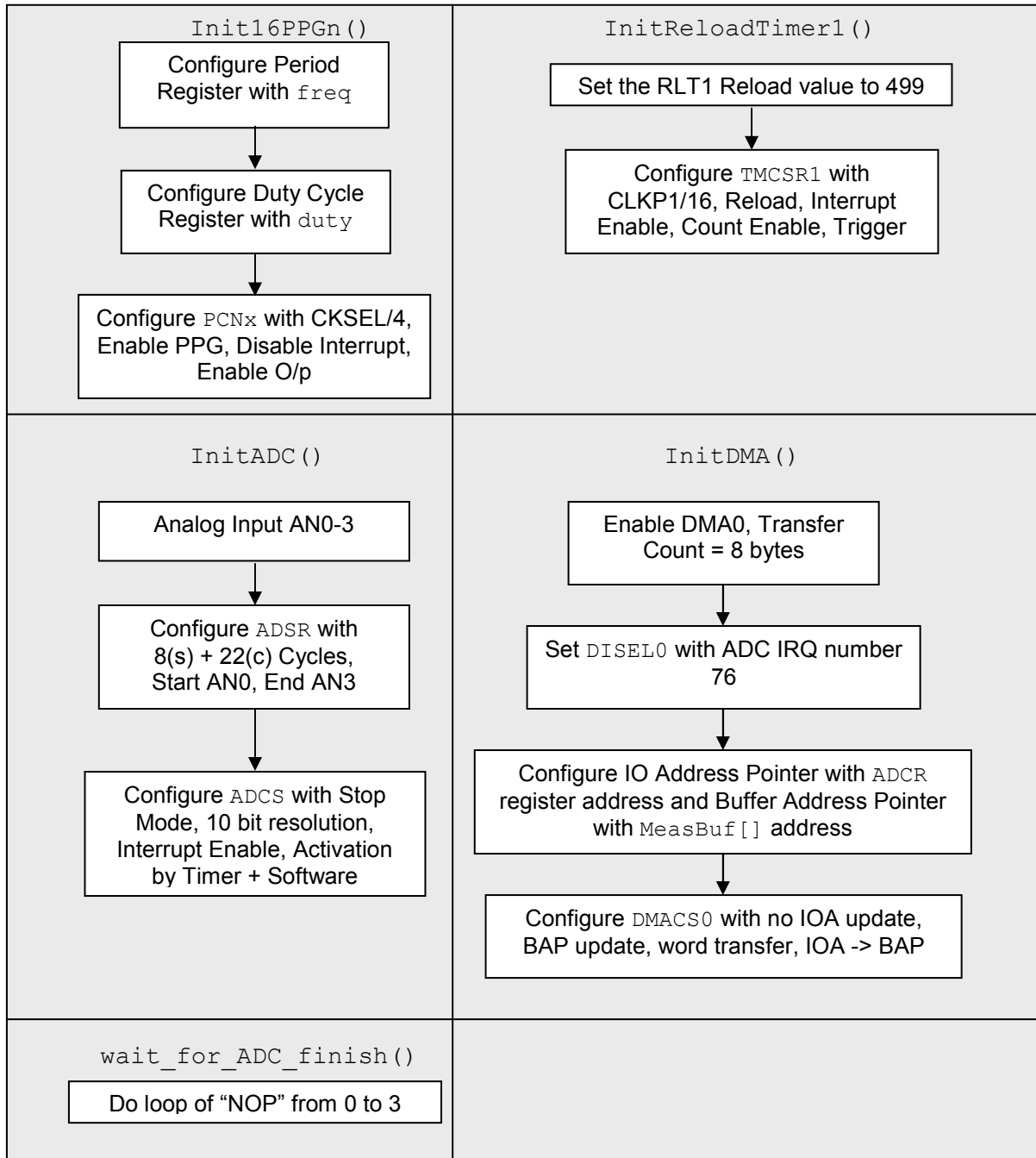
² The cycle setting for Compare Time depends on the supply voltage AV_{CC}.

