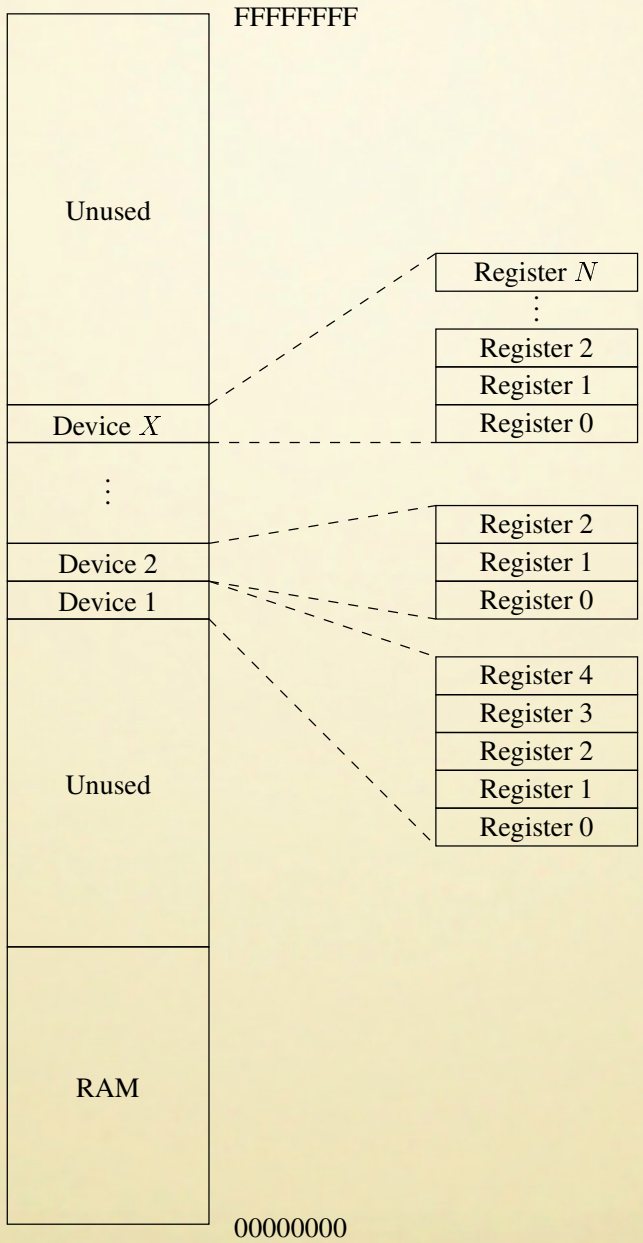
Modern Assembly Language Programming with the ARM processor

Chapter 11: Devices

Memory-Mapped Devices



Raw Access to Device Registers

```
NAME
       mmap, munmap - map or unmap files or devices into memory

SYNOPSIS
       #include <sys/mman.h>

       void *mmap(void *addr, size_t length, int prot, int flags,
                  int fd, off_t offset);
       int munmap(void *addr, size_t length);

       See NOTES for information on feature test macro requirements.

DESCRIPTION
       mmap()  creates a new mapping in the virtual address space of the callâĂŘ
       ing process.  The starting address for the new mapping is specified  in
       addr.  The length argument specifies the length of the mapping.

       If addr is NULL, then the kernel chooses the address at which to create
       the mapping; this is the most portable method of creating  a  new  mapâĂŘ
       ping.   If  addr  is not NULL, then the kernel takes it as a hint about
       where to place the mapping; on Linux, the mapping will be created at  a
       nearby  page  boundary.   The address of the new mapping is returned as
       the result of the call.
```

Mapping a Device in C

The program must have permissions to read and write `/dev/mem`.

