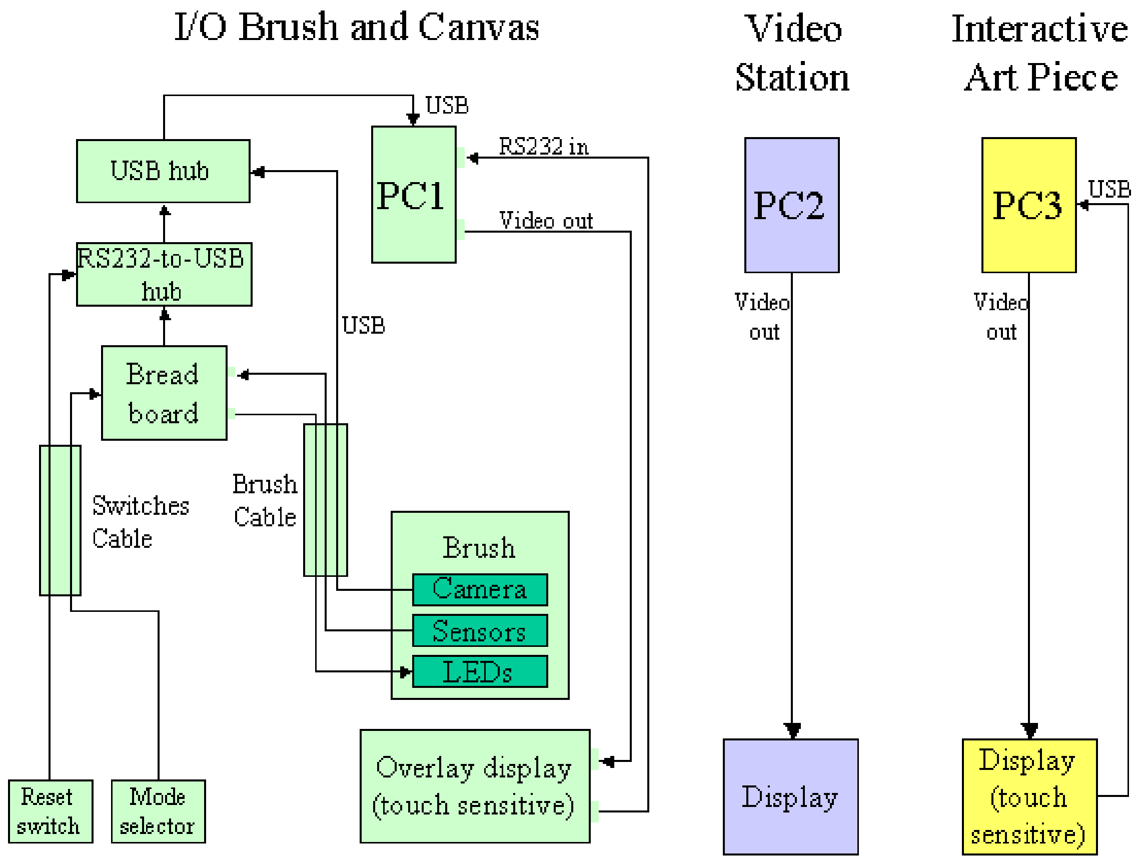
I/O Brush Technical Manual for Ars Electronica Center

Version of October 4, 2004

Kimiko Ryokai and Stefan Marti
kimiko@media.mit.edu, stefanm@media.mit.edu

1.1 System diagram



1.2 Components

The whole I/O brush installation consists of these main components:

- Brush
- Breadboard, USB hub, RS232-to-USB hub
- PC
- Overlay touch panel
- Interactive art piece
- Video station

In the following, we will describe all components in detail.

1.2.1 Brush

The brush is a wooden body that contains all the electronics, sensors, camera, bristles, etc. It also has a cable that connects these components to a breadboard.

The body of the brush is hand made from hard maple wood, professionally wood-turned, a process that took several days to complete because of the complex inner and outer shape of the wooden piece.

The bristles are original brush bristles, hot glued to a laser cut outer ring made of transparent acrylic.

The outer ring is attached to an acrylic inner ring, which is slightly bigger diameter. The two rings are attached to each other via the four force sensors, to guarantee maximum sensitivity.

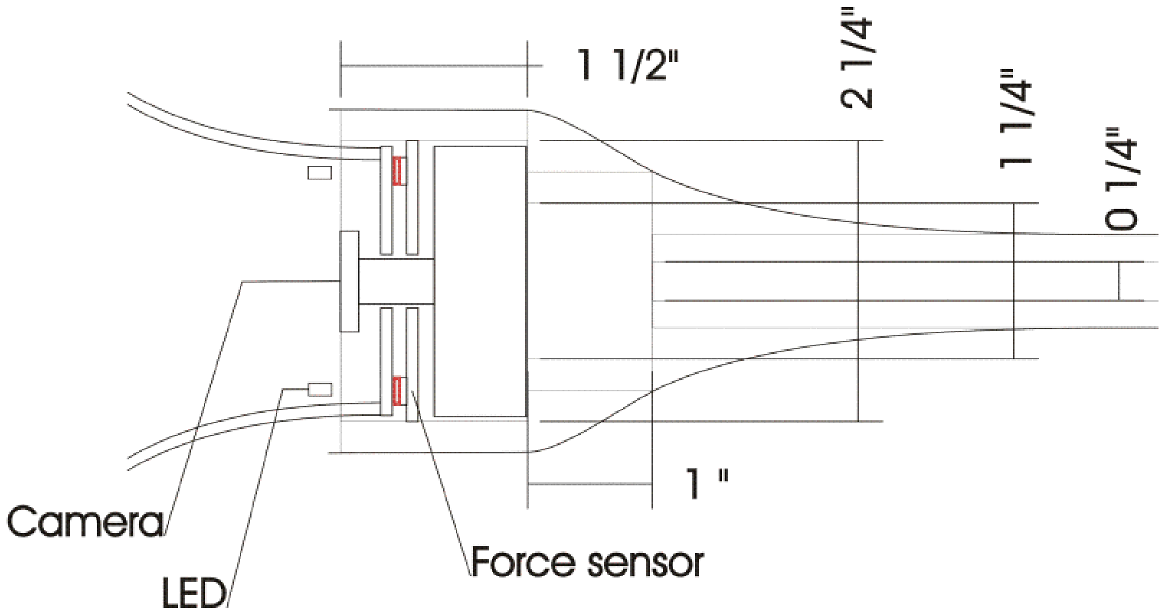
The force sensors are attached with thin double-sided tape to the inner ring, and with double-sided foam tape to the outer ring. The foam tape is necessary to (1) distribute the force evenly over the sensor, and (2) buffer the pushing force (vertical travel of the outer ring) that gets applied when the user presses the brush to an object or surface. To prevent the two rings from getting pulled apart (negative pressure), there are two safety screws that span the two rings, but do not apply force to them. In order to not influence the sensor readings, the holes in the rings are larger in diameter than the diameter of the screws, and the nuts are not tightened completely: the nuts—two for each screw on the outer ring side—are secured with a drop of epoxy glue, instead.

