Some information on Project #4 – the QNX Interrupt Response Time Project

```
#define PORT_LENGTH 1

#define DATA_ADDRESS 0x378
#define STATUS_ADDRESS 0x379
#define CTRL_ADDRESS 0x37a
/* bit 2 = printer initialization (high to initialize) */
/* bit 4 = hardware IRQ (high to enable) */
#define INIT_BIT 0x04
#define INTR_BIT 0x10
#define PARALLEL_IRQ 0x07      /* parallel port's interrupt vector */

      uintptr_t data_handle;
      uintptr_t status_handle;
      uintptr_t ctrl_handle;

/* Give this thread root permissions to access */
      /* the hardware */
      privity_err = ThreadCtl( _NTO_TCTL_IO, NULL );
      if ( privity_err == -1)
      {
            printf( "Can't get root permissions\n");
            return -1;
      }

/* Get a handle to the parallel port's control register */
            ctrl_handle = mmap_device_io( PORT_LENGTH, CTRL_ADDRESS );
      /* Initialize the parallel port */
      out8( ctrl_handle, INIT_BIT );

/* Get a handle to the parallel port's Status Register */
            data_handle = mmap_device_io( PORT_LENGTH, STATUS_ADDRESS );

/* Get a handle to the parallel port's Data Register */
            data_handle = mmap_device_io( PORT_LENGTH, DATA_ADDRESS );

/* eventually you'll need to: */

/* Enable interrupts on the parallel port */
      out8( ctrl_handle, INTR_BIT );
```

/* I believe that the parallel port's interrupt handler must clear the interrupt each time by *reading* the STATUS port and the DATA port. */

/* two files built from the QNX Help facility each contain several Help topics that should be of help on this project. There are about 20 pages of relevant material beginning on page 230 of the book Getting Started with QNX Neutrino 2 available in the lab */