Using Matrix Orbital Displays on Linux

Rev 1.1

Document revisions

Version	Date	Author	Remarks
1.0	30-Apr-08	Matrix Orbital	Initial Version
1.1	17-Nov-08	Matrix Orbital	Added inclusion in 2.6.26

Table of Contents

Introduction

Purpose of This Document

We regularly get questions on how to use our displays with Linux. This application note will outline all steps needed to be taken and provide a small programming sample to get your display working on Linux.

Getting the Device Recognized

Serial

If you are using a serial module there are no extra drivers needed, the serial port usually shows up as /dev/ttySx where x is a sequential number for each physical comport available to the system.

USB

Our -24-USB line of products will be recognized automatically, the newer -25-USB line of products have moved to our new Vendor and Product ID's and are therefore not automatically detected by the system if you are running an older kernel. Starting kernel 2.6.26 all our devices are supported by default and no manual steps are needed, if you run an older kernel the "Manually loading the driver" section of this manual will explain the needed steps to do so.

Manually Loading the Driver

To manually load the USB Driver you will need to know the vendor and product ID for which you want to load the driver. The Matrix Orbital Vendor ID is 0x1B3D. The following product ID's have currently been assigned:

ProductID	Product Description
0x0155	LK202-25-USB
0x0156	VK202-25-USB
0x0157	LK204-25-USB
0x0158	VK204-25-USB
0x0127	GLK19264-7T-1U-USB
0x012C	LK204-7T-1U-USB
0x0153	MOU-Axxxx
0x0154	XBoard -U series

To manually load the driver issue the following command :

```
modprobe ftdi_sio vendor=0x1b3d product=????
```

Lookup the product ID in the table above. For instance for the *LK204-25-USB* the command would be :

```
modprobe ftdi_sio vendor=0x1b3d product=0x0157
```

Finding the Device Name

Once the driver is loaded and you plug in your display the following line should appear in your system message log:

```
kernel: usbcore: registered new interface driver usbserial
kernel: /build/buildd/linux-source-2.6.22-2.6.22/drivers/usb/serial/usb-serial.c: USB Serial
support registered for generic
kernel: usbcore: registered new interface driver usbserial_generic
kernel: /build/buildd/linux-source-2.6.22-2.6.22/drivers/usb/serial/usb-serial.c: USB Serial
Driver core
kernel: /build/buildd/linux-source-2.6.22-2.6.22/drivers/usb/serial/usb-serial.c: USB Serial
support registered for FTDI USB Serial Device
kernel: ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
kernel: /build/buildd/linux-source-2.6.22-2.6.22/drivers/usb/serial/ftdi_sio.c: Detected
FT232BM
kernel: usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
kernel: usbcore: registered new interface driver ftdi_sio
```

We can see the new device is called **ttyUSB0** which we then can find in /dev/ttyUSB0

Programming the Display

"hello world" Application

```
//
//DislayTest.cpp
//
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <sys/types.h>

#define BAUDRATE B19200
#define device "/dev/ttyUSB0"

unsigned char ClearScreen[] = {254,88};
unsigned char WelcomeMessage[] = {'H','e','l','l','o',' ','W','o','r','l','d','!'};

int main(int argc, char *argv[])
{
	struct termios term;
	//Open the serial port
	int lcd = open(device, O_RDWR | O_NOCTTY | O_NONBLOCK);
	//handle errors
	if (lcd < 0)
	{
		printf("Error opening serial port %s\n",device);
		return -1;
	}

	//Setup the serial port 8 data bits, no parity ,1 stopbit, no flow control
	term.c_cflag = BAUDRATE | CS8 | CSTOPB | CLOCAL | CREAD;
	term.c_iflag = 0;
	term.c_oflag = 0;
	term.c_lflag = 0;
	tcflush(lcd, TCIFLUSH);
	tcsetattr(lcd,TCSANOW,&term);

	//Write Welcome Message to the display
	write(lcd,ClearScreen,sizeof(ClearScreen));
	write(lcd,WelcomeMessage,sizeof(WelcomeMessage));

	//Close the display
	close(lcd);
	return 0;
}
```

Compiling the “hello world” Application

To compile the application, issue the following command:

```
g++ DislayTest.cpp –o DisplayTest
```

Running the “hello world” Application

To run the hello world application, issue the following command:

```
./DisplayTest
```

Known Issues

Units produced before April 2008 might be incompatible with Linux. To diagnose if your unit has Linux incompatibility issues issue the following command:

```
lsusb –v
```

and search for the following lines in the output:

```
  Endpoint Descriptor:
    bLength             7
    bDescriptorType     5
    bEndpointAddress    0x81  EP 1 IN
    bmAttributes        2
      Transfer Type          Bulk
      Synch Type             None
      Usage Type             Data
    wMaxPacketSize     0x0
    bInterval           0
```

wMaxPacketSize should be 0x40 (64 bytes), if this is not the case please contact support@matrixorbital.ca for instructions how to resolve this issue and get your display working on Linux.