The camera used currently in the brush is a standard USB Logitech webcam.

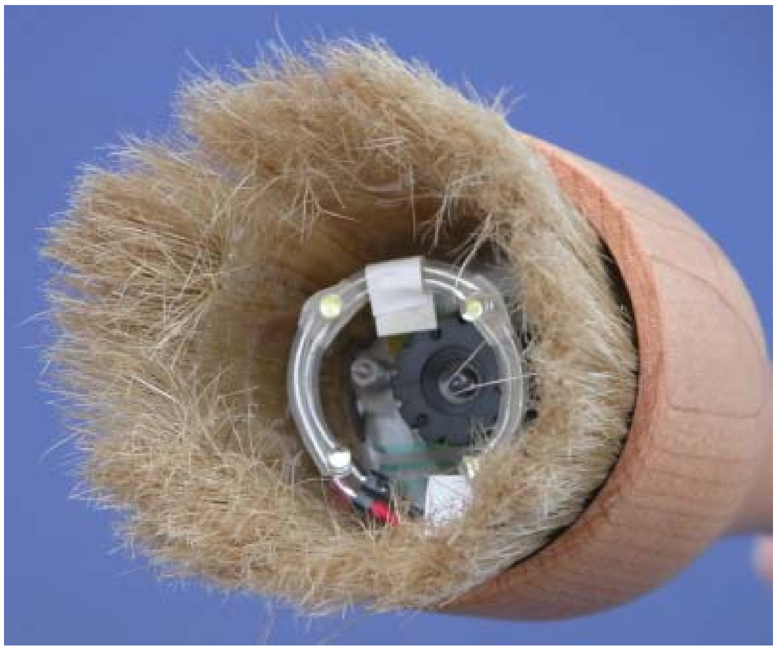
The sensors used are Force Sensing Resistors (FSR) from Interlink Electronics (model 400: 0.3" circle with solderable tabs)

http://www.interlinkelec.com/documents/datasheets/fsrdatasheet.pdf
http://www.interlinkelec.com/documents/usersguides/fsrguide.pdf

These sensors are very thin, since they are made of several layers of polymer film. They are very sturdy and reliable, but (as mentioned before) *they are do not like negative forces*. If such forces occur (pulling forces), the components of the film might separate. In order to prevent negative forces, there are the two (not tightened) screws described above. Note that negative forces do only happen if somebody PULLS the bristles of the brush—something only little children might do when unattended…

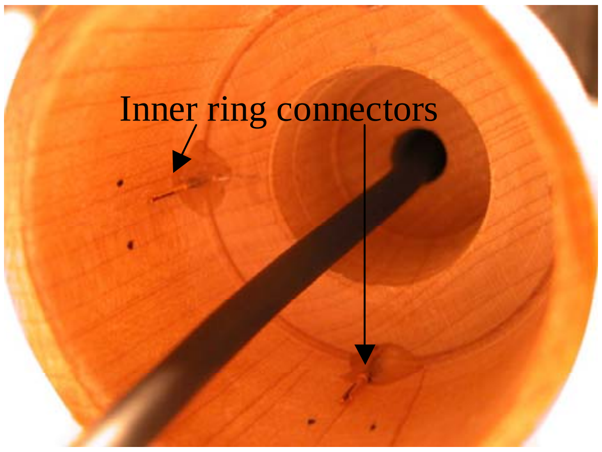


A construction sketch of the head of the brush, showing the location of the force sensors (red), the camera housing, the camera lens, the white, the LEDs, the two acrylic rings, the bristles, etc.

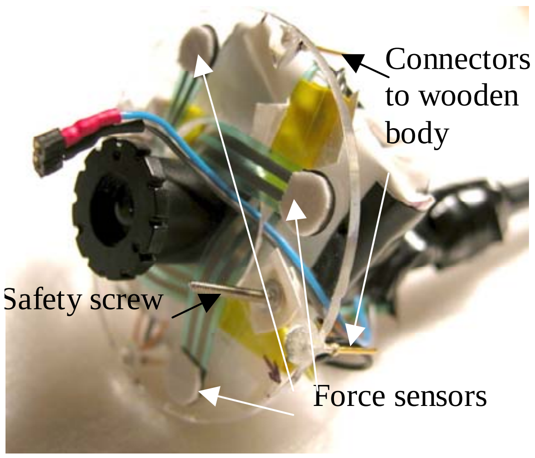


Finished head of the brush. Note that in this picture, the ring of white LEDs is not yet glued to the side of the bristles.

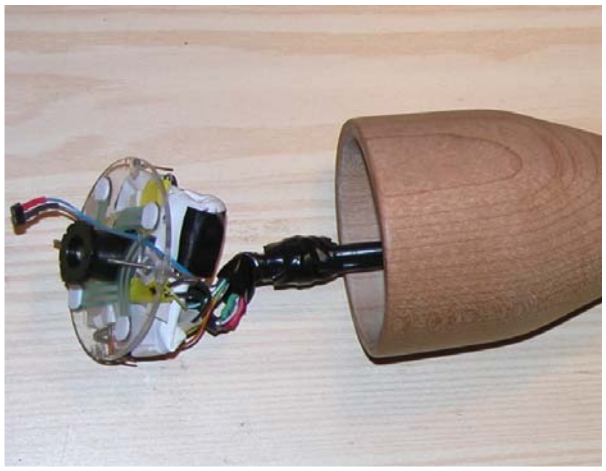
The outer ring with the bristles.



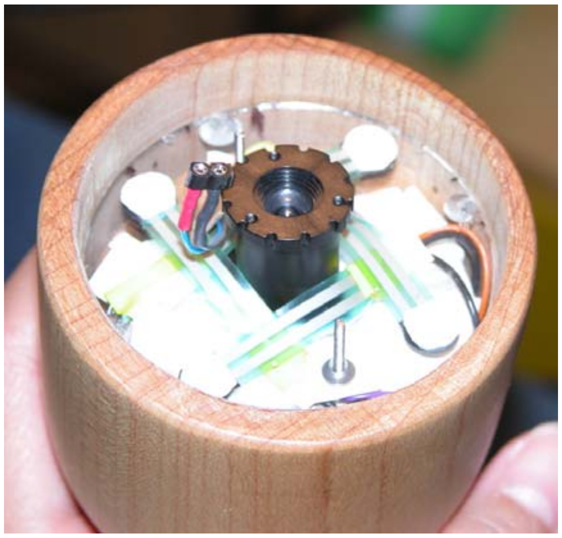
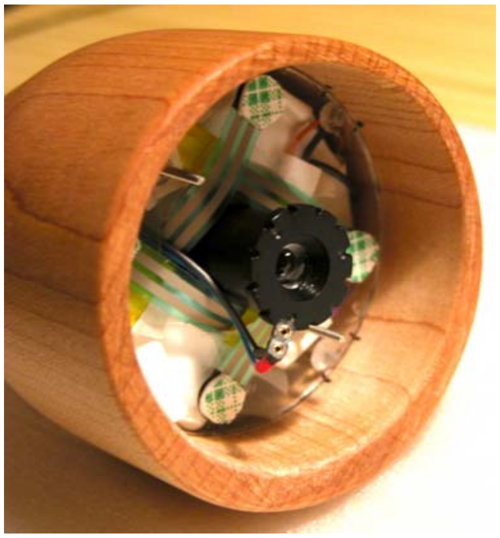
Inside view of the wooden brush head. Visible are two connectors (out of four) that the inner ring snaps into. In the current version of the brush, the inner ring additionally is glued to the wooden walls with epoxy at the location of the four connectors. This means the inner ring (and therefore the whole assembly with the bristles) cannot be detached easily. If it has to be removed, then the glued parts have to be cut out first, which can be done only by Kimiko Ryokai.