After every conversion the DMA transfers the converted ADC value to `MeasBuf[]` array. After 4 such transfers the ADC ISR would be executed, which re-initializes the DMA. In the main routine the ADC data can be analyzed with the above discussed logic to monitor the lamp. Nonetheless, the logic discussed above is just an example. In an application this logic may not be used as is and may need enhancements.

3 Example Code

3.1 Initialization Functions

3.1.1 Flowchart



3.1.2 C Code

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/
char InitState; // State Flag for sequential PPG init and ADC init
volatile int MeasBuf[4]; // ADC Measurement Buffer
int Results[4]; // Result Buffer: Takes data from (MeasBuf[] & 0x3FF)

unsigned char LampStat[4]; // Status of the Lamp

/*-----*/
/* Initialize PPGs */
/*-----*/

void Init16PPG8 (unsigned int freq, unsigned int duty)
{
    PCSR8 = freq; // always set cycle value PERIOD 1st
    PDUT8 = duty; // set duty value DUTY CYCLE
    PCN8 = 0xD402; // clk/4, enable PPG8, disable interrupts, enable output
}

void Init16PPG9 (unsigned int freq, unsigned int duty)
{
    PCSR9 = freq; // always set cycle value PERIOD 1st
    PDUT9 = duty; // set duty value DUTY CYCLE
    PCN9 = 0xD402; // clk/4, enable PPG9, disable interrupts, enable output
}

void Init16PPG10 (unsigned int freq, unsigned int duty)
{
    PCSR10 = freq; // always set cycle value PERIOD 1st
    PDUT10 = duty; // set duty value DUTY CYCLE
    PCN10 = 0xD402; // clk/4, enable PPG10, disable interrupts, enable output
}

void Init16PPG11 (unsigned int freq, unsigned int duty)
{
    PCSR11 = freq; // always set cycle value PERIOD 1st
    PDUT11 = duty; // set duty value DUTY CYCLE
    PCN11 = 0xD402; // clk/4, enable PPG11, disable interrupts, enable output
}

/*-----*/
/* Initialize RLT1 */
/*-----*/

void InitReloadTimer1 (void)
{
    TMRLR1 = 499; // set reload value 499
                // One Cycle: 500 ticks * 1 µs = 0.5 ms
    TMCSR1 = 0x041B; // prescaler CLKP1/2^4, reload, interrupt enable, count enable,
                // trigger
}

/*-----*/
/* Initialize ADC */
/*-----*/

void InitADC (void)
{
    ADER0 = 0x0F; // ADC Input 0-3

    ADSR = 0x4003; // 8 cyc. sample, 22 cyc. conversion, Start AN0, End AN3
    ADCSL = 0xC0; // Stop Mode, 10 Bit
    ADCSH = 0xA8; // Interrupts, Activation by Timer and Software
}

```

```
/*-----*/
/*              (Re-)Initialize DMA              */
/*-----*/

void InitDMA (void)
{
    DER = 0x0001;    // DMA 0 enable

    DCTH0 = 0;      // 8 Bytes = 4 words
    DCTL0 = 8;
    DISEL0 = 76;    // ADC irq number
    IOAH0 = (unsigned char) &ADCR >> 8; // I/O Bank 00
    IOAL0 = (unsigned char) &ADCR & 0xFF;
    DMACS0 = 0x18;  // no IOA update, BAP update, word transfer, IOA -> BAP
    BAPH0 = (__far unsigned long) &MeasBuf[0] >> 16;
    BAPM0 = (__far unsigned long) &MeasBuf[0] >> 8;
    BAPL0 = (__far unsigned long) &MeasBuf[0] & 0xFF;
}

