第1个文件Adc.C

/\*\*

ADC Generated Driver File

@Company

Microchip Technology Inc.

@File Name

adc.c

@Summary

This is the generated driver implementation file for the ADC driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This source file provides implementations for driver APIs for ADC.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include "adc.h"

#include "mcc.h"

/\*\*

Section: Macro Declarations

\*/

#define ACQ\_US\_DELAY 5//5

/\*\*

Section: ADC Module APIs

\*/

void ADC\_Initialize(void)

{

// set the ADC to the options selected in the User Interface

// GO\_nDONE stop; ADON enabled; CHS AN0;

ADCON0 = 0x01;

// ADFM right; ADNREF VSS; ADPREF VDD; ADCS FOSC/8;

ADCON1 = 0x80;

// ADRESL 0;

ADRESL = 0x00;

// ADRESH 0;

ADRESH = 0x00;

}

void ADC\_SelectChannel(adc\_channel\_t channel)

{

// select the A/D channel

ADCON0bits.CHS = channel;

// Turn on the ADC module

ADCON0bits.ADON = 1;

}

void ADC\_StartConversion()

{

// Start the conversion

ADCON0bits.GO\_nDONE = 1;

}

bool ADC\_IsConversionDone()

{

// Start the conversion

return ((bool)(!ADCON0bits.GO\_nDONE));

}

adc\_result\_t ADC\_GetConversionResult(void)

{

// Conversion finished, return the result

return ((adc\_result\_t)((ADRESH << 8) + ADRESL));

}

adc\_result\_t ADC\_GetConversion(adc\_channel\_t channel)

{

// select the A/D channel

ADCON0bits.CHS = channel;

// Turn on the ADC module

ADCON0bits.ADON = 1;

// Acquisition time delay

//\_\_delay\_us(ACQ\_US\_DELAY);

// Start the conversion

ADCON0bits.GO\_nDONE = 1;

// Wait for the conversion to finish

while (ADCON0bits.GO\_nDONE)

{

}

// Conversion finished, return the result

return ((adc\_result\_t)((ADRESH << 8) + ADRESL));

}

/\*\*

End of File

\*/

第1个文件结束

第2个文件Adc.H

/\*\*

ADC Generated Driver API Header File

@Company

Microchip Technology Inc.

@File Name

adc.h

@Summary

This is the generated header file for the ADC driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This header file provides APIs for driver for ADC.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

#ifndef \_ADC\_H

#define \_ADC\_H

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include <stdint.h>

#include <stdbool.h>

#ifdef \_\_cplusplus // Provide C++ Compatibility

extern "C" {

#endif

/\*\*

Section: Data Types Definitions

\*/

/\*\*

\* result size of an A/D conversion

\*/

typedef uint16\_t adc\_result\_t;

/\*\*

\* result type of a Double ADC conversion

\*/

typedef struct

{

adc\_result\_t adcResult1;

adc\_result\_t adcResult2;

} adc\_sync\_double\_result\_t;

/\*\* ADC Channel Definition

@Summary

Defines the channels available for conversion.

@Description

This routine defines the channels that are available for the module to use.

Remarks:

None

\*/

typedef enum

{

channel\_AN3 = 0x3,

channel\_AN4 = 0x4,

channel\_AN5 = 0x5,

channel\_Temp = 0x1D,

channel\_DAC = 0x1E,

channel\_FVR = 0x1F

} adc\_channel\_t;

/\*\*

Section: ADC Module APIs

\*/

/\*\*

@Summary

Initializes the ADC

@Description

This routine initializes the Initializes the ADC.

This routine must be called before any other ADC routine is called.

This routine should only be called once during system initialization.

@Preconditions

None

@Param

None

@Returns

None

@Comment

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

convertedValue = ADC\_GetConversionResult();

</code>

\*/

void ADC\_Initialize(void);

/\*\*

@Summary

Allows selection of a channel for conversion

@Description

This routine is used to select desired channel for conversion.

available

@Preconditions

ADC\_Initialize() function should have been called before calling this function.

@Returns

None

@Param

Pass in required channel number

"For available channel refer to enum under adc.h file"

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

ADC\_SelectChannel(AN1\_Channel);

ADC\_StartConversion();

convertedValue = ADC\_GetConversionResult();

</code>

\*/

void ADC\_SelectChannel(adc\_channel\_t channel);

/\*\*

@Summary

Starts conversion

@Description

This routine is used to start conversion of desired channel.