The inner ring with the camera attached. Visible two of the four pins that snap into the plugs on the picture left. Also clearly visible are the four force sensors, which are attached to the inner ring and (with double sided foam tape) will be attached to the outer ring. They are the only connection between inner and outer ring, which guarantees that all the forces applied to the bristles are measured by the four pressure sensors. Also visible is one of two screws that prevent the outer ring from being pulled off.



In the above pictures, the brush cable is inserted into the brush and the inner ring is ready to get mounted to the wooden brush head.



Now the inner ring and the camera are mounded. The two wires and connectors sticking out beside the lens will eventually provide power for the white LEDs. The 2-sided foam tape is still covered with the protective layer.

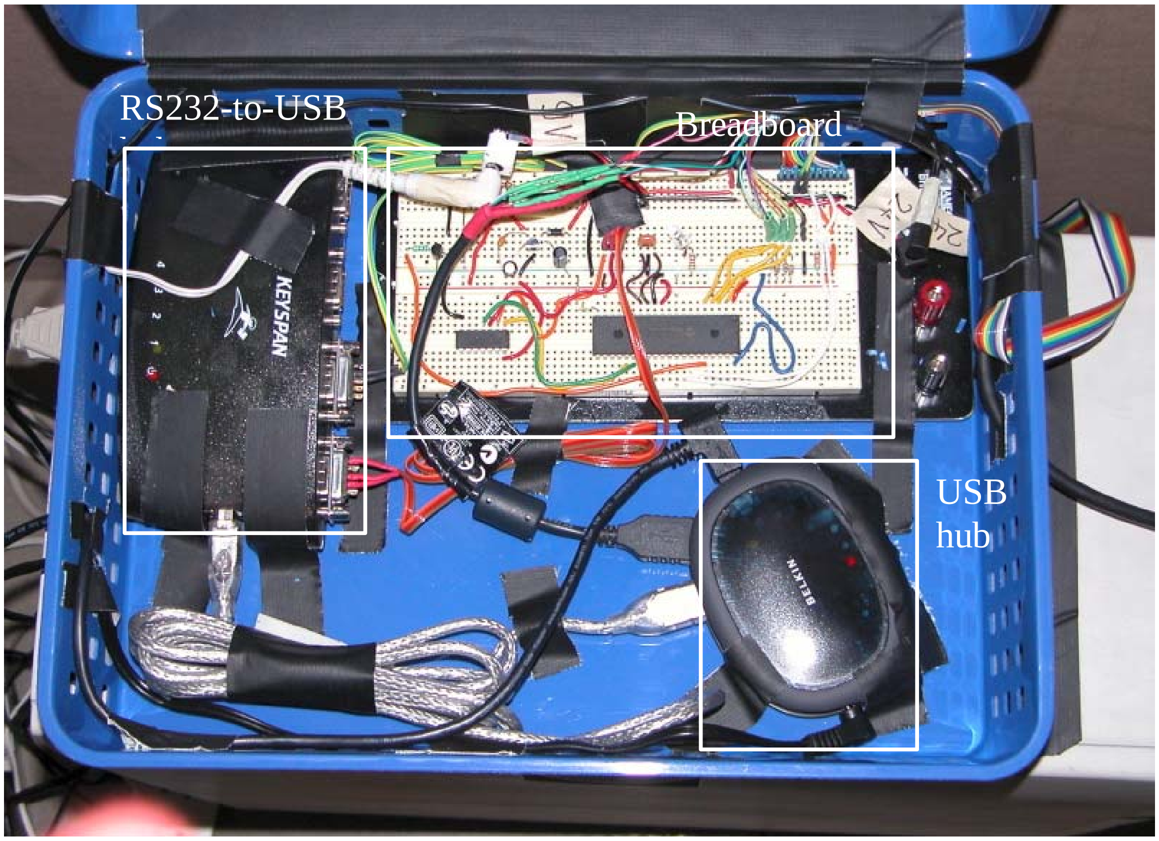
The outer ring with the bristles is ready to be attached to the inner ring.

1.2.2 Breadboard, USB hub, RS232-to-USB hub

The breadboard acts as a connection between the brush and the PC. Its function is to digitize the sensors of the brush and send them via serial port to the PC. It also lights up the white LEDs in the brush when the brush touches an object to pick up.

The serial connector of the breadboard (3 wires, brown/red/orange) is connected to an RS232-to-USB hub. Also connected to the RS232-to-USB hub is the reset button (2 wires, green/yellow).

