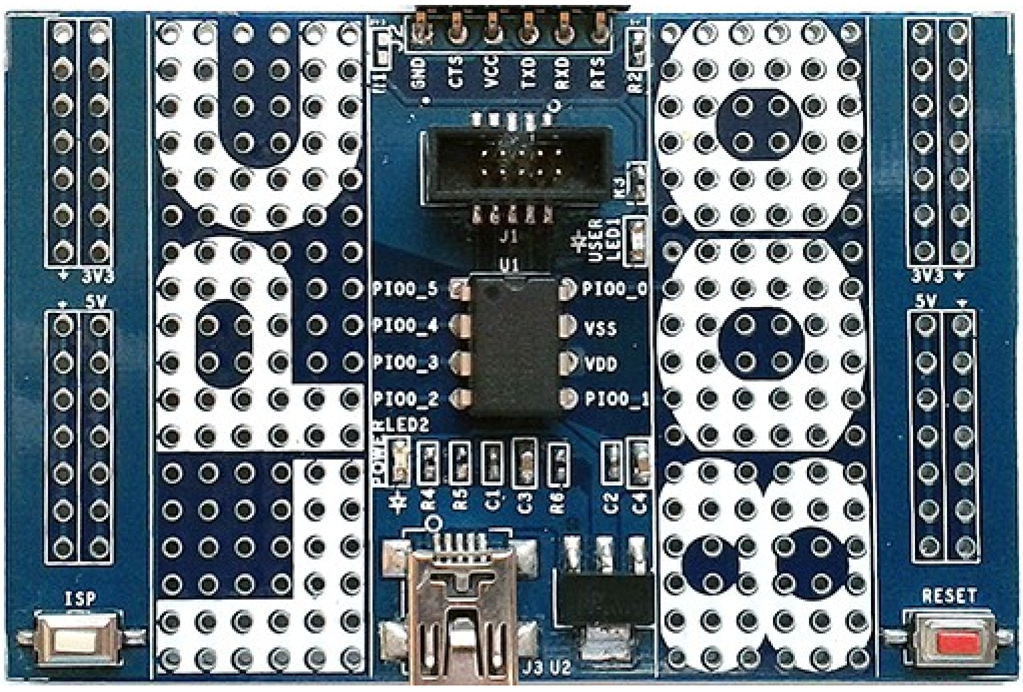
Taking the LPC800 into the delta quadrant :)

Background

One of our boys received a Revell model of the starship Voyager (from Star Trek) as a gift. It sat on his shelf for a least a year crying out for someone to build it.

Then along came a free LPC800 mini kit in the post.



I wondered what I could do with this given the limited number of pins for I/O and then remembered the Voyager kit. A plan began to emerge...

Spacedock

Ours is principally a Linux house in that all computers run Linux by default. This can present problems sometimes when vendors fail to supply a Linux version of their Windows application. “Flash Magic” is a case in point. This program allows you download firmware to NXP processors using their built-in serial boot loader. There is no Linux version of this program however it will run fine under the “wine” utility in Linux. We were all set now for firmware download.

We downloaded sample code from http://lpcware.com/lpc800-mini-kit (the LPC810 Code base) to give us some idea of the development process for this particular kit. As it happens there was also a recent version of LPCXPresso on our home computer. A little testing showed that our “toolchain” was working and we were able to download the sample application onto the board. The next step was to customize the application to suit Voyager.

We decided on the following pin configuration:

PIO0_5 = RESET
PIO0_4 = U0_TXD
PIO0_3 = GPIO – Wired to port and starboard running lights.
PIO0_2 = GPIO – Wired to main deflector dish light
PIO0_1 = GPIO – Wired to red lights on the front of nacelles
PIO0_0 = U0_RXD

This leaves TX and RX for future development. In addition to the above, the sides of the nacelles were to be illuminated by LED's hardwired to the power supply.

