Drivers PicTrainer

Generado por Doxygen 1.8.11

Índice

1. Driver de la tarjeta PicTrainer V3

1.1. Introducción

Este driver facilita el manejo de la tarjeta PicTrainer V3. El driver incluye módulos para manejar el ADC, el PWM, la UART y para implantar un bucle de scan con una tarea idle que controla el periodo del bucle.

En la pestaña Archivos encontrará la documentación de cada uno de estos módulos.

1.2. Instalación

Se recomienda descomprimir la carpeta en el directorio raiz de sus proyectos y añadir la carpeta al entorno de desarrollo MPLAB X. En este caso para usar cada módulo ha de incluir su .h de la forma: #include "../DriverPic↩ Trainer/adc.h"

2. Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

3. Documentación de archivos

3.1. Referencia del Archivo adc.c

Módulo encargado de gestionar el conversor AD del microcontrolador.

```
#include <xc.h>
#include "adc.h"
```

Funciones

- void inicializarADCPolling (unsigned int input_pins)
- unsigned int leerADCPolling (unsigned int canal)

3.1.1. Descripción detallada

Módulo encargado de gestionar el conversor AD del microcontrolador.

Autor

José Daniel Muñoz Frías

Versión

1.0.1

Fecha

08/09/2015

3.1.2. Documentación de las funciones

3.1.2.1. void inicializarADCPolling (unsigned int *input_pins*)

Inicializar el conversor AD en modo pooling

Inicializa el conversor AD para funcionar por polling. Es necesario pasarle a la función un parámetro que indique qué entradas analógicas se van a convertir. Para ello se usa el argumento poniendo a 1 los bits cuyas entradas analógicas queramos usar. Por ejemplo si se van a usar las entradas AN0 y AN2 el argumento input_pins será igual a 0x05.

Parámetros

input_pins	bitmap con los pines que se van a usar como entradas analógicas

3.1.2.2. unsigned int leerADCPolling (unsigned int *canal*)

Lee el canal indicado en el parámetro. Para ello lanza una conversión y se bloquea a la espera de que finalice dicha conversión.

Parámetros

canal	Número del canal que se desea leer

Devuelve

Valor leído del ADC

3.2. Referencia del Archivo adc.h

Módulo encargado de gestionar el conversor AD del microcontrolador.

Funciones

- void inicializarADCPolling (unsigned int input_pins)
- unsigned int leerADCPolling (unsigned int canal)

3.2.1. Descripción detallada

Módulo encargado de gestionar el conversor AD del microcontrolador.

Autor

José Daniel Muñoz Frías

Versión

1.0.1

Fecha

08/09/2015

3.2.2. Documentación de las funciones

3.2.2.1. void inicializarADCPolling (unsigned int *input_pins*)

Inicializar el conversor AD en modo pooling

Inicializa el conversor AD para funcionar por polling. Es necesario pasarle a la función un parámetro que indique qué entradas analógicas se van a convertir. Para ello se usa el argumento poniendo a 1 los bits cuyas entradas analógicas queramos usar. Por ejemplo si se van a usar las entradas AN0 y AN2 el argumento input_pins será igual a 0x05.

Parámetros

input_pins	bitmap con los pines que se van a usar como entradas analógicas

3.2.2.2. unsigned int leerADCPolling (unsigned int *canal*)

Lee el canal indicado en el parámetro. Para ello lanza una conversión y se bloquea a la espera de que finalice dicha conversión.

Parámetros

canal	Número del canal que se desea leer

Devuelve

Valor leído del ADC

3.3. Referencia del Archivo config.c

Funciones de configuración del microcontrolador.

```
#include <xc.h>
#include "config.h"
```

Funciones

- void inicializarReloj (void)

3.3.1. Descripción detallada

Funciones de configuración del microcontrolador.