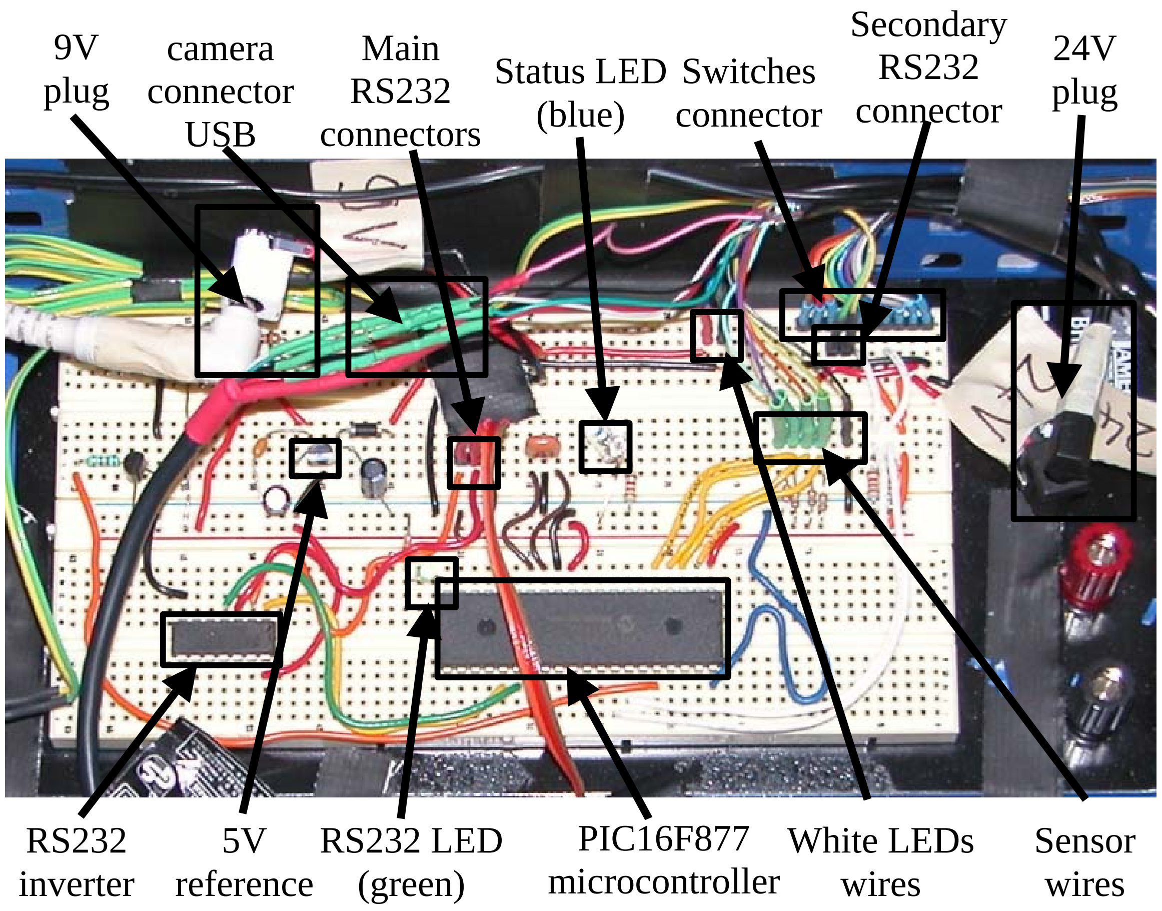
The USB signals from the RS232-to-USB hub (breadboard) and the USB camera are collected in the USB hub.



The main components on the breadboard are:

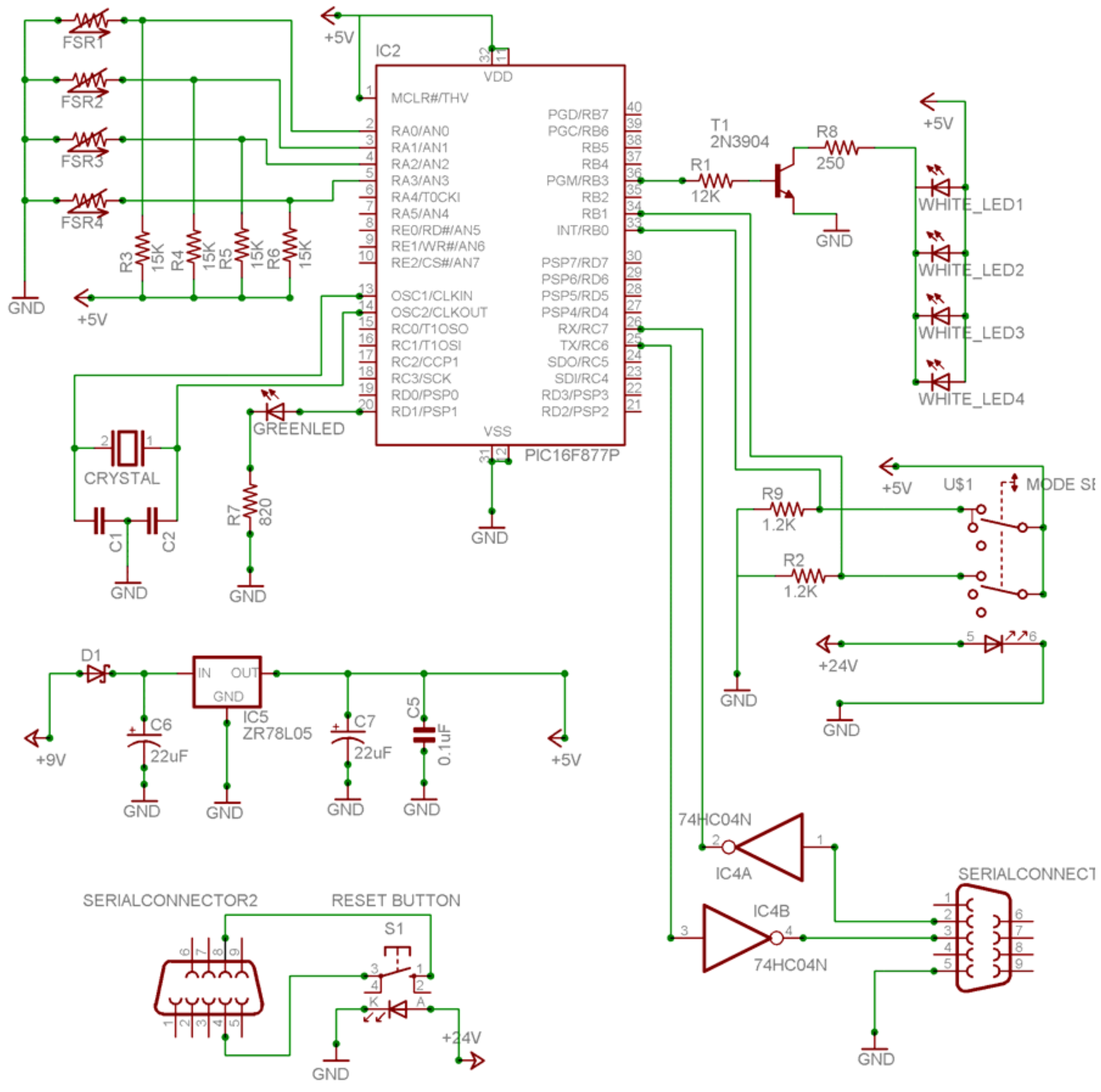
- Micro controller (PIC16F877) that digitizes the analog sensor signals
- Power-conditioning circuitry, to create the stable internal 5V from the 9V external power supply
- Components for the serial communication with the PC, based on a MAX233 chip
- Connectors for all the cables and connectors, such as:
    - Power for breadboard (from 9V power supply: black plugs)
    - Mode selector switch and reset button (rainbow colored cable band)
    - Power for orange lights on mode selector switch and reset button (from external 24V power supply, white plugs)
    - Sensors (5 wires: yellow/orange/light green/purple for sensors 1-4, and black with white stripe for ground).
    - RS232 cable to RS232-to-USB hub (which goes to USB hub and to PC), for communication with PC (orange/red/brown wires)
    - Connection to second RS232 port, for sensing the reset button (two wires, green/yellow). Note that there is no analog-to-digital conversion of any kind on this port. The reset button merely closes two lines on the serial port (DTE ready line 4, and RTS line 8), which is detected by the iomon.exe code that listens on this port. This is of course not serial communication according to RS232 standard.

It also contains status LEDs: a blue LED lights permanently when the 5V reference is ready, and a green LED that blinks if the communication between windows code and microcontroller it running. Note that the green LED only blinks when the windows code is actually running.