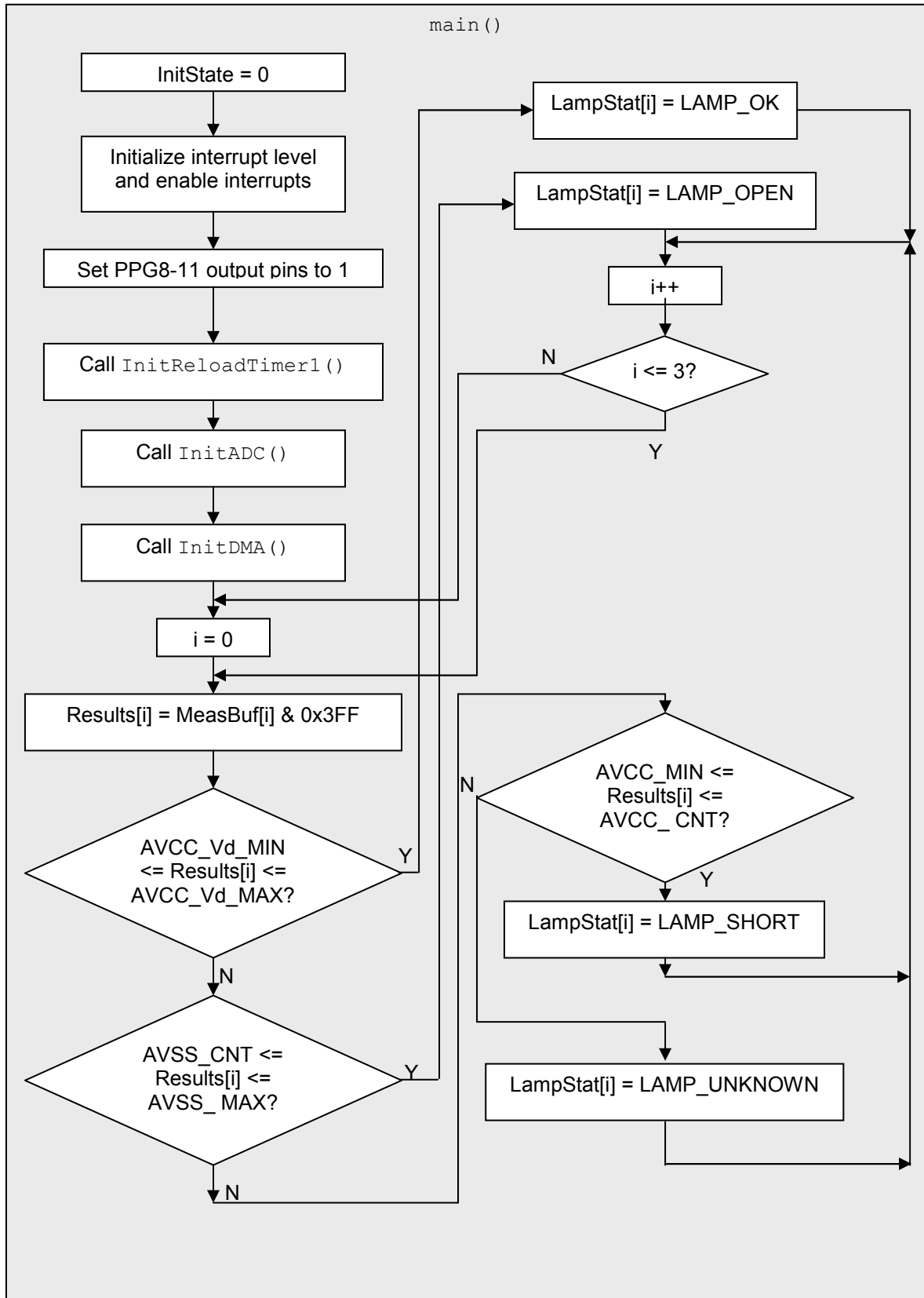
/*-----*/
/*              Wait delay              */
/*-----*/