Autor

Jaime Boal Martín-Larrauri

Versión

2.0.0

Fecha

26/08/2016

3.3.2. Documentación de las funciones

3.3.2.1. void inicializarReloj (void)

Inicializa el reloj interno FRC con PLL.

Configura la frecuencia del oscilador FRC (FOSC), cuya frecuencia nominal (Fin) son 7.37 MHz, para que el microprocesador opere a 40 MIPS (FCY). FOSC = Fin $*$ M/(N1$*$N2) FCY = FOSC/2 FOSC = 79.2275 MHz FCY = 39.61375 MHz

3.4. Referencia del Archivo config.h

Funciones de configuración del microcontrolador.

'defines'

- #define FCY 39613750

  Frecuencia de operacion del microprocesador (Hz)

Funciones

- void inicializarReloj (void)

3.4.1. Descripción detallada

Funciones de configuración del microcontrolador.

Autor

Jaime Boal Martín-Larrauri

Versión

2.0.0

Fecha

26/08/2016

3.4.2. Documentación de las funciones

3.4.2.1. void inicializarReloj (void)

Inicializa el reloj interno FRC con PLL.

Configura la frecuencia del oscilador FRC (FOSC), cuya frecuencia nominal (Fin) son 7.37 MHz, para que el microprocesador opere a 40 MIPS (FCY). FOSC = Fin $*$ M/(N1$*$N2) FCY = FOSC/2 FOSC = 79.2275 MHz FCY = 39.61375 MHz

3.5. Referencia del Archivo idle.c

Funciones de manejo de la tarea Idle para un sistema basado en bucle de scan. Dicha tarea usa el timer 1 para llevar a cabo la temporización del bucle.

```
#include "xc.h"
#include "config.h"
#include "idle.h"
```

Funciones

- void inicializarTareaIdle (unsigned int t_s)
- void tareaIdle (void)

3.5.1. Descripción detallada

Funciones de manejo de la tarea Idle para un sistema basado en bucle de scan. Dicha tarea usa el timer 1 para llevar a cabo la temporización del bucle.

Autor

José Daniel Muñoz Frías

Versión

1.0.1

Fecha

08/09/2015

3.5.2. Documentación de las funciones

3.5.2.1. void inicializarTareaIdle (unsigned int *t_s*)

Inicializa la tarea Idle para usar el periodo de muestreo t_s

En un sistema basado en bucle de scan es necesario usar un timer para que las iteraciones del bucle duren siempre lo mismo. Esta función inicializa el timer 1 con un periodo igual a t_s. Cuando se llame a la función TareaIdle, ésta esperará al final de dicho periodo. Ojo, el valor máximo del periodo de muestreo es de 423,5 ms; ya que es el máximo valor que puede contar el timer 1 sin rebosar.

Parámetros

in	*t_s*	periodo de muestreo en décimas de ms. El valor máximo es de 4235, ya que es el valor máximo que puede contar el timer 1 usando un prescaler de 256 sin rebosar.

3.5.2.2. void tareaIdle (void)

Tarea Idle del bucle de scan

Esta tarea se queda bloqueada hasta el final del periodo de muestreo, marcado por el final de cuenta del timer 1. El periodo de muestreo se define con la llamada a InicializarTareaIdle().

3.6. Referencia del Archivo idle.h

Funciones de manejo de la tarea Idle para un sistema basado en bucle de scan. Dicha tarea usa el timer 1 para llevar a cabo la temporización del bucle.

Funciones

- void inicializarTareaIdle (unsigned int t_s)
- void tareaIdle (void)

3.6.1. Descripción detallada

Funciones de manejo de la tarea Idle para un sistema basado en bucle de scan. Dicha tarea usa el timer 1 para llevar a cabo la temporización del bucle.