1.2.2.1 Schematic of circuit on breadboard

Below is a schematic of the circuit, which includes the breadboard, the brush, and the switches.



1.2.3 PC

The PC runs the actual software that generates the images on the screen.
Connected to the PC are:

- Touch screen, via serial connector (COM2)
- USB hub (breadboard, camera)
- Projector

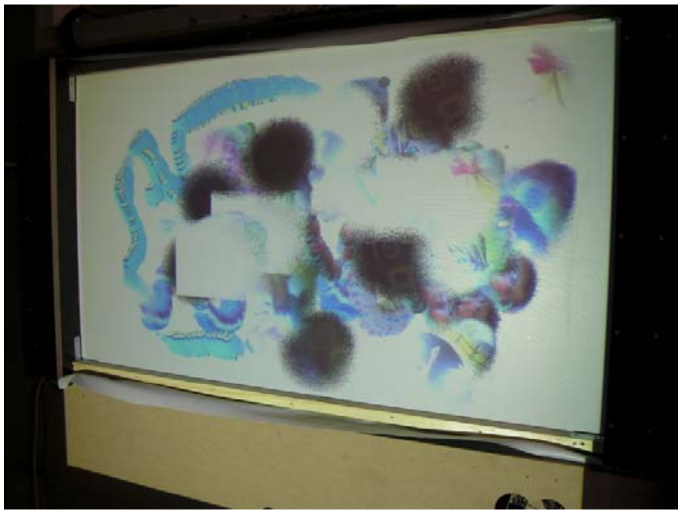
Graphic resolution of this PC is set to 1280x1024. Although the projector doesn't support this resolution natively, it is better to run it like that, because the pixels are (due to interpolation) a bit fuzzy, which is good for our purpose.

1.2.4 Touch panel overlay

The large *NextWindow* touch panel overlay is using vision to detect objects on the screen (http://www.nextwindow.com/products/plasma.html). It is the only technology that allows measuring the presence of the soft bristles on a surface. We tested both inductive and resistive technologies, but neither of them worked reliably.

The panel uses two cameras to detect the position of the objects on the screen. Although theoretically possible, it cannot detect more than one object, though. It has been calibrated with custom calibration points, since the panel is 16-to-9 and the projector can only provide 4-to-3. Although it should not be necessary to re-calibrate it, there is an *AutoCalibraor.exe* icon on the desktop that will repeat the calibration steps, including the custom calibration points, which are stored in the file C:\Program Files\Next Window\2400\Calibrator.ini.

The projector has to be set up so that the upper left corner of the Windows desktop is also on the upper left corner of the overlay touch panel. This means, the lower part of the windows desktop is not visible, which is good, because that's where the few Windows icons are that cannot be removed from the desktop. These icons are not visible anymore in projector mode, of course.



Back view of the touch panel overlay and projection. Note that the projection aligns on the top right corner (view from behind) with the overlay touch panel. The top right corner is also where the preview window is visible.

1.2.5 Interactive art piece

The interactive art piece is an executable on a Windows PC (.EXE file) with a touch screen. Although it is originally written in Macromedia Director MX (Lingo), it is converted to the executable file that just has to be clicked on.
The latest art piece is in the Startup folder, and is called

```
artpiece_Projector_090304_v1.exe
```

In the same Startup folder, there is also a folder "quicktime" that contains all 24 QuickTime movies that are necessary for the interactive art piece.

The graphic resolution of the PC has to be set to 1280x1024. (The executable expects this resolution.)

1.2.6 Video station

The video station is a normal PC that runs a movie of an I/O brush demonstration in a loop. The file is AVI and is played by a Windows Media Player *full screen* mode. A link to the AVI file is in the Startup folder, so it will start up automatically, but still has to be switched to *full screen* mode manually by pressing “Alt-Enter”.

Note that the file is quite large. The current version, “iobrush_movie_090204_v2.avi” is 476MB large. We also noted that if the video is played in non-full-screen mode, there are some artifacts visible on the right side of the video. They are not present in the full-screen version, though.

1.3 Start-up procedure

Make sure that all components are plugged in:

- Touch panel power: it makes a few beeps when powered up
- Touch panel serial plug into Windows PC serial port 2 (the lower of the two, on the back of the computer)
- USB hub into USB port of PC: we are using currently the left one in the back of the computer.
- USB hub power supply: this has to be done before the PC starts up, or the PC will complain about not having enough power to power all devices connected. A red power LED must be on, and two orange LED’s for each cable plugged into it (serial-to-USB hub, and USB cable from camera)
- Brush cables into breadboard. These are single pins: 5 for the sensors, 2 for the white LEDs, and 5 for the USB camera)
- Switch cables into breadboard: this is the rainbow colored cable with the wide 10-pin header at the end.

Then power up in the following sequence:

1. Power to breadboard (9V power supply, WHITE plug on breadboard): after plugging in (or resetting) this connector, all LED’s on the board (green and blue) as well as the white brush LEDs should blink rapidly for 1 second, and then all go off except the blue LED.
2. On the PC, start the system with the link “IOBrush Start aec083004”. The link contains the following file name, path and arguments:
   ```
   C:\AEC\iomon.exe COM4 “C:\AEC\projector.exe”
   ```
   This will start a command line window (iomon.exe), which in turn will start the actual projector EXE file. The screen will turn completely white, except where the mouse pointer was, there will appear round “stain marks.” This means, the system is ready.

If the mode selector switch and reset button do not light up in orange color, also connect the 24V power supply to the jack on the breadboard marked with “24V” (black connector). This can be done at any time, since the power goes to the switches' lights directly, and is independent from what is going on on the PC or on the breadboard. However, DO NOT CONFUSE THE 9V AND THE 24V CONNECTORS! The 9V connector is white, where as the 24V connector is black.

Also note that the white 9V power supply is actually a multi voltage supply. DO NOT change the settings on the power supply, and do not use any other voltage than 9V!

1.4 Tips for info-trainers: Do's and Don'ts for visitors

1.4.1 General treatment:

1. First and most importantly: **the brush is fragile**. Please tell the visitors to treat it gently, and supervise them when they use the brush!
2. Do NOT drop the brush. It will probably break, but we don't want to find out, right?
3. Do NOT pull the cord, neither on the brush side nor on the wall side. Avoid tripping over the cord: this will pull the cable very much, which might damage the brush.
4. Do NOT pull the bristles.
5. Do NOT throw the brush, or swing it around holding just the cord. Some very young visitors will probably try doing that…
6. Do NOT knock with brush on the screen: the wooden body of the brush might actually break the glass, which is very dangerous.
7. After use, please put the brush back on its tray. Do NOT leave it on the table with the objects to pick up: somebody certainly will trip over the cable.