@Preconditions

ADC\_Initialize() function should have been called before calling this function.

@Returns

None

@Param

None

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

ADC\_StartConversion();

convertedValue = ADC\_GetConversionResult();

</code>

\*/

void ADC\_StartConversion();

/\*\*

@Summary

Returns true when the conversion is completed otherwise false.

@Description

This routine is used to determine if conversion is completed.

When conversion is complete routine returns true. It returns false otherwise.

@Preconditions

ADC\_Initialize() and ADC\_StartConversion(adc\_channel\_t channel)

function should have been called before calling this function.

@Returns

true - If conversion is complete

false - If conversion is not completed

@Param

None

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

ADC\_StartConversion(AN1\_Channel);

while(!ADC\_IsConversionDone());

convertedValue = ADC\_GetConversionResult();

</code>

\*/

bool ADC\_IsConversionDone();

/\*\*

@Summary

Returns the ADC conversion value.

@Description

This routine is used to get the analog to digital converted value. This

routine gets converted values from the channel specified.

@Preconditions

This routine returns the conversion value only after the conversion is complete.

Completion status can be checked using

ADC\_IsConversionDone() routine.

@Returns

Returns the converted value.

@Param

None

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

ADC\_StartConversion(AN1\_Channel);

while(ADC\_IsConversionDone());

convertedValue = ADC\_GetConversionResult();

</code>

\*/

adc\_result\_t ADC\_GetConversionResult(void);

/\*\*

@Summary

Returns the ADC conversion value

also allows selection of a channel for conversion.

@Description

This routine is used to select desired channel for conversion

and to get the analog to digital converted value.

@Preconditions

ADC\_Initialize() function should have been called before calling this function.

@Returns

Returns the converted value.

@Param

Pass in required channel number.

"For available channel refer to enum under adc.h file"

@Example

<code>

uint16\_t convertedValue;

ADC\_Initialize();

conversion = ADC\_GetConversion(AN1\_Channel);

</code>

\*/

adc\_result\_t ADC\_GetConversion(adc\_channel\_t channel);

#ifdef \_\_cplusplus // Provide C++ Compatibility

}

#endif

#endif //\_ADC\_H

/\*\*

End of File

\*/

第2个文件结束

第3个文件Epwm1.C

/\*\*

ECCP1 Generated Driver File

@Company

Microchip Technology Inc.

@File Name

eccp1.c

@Summary

This is the generated driver implementation file for the ECCP1 driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This source file provides APIs for ECCP1.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include "epwm1.h"

/\*\*

Section: Macro Declarations

\*/

#define PWM1\_INITIALIZE\_DUTY\_VALUE 0

/\*\*

Section: EPWM Module APIs

\*/

void EPWM1\_Initialize (void)

{

// Set the PWM to the options selected in PIC10 / PIC12 / PIC16 / PIC18 MCUs

// CCP1M P1A,P1C: active high; P1B,P1D: active high; DC1B 0; P1M halfbridge;

CCP1CON = 0b10001100;

// CCP1ASE operating; PSS1BD low; PSS1AC low; CCP1AS disabled;

//ECCP1AS = 0x00;

// P1RSEN automatic\_restart; P1DC 0;

//PWM1CON = 0x80;

// PWM1CON = 0b10000100;

// STR1D P1D\_to\_port; STR1C P1C\_to\_port; STR1B P1B\_to\_CCP1M; STR1A P1A\_to\_CCP1M; STR1SYNC start\_at\_begin;

//PSTR1CON = 0x03;

T2CON = 0x00;

// PR2 255;

PR2 =255;

// Clearing IF flag.

PIR1bits.TMR2IF = 0;

// CCPR1H 0;

CCPR1H = 0x00;

// Selecting Timer2

CCPTMRSbits.C2TSEL = 0x0;

// T2CON = 0b00001100;

T2CONbits.T2CKPS = 0b00;

T2CONbits.TMR2ON = 1;

//

}

第3个文件结束

第4个文件Epwm1.H

/\*\*

ECCP1 Generated Driver File

@Company

Microchip Technology Inc.

@File Name

eccp1.c

@Summary

This is the generated driver implementation file for the ECCP1 driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This source file provides APIs for ECCP1.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