The lighting scheme is as follows:
The running lights (red for port, green for starboard) should toggle every second
The main dish should pulsate slowly
The red lights on the front of the nacelles (warp engines?) should pulsate quickly.

It was decided to use software pulse width modulation (PWM) to achieve the pulsating light effects. The demo code used the multi-rate timer running at a slower rate than we needed. We increased its interrupt rate to 10kHz which gave us plenty of resolution for our software PWM.

The software PWM operates as follows:
At each interrupt, increment a counter (PWM_Counter).
If the count reaches 100, reset it to zero and turn on all output PWM channels.
In subsequent interrupts, examine the counter and, if it exceeds the Duty cycle (percent) for a PWM controlled output then turn off that output. We have two PWM outputs and the Duty Cycles for these are stored in PWM_Match0 and PWM_Match1. Using this mechanism we can control the average brightness of the LED's behind the main deflector dish and the warp engines. In order to achieve a pulsating effect on these LED's we raise and lower the brightness of these LED's periodically. This is done in the main program loop.

Operation

You can view a short video clip of Voyager here:

http://youtu.be/mCVECk-OUck

As you can see, the lights all work. There is some crude looking wiring near the back and we didn't do such a great decorating job but hey, it all works. The device is powered using a trailing USB cable that emerges from the shuttle craft launch door.

The code

The code is a slightly modified version of that found in the LPC810 code base.
Modified version of mrt.c

```
/**************************************************************************/
/*!
    @file     mrt.c
    @author   K. Townsend
    @brief    Multi-rate timer (mrt) helper functions

    @section LICENSE

    Software License Agreement (BSD License)

    Copyright (c) 2013, K. Townsend (microBuilder.eu)
    All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are met:
    1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
    3. Neither the name of the copyright holders nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ''AS IS'' AND ANY
    EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
    DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
    ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
/**************************************************************************/
#include "LPC8xx.h"
#include "mrt.h"

volatile uint32_t mrt_counter = 0;
int RunningLightCounter=0;
int RunningLightsOn=0;
int PWM_Counter=0;
int PWM_Match0=50;
int PWM_Match1=0;

void MRT_IRQHandler(void)
{
       // the interrupt rate here should be approx. 10kHz
  if ( LPC_MRT->Channel[0].STAT & MRT_STAT_IRQ_FLAG )
  {

    LPC_MRT->Channel[0].STAT = MRT_STAT_IRQ_FLAG;      /* clear interrupt flag */
    mrt_counter++;
    // Going to extend the functioning of this isr to facilitate software PWM.
    PWM_Counter++;
    if (PWM_Counter==100)
    {
       PWM_Counter=0;
       // reset the counter and set all associated outputs high
       LPC_GPIO_PORT->SET0 = 1<<2;
       LPC_GPIO_PORT->SET0 = 1<<1;
    }
    if (PWM_Counter>=PWM_Match0)
    {
       // clear channel output (main dish)
       LPC_GPIO_PORT->CLR0 = 1<<2;
    }
    if (PWM_Counter>=PWM_Match1)
    {
       // clear channel output (warp engines)
       LPC_GPIO_PORT->CLR0 = 1<<1;
    }
    if (RunningLightCounter++>=10000)
    {
       RunningLightsOn = ~RunningLightsOn; // toggle running lights
       RunningLightCounter = 0;
       if (RunningLightsOn)
              LPC_GPIO_PORT->CLR0 = 1<<3;
       else
              LPC_GPIO_PORT->SET0 = 1<<3;
    }
  }
  return;
}
```