1.4.2 General handling:

8. In order to pick up something, press the brush towards an object. As soon as the white lights come up in the brush, it will start picking up. However, in normal mode, only the LAST picture will be kept. Remember, if the white lights inside the brush don't come on, or flicker, then you do not pick up the object properly. Instead, press a little harder when picking up, so that the lights are on as long as you touch the object.
9. Once you have picked up an 'ink', you can paint on the screen. Best is to move the brush slowly with light or medium touch. The lighter you press, the thinner the line will be.
10. Avoid pressing the brush very hard (on objects or on screen): it will eventually damage the bristles, and will eventually scratch the glass of the overlay panel!
11. If you have picked up an 'ink' and decided to stamp it on the screen (not moving when touching the screen), avoid staying on the same spot on the screen for more than two seconds. If you do, the brush might create green circles and overwrite the ink you just picked up. This is because the brush on the screen picks up the screen itself after some time of no movements, which is not what you want.
12. I/O brush has two states: picking up something, and drawing with that something on screen. This means, normally do not pick up ink and AT THE SAME TIME

draw on the screen. This happens when two or more visitors are using I/O brush at the same time. Or rather, you can pick up and draw on the screen simultaneously, but then the person who draws on the screen will get the color that is changing continuously (which can be fun).

13. Do NOT put objects on the screen to pick them up from there, like a book. The screen will see the object(s) on the screen and get confused of weather the user wants to pick up or draw. Remember, I/O brush can only pick up objects OR draw, but not at the same time.
14. Don’t forget that you can smudge the color. However, after each time you smudge, you have to wait a few seconds to smudge again.

1.4.3 Movie mode

15. If you turn the knob on the front wall to the RIGHT, the brush goes into movie mode.
16. Picking up the eye is not a simple as it looks: it takes some practice! The most common error is that the lights do not light up continuously. If the lights flicker, then you have to press a little harder.
17. I/O brush can only pick up three movies, each maximum 1.5 seconds long. The fourth movie will overwrite the first one, etc.
18. If the movie you pick up is less than 1.5 seconds, you will see leftover frames from the older movie that you just over wrote. That is normal.
19. There are other ways that the eye to show the movie mode: for example, brush over text (from the colorful books provided), and then paint on the canvas: you will see that the line is changing colors as you move along the canvas: this is a nice effect, too, and much easier to demo!

1.4.4 Color mode

20. If you turn the right orange knob on the front wall to the LEFT, the brush goes into color mode. This means, you will pick up only the main color of what the brush sees.
21. If you don’t get the color that you expected from an object, keep the brush where it was and turn the knob to the middle position, and you will see what the brush sees right now. Adjust the position of the brush so that the brush sees as much as possible of the color you want, and turn the knob back to left. Then you should get the color you expect to get—but it is harder than you think! Very often, although the colorful object is in the middle, there is black or white color around it, which makes it difficult for the brush to find the real color. Note that the brush does not give you the *average* color of all the colors it sees, but the *single color* that it sees the most. In other words: if you brush over a yellow and red striped pattern, you will NOT get orange!

1.4.5 Restarting

22. You can clean the screen by hitting the left orange button. Do hit only once—that is enough. Restarting takes a few seconds, during which you see the text “Einen Moment, bitte!” As soon as the hourglass mouse driver disappears and some stain rings show up, the system is ready again.

23. IMPORTANT: Restart only when you are in NORMAL mode (when the button is in the middle position).
24. Do NOT restart when you are in COLOR mode (knob turned to the left). If you do so, it just won't paint. Just turn the knob to the middle (NORMAL mode), and it will work again. After you have painted once in NORMAL mode, you can turn the knob back to left, and paint in COLOR mode.
25. If you restart in MOVIE mode, you will NOT see a preview of the first movie on the top left of the screen. However, you can still picks up a movie. Just paint with the first movie, and from the second movie on you will see the preview on the top left corner of the screen. But remember, we recommend that you restart only in NORMAL mode (knob in middle position).

1.4.6 Other hints

26. You CAN paint over the things that are already on the screen. Some visitors don't realize that.
27. You CANNOT paint over movies on the screen.

1.5 Software

1.5.1 PC software

The PC software contains of two executables: a monitor program (iomon.exe) that starts up automatically the actual program (projector.exe). This means, the actual program (projector.exe) should not be started manually: the monitor program does that automatically. (If you start projector.exe manually, the system works normally, except that the reset button does not work.)

The idea behind this is that when the user presses the orange reset button (the round button on the left side under the large screen), the monitor program (iomon.exe) kills the actual I/O brush program (projector.exe), and after a few seconds, restarts it.

The iomon.exe is a command line program that has to know to which serial port the reset button is connected to: this is currently port 2 of the *Keyspan* RS232-to-USB hub, which appears as COM4 on the PC. It also needs to know which program it has to start up, kill, restart etc. The current command, stored as a link in "IOBrush Start aec083004", is:

```
    C:\AEC\iomon.exe COM4 "C:\AEC\projector.exe"
```

Usually, we run the compiled version of the Macromedia Director code. In order to debug the sensor values (the most likely thing to do), we have to run the original Director script. At this point, basic knowledge about Macromedia Director MX and its programming environment is necessary.

To test the sensor values:

1. Open the most recent .dir file in "My Documents\IOBrush\AEC_debug". Currently, that is "timedelay_iobrush_082004_modified_aec_090304.dir"

2. In script 10, uncomment the two lines 177 and 178 by removing the double hyphen at the beginning of the lines:
   ```
     --  put "sensors : "&sensor
     --  put "mode:  "&mode_selected
   ```
   The lines will look like that afterwards:
   ```
       put "sensors : "&sensor
       put "mode:  "&mode_selected
   ```
3. Click on the *Message Window* so that it is visible.
4. Rewind the movie, and run it.
5. You should see the sensor values (4 bytes, each between 0 and 1024), and the next line the mode (1 value, 0-2)

Note that you most likely will see that sensor 1 has a somewhat erratic value. If the brush is not pressed, sometimes its value is close to 0, but sometimes it stands closer to 400. We decided that sensor 1 is not reliable, and disregard its value in the most recent director code. This does not influence the performance of the system at all, since in praxis just one of the sensors has to be working reliably! We have four pressure sensors for two reasons:

1. Four-times redundancy: if a sensor stops being reliable, it can be disabled. We actually only need one working sensor! Note that an elevated base level (as with sensor 1) does not mean the sensor is broken. It just means the sensor encounters a pressure even without the bristles being pressed. Such an offset could also be calibrated for, if necessary. We do not know, however, what causes the standby pressure that sensor 1 sometimes shows.
2. In later versions of the brush, we intend on using the direction of the pressure for refined drawing expressivity. The current brush, however, does not have these capabilities implemented.

The most likely change that you might want to make is **adjusting the sensor threshold of the brush**. This might be necessary if a sensor starts to act “weird” by either not giving a value anymore (very unlikely), or having an elevated base level (likely), due to mechanical damage of the brush.

In script 10, find the line 187:
```
    if sum > 600 then
```

As you can guess, the current threshold is set to 600, which means that for three sensors, each has to have an average value of 200 to trigger the picking up behavior (indicated by the white LEDs coming on), or painting on canvas. Adjust this value, up or down, but BE CAREFUL WHEN YOU CHANGE THIS VALUE! You will have to test the brush thoroughly to see if it works the way you expect: test picking up different colors from different objects, and paint a lot to see if the change is better or worse!

