Serial Communication and Data Collection

- How to collect data on the Handy Board and upload it to a host computer for processing, using the IC environment.
- There are at least two ways upload sensor data from the Handy Board:

- **Sensor data is printed to serial line in real time**

  Ideal for when data does not need to be sampled too quickly, since sampling rate is limited by the serial transfer speed

  or when the amount of data being collected is so great as to exceed the Handy Board's memory

- **Sensor data is collected & stored in HB's memory and later uploaded to serial line**

  Best for capturing a short burst of rapidly changing data

  or when it is desired to leave the Handy Board in a remote location for the data collection process

Serial Communication and Data Collection

Serial Line Interaction

- IC interacts with HB via low-level protocol that allows it to do things like write to/from HB's memory while HB is executing IC programs
  - The runtime system resident on HB responds to this protocol at all times
  - Because of this, characters cannot simply be sent to the HB's serial line
- **Solution:** temporarily disable runtime system's responses to serial activity
  - From the host computer side, it will appear to IC that HB is not connected, because it will no longer respond to the built-in serial protocol
- Then, writing to the serial line is done by interacting with built-in 6811 serial port registers
  - *Serial Communications Data Register* (SCDR) is located at address 0x102f
    - If data is stored to this register, the 6811 transmits it as serial output; when a serial character is received, it is retrieved by reading from the same register
  - *Serial Communications Status Register* (SCSR) is located at address 0x102e
    - Bits in this register indicate when the serial port is busy (e.g., whether it is in the middle of receiving or transmitting a character)
- IC library file: **serialio.c**
  - Wrapper functions for interacting with serial port

Serial Communication and Data Collection

Connecting to a Terminal Program

- *Terminal emulator program:* used for receiving serially-transmitted data
- Test program for establishing a connection between the HB and a terminal emulator on the host computer
- **Load serxmit.c serialio.c**
- **`disable_pcode_serial()`:** so that HB does not interpret any accidental characters that might be sent from the host computer
- Infinite loop:
  - Prints a message to the LCD screen telling the user to press the Start button
  - Calls the library function **`start_press()`,** which waits for the Start button to be pressed
  - Transmits the 96 printable characters of the ASCII 1 character set, beginning with code 32, a space, and ending with code 127, a tilde
- To restart HB in normal mode: hold down Start button while turning it on

```
/* serxmit.c
Each time start button is
pressed,  transmits the 96-
character ASCII  set */
void main()
{
     int i;

disable_pcode_serial();
       while (1) {
        printf("Press Start
button to begin\n");
        start_press();

printf("Transmitting...\n");
        for (i= 32; i< 128;
i++)
        serial_putchar(i);
       }
}
```


Serial Communication and Data Collection

Printing to Serial Line

- `IC` library file **`printdec.c`** provides **`printdec()`,** which takes an integer as input and prints its value as a decimal number over the serial line
- Example: **analogpr.c** program to continuously print value of analog sensor 0 to serial line
  - After calling **`printdec()`** to print the sensor value, the program outputs the values 10 and 13 to the serial line
  - This is done using `serial` **`putchar()`** so that the data is sent as control characters. When interpreted by the terminal emulator, the 10 causes a line feed and the 13 causes a carriage return.
  - **`msleep()`** function in the inner loop of the display routine slows down the rate at which the HB broadcasts the sensor data to allow terminal emulator program to keep up on its screen display.
  - Sensor data is continuously displayed on the host computer screen

```
/* analogpr.c
requires printdec.c,
serialio.c */
void main()
{

disable_pcode_serial();
     while (1) {

printdec(analog(0));

serial_putchar(10);

serial_putchar(13);
/* wait 0.1 sec between
each print */
       msleep(100L);
     }
}
```

Serial Communication and Data Collection

Capturing Data

- For quickly changing data, the final piece of the puzzle is storing sensor data in the HB's memory for later printing to the serial line
- This allows a much faster capture rate since the speed is limited only by the speed of IC, rather than the relatively slow serial communications rate
- **datacoll.c:** IC program for capturing data
  - **`data[]`** array of 1000 elements
  - **`main()`** allows user to trigger data-collection and data-dump modes by pressing the Start button

```
/* datacoll.c
requires printdec.c,
serialio.c */
int SAMPLES=1000;
char data[1000];
void main()
{
     disable_pcode_serial();
     printf("press Start to
collect data\n");
     start_press();
     collect_data();
     beep();
     printf("press Start to
dump data\n");
     start_press();
    dump_data();
     beep();
     printf("done.\n");
}
```


Serial Communication and Data Collection

Capturing Data

- **`collect_data()`** iterates through the elements of the array, storing a successive data sample in each one (takes 2 sec - 500 samples/sec - may slow down)
- **`dump_data()`** outputs the data stored in the array to the serial line, using the line-feed/ carriage-return technique
- Save data; load into spreadsheet program for graphing and analysis

```
void collect_data()
{
     int i;
     for (i= 0; i< SAMPLES;
i++) {
     data[i]= analog(0);
/* to slow down capture rate,
add msleep here */
     }
}
void dump_data()
{
     int i;
     for (i= 0; i< SAMPLES;
i++) {
     printdec(data[i]);
     serial_putchar(10); /*
line feed */
     serial_putchar(13); /*
carriage return */
     }
}
```