Blue Screen

Application Note ANBS1.00.04 [Issued date April 4, 2009]

Keypad demo tutorial

Keypad example code is the simplest one suited for learning how the board works. Then users can apply to other more complex application (for example MP3 player). To read this document users should had already read BlueScreen's user manual first.

1. Screen initialization

We will start at the line ScrObjInit(KeyPadScreenInit) in app_blue_screen_demo.c. This is the start of the application. The body of ScrObjInit() is in screen_obj.c (note that we will not express the suffix version of files in this place, the full name may be screen_obj_v1_00.c or higher).

```
void ScrObjInit(void (*first_screen_init_func)(void))
{
	int i;

	for (i=0; i<MAX_SCR_OBJ; i++)
	{
		layer[i] = i;
		so_obj[i].stat = SO_ST_OFF;
	}

	first_screen_init_func();

	ScrObjDraw();
}
```

Firstly, the function clear status of all objects (in for loop). In our newer version of screen_obj.c, null functions is assigned to so_obj[i].do_, .draw, and other functions of the structure. This will protect the board from

resetting when these functions are executed without pre-assigned address and reduce rubbish code that users need to provide in first version of screen_obj.c (in this keypad example, many null functions like key_task100ms(), key_is_white() are written with doing nothing, but with newer version users have just need to write only what they need to do).

Secondly, first_screen_init_func() do initializing objects including 12 buttons on screen. The original code is KeyPadScreenInit() in app_screen_obj.c (since this is the input parameter of ScrObjInit()), in this function origin of each buttons is assigned and also what they do in each event. Back to ScrObjInit, after initializing, buttons are drawn on screen with ScrObjDraw(). In fact, this is the only time that so_obj[x].draw() is executed in background. In other case, for example when buttons are pressed and there color change, users have to call draw function themselves.

2. Scanning the 'Pen'

Now, we've got 12 buttons on the screen. When users press on screen the pin "TC_PEN" from touch screen controller IC AD7843 goes low. And every 10 ms we scan this pin in AppScanPen() (in app_bluescreen_demo.c), if it goes low, analog value is read through functions TCRead(), TCGetH() and TCGetV() (H refers to horizontal and V refers to vertical) for four times. This is to reduce the error caused by nature of the touch screen (just like de-bouncing mechanical switches).

3. Converting to position and calibration

With help of some mathematic methods, we got accurate value from AD7843 and convert them to position of screen. Ideally, analog value we got is linear proportion to position. So we calculate this with simple linear equation (Y = mX+c). These are done in cal_posx() and cal_posy(). Those constant values are from the pre-calibration stored in EEPROM.

Talking about calibration, least square method is used. In original version of AppCalibrateScreen(), users need to press for 50 times to recalibrate. However, with our current firmware, 5 times are need. Now it depends on user how accurate they press. If it's not good enough, just reset the board and do it again. And also in this new function, recalibration wait time can be

set (in seconds). To skip, just put zero to the function but don't comment it out! Because loading old parameters is done in it.

4. Checking which button is pressed

The global position (as the reference is at top-left of the whole screen) is passed into ScrObjDo(). As the origin of each object is initialized, we can check which object's area the pressing position is in. Then the local position (the reference is at top-left of pressing object) is calculated and is passed through function scr_obj[x].do_(). Anyway, in a button-style object, this value is not used.

The address of scr_obj.do_() is initialized to be keyX_do(). So when button '0' is pressed, key0_do() is called. At this point, users write what they want to be done with each button. Don't forget to check state of the pen (p_stat), unless the process is done every 10 ms whenever that button is pressed (see more detail about Pen status at the appendix of user manual).

5. Switching between screen

In those application which multi-screen are needed. We recommend separating each screen to individual .c files. To switch between screen, just call ScrObjInit(). And we also recommend using version 1.02 (current version) or higher of screen_obj.c. In which, switching is just about setting a flag and done in 100 ms task. This is about to isolate each screen, unless unintentional may occurs since every screen use the same scr_obj[].