If you have make changes to the Director code, store it as a different file name. Then you have to ‘recompile’ it, or rather, make a *projector* file: In the menu, choose “File→Create Projector”. Then locate the director file you have just modified, and add it as “Director Movie” file type. The locate the following extras, and add them as “Xtra” file type:

- IOBrush\DirectorMX\Xtras\DirectComm\Windows\Runtime\DirectComm.x32

- IOBrush\DirectorMX\Xtras\VideoMask1.1\VideoMask.x32

Then “Create” the projector.
Rename the file to “projector.exe”.
Then go to the directory

```
C:\AEC\
```

There you will find another “projector.exe” file there, the old one. RENAME THIS FILE SO THAT YOU HAVE A BACKUP! (You can also move it to a safe location instead, but remember where it is, because this one works!)
Then copy the new projector.exe file to:

```
C:\AEC\projector.exe
```

That’s it! Of course you have to check now if everything still works! You don’t want to run the projector.exe file by itself, because you can’t test in the reset button works fine. Use the Iomon link (“IOBrush Monitor aec083004”) instead, which will start the projector.exe file in this directory.

1.5.2 Microcontroller software

The software is written in C and compiled with a CCS compiler. It is burned onto the PIC with a PICstart burner, so cannot be accessed externally. However, since it also contains a so-called boot loader, the PIC can be reprogrammed via serial port. This is an extreme measure, needs special reprogramming software on the PC, and should only be done by Stefan Marti.

If you want to test the PIC software separately from the Macromedia Director code, you can use a normal serial terminal program like Windows *Hyperterm* or *SecureCRT 4.0* to test it. The serial port settings are: 115200 baud, 8N1, no flow control (neither hardware nor software), COM3 (with the current configuration).

Turn off the breadboard power supply, start the terminal program, then turn on the power to the breadboard. You should see the following line:

```
I/O Brush prototype 2, polling, v3.4 starting up....
```

Then nothing will happen until you press the numbers 0, 1 or 2 on the PC. Each time you press one of these numbers, the microcontroller will return the values of the four sensors (values between 0 and 1024), and a fifth number that represents the position of the mode selector switch.

We are using a polling method to communicate between Windows code and microcontroller code: the microcontroller does not send data unless requested by the Windows code. The Windows code currently asks the microcontroller 20 times per second for its values.

If you want to see a more continuous string of sensor values, just keep pressing “0” on the keyboard, and the microcontroller will spit out data more continuously.

1.5.3 Interactive art piece software

The interactive art piece is written in Director MX Lingo, and then compiled into an executable (the projector). The current version is called

```
artpiece_Projector_090304_v1.exe
```

It currently contains a painting with links to 24 embedded QuickTime movies.

This software is written for a touch screen, but can also be tested with a normal mouse. Note that the mouse cursor is hidden, though.

We have found out that the smoothness of the videos depends greatly on the speed of the CPU. We do not recommend using less than a 1.4GHz computer. It will run much nicer on a faster computer, though!

1.6 FAQ

Q: What if the PC complains that the USB port does not have enough power?
A: The USB hub must have the power cable plugged in before the PC starts up, or the PC will not have enough current to power the things plugged into the USB hub. Unplug and plug in back the power supply. If the PC still complains,

1. Unplug the power to the USB hub
2. Unplug the two USB cables that go into the hub (REMEMBER which one was plugged in where!)
3. Plug back in the power to the USB hub
4. Plug back in the two USB cables that go in to the hub: if looking at the hub with the writing "Belkin" horizontal, the black USB cable (the one from the camera) goes to the right most plug. The silver USB cable (from the serial-to-USB hub) goes to the second to right plug of the hub.

Q: What if I start up the PC programs, and it starts up fine, producing the little round stain marks where the mouse was, but I still can't draw on the screen?
A: Make sure the green LED on the breadboard is blinking. If the LED is not blinking after the windows program has started, then the breadboard is not communicating with the PC. Do the following:

1. Unplug the power of the breadboard (9V plug).
2. Stop the PC program (Control-Alt-'dot', or if this doesn't work, Alt-Tab to the Windows command line window, and then ESCAPE several times until you see the actual window, then Control-C to kill it.
3. Then plug back in the 9V power supply for the breadboard: all it's LEDs will blink rapidly, then stop.
4. Then restart the PC program with the Iomon link ("IOBrush Monitor aec083004"). The green LED on the breadboard shout blink about twice a second.

Sometimes it is necessary to go through the above procedure 2 or 3 times (probably because the serial port is not always freed properly when the PC program is killed).

Q: How can I open the brush?

A: You can’t! The current prototype is glued together with epoxy resin glue, and only the artists can take it apart. Next versions will have screws, though.

Q: What is the most likely way the brush breaks?
A: That’s very hard to predict. However, we estimate that because the brush is a mechanical part, it will break mechanically in some way before the electronics or the software will cause errors. In our opinion, a failure will not be catastrophic, but rather gradual. We think that the first thing might happen is that the brush lights do not go off easily anymore, or in other words: the white LEDs might be on sometimes even though not object or surfaces is touched. This means probably that mechanical parts in the brush will influence the sensor values. This might come from accidents, like when the brush fell down to the ground and the outer acrylic ring got moved or even damaged. Such a failure could be counteracted by adjusting the threshold level of the sensors. This procedure is relatively simple, but requires knowledge in Macromedia Director (because the code later has to be transformed to an EXE file, a process described above). HOWEVER, it is better (and easier) to send back the brush to us, and we will fix it for you.

