For the given code below,
• create a makefile,
• cross-compile for arm,
• transfer the executable to mini2440, and
• set LD_LIBRARY_PATH and
• run on mini2440.
• Also do “cat /dev/input/event0” on mini2440, touch the display, see what happens.

/ * 
* This program tests both the touchscreen and the framebuffer interface 
* To run the program don’t forget to set TSLIB_TSDEVICE and LD_LIBRARY_PATH for 
* MINI2440. 
*/ 
#include <stdio.h> 
#include<stdlib.h> 
#include <signal.h> 
#include <sys/fcntl.h> 
#include <sys/ioctl.h> 
#include <sys/mman.h> 
#include <sys/time.h> 
#include <unistd.h> 
#include <linux/fb.h> 
#include <sys/mman.h> 
#include “tslib.h” 
int main() 
{ 
  struct tsdev *ts; 
  char *tsdevice=NULL; 
  if( (tsdevice = getenv(“TSLIB_TSDEVICE”)) != NULL ) 
  { 
    ts = ts_open(“/dev/input/event0”, 0); 
  
  } 
  if (!ts) 
  { 
    perror(“ts_open”); 
    exit(1); 
  } 
  if (ts_config(ts)) 
  { 
    perror(“ts_config”); 
    exit(1); 
  } 
  int fbfd = 0;
struct fb_var_screeninfo vinfo;
struct fb_fix_screeninfo finfo;
long int screensize = 0;
char *fbp = 0;
int x = 0, y = 0;
long int location = 0;
// Open the file for reading and writing
fbfd = open("/dev/fb0", O_RDWR);
if (!fbfd) {
    printf("Error: cannot open framebuffer device.\n");
    exit(1);
}
printf("The framebuffer device was opened successfully.\n");
// Get fixed screen information
if (ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo)) {
    printf("Error reading fixed information.\n");
    exit(2);
}
// Get variable screen information
if (ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo)) {
    printf("Error reading variable information.\n");
    exit(3);
}
printf("%dx%d, %dbpp\n", vinfo.xres, vinfo.yres, vinfo.bits_per_pixel);
// Figure out the size of the screen in bytes
screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
// Map the device to memory
fbp = (char *)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
if ((int)fbp == -1) {
    printf("Error: failed to map framebuffer device to memory.\n");
    exit(4);
}
printf("The framebuffer device was mapped to memory successfully.\n");
// Where we are going to put the pixel
// Figure out where in memory to put the pixel
while (1) {
    struct ts_sample samp;
    int ret;
    ret = ts_read(ts, &samp, 1);
    if (ret < 0) {
        perror("ts_read");
        exit(1);
    }
    if (ret != 1)
        continue;
printf("%ld.%06ld: %6d %6d %6d\n", samp.tv.tv_sec, samp.tv.tv_usec, samp.x, samp.y, samp.pressure);
for ( y = samp.y; y < samp.y+10; y++ )
  for ( x = samp.x; x < samp.x+10; x++ ) {
    location = (x+vinfo.xoffset) * (vinfo.bits_per_pixel/8) +
               (y+vinfo.yoffset) * finfo.line_length;
    int b = 10;
    int g = (x-samp.x)/6; // A little green
    int r = 31-(y-samp.y)/16; // A lot of red
    unsigned short int t = r<<11 | g << 5 | b;
    *((unsigned short int*)(fbp + location)) = t;
  }
}