.cpu arm7tdmi

.eabi\_attribute 20, 1

.eabi\_attribute 21, 1

.eabi\_attribute 23, 3

.eabi\_attribute 24, 1

.eabi\_attribute 25, 1

.eabi\_attribute 26, 1

.eabi\_attribute 30, 6

.eabi\_attribute 34, 0

.eabi\_attribute 18, 4

.file "tec\_setup.c"

.text

;declaración funcion TEC\_cfg

.align 2

.global TEC\_cfg

.syntax unified

.arm

.fpu softvfp

.type TEC\_cfg, %function

TEC\_cfg:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 16

@ frame\_needed = 1, uses\_anonymous\_args = 0

push {fp, lr} ;guarda el “frame pointer” y el “Link register” en el stack

add fp, sp, #4 ;espacio para el puntero

sub sp, sp, #16 ;espacio para lo que ocupa el frame

mov r3, r0 ;copia el argumento de la funcion (tec) en r3

strb r3, [fp, #-13] ;guarda en la posicion fp-13,que ahora corresponde a la

;variable tec

mov r3, #1 ;copia 1 en r3 correspondiente a port

strb r3, [fp, #-8] ;lo guarda en fp-8, es decir fp-8 corresponde a la variable port

mov r3, #0 ;copia un 0 en r3 correspondiente a function.

strb r3, [fp, #-9] ;fp-9 ahora corresponde a la variable function

mov r3, #6 ;copia un 6 en r3 corresponde a ezi

strb r3, [fp, #-10] ;fp-10 ahora corresponde a la variable ezi

;if(tec==TEC1 || tec==TEC2 || tec==TEC3)

ldrb r3, [fp, #-13] @ zero\_extendqisi2 ;carga tec

cmp r3, #0 ;compara con 0 (TEC1)

beq .L2 ;si la comparación es true → branch dentro del if

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #1 ;compara con 1 (TEC2)

beq .L2 ;si la comparación es true → branch dentro del if

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #2 ;compara con 2 (TEC3)

bne .L3 ;si la comparación es false → branch al siguiente else if

.L2:

mov r3, #0

strb r3, [fp, #-6] ;guarda un 0 en fp-6 que corresponde a la variable port\_gpio

;if(tec==TEC1)

ldrb r3, [fp, #-13] @ zero\_extendqisi2 ;carga tec

cmp r3, #0

bne .L4 ;si el compare con TEC1 es false → branch siguiente else if

mov r3, #0

strb r3, [fp, #-5] ;guarda un 0 en fp-5 que corresponde a pin

mov r3, #4

strb r3, [fp, #-7] ;guarda un 4 en fp-7 que corresponde a pin\_gpio

b .L8 ;branch incondicional fuera del if

.L4:

;else if(tec==TEC2)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #1

bne .L6 ;si cmp con TEC2 da false → branch al siguiente else if

mov r3, #1

strb r3, [fp, #-5] ;guarda un 1 en fp-5 que corresponde a pin

mov r3, #8

strb r3, [fp, #-7] ;guarda un 8 en fp-7 que corresponde a pin\_gpio

b .L8 ;branch incondicional fuera del if

.L6:

else if(tec==TEC3)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #2

bne .L8 ;si cmp con TEC3 da false → branch fuera del if

mov r3, #2

strb r3, [fp, #-5] ;guarda un 5 en fp-5 que corresponde a pin

mov r3, #9

strb r3, [fp, #-7]] ;guarda un 9 en fp-7 que corresponde a pin\_gpio

b .L8 ;branch incondicional fuera del if

.L3:

;else if(tec==TEC4)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #3

bne .L7 ;si cmp con TEC4 da false → branch fuera del if

mov r3, #6

strb r3, [fp, #-5] ;guarda un 6 en fp-5 que corresponde a pin

mov r3, #1

strb r3, [fp, #-6] ;guarda un 1 en fp-6 que corresponde a la variable port\_gpio

mov r3, #9

strb r3, [fp, #-7] ;guarda un 9 en fp-7 que corresponde a pin\_gpio

b .L7 ;branch incondicional fuera del if

.L8:

nop

.L7:

ldrb r3, [fp, #-9] @ zero\_extendqisi2

ldrb r2, [fp, #-5] @ zero\_extendqisi2

ldrb r1, [fp, #-8] @ zero\_extendqisi2

ldr r0, .L9 ;carga function, pin, port, y la direccion de SCU

SCU\_SetPin(SCU, port, pin, function);