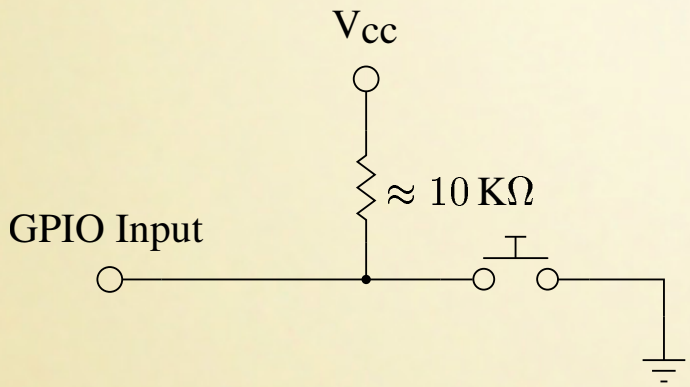
```
#include <sys/mman.h>
#include <fcntl.h>
    :
  void *devaddress = HARDWARE_ADDRESS;
  int memfd;
    :
  memfd = open("/dev/mem",O_RDWR);
  if(memfd<0) exit(1);
  devaddress = mmap(devaddress,4096,PROT_READ|PROT_WRITE,
                    MAP_SHARED,memfd,0);
  /* devaddress now points to the mapped device registers */
    :
  /* Finished accessing the device registers .  Clean up. */
  munmap(devaddress,4096);
  close(memfd);
    :
```

General Purpose Input/Output Device (GPIO)

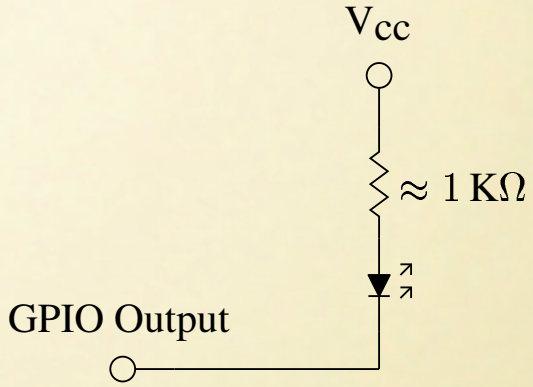
For most GPIO devices:

- individual pins or groups of pins can be configured,
- pins can be configured to be input or output,
- pins can be disabled so that they are neither input nor output,
- input values can be read by the CPU (typically high=1, low=0),
- output values can be read or written by the CPU, and
- input pins can be configured to generate interrupt requests.

Input and Output



GPIO pin being used as input to read the state of a push-button switch.



GPIO pin being used as output to drive an LED.

Raspberry Pi GPIO

The Raspberry Pi is based on the Broadcom BCM2835.

- The GPIO device controls 54 pins.
- Pins are named GPIOx, where x is a number between 0 and 53.
- Each pin can be used
  - for general purpose input,
  - for general purpose output, or
  - for one of six pre-defined alternate functions.

