MEMU-Pi Hardware Config

Introduction

As well as providing a version of Andy Key's excellent MTX emulator on the Raspberry Pi, MEMU-Pi provides the ability to interface with legacy hardware to provide an even more authentic experience.

Since different people may have different requirements, and may wish to connect hardware in different fashions, the software has been made flexible in terms of hardware configuration. This is achieved via a hardware configuration file, which is described in this document.

The hardware configuration file consists of blocks defining a particular interface, introduced by a name in square brackets, followed by the definition of that interface.

Pin Definitions

Most of the hardware definition file is defining which digital I/O pins on the Raspberry Pi are connected to what hardware. These may be the Raspberry Pi's built in GPIO pins. Or, since some applications may require more I/O than is built-in, there is also support for one or more MCP23017 I2C port expanders.

These pin definitions take the form:

```
GPIO, <bcm pin no>

MCP23017, <i2c dev>, <i2c addr>, <mcp pin no>
```

The first form is used for the Raspberry Pi's built-in GPIO pins. The pin number is specified according to the Broadcom convention.

The second form is used for a port expander, where:

<i2c dev> =	The device name of the I2C bus used. This will typically be "/dev/i2c-1" on more recent Raspberry Pis. It may be "/dev/i2c-0" for Rev.1 Raspberry Pis, or if the nominally reserved GPU I2C bus is being used.
<i2c addr> =	The I2C address of the port expander. This will typically be 0x20, although up to 8 simultaneous port expanders on different addresses are supported.
<mcp pin no> =	The number of the pin on the port expander. This may either be a number in the range 0-15, or a letter and number in the ranges A0-A7 or B0-B7.

For speed reasons, it is recommended that, when possible, keyboard or joysticks are connected to built-in GPIO pins.

Keyboard

This block of the configuration file is used to define the connection of a MTX matrix keyboard or equivalent. It takes the form:

```
[keyboard]
kb0 = <pin definition>
     :
kb9 = <pin definition>
dr0 = <pin definition>
     :
dr7 = <pin definition>
reset = <pin definition>
reset2 = <pin definition>
dr_reset = <gnd | dr0-dr7>
```

The lines kb0 – kb9 define the 10 keyboard sense lines, and the lines dr0-dr7 define the 8 keyboard drive lines.

For an unmodified MTX keyboard, the reset line defines the I/O pin that one of the keyboard reset lines is connected to. The other reset line should be connected to ground (0v). In that case the reset2 and dr_reset lines would be omitted.

The MTX keyboard may be modified by soldering an additional connection to the track joining the two reset keys. In that case both the existing keyboard reset lines should be connected to IO pins, and defined by the reset and reset2 lines. The additional connection between the reset keys should either be connected to ground (0v) or to one of the keyboard drive lines. The dr_reset line specifies how this is connected (gnd is the default).

Joysticks

If a matrix keyboard is fitted, then Atari style joysticks may be connected to the drive and sense lines, in parallel with the keyboard, as per the MTX. In that case no joystick definition is required. Alternately, the separate joystick switches may be connected to I/O pins, and the common connection to ground, as per earlier versions of MEMU-Pi. In that case, the joystick connections are defined by a block of the form:

```
[joystick_1]
left = <pin definition>
right = <pin definition>
up = <pin definition>
down = <pin definition>
fire = <pin definition>
```

The second joystick, if fitted, is defined similarly in a [joystick_2] block.

Printer

A Centronics style printer port may be provided. It should be noted that the Centronics connector uses 5v logic, while the Raspberry Pi GPIO connections are only 3.3v and are not 5v tolerant. Therefore some form of level shifting is required, Probably the simplest solution is to use an MCP23017 powered from 5v, and then use I2C compatible level shifters between that and the Raspberry Pi. The printer hardware block takes the form:

```
[printer]
d0 = <pin definition>
     :
d7 = <pin definition>
strobe = <pin definition>
busy = <pin definition>
error = <pin definition>
pe = <pin definition>
slct = <pin definition>
```

Parallel Input/Output Port

Note that the MTX PIO port is 5v logic while the Raspberry Pi GPIO connections are only 3.3v tolerant. Also it is not practical to implement the INSTB and OTSTB lines in software. HCT244, HCT245 or HCT373 are examples of devices that could provide an OTSTB function and 3.3v to 5v step-up (not HC chips). A HC373 or HCT373 are examples that could provide INSTB, but the outputs would need resistor dividers to connect to the Raspberry Pi GPIO. The PIO hardware definition block takes the form:

```
[pio]
pot0 = <pin definition>
     :
pot0 = <pin definition>
pin0 = <pin definition>
     :
pin7 = <pin definition>
```

Example

An example of a complete hardware configuration file for a matrix keyboard and Centronics printer is shown below:

```
[keyboard]
kb9 = gpio,  4
kb8 = gpio, 17
kb7 = gpio, 27
kb6 = gpio, 22
kb5 = gpio, 10
kb4 = gpio,  9
kb3 = gpio, 11
kb2 = gpio,  5
kb1 = gpio,  6
kb0 = gpio, 13
dr0 = gpio, 19
dr1 = gpio, 26
dr2 = gpio, 21
dr3 = gpio, 20
dr4 = gpio, 16
dr5 = gpio, 12
dr6 = gpio,  7
dr7 = gpio,  8
reset = gpio, 25
[printer]
d0 = mcp23017, “/dev/i2c-1”, 0x20, a0
d1 = mcp23017, “/dev/i2c-1”, 0x20, a1
d2 = mcp23017, “/dev/i2c-1”, 0x20, a2
d3 = mcp23017, “/dev/i2c-1”, 0x20, a3
d4 = mcp23017, “/dev/i2c-1”, 0x20, a4
d5 = mcp23017, “/dev/i2c-1”, 0x20, a5
d6 = mcp23017, “/dev/i2c-1”, 0x20, a6
d7 = mcp23017, “/dev/i2c-1”, 0x20, a7
busy = mcp23017, “/dev/i2c-1”, 0x20, b0
error = mcp23017, “/dev/i2c-1”, 0x20, b1
pe = mcp23017, “/dev/i2c-1”, 0x20, b2
slct = mcp23017, “/dev/i2c-1”, 0x20, b3
strobe = mcp23017, “/dev/i2c-1”, 0x20, b7
```