bl SCU\_SetPin

ldrb r3, [fp, #-10] @ zero\_extendqisi2

ldrb r2, [fp, #-5] @ zero\_extendqisi2

ldrb r1, [fp, #-8] @ zero\_extendqisi2

ldr r0, .L9 ;carga ezi, pin, port, y la direccion de SCU

SCU\_SetEZI(SCU, port, pin, ezi);

bl SCU\_SetEZI

ldrb r2, [fp, #-7] @ zero\_extendqisi2

ldrb r1, [fp, #-6] @ zero\_extendqisi2

mov r3, #0

ldr r0, .L9+4 ;carga pin\_gpio, port\_gpio, 0 (INPUT) y la direccion de GPIO

GPIO\_SetPinDIR(GPIO, port\_gpio, pin\_gpio, INPUT);

bl GPIO\_SetPinDIR

nop

sub sp, fp, #4

@ sp needed

pop {fp, lr} ;saca el frame pointer y el lr del stack

bx lr ;branch a la direccion del lr que se habia guardado en la pila

.L10:

.align 2

.L9:

.word 1074290688

.word 1074741248

.size TEC\_cfg, .-TEC\_cfg

;declaracion de la funcion TEC\_GetState

.align 2

.global TEC\_GetState

.syntax unified

.arm

.fpu softvfp

.type TEC\_GetState, %function

TEC\_GetState:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 16

@ frame\_needed = 1, uses\_anonymous\_args = 0

push {fp, lr} ;guarda el “frame pointer” y el “Link register” en el stack

add fp, sp, #4

sub sp, sp, #16

mov r3, r0

strb r3, [fp, #-13] ;guarda tec en fp-13

;if(tec==TEC1 || tec==TEC2 || tec==TEC3)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #0

beq .L12

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #1

beq .L12

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #2

bne .L13

.L12:

mov r3, #0

strb r3, [fp, #-5]

;if(tec==TEC1)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #0

bne .L14

mov r3, #4

strb r3, [fp, #-6]

b .L19

.L14:

;else if(tec==TEC2)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #1

bne .L16

mov r3, #8

strb r3, [fp, #-6]

b .L19

.L16:

;else if(tec==TEC3)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #2

bne .L19

mov r3, #9

strb r3, [fp, #-6]

b .L19

.L13:

;else if(tec==TEC4)

ldrb r3, [fp, #-13] @ zero\_extendqisi2

cmp r3, #3

bne .L17

mov r3, #1

strb r3, [fp, #-5]

mov r3, #9

strb r3, [fp, #-6]

b .L17

.L19:

nop

.L17:

ldrb r2, [fp, #-6] @ zero\_extendqisi2

ldrb r3, [fp, #-5] @ zero\_extendqisi2

mov r1, r3

ldr r0, .L20

;GPIO\_GetPinState(GPIO, port\_gpio, pin\_gpio);

bl GPIO\_GetPinState

mov r3, r0

mov r0, r3

sub sp, fp, #4

@ sp needed

pop {fp, lr} ;saca el frame pointer y el lr del stack

bx lr ;branch a la direccion del lr que se habia guardado en la pila

.L21:

.align 2

.L20:

.word 1074741248

.size TEC\_GetState, .-TEC\_GetState

Declaracion de la funcion TEC\_ALL

.align 2

.global TEC\_ALL

.syntax unified

.arm

.fpu softvfp

.type TEC\_ALL, %function

TEC\_ALL:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 0

@ frame\_needed = 1, uses\_anonymous\_args = 0

push {fp, lr} ;guarda el “frame pointer” y el “Link register” en el stack

add fp, sp, #4

mov r0, #0

bl TEC\_cfg ;branch a la funcion TEC\_cfg con TEC1 como argumento

mov r0, #1

bl TEC\_cfg ;branch a la funcion TEC\_cfg con TEC2 como argumento

mov r0, #2

bl TEC\_cfg ;branch a la funcion TEC\_cfg con TEC3 como argumento

mov r0, #3

bl TEC\_cfg ;branch a la funcion TEC\_cfg con TEC4 como argumento

nop

sub sp, fp, #4

@ sp needed

pop {fp, lr} ;saca el frame pointer y el lr del stack

bx lr ;branch a la direccion del lr que se habia guardado en la pila

.size TEC\_ALL, .-TEC\_ALL

.ident "GCC: (15:6.3.1+svn253039-1build1) 6.3.1 20170620"