Outline
- Input/Output
- Interrupts
- Timers

References
- Mod5270 PinIO Class Application Note
- Mod5270 Interrupts Application Note
- Mod5270 DMA Timers Application Note
- MCF5270 Reference Manual
Netburner Hardware
Input/Output

- The Netburner module brings out 100 pins from the processor to the two 50 pin connectors J1 and J2.

- Many of the signal pins have multiple functions (interrupt, uart, I/O, etc). 46 of the pins can be used for general purpose I/O (GPIO). The GPIO pins are grouped in 8 bit ports (not all ports use all 8 bits). There are 12 ports (ADDR, DATAL, DATAH, etc.)
Netburner Hardware
Input/Output
Netburner Hardware
Input/Output

PORT BUSCTL

- OE / BUSCTL[7]
- TA / BUSCTL[6]
- R/W / BUSCTL[4]
- TIP / DREQ[0] / BUSCTL[0]

PORT BS


PORT CS


PORT SDRAM

- SD_WE / PSDRAM[5]
- SD_SCS / PSDRAM[4]
- SD_SRAS / PSDRAM[3]
- SD_SCKE / PSDRAM[2]
- SD_CS[1:0] / PSDRAM[1:0]

PORT QSPI

- QSPI_PCS1 / SD_SCKE / PQSPI[4]
- QSPI_PCS0 / PQSPI[3]
- QSPI_SCK / I2C_SCL / PQSPI[2]
- QSPI_DIN / I2C_SDA / PQSPI[1]
- QSPI_DOUT / PQSPI[0]

PORT TIMER

- T3IN / U2CTS / QSPI_PCS2 / PTIMER[7]
- T1IN / DREQ[1] / T1OUT / PTIMER[3]
- T0IN / DREQ[0] / PTIMER[1]
- T0OUT / DACK[0] / PTIMER[0]
Netburner Hardware
Input/Output

- To use a pin for GPIO you must configure the corresponding port correctly. This typically involves configuring 1) the port assignment register (to select the pin function), and 2) the port data direction register (to set up a GPIO pin for input or output). We set/clear a pin through either the port output data register, the port set data register or the port clear data register. (There are 12 sets of these registers corresponding to the 12 ports.)
To configure pin J2-44 as an output GPIO pin and then drive it high, you could use code similar to this:

```c
sim.gpio.par_uart &= ~0x1000; // Configure pin J244 for GPIO
sim.gpio.pddr_uarth |= 0x02; // Set signal direction as output
sim.gpio.ppdsdr_uarth = 0x02; // Set bit to be driven out on pin
```
Netburner Hardware
Input/Output

• Alternatively, Netburner provides the PinIO class which allows us to accomplish the previous task using:

```cpp
#include <pins.h>
// Configure pin J2-44 for GPIO
J2[44].function( PINJ2_44_GPIO );
J2[44] = 1; // Set pin as output high
```

• A pin's function is set using the `function()` method in the pin class. You can set a pin HI or LO by setting it to 1 or 0.
Netburner Hardware
Input/Output

• The functions of many of the pins can be set on a pin-by-pin basis, but the PDATA port pins (there are 16 of them) must all be set to GPIO or none of them can be. (The LEDs and switches on the carrier board are connected to the PDATA port pins.)

• You can set all PDATA pins to GPIO by setting any one of the pins to GPIO:

  J2[15].function(PINJ2_PD_GPIO);
Netburner Hardware
Input/Output

• In addition to the `function()` method the PinIO class provides the `set()` and `clr()` methods to set an output pin high or low. (Note: `J2[44] = 1` is equivalent to `J2[44].set()` and `J2[44] = 0` is equivalent to `J2[44].clr()`).

• The `read()` method will return a 1 or 0 indicating the state of an input pin. (Assignment is overloaded so that `s = J2[44]` is equivalent to `s = J2[44].read()`).
The `hiz()` method forces a pin into a high impedance state (disconnected) while the `drive()` method makes the output pin active.

Refer to the MOD-DEV-70-SCH-R1p7.pdf schematic (page 2) to determine to which connector pins the LEDs and switches are connected. This document is in the `docs/platform/schematics` folder.
Netburner Hardware Interrupts

- The MCF5270 processor has a very flexible interrupt system with 41 different interrupt sources. 7 levels (IRQ1 – IRQ7) of interrupt are supported with level 1 being the lowest and level 7 the highest. Level 7 is a nonmaskable interrupt and is not typically used with uC/OS applications.

- Priorities (0-8) may be assigned at each interrupt level to prioritize interrupts at the same IRQ level.
Netburner Hardware
Interrupts

• Configuring interrupts for the Coldfire processor is similar to configuring interrupts for other microcontrollers. The Netburner API provides a few helper functions, but to use interrupts effectively you need to read Chapter 13 of the MCF5271 manual.

• There are two general steps for working with interrupts. First configure the interrupt source (timer, serial I/O, etc) and then configure the interrupt controller.
Netburner Hardware
Interrupts

- To configure the interrupt controller you must write an interrupt service routine (ISR). You can use any uC/OS routines that do not block (POST routines generally) in the ISR. Netburner provides the `INTERRUPT()` macro to simplify writing of uC/OS compatible ISRs.

- Before an interrupt occurs you must set up the interrupt vector, level and priority. Netburner provides the `SetIntc()` function to make this easier.
Netburner Hardware
Timers

- The MCF5270 has four 16-bit program interrupt timer (PITs) modules that provide precise interrupts at regular intervals. (PIT0 is used by uC/OS as the tick timer, so do not use it in your application.)

- The MCF5270 also has four 32-bit DMA timer modules that provide input capture and compare capabilities. They can optionally signal events using interrupts or triggers.
The DMA timer module could be used for input pulse measurement. Either the PIT or DMA could be used for PWM, but the DMA will provide better resolution. Only the DMA module will be discussed further here.

DMA timer modules can be clocked from the system clock or externally. When operating from the system clock a DMA timer can be programmed to have a value between 13 ns and 234,562 s (5 hours).
Netburner Hardware
Timers

• The four DMA timer modules are denoted DTIM0-DTIM3. There are 6 configuration and control registers for each timer module.

• (1) The Timer Mode Register (TMR) is a 16-bit register that does most of the configuration. It is used to select between the system clock or external source and to divide the system clock by 1 or 16. The TMR also controls timer reset and enable.
Netburner Hardware

Timers

- (2) The Timer Extended Mode Register (TXMR) is an 8-bit register that can be used to enable a DMA request.

- (3) The Timer Event Register (TER) is an 8-bit register that reports capture or reference events by setting the reference/capture bit.

- (4) The Timer Reference Register (TRR) is a 32-bit register that contains a value that can be compared to the DMA counter (TCN).
Netburner Hardware
Timers

• (5) The Timer Capture Register (TCR) is a 32-bit register that latches the DMA counter (TCN) on a capture event.

• (6) The Timer Counter Register (TCN) is the actual up-counter register. It may be read without affecting the count. Writing to the register clears it.
Netburner Hardware Timers

- Refer to the `dma_timer.cpp` program for an example of how to set up a DMA timer and corresponding interrupt. (This program is from the DMA Timer Application Note.)

- Complete details on the DMA timers can be found in Chapter 22 of the MCF5270 Reference Manual.