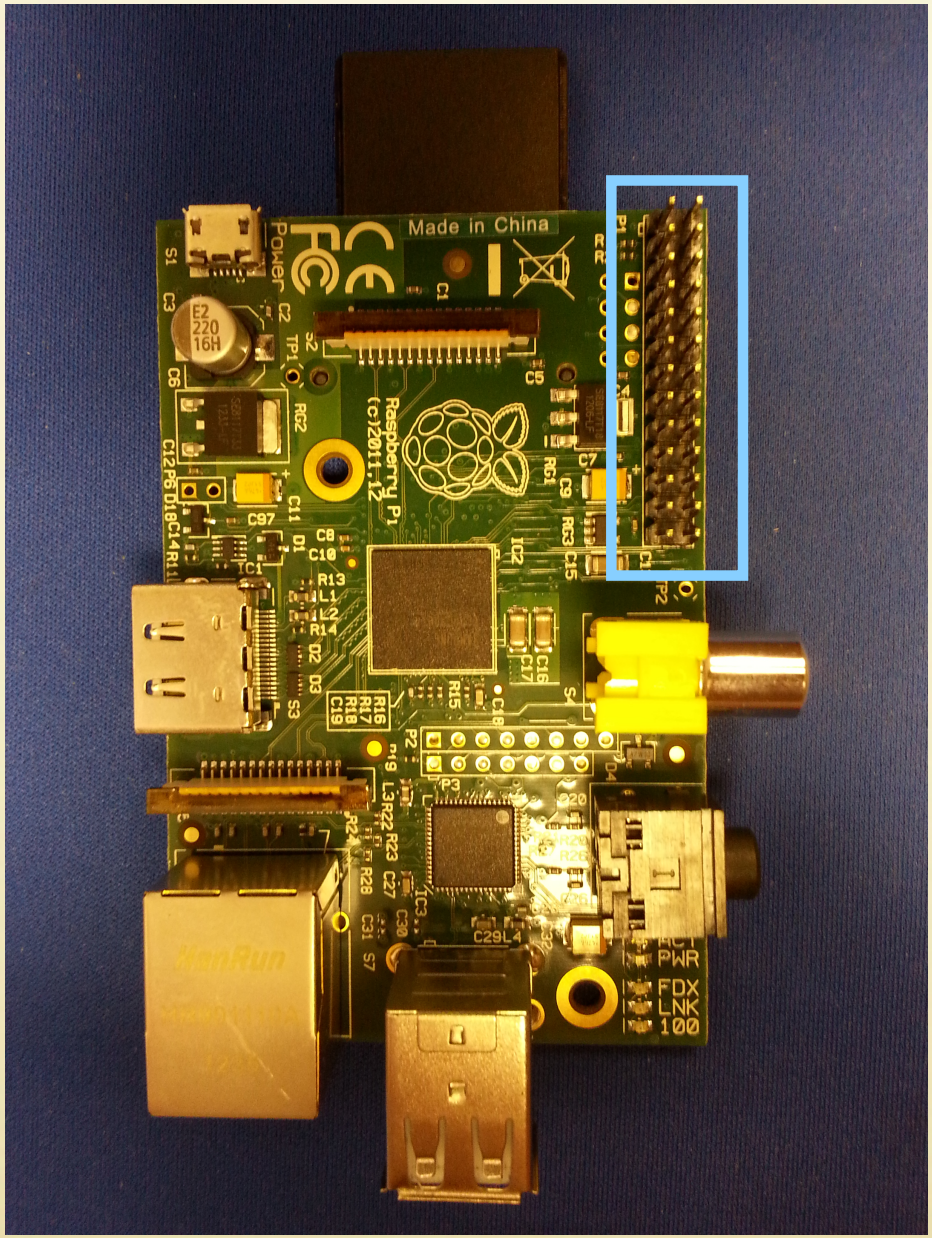
Raspberry Pi GPIO

For example, GPIO4 can be used

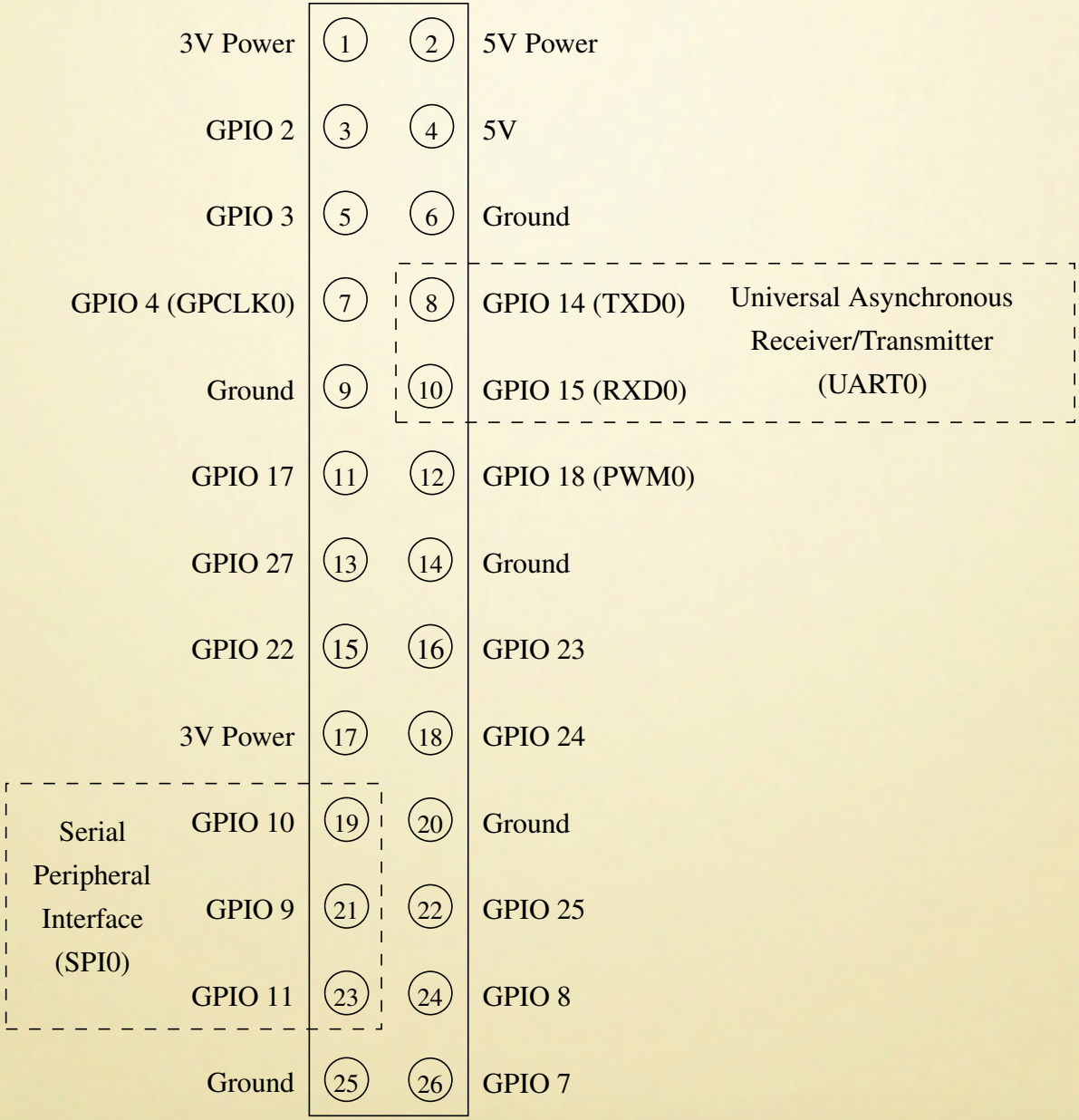
- for general purpose I/O,
- to send the signal generated by General Purpose Clock 0 to external devices,
- to send bit one of the Secondary Address Bus to external devices, or
- to receive JTAG data for programming the firmware of the device.

The last 8 GPIO pins, GPIO46–GPIO53 have no alterate functions, and are used only for GPIO.

Raspberry Pi GPIO



Raspberry Pi GPIO



pcDuino GPIO

The pcDuino is based on the AllWinner A10/A20.

- The GPIO device controls 175 pins.
- Pins are arranged in seven ports.
- Each of the seven ports is identified by a letter between “A” and “I”.
- Some ports have as many as 28 physical pins, while others have as few as six.
- Regardless of the number of pins, the port registers are all laid out the same way on every port.
- Each pin can be used
  - for general purpose input,
  - for general purpose output, or
  - for one of six pre-defined alternate functions.

pcDuino GPIO

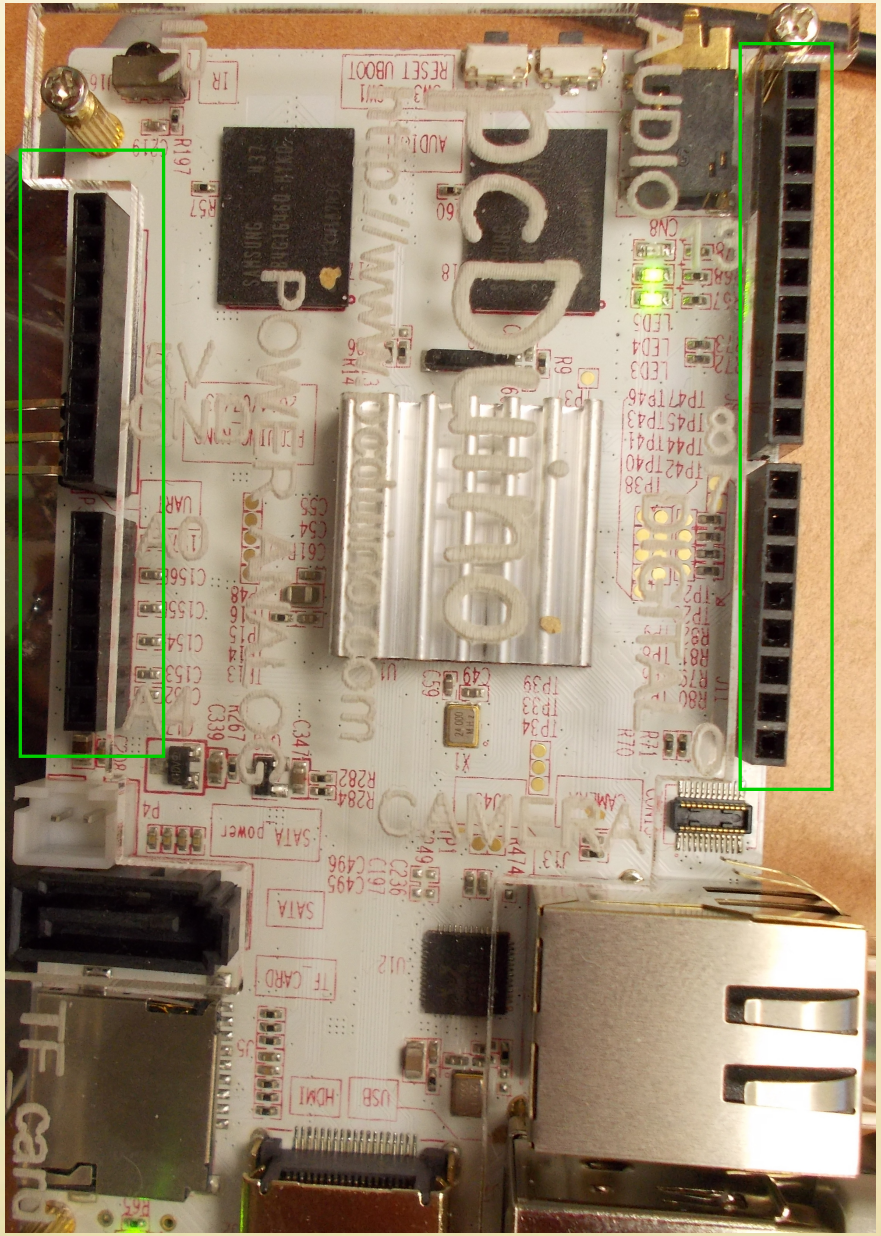
For example, Port B, Pin 2 can be used

- for general purpose I/O, or
- to send the output of Pulse Width Modulator 0 to an external device.

Port I, Pin 19 can be used

- for general purpose I/O,
- to receive RS232 serial data, or
- to route an interrupt signal from an external device to the interrupt controller.

pcDuino GPIO



pcDuino GPIO