

```
/******
```

```
*      comaidsystem.c
```

```
*
```

```
*      Communication Aid System:  Designed to assist on-road communication with deaf driver
```

```
*              Hardware specs: Atmega168p microcontroller
```

```
*
```

```
*      Authors: Timmy Mbaya, Brendan Davis, Joseph Cohen
```

```
*
```

```
*      Under supervision from Betty O'Neil
```

```
*
```

```
*      Spring 2010 Real-Time Systems Independent Study, UMass Boston
```

```
*
```

```
*****/
```

```
/* Copyright (c) 2010 Timmy Mbaya, Brendan Davis, Joseph Cohen
```

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

* 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.

* Neither the name of the copyright holders nor the names of 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 AND CONTRIBUTORS "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 OWNER OR CONTRIBUTORS 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. */

/* \$Id: comaidssystem.c, version 1.0 2010/31/04 09:26:08 */

//

```
//
```

```
//
```

```
#include "delay.h"
```

```
#include "keyboard.h"
```

```
#include "nlcd.h"
```

```
#include "comaidsystem_functions.h"
```

```
#include <avr/interrupt.h>
```

```
#include <avr/sfr_defs.h>
```

```
#include <avr/pgmspace.h>
```

```
#include <avr/eeprom.h>
```

```
unsigned char bitcount;           //variable to parse scancode data bits from bits recived from keyboard
```

```
//MAIN
```

```
int main()
```

```
{
```

```
    bitcount = NUM_BITS;          //initializes bitcount to 11 (1 start, 1 parity, 8 data, 1 stop
```

```
    init_sysvarStates();          //initializes system global variables
```

```
    enable_pcint(KB_PCINT);       //Enable pin change interrupts from the keyboard clock
```

```
initTables();      //Initialize scancode tables
```

```
keyboard_setup();  //Sets up the keyboard after BAT test
```

```
sei();
```

```
nlcd_init();
```

```
nlcd_enable_scrolling(); //Must enable: nlcd_init does not by default
```

```
deletefn(NO_USE_CHAR);
```

```
while(1)
```

```
;
```

```
return 0;
```

```
}
```

```
//Interrupt Service Routine for pin change interrupts from PCINT8-14
```

```
ISR(PCINT1_vect)
```

```
{
```

```
static unsigned char char_data = 0;
```

```
//If negative edge, ignore start, parity and stop bits and read bit
```

```

if ( (PINC & (1 << PINC3)) == 0 ) {

    if (bitcount < NUM_BITS && bitcount > 2) {        //we only take the 8 data bits

        char_data = (char_data >> 1);

        if (PINC & 0x04)

            char_data = (char_data | 0x80);

    }

    //If we received a byte...

    if (--bitcount == 0) {

        bitcount = NUM_BITS;

        process_scancode(char_data);                //Decode and process received scancode

    }

}

//Enables pin change interrupts

void enable_pcint(int pcintnum)

{

    if ((pcintnum >= 0) && (pcintnum < 8))

        ;

    if ((pcintnum >= 8) && (pcintnum <= 14)) {

        PCICR |= 0x2; //Enables pin change interrupts from PCINT8-14

```

```
PCMSK1 |= 0x8; //Unmasks pin change interrupt from PCINT11 only
```

```
DDRC &= ~(1 << DDC3); //Sets PINC3 as input for data
```

```
PORTC |= (1 << PORTC3); //Sets pull-up on PINC3
```

```
MCUCR |= (1 << PUD); //Completes Tri-state (Hi-Z) DDxn:0 PORTxn:1 PUD:1 (MCUCR)
```

```
DDRC &= ~(1 << DDC2); //Sets PINC2 as input for data
```

```
PORTC |= (1 << PORTC2); //Sets pull-up on PINC2
```

```
MCUCR |= (1 << PUD); //Completes Tri-state (Hi-Z) DDxn:0 PORTxn:1 PUD:1 (MCUCR)
```

```
}
```

```
if ((pcintnum >= 16) && (pcintnum <= 23))
```

```
;
```

```
}
```

```
//Sets up the keyboard after BAT test, clears keyboard buffer and resets
```

```
void keyboard_setup(void)
```

```
{
```

```
    //Set up for output
```

```
    DDRC |= (1 << DDC2);
```

```
    PORTC |= (1 << PORTC2);
```

```
    PINC = 0xFF; //Tell keyboard to reset.
```

```
delay_ms(100);
```

```
//Set up for input again
```

```
DDRC &= ~(1 << DDC2);
```

```
PORTC |= (1 << PORTC2);
```

```
}
```