void wait_for_ADC_finish(void) // inclusive CALLP and RETP about 6 us
{
    volatile unsigned char i;

    for (i = 0; i < 4; i++)
        __wait_nop();
}
```

3.2 Main Function

3.2.1 Flowchart



3.2.2 C Code

The configurable parameters can be edited by the user depending upon the application requirement. It should be noted that the lamp voltage is compared to range of voltage rather than a fixed value so as to accommodate the tolerance in the supply voltage and the voltage drop across the lamp.

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/
// Configurable Parameters
#define AVCC          5          // ADC Supply Voltage in volts
#define AVSS          0          // ADC Ground Voltage in volts
#define Vd            0.1       // voltage drop across lamp in volts
#define Margin        2          // Tolerance in voltage in terms of percentage

// Derived Parameters
#define AVCC_CNT      (1024)     // ADC Count equi. to Vcc at 10 bit resolution
#define AVSS_CNT      ((AVSS/AVCC)*1024) // ADC Count equi. to Vss at 10 bit resolution
#define Vd_CNT        ((Vd/AVCC)*1024) // ADC Count equi. to Vd at 10 bit resolution
#define AVCC_MIN      (AVCC_CNT - ((AVCC_CNT*Margin)/100))
#define AVSS_MAX      (AVSS_CNT + ((AVSS_CNT*Margin)/100))
#define AVCC_Vd       (AVCC_CNT - Vd_CNT)
#define AVCC_Vd_MAX   (AVCC_Vd + ((AVCC_Vd*Margin)/100))
#define AVCC_Vd_MIN   (AVCC_Vd - ((AVCC_Vd*Margin)/100))
#define LAMP_DEFAULT  0
#define LAMP_OK        1
#define LAMP_OPEN     2