Autor

José Daniel Muñoz Frías

Versión

1.0.1

Fecha

08/09/2015

3.6.2. Documentación de las funciones

3.6.2.1. void inicializarTareaIdle (unsigned int *t_s*)

Inicializa la tarea Idle para usar el periodo de muestreo t_s

En un sistema basado en bucle de scan es necesario usar un timer para que las iteraciones del bucle duren siempre lo mismo. Esta función inicializa el timer 1 con un periodo igual a t_s. Cuando se llame a la función TareaIdle, ésta esperará al final de dicho periodo. Ojo, el valor máximo del periodo de muestreo es de 423,5 ms; ya que es el máximo valor que puede contar el timer 1 sin rebosar.

Parámetros

in	*t↩ _↩ s*	periodo de muestreo en décimas de ms. El valor máximo es de 4235, ya que es el valor máximo que puede contar el timer 1 usando un prescaler de 256 sin rebosar.

3.6.2.2. void tarealdle (void)

Tarea Idle del bucle de scan

Esta tarea se queda bloqueada hasta el final del periodo de muestreo, marcado por el final de cuenta del timer 1. El periodo de muestreo se define con la llamada a InicializarTareaIdle().

3.7. Referencia del Archivo pwm.c

Módulo encargado de generar señales de PWM usando el módulo PWM del microcontrolador.

```
#include <xc.h>
#include "pwm.h"
#include "config.h"
```

Funciones

- void inicializarPWM (unsigned int bit_map, unsigned int frecuencia)
- void setFrecuencia (unsigned int frecuencia)
- void setDcPWM (unsigned int bit_map, unsigned int dc)
- void activarPWM (unsigned int bit_map)
- void desactivarPWM (unsigned int bit_map)

3.7.1. Descripción detallada

Módulo encargado de generar señales de PWM usando el módulo PWM del microcontrolador.

Autor

Jaime Boal Martín-Larrauri, José Daniel Muñoz Frías

Versión

1.0.1

Fecha

07/09/2015

3.7.2. Documentación de las funciones

3.7.2.1. void activarPWM (unsigned int *bit_map*)

Activa las salidas PWM indicadas mediante el parámetro bit_map.

Parámetros

bit_map	Indica qué pines se activan. Por ejemplo si se desean activar los pines RB15 y RB10, bit_map será igual a (1<<15)|(1<<10).

3.7.2.2. void desactivarPWM (unsigned int *bit_map*)

Desactiva las salidas PWM indicadas mediante el parámetro bit_map. Nótese que la desactivación hace que la salida pase a estar controlada por el módulo de E/S digital, por lo que tomará el valor indicado por el bit correspondiente de PORTB.

Parámetros

bit_map	Indica qué pines se desactivan. Por ejemplo si se desean desactivar los pines RB15 y RB10, bit_map será igual a (1<<15)|(1<<10).

3.7.2.3. void inicializarPWM (unsigned int *bit_map,* unsigned int *frecuencia*)

Incializa el módulo PWM. La función configura los pines indicados mediante el parámetro bit_map como salidas del módulo PWM. Dichas salidas se configuran de modo independiente. Además el módulo configura el módulo PWM en el modo free running a la frecuencia indicada mediante el parámetro frecuencia.

Parámetros

bit_map	Indica qué pines se conectan a las salidas del módulo PWM (1) o se dejan como E/S digital (0). Por ejemplo si se desea usar el pin RB15 y el RB10 como salidas PWM, bit_map será igual a (1<<15)|(1<<10).
frecuencia	Frecuencia en Hz de la señal PWM.

3.7.2.4. void setDcPWM (unsigned int *bit_map,* unsigned int *dc*)

Define el factor de servicio de la señal PWM de una o varias salidas, las cuales se definen mediante el parámetro bit_map. Nótese que las salidas RB15- RB14, RB13-RB12 y RB11-RB10 comparten el mismo canal PWM, por lo que su factor de servicio no puede ser distinto.

