Entradas -> alta impedancia / resistencias pull up/down

Salidas -> push/pull / open collector

**REGISTROS PRINCIPALES PARA CONFIGURAR CADA PIN**

Pinsel - Funcion que se desea (creo)

Pinmode - Tipo de entrada (si lo es)

Pinmode\_od - Tipo de salida (si lo es)

**PINSEL**

Los registros pinsel son para alternar, elegir las funciones de cada pin. Entre 0 y 3

Por cada pin, nesecito 2 bits en el registro para que cada pin tenga 2bits. Ya que puede tener 4 funciones cada pin entonces con dos bits le asigno 00 01 10 11.

Puerto 0 tiene 32 pines por lo q uso pinsel 0 y pin sel 1 (cad uno tiene un registro de 32bits)

PINSEL son como los “nombres” de los bloques de registros, cada pinsel tiene 32 bits.

Tras reset, todos amanecen x defecto como GPIO pero por buena practica debo siempre forzar el valor que nesecito

**PINMODE**

Los registros pinmode son para alternar entre que tipo de entrada sera cada pin.

Tambien, 2 bits por cada pin, y se manejan igual q e registro pinsel. (para pin 1 uso bits 0 y 1 de pinmode0, y para pin 16 uso bits 2 y 3 de pinmode1).

Todos amanecen tras reset como pullup interno pero igual q antes, debo forzar lo q nesecito.

Si lo voy a manjear como salida solo uso pinmodeod, no pinmode

**PINMODE\_OD**

Registros para asignar tipo de salida a cada pin.

2 bits por pin, ya que solo hay 2 tipos de salidas, normal y open drain

**En pinsel y pinmode**

Para usar mascaras, si x ej estoy en pinsel1 debo hacer 2\*(pin - 16)

**OTROS REGISTROS**

**FIODIR**

Lo toco solo una vez para decir si tal pin es entrada o salida

**FIOMASK**

Mascara (0=enable)

**FIOPIN**

Lectura/escritura de pines (fiomask=0 => todo habilitado)

**FIOSET, FIOCLEAR**

Son para poner 0 o 1 en un bit de manera mas facil. Fioclear ponemos 1 en tal bit y lo deja en 0, fioset pongo 1 en tal pin y se pone en 1.

Estos dos reg. modifican a fiopin, pero facilitandonos el trabajo.

**INTERRUPCIONES**

ISER 0 y 1 -> registros de 32 bits para habilitar interrupciones

ICER 0 y 1-> registros de 32 bits para deshabilitar interrupciones

EXTINT -> registro de 32 bits, con 4 habilitados, uno por interrupcion, cuyos bits son flags para marcar si hay una interrupcion, y si se la atendio. Al atender la inter. Se pone el bit en 1

EXMODE -> reg de 32 bits con 4 habilitados para configurar si la interrupcion se activara por nivel(0) o por flanco(1)

EXTPOLAR -> reg de 32bits con 4 habilirados para configurar si se trabaja por nivel bajo/alto o por flanco ascen/descen

Ejemplo led:

Configuro fuente de interrupcion:

Uso Setpinsel y selecciono funciond e interrupcion -> extint (limpio flags con 1) -> Extmode -> Extpolar -> iser

Luego para lo que hara el handler:

Uso setpinsel y selecciono gpio -> setdir en salida (led)

El lpc tiene 4 interrupciones,

**CAPAS**

Aplicadion -> maquina de estados

Primitivas -> prenderLED, apagarLED, etc. Funciones q hacen cosas q se usan en la app

Buffers-> variables globales comunes a varios modulos

Drivers-> funciones q se usan en las primitivas q ya manejan el hardware a bajo nivel. Setpin getpin y las ISR de las interrupciones

INTERRUPCIONES

EXMODE -> configura si la interrupcion es por nivel o por flanco. Por nivel importa el valor q tenga la senal de entrada (si esta en uno interrumpo, puede llevar a infinita interrupcion). Por flanco interesa la transicion (ascendente seria siempre q la senal de entrada cambio de 0 a 1. Pero interesa el momento de la transc., no el estado en el que queda).

EXPOLAR -> si es por nivel, si sera alto o bajo, y si es por flanco, si sera ascndente o descendente

Rutinas ISR -> rutinas de atencion al servicio de interrupcion

Tabla de vectores -> ahi asignamos cada interrupcion de los perifericos a un handler especifico

Las funciones handler de esta tabla son void void porq como se las llama asincronicamente, en un instante “randon”, no tengo algo para pasarle, o un lugar donde guardar el retorno

**TIMERS**

Pasos para inicializar: cuenta que se desea cargar, modificar systick ctrl (enable, tickint, clksource)