#define LAMP_SHORT    3
#define LAMP_UNKNOWN  4

char InitState;           // State Flag for sequential PPG init and ADC init
int MeasBuf[4];          // ADC Measurement Buffer
int Results[4];          // Result Buffer: Takes data from (MeasBuf[] & 0x3FF)
unsigned char LampStat[4]; // Status of the Lamp

void main(void)
{
    unsigned int i;
    InitState = 0;
    InitIrqLevels();
    __set_il(7);          // allow all levels
    __EI();               // globally enable interrupts

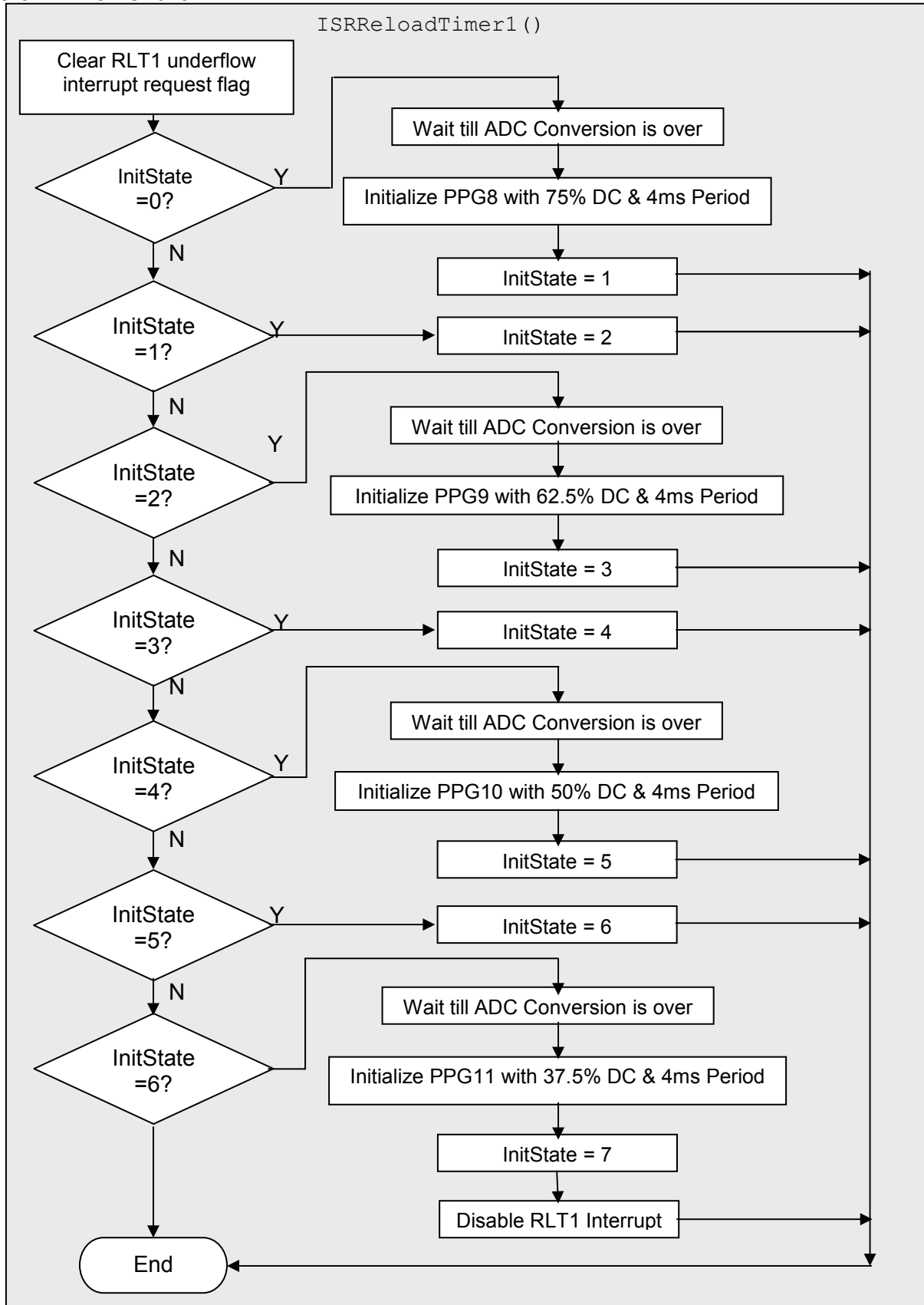
    PDR09 = 0x0F;        // Set PPG11-8 output pins to "1" for 1st ADC measurement
    DDR09 = 0x0F;        // -> output "1"
    InitReloadTimer1();  // Start RLTL/ADC and PPG sequence
    InitADC();           // Init ADC
    InitDMA();           // Init ADC-DMA transfer

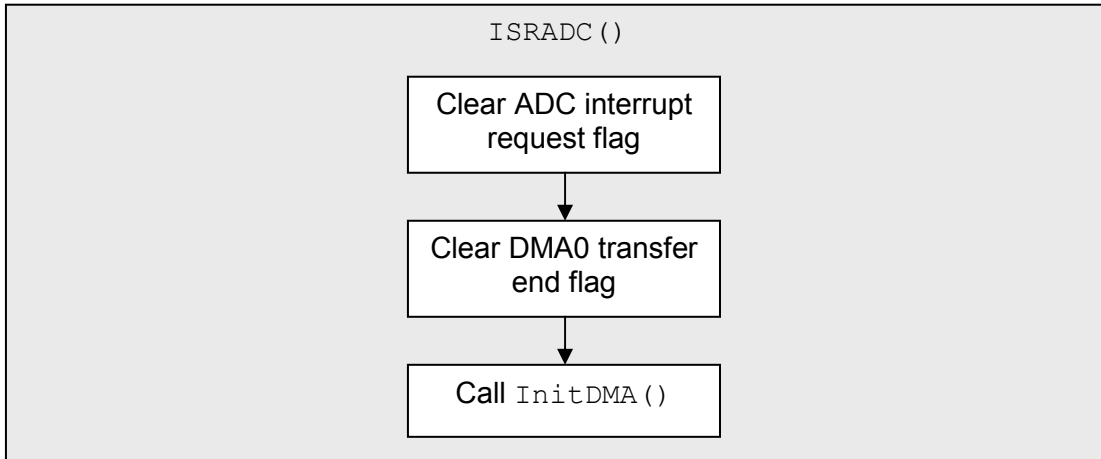
    while(1)
    {
        for (i=0; i<=3 ; i++)
        {
            // Mask results from buffer and update array
            Results[i] = MeasBuf[i] & 0x3FF;
            //determine the lamp status
            if (Results [i] >= AVCC_Vd_MIN && Results [i] <= AVCC_Vd_MAX)
                LampStat[i] = LAMP_OK;
            else if (Results [i] >= AVSS_CNT && Results [i] <= AVSS_MAX)
                LampStat[i] = LAMP_OPEN;
            else if (Results [i] >= AVCC_MIN && Results [i] <= AVCC_CNT)
                LampStat[i] = LAMP_SHORT;
            else
                LampStat[i] = LAMP_UNKNOWN;
        }
    }
}