Parámetros

bit_map	Indica qué pines se conectan a las salidas del módulo PWM (1) o se dejan como E/S digital (0). Por ejemplo si se desea usar el pin RB15 y el RB10 como salidas PWM, bit_map será igual a (1<<15)|(1<<10).
dc	Factor de servicio en tanto por 10000. Por ejemplo si se desea un factor de servicio del 50 %, el parámetro dc ha de valer 5000.

3.7.2.5. void setFrecuencia (unsigned int *frecuencia*)

Define la frecuencia del módulo PWM. Nótese que la frecuencia es común para las seis salidas del módulo PWM

Parámetros

frecuencia	Frecuencia en Hz de la señal PWM.

3.8. Referencia del Archivo pwm.h

Módulo encargado de generar señales de PWM usando el módulo PWM del microcontrolador.

Funciones

- void inicializarPWM (unsigned int bit_map, unsigned int frecuencia)
- void setFrecuencia (unsigned int frecuencia)
- void setDcPWM (unsigned int bit_map, unsigned int dc)
- void activarPWM (unsigned int bit_map)
- void desactivarPWM (unsigned int bit_map)

3.8.1. Descripción detallada

Módulo encargado de generar señales de PWM usando el módulo PWM del microcontrolador.

Autor

Jaime Boal Martín-Larrauri, José Daniel Muñoz Frías

Versión

1.0.1

Fecha

07/09/2015

3.8.2. Documentación de las funciones

3.8.2.1. void activarPWM (unsigned int *bit_map*)

Activa las salidas PWM indicadas mediante el parámetro bit_map.

Parámetros

bit_map	Indica qué pines se activan. Por ejemplo si se desean activar los pines RB15 y RB10, bit_map será igual a (1<<15)|(1<<10).

3.8.2.2. void desactivarPWM (unsigned int *bit_map*)

Desactiva las salidas PWM indicadas mediante el parámetro bit_map. Nótese que la desactivación hace que la salida pase a estar controlada por el módulo de E/S digital, por lo que tomará el valor indicado por el bit correspondiente de PORTB.

Parámetros

bit_map	Indica qué pines se desactivan. Por ejemplo si se desean desactivar los pines RB15 y RB10, bit_map será igual a (1<<15)|(1<<10).

3.8.2.3. void inicializarPWM (unsigned int *bit_map,* unsigned int *frecuencia*)

Incializa el módulo PWM. La función configura los pines indicados mediante el parámetro bit_map como salidas del módulo PWM. Dichas salidas se configuran de modo independiente. Además el módulo configura el módulo PWM en el modo free running a la frecuencia indicada mediante el parámetro frecuencia.

Parámetros

bit_map	Indica qué pines se conectan a las salidas del módulo PWM (1) o se dejan como E/S digital (0). Por ejemplo si se desea usar el pin RB15 y el RB10 como salidas PWM, bit_map será igual a (1<<15)|(1<<10).
frecuencia	Frecuencia en Hz de la señal PWM.

3.8.2.4. void setDcPWM (unsigned int *bit_map,* unsigned int *dc*)

Define el factor de servicio de la señal PWM de una o varias salidas, las cuales se definen mediante el parámetro bit_map. Nótese que las salidas RB15- RB14, RB13-RB12 y RB11-RB10 comparten el mismo canal PWM, por lo que su factor de servicio no puede ser distinto.

Parámetros

bit_map	Indica qué pines se conectan a las salidas del módulo PWM (1) o se dejan como E/S digital (0). Por ejemplo si se desea usar el pin RB15 y el RB10 como salidas PWM, bit_map será igual a (1<<15)|(1<<10).
dc	Factor de servicio en tanto por 10000. Por ejemplo si se desea un factor de servicio del 50 %, el parámetro dc ha de valer 5000.

3.8.2.5. void setFrecuencia (unsigned int *frecuencia*)

Define la frecuencia del módulo PWM. Nótese que la frecuencia es común para las seis salidas del módulo PWM

Parámetros

frecuencia	Frecuencia en Hz de la señal PWM.