1.7 Appendix: Microcontroller code

```
// I/O Brush prototype 2, polling version
// Copyright (C) 2004 MIT Media Laboratory
// Concept, design, and programming by Stefan Marti stefanm@media.mit.edu
// Code fragments by Vadim Gerasimov

#include <16F877.H>
#list

// Configure PIC to use: HS clock, no Watchdog Timer,
// no code protection, enable Power Up Timer,
// and disable brownout
#fuses HS,NOWDT,NOPROTECT,PUT,NOLVP,NOBROWNOUT

// Tell compiler clock is 20MHz.  This is required for delay_ms()
// and for all serial I/O.  These functions use software delay
// loops, so the compiler needs to know the processor speed.
#use Delay(Clock=20000000)

//serial stuff
#use RS232(Baud=115200,xmit=PIN_C6,Rcv=PIN_C7,parity=N,bits=8)

// Declare that we'll manually establish the data direction of
// each I/O pin on ports A to E.
#use fast_io(a)
#use fast_io(b)
#use fast_io(c)
#use fast_io(d)
#use fast_io(e)

// Set variable that maps to memory
#byte PORTA = 5
#byte PORTB = 6
#byte PORTC = 7
#byte PORTD = 8
#byte PORTE = 9

//=========================================================================================
// Define this ONLY for initial PIC programming to include the serial programming function
// After that, comment it out!
```

```
//#define InitialProgram = "true"
//======================================================================================


// LED pins
#define GREEN_LED  PIN_D1       // (output)

// Serial programming switch
#define PROGR_ON   PIN_B7       // (input)

//A/D pins
#define SENSOR1    PIN_A0       // (input analog)
#define SENSOR2    PIN_A1       // (input analog)
#define SENSOR3    PIN_A2       // (input analog)
#define SENSOR4    PIN_A3       // (input analog)

//Mode switch
#define MODE_SWITCH1    PIN_B0  // (input) mode switch 1
#define MODE_SWITCH2    PIN_B1  // (input) mode switch 2

//Brush LEDs
#define BRUSHLIGHT      PIN_B3  // (output) lights on the brush

// Macros to simplify I/O operations
#define GREEN_LED_ON    output_high(GREEN_LED)
#define GREEN_LED_OFF   output_low(GREEN_LED)

#define BRUSHLIGHT_ON   output_high(BRUSHLIGHT)
#define BRUSHLIGHT_OFF  output_low(BRUSHLIGHT)

// How fast does the serial communication LED blink?
#define BLINK_HOW_FAST 10

//threshold--PIC internal, obsolete now
#define THRESHOLD   1000

// Function Prototypes
void blinky();
#inline void reprogram();
void do_adc();

// Global variables
char cmd;      // last command character
long value[4]; //array with sample values
long sum;
long value_inv[4];

char char1, char2;
char user_mode;
int  picked_up_something = 0;
int  serial_traffic = 0;
int  green_led_status = 0;

void blinky()
{
   int j;
   for (j=0; j<10; j++) {
      GREEN_LED_ON;
      BRUSHLIGHT_ON;
      delay_ms(50);

      GREEN_LED_OFF;
      BRUSHLIGHT_OFF;
      green_led_status = 0;
      delay_ms(50);
   }
}

#ifdef InitialProgram
#ORG 0x1F00, 0x1FFF
void self_program()
```

```
{
  long addr, count, val, i;
  disable_interrupts(GLOBAL);
  while(getc()==0xAB) {
    *(&addr)=getc();
    *(&addr+1)=getc();
    *(&count)=getc();
    *(&count+1)=getc();
    for (i=0; i<count; i++) {
      *(&val)=getc();
      *(&val+1)=getc();
      if (val==read_program_eeprom(addr)) putc(1);
      else {
        write_program_eeprom(addr, val);
        if (val==read_program_eeprom(addr)) putc(1);
        else putc(0);
      }
      addr++;
    }
  }
  #asm
  movlw    0
  movwf    0x0A
  goto     0
  #endasm
}
#endif

// Jumps to self_program()
// This is normally called from command processor.
// Note: If you want serial programming to work make sure you can always interrupt
//       your program from serial port and have a command that calls this function
#inline
void reprogram()
{
  putc(cmd);  // Return the 0xA5!! This is necessary for the CF.exe,
              // the Windows code written by Vadim--it expects that
  puts("");
#ifdef InitialProgram
  self_program();
#else
  #asm
  movlw 0x18
  movwf 0x0A
  goto  0x700
  #endasm
#endif
}

// Self-programming function, reprograms PIC through RS232.
// The host sends pieces of code as:
//   1 byte header 0xAB
//   2 bytes start address
//   2 bytes (N) number of words to program
//   N words (N*2 bytes) program words
//
// After each program word function sends back
//   1 if word programmed successfully
//   0 if not
//
// Send non 0XAB byte to exit the cycle and restart


void main()
{
  setup_adc(ADC_CLOCK_DIV_32);   // set ADC clock to 20MHZ/32
  setup_adc_ports(ALL_ANALOG);
  // Since we've declared #use fast_io(A to E) above, we MUST
  // include a call to set_tris_a to e at startup.
  set_tris_a(0b00001111);  // all As are outputs, except A0-A3 (SENSOR1-4)
  set_tris_b(0b10000011);  // all Bs are outputs, except B0, B1 (mode
                         //   switches) and  B7 (programming switch)
  set_tris_c(0b10000000);  // all Cs are outputs, except C7 (serial receive)
  set_tris_d(0b00000000);  // all Ds are outputs
  set_tris_e(0b00000000);  // all Es are outputs

  // Enable_interrupts(INT_RB);
  enable_interrupts(INT_TIMER2);
  enable_interrupts(GLOBAL);

  setup_timer_2(T2_DIV_BY_4, 250, 5); // 1mS interval @ 20 MHz

  // Startup blinking
  blinky();

  printf("\r\n\r\nI/O Brush prototype 2, polling, v3.4 starting up....\r\n");

  if (input(PROGR_ON)) {    // if the 'programming switch' is on
       printf("--> I am now in serial programming mode--send your stuff!...\r\n");
       while (1) {         // wait for the serial signal
           cmd = getc();
            if (cmd== 0xA5) reprogram();
        }
   }

  while(1) {
        if (kbhit()) {
           char1 = getc();
           user_mode = '0'; // default, just in case
           if (!(input(MODE_SWITCH1)) &&  (input(MODE_SWITCH2))) {
               user_mode = '0';
           }
           if (!(input(MODE_SWITCH1)) && !(input(MODE_SWITCH2))) {
               user_mode = '1';
           }
           if ( (input(MODE_SWITCH1)) &&  (input(MODE_SWITCH2))) {
               user_mode = '2';
           }
           do_adc();
           printf("%lu\t%lu\t%lu\t%lu\t%C\r\n", value_inv[0], value_inv[1], value_inv[2],
value_inv[3], user_mode);

           //  old   new    meaning
           //  "00"  "0"    picking up color, not touching
           //  "01"  "1"    picking up color, touching (white LED on)
           //  "1x"  "2"    painting on canvas (all LED off)
           if (char1 == '0') {  // brush is PICKING UP COLOR (not on canvas), but NOT TOUCHING
                BRUSHLIGHT_OFF;
                if (picked_up_something == 1) {
                      // used for laser fiberoptics
                }
           }

           if (char1== '1') { // brush is PICKING UP COLOR (not on canvas), TOUCHING something
               BRUSHLIGHT_ON;
               picked_up_something = 1;
           }

           if (char1 == '2') {  // this means, brush is PAINTING ON CANVAS
               BRUSHLIGHT_OFF;
               picked_up_something = 0;
           }

           // now we blink the green LED on and off, after about 20 requests from the PC lingo code
           serial_traffic++;
           if (serial_traffic >= BLINK_HOW_FAST) {
               serial_traffic = 0; // reset counter

               // now toggle the green LED
               if (green_led_status == 0) { // green LED is currently off
                   GREEN_LED_ON;
                   green_led_status = 1;
               }else{                       // green LED is currently on
                   GREEN_LED_OFF;
                   green_led_status = 0;
               }
           }
        } // end of KBHIT
  } // end of while loop
}  // end of Main

void do_adc ()
{
    int i, j;
    for (i=0; i<4; i++) {
      set_adc_channel(i);
      delay_us(10);
      value[i] = read_adc();
      value_inv[i] = 1023 - value[i];
    }
}
```