Modified version of main.c

```
/**************************************************************************/
/*!
    @file     main.c

    @section LICENSE

    Software License Agreement (BSD License)

    Copyright (c) 2013, K. Townsend (microBuilder.eu)
    All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are met:
    1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
    3. Neither the name of the copyright holders nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ''AS IS'' AND ANY
    EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
    DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
    ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
/**************************************************************************/

/* Modifications for Voyager spacecraft model
 * PIO0_1 : assigned to red lights at end of Nacelles
 * PIO0_2 : assigned to reflector dish
 * PIO0_3 : assigned to port and starboard running lights
 */
#include <stdio.h>
#include "LPC8xx.h"
#include "gpio.h"
#include "mrt.h"
```

```
#include "uart.h"

#if defined(__CODE_RED)
  #include <cr_section_macros.h>
  #include <NXP/crp.h>
  __CRP const unsigned int CRP_WORD = CRP_NO_CRP ;
#endif

#define LED_LOCATION    (2)

/* This define should be enabled if you want to      */
/* maintain an SWD/debug connection to the LPC810,   */
/* but it will prevent you from having access to the */
/* LED on the LPC810 Mini Board, which is on the     */
/* SWDIO pin (PIO0_2).                               */
// #define USE_SWD

void configurePins()
{
  /* Enable SWM clock */
  LPC_SYSCON->SYSAHBCLKCTRL |= (1 << 7);

  /* Pin Assign 8 bit Configuration */
  /* U0_TXD */
  /* U0_RXD */
  LPC_SWM->PINASSIGN0 = 0xffff0004UL;

  /* Pin Assign 1 bit Configuration */
  #if !defined(USE_SWD)
    /* Pin setup generated via Switch Matrix Tool
       ------------------------------------------------
       PIO0_5 = RESET
       PIO0_4 = U0_TXD
       PIO0_3 = GPIO            - Disables SWDCLK
       PIO0_2 = GPIO (User LED) - Disables SWDIO
       PIO0_1 = GPIO
       PIO0_0 = U0_RXD
       ------------------------------------------------
       NOTE: SWD is disabled to free GPIO pins!
       ------------------------------------------------ */
    LPC_SWM->PINENABLE0 = 0xffffffbfUL;
  #else
    /* Pin setup generated via Switch Matrix Tool
       ------------------------------------------------
       PIO0_5 = RESET
       PIO0_4 = U0_TXD
       PIO0_3 = SWDCLK
       PIO0_2 = SWDIO
       PIO0_1 = GPIO
       PIO0_0 = U0_RXD
       ------------------------------------------------
       NOTE: LED on PIO0_2 unavailable due to SWDIO!
       ------------------------------------------------ */
    LPC_SWM->PINENABLE0 = 0xffffffb3UL;
  #endif
}
extern int PWM_Match0;
extern int PWM_Match1;
int MainDishStep=1;
int NacelleStep=10;
int main(void)
{
  /* Configure the core clock/PLL via CMSIS */
  SystemCoreClockUpdate();

  /* Initialise the GPIO block */
  gpioInit();

  /* Initialise the UART0 block for printf output */
  uart0Init(115200);

  /* Configure the multi-rate timer for 100 microsecond ticks */
  mrtInit(SystemCoreClock/10000);

  /* Configure the switch matrix (setup pins for UART0 and GPIO) */
  configurePins();

  /* Set the LED pins to output (1 = output, 0 = input) */
  #if !defined(USE_SWD)
    LPC_GPIO_PORT->DIR0 |= (1 << 1);
    LPC_GPIO_PORT->DIR0 |= (1 << 2);
    LPC_GPIO_PORT->DIR0 |= (1 << 3);
  #endif

  while(1)
  {
    #if !defined(USE_SWD)

      mrtDelay(100);

      // Manage pulsing of main reflector dish
      PWM_Match0+=MainDishStep;
      if ( (PWM_Match0>99) || (PWM_Match0 < 1))
         MainDishStep=-MainDishStep;
      // Manage pulsing of red light at end of nacelles dish
      PWM_Match1+=NacelleStep;
      if ( (PWM_Match1>90) || (PWM_Match1 < 10))
         NacelleStep=-NacelleStep;

    #else
      /* Just insert a 1 second delay */
      mrtDelay(100);
    #endif

    /* Send some text (printf is redirected to UART0) */
    printf("Space, the final frontier..\n");
  }
}
```