How to interface the BMCM USB-PIO device to a Western Digital MyBook World edition external HDD

(A. Christopoulos, email a.christopoulos@solar-photon.gr)

Introduction

This document describes how to interface the USB-PIO device of BMC Messsysteme GmbH to a Western Digital MyBook World edition (WDMB) device. The How-To describes as detail as I can the whole procedure and may be used as a guideline for interfacing the USB-PIO to other embedded Linux based systems.

The USB-PIO is recognised by the Linux OS as a USB serial communication device. By connecting the USB-PIO to a USB port a device name is assigned to it by the CDC-ACM driver. The CDC ACM driver exposes the USB device as a virtual modem or a virtual COM port to the operating system. The driver enables sending both data and AT commands, either through ACM (separating data and AT commands over different channels) or through Serial Emulation (passing the AT commands as is and as part of the data stream). By attaching the USB-PIO to a USB port the driver will assign a device name to it (normally ttyACM0) which can be also seen in the device directory (/dev).

However the situation in the Western Digital MyBook World edition is a bit different than a normal Linux distribution. In the rest of this document I will explain how to interface the USB-PIO device to WD MyBook assuming that you have already gain access through SSH. If you haven't you can do it by following the instructions of Mr. Martin Hinner (http://martin.hinner.info/mybook/) or by mounting the MyBook HDD to a Linux machine and enable SSH.

The WDMB runs a striped down version of Linux and doesn't have a CDC-ACM driver. By connecting the USB-PIO to a USB port the device should be recognized but not be accessible. To see if the WDMB is visible as a USB device type *lsusb*.

The next step is to build the CDC-ACM for our Linux distribution. For this a Linux PC should be used to build the necessary Kernel modules.

Cross compiling kernel modules using gcc 4.1

As the gcc version used to compile the modules should match the one to compile the kernel, the following steps should be followed.

- First download the sources distributed by WD. (beware: 400MB download!)
- Unpack the WD-GPL-v1.18.tar.bz2 archive
- Unpack the buildroot sources, run "make menuconfig" and set the following options:

    - Target architecture = arm
    - Target architecture variant = arm926t
    - Toolchain/uCLibC library version = daily snapshot
    - GCC compiler version = 4.1.2 (4.1.1 was not available but this one works fine)
    - Any other options can be set according to your requirements

- Use the 4.1.3 gcc version by setting the following environment variables:

    - export HOSTCC=gcc-4.1
    - export HOSTCXX=g++-4.1

- Now run make. Some questions will be asked on the uCLibC configuration. The default answer can be given to all of them, except when it asks whether there is a floating point unit, the answer should be No.
- Run make again, now it should build fine.
- For the following steps, we need the just compiled toolchain binaries to be in the path. This can be done by running:

    - Export PATH=$PATH:<your_buildroot_dir>/build_arm_nofpu/staging_dir/bin

We are almost ready to start building the kernel but before we proceed a change in the source code may be necessary to provide support for the USB-PIO device (this is not always the case as the USB-PIO device is normally supported by the CDC-ACM driver). First of all, we need to make sure that the ID as reported by *lsusb* is matching our modifications. To do so type:

```
# lsusb
Bus 001 Device 002: ID 09ca:5544
```

You will see that its `0x09ca, 0x5544`. This is what we will use in our modification. Modify the cdc-acm.c file:

```
# vi drivers/usb/class/cdc-acm.c
```

scroll down about 97%... put this in similar place, next to all the other devices

```
{ USB_DEVICE(0x09ca, 0x5544), /* BMCM USB-PIO Digital I/O Module */
.driver_info = NO_UNION_NORMAL, /* has no union descriptor */
},
```

Save, and that's all with the source code modification. Now we are ready to build the kernel.

- Go to the directory vendor/linux-kernel in the WD archive.
- Run "make CROSS_COMPILE=arm-linux- oxnas_wd2nc_defconfig"
- Run "make CROSS_COMPILE=arm-linux- menuconfig" and enable all desired modules (in our case we enabled the CDC-ACM module)
- Run "make CROSS_COMPILE=arm-linux- modules"
- Copy the cdc-acm.ko module to your Mybook (e.g., to /lib/modules/2.6.17.14/kernel/drivers/usb/class)

Installing the kernel module

- Edit /lib/modules/2.6.17.14/modules.dep and add the following line: /lib/modules/2.6.17.14/kernel/drivers/usb/class/cdc-acm.ko: /lib/modules/2.6.17.14/kernel/drivers/usb/core/usbcore.ko
- Go to /lib/modules/2.6.17.14/kernel/drivers/usb/class folder and run "modprobe cdc-acm"
- To check if the module is loaded run "lsmod"

Now everything should be ok and by inserting the USB-PIO the device should be assigned a device name and be visible in the /dev folder. If this is not the case you should check if the device has been recognised by running "tail -n 20 /var/log/messages | grep kernel" and if it is create the device listing manually by typing "mknod /dev/ttyACM0 c 166 0".

Installing Optware and Minicom

Optware is software package repositories maintained by NSLU2 project. They offer access to hundreds of precompiled packages with the latest and greatest software, all at your fingertips. To start using Minicom, you need first to install package manager. Optware uses ipkg**,** which is standard package manager for many embedded distributions. Following commands manually bootstrap ipkg from the feed.

```
# wget http://mybookworld.wikidot.com/local--files/optware/setup-optware.sh
# sh setup-optware.sh
# echo "/opt/lib" >>/etc/ld.so.conf
# ldconfig
# export LD_LIBRARY_PATH=/opt/lib
```

Update local feed lists

```
# /opt/bin/ipkg update
```

Check if Minicom is available

```
# /opt/bin/ipkg list |grep minicom
```

Install Minicom

```
# /opt/bin/ipkg install minicom
```

Communicating to USB-PIO

After configuring Minicom to use the ttyACM0 device we are ready to start communicating with the USB-PIO.

To set the digital outputs you have to send something like @00Pnxx\r to the device. n must be replaced by the port number (0 to 2) and xx by the value.

Maybe a simple example might help. To set the first output to 0xa5 your communication should look like this (=> denotes what you send to the USB-PIO, <= denotes what you get back from the device, \r is carriage return):

```
=> @00P0A5\r
<= !00A5\r
```

To read the digital port you have to send @00Pn?. So reading the second port looks like this:

```
=> @00P1?\r
<= !00A5\r
```

Changing the port's direction can be done by sending @00Dnxx\r. As you might have guessed, n gets replaced by the port number (0 to 2) and xx gets replaced by the direction. There are two direction codes: 0xff denotes an input port and 0x00 denotes an output port.

Setting port 3 to output:

```
=> @00D200\r
<= !0000\r
```

And back to input:

```
=> @00D2FF\r
<= !00FF\r
```

Learning the port's current direction can be done by sending @00Dn?\r.

Loading CDC-ACM module and Environmental variables at start up

Run vi /root/.bashrc and add the following two lines at the end of the file.

```
export LD_LIBRARY_PATH=/opt/lib
export PATH=$PATH:/opt/bin
```

Then run vi /etc/profile and add once again the two lines as before.

Go to /etc/init.d and run "vi S90cdc-acm". Type in "modprobe cdc-acm", save and exit.
Reboot your MyBook!

References and Useful links

http://www.bmcm.de
http://support.wdc.com/product/download.asp?groupid=107&sid=64&lang=en
http://martin.hinner.info/mybook/
http://mybookworld.wikidot.com
http://yoonkit.blogspot.com/2008/09/getting-aztech-um-3100-usb-modem-to.html