3.9. Referencia del Archivo uart.c

Módulo encargado de gestionar las comunicaciones USB.

```
#include <xc.h>
#include "uart.h"
#include "config.h"
```

'defines'

- #define TAM_TR_UART 250

  Tamaño de los vectores y colas.
- #define **TAM_REC_UART** 250
- #define PR_INT_TX 5

  Prioridad de las interrupciones (máx. 7 - mín. 1)
- #define **PR_INT_RX** 6

Funciones

- void __attribute__ ((interrupt, no_auto_psv))
- void procesarUART (void)
- void putsUART (char *pcad)
- char getcharUART (void)
- void ponerEnColaTransmisionUART (unsigned char uc_caracter)
- void transmitirUART (void)

3.9.1. Descripción detallada

Módulo encargado de gestionar las comunicaciones USB.

Autor

Jaime Boal Martín-Larrauri, José Daniel Muñoz Frías

Versión

1.1.0

Fecha

08/09/2015

3.9.2. Documentación de las funciones

3.9.2.1. void __attribute__ ((interrupt, no_auto_psv))

Rutina de atención a la interrupción de la UART asociada a la transmisión.

Rutina de atención a la interrupción de la UART asociada a la recepción.

La rutina introduce el caracter recibido en la cola sólo si ésta no está llena. Si lo está enciende el LED RB12 de la tarjeta para avisar al usuario. en este caso el caracter recibido por la interrupción se pierde.

3.9.2.2. char getcharUART (void)

Obtiene un caracter de la UART si hay alguno disponible. Si no obtiene un \0

Devuelve

char leído de la cola de recepción de la UART o \0 si la cola está vacía.

3.9.2.3. void ponerEnColaTransmisionUART (unsigned char *uc_caracter*)

Coloca un dato en la cola de transmisión.

La función introduce el carácter en la cola sólo si ésta no está llena. Si lo está enciende el LED RB12 de la tarjeta para avisar al usuario. En este caso el caracter enviado a la función se pierde.

Parámetros

in	*uc_caracter*	Caracter que se quiere poner en cola.

3.9.2.4. void procesarUART (void)

Procesa los mensajes recibidos a través del puerto USB.

3.9.2.5. void putsUART (char * *pcad*)

Transmite una cadena de caracteres por la UART.

Parámetros

pcad	cadena de caracteres a transmitir

3.9.2.6. void transmitirUART (void)

Envía todos los datos almacenados en la cola de transmisión.

3.10. Referencia del Archivo uart.h

Módulo encargado de gestionar las comunicaciones USB.

Funciones

- void inicializarUART (unsigned long baudrate)
- void procesarUART (void)
- void putsUART (char *pcad)
- char getcharUART (void)

3.10.1. Descripción detallada

Módulo encargado de gestionar las comunicaciones USB.

Autor

Jaime Boal Martín-Larrauri, José Daniel Muñoz Frías

Versión

1.1.0

Fecha

08/09/2015

3.10.2. Documentación de las funciones

3.10.2.1. char getcharUART (void)

Obtiene un caracter de la UART si hay alguno disponible. Si no obtiene un \0

Devuelve

char leído de la cola de recepción de la UART o \0 si la cola está vacía.

3.10.2.2. void inicializarUART (unsigned long *baudrate*)

Inicializa la UART.

Se inicializa la UART para usar una trama de 8 bits de datos sin paridad y con un bit de stop. El módulo usa interrupciones tanto para la recepción como para la transmisión. La comunicación con las rutinas de interrupción se realiza mediante dos colas.

Parámetros

baudrate	Baudrate de la uart en baudios

3.10.2.3. void procesarUART (void)

Procesa los mensajes recibidos a través del puerto USB.

3.10.2.4. void putsUART (char * *pcad*)

Transmite una cadena de caracteres por la UART.

Parámetros

pcad	cadena de caracteres a transmitir

Índice alfabético