```

3.3 Interrupt Service Routines

3.3.1 Flowchart





3.3.2 C Code

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/
/* Reload 1 Timer ISR */
/*-----*/

__interrupt void ISRReloadTimer1(void)
{
    TMC SR1_UF = 0; // reset underflow interrupt request flag

    // Sequential Initialization of PPG11 - PPG8
    switch (InitState)
    {
        case 0:
            wait_for_ADC_finish(); // wait for about 6us
            Init16PPG8(15999, 11999); // 1 ms + 3 ms = 4 ms
            InitState = 1;
            break;

        case 1:
            InitState = 2;
            break;

        case 2:
            wait_for_ADC_finish(); // wait for about 6us
            Init16PPG9(15999, 9999); // 1.5 ms + 2.5 ms = 4 ms
            InitState = 3;
            break;

        case 3:
            InitState = 4;
            break;

        case 4:
            wait_for_ADC_finish(); // wait for about 6us
            Init16PPG10(15999, 7999); // 2 ms + 2 ms = 4 ms
            InitState = 5;
            break;

        case 5:
            InitState = 6;
            break;

        case 6:
            wait_for_ADC_finish(); // wait for about 6us
            Init16PPG11(15999, 5999); // 2.5 ms + 1.5 ms = 4 ms
            InitState = 7;
            TMC SR1 = 0x0412; // disable RLT1 interrupt (ADC/PPG configuration done)
            break;

        case default:
            break;
    }
}

/*-----*/
/* ADC ISR */
/*-----*/

__interrupt void ISRADC(void)
{
    ADCSH = 0xA8; // clear INT
    DSR = 0x0000; // Clear DMA request

    InitDMA(); // Reinitialize DMA
}

```

3.4 Interrupt Vector

3.4.1 Code

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/
ICR = (52 << 8) | 2; /* Priority Level 2 for RLT1 of MB9634x Series */
ICR = (76 << 8) | 2; /* Priority Level 2 for ADC of MB9634x Series */

. . .

/* ISR prototype */
__interrupt void ISRReloadTimer1(void);
__interrupt void ISRADC (void);

. . .
#pragma intvect ISRReloadTimer1 52 /* RLT1 of MB9634x Series */
#pragma intvect ISR_ADC 76 /* ADC of MB9634x Series */
. . .
```

4 Additional Information

Information about FUJITSU Microcontrollers can be found on the following Internet page:

<http://mcu.emea.fujitsu.com/>

The software example related to this application note is:

96340_ppg_rt_adc_dma

It can be found on the following Internet page:

http://mcu.emea.fujitsu.com/mcu_product/mcu_all_software.htm