STM32 GPIO

The STM32 GPIO (general purpose i/o) pins are very flexible, and, like every silicon vendor's, rather idiosyncratic. After perusing the STM32F4 reference manual I decided to do a quick write-up of my understanding of how to use the pins.

These are probably the same across *all* the STM32 parts, but at the moment I have not verified this.

A few quick points:

- Each port is 16 bits wide
- Any pin of any port can be connected to *any* of 16 alternate functions (AFs)
- Any combination of bits in the output data register can be *set* or *reset* with a single, word-wide store (of two 16-bit masks) to a set/reset register
- Every pin can be configured as input, output, alternate function (AF), or analog;
- .. can be push-pull (ie, totem-pole) or open-drain;
- .. can have a weak pull-up, weak pull-down, or none;
- .. and can have a configured slew rate (ie, toggle speed)

With that in mind, here is the register layout for each port – the base addresses are at 1 KiB (0x0400) boundaries.

offset	register	description
0x00	GPIO_MODER	port mode register
0x04	GPIO_OTYPER	output type register
0x08	GPIO_OSPEEDR	output speed (slew-rate) register
0x0c	GPIO_PUPDR	pull-up/pull-down register
0x10	GPIO_IDR	input data register
0x14	GPIO_ODR	output data register
0x18	GPIO_BSRR	bit set/reset register

0x1c GPIO_LCKR configuration lock register

0x20 GPIO_AFRL alternate function low register (pins 0-7)

0x24 GPIO_AFRH alternate function high register (pins 8-15)

We're going to ignore the GPIO_LCKR.

GPIO_MODER

This register consists of 16 two-bit fields, one for each port pin. Bits 0 and 1 set the mode for pin 0; bits 2 and 3 for pin 1, etc.

Here is the mode table for each pin:

00 input

01 output

10 alternate function (AF)

11 analog

GPIO_OTYPER

This register has 1 bit per pin; it is only 16 bits wide.

For any port pin that GPIO_MODER has configured as an output, OTYPER sets the drive configuration:

0 push-pull (totem-pole) output

1 open-drain output

GPIO_OSPEEDR

This register consists of 16 two-bit fields, one for each port pin. Bits 0 and 1 set the mode for pin 0; bits 2 and 3 for pin 1, etc.

Here is the output speed table for each pin:

00 2 MHz (low speed)

01 25 MHz (medium speed)

10 50 MHz (fast speed)

11 100 MHz (fucking ridiculous!)

GPIO_PUPDR

This register consists of 16 two-bit fields, one for each port pin. Bits 0 and 1 set the mode for pin 0; bits 2 and 3 for pin 1, etc.

Here is the pull-up/pull-down table for each pin:

00 No pull-up or pull-down

01 Pull-up

10 Pull-down

11 Reserved (don't try this at home!)

GPIO_IDR

The input data register has 16 bits; the high 16 bits are reserved, and read as zero. When read, the register returns the logic level present on the pin (after synchronisation with the bus clock).

GPIO_ODR

The output data register has 16 bits; the high 16 bits are reserved, and should be preserved as zero (the reset value).

If the corresponding pin is configured as an output, the value of each of the low 16 bits is driven onto the pin.

GPIO_BSRR

This register consists of two write-only bit masks, each 16-bits wide. The high half – bits 16 to 31 – is the *reset* mask. Writing a 1 to any of these bits *clears* bit N-16 in the GPIO_ODR.

The low half – bits 0 to 15 – is the *set* mask. Writing a 1 to any of these bits *sets* the corresponding bit in the GPIO_ODR.

By judiciously choosing a combined mask, it is possible to *set* and *reset* a combination of bits in the GPIO_ODR in one word-wide write.

GPIO_AFRL and GPIO_AFRH

These two registers, each of which has 8 four-bit fields, describe which alternate function to attach to each port pin, when the corresponding bits in the GPIO_MODER configure the pin for the alternate function.

GPIO_AFRL contains the multiplexer settings for port bits 0 to 7; GPIO_AFRH, for bits 8 to 15. The actual map from AF number to peripheral function is different for each chip, so I won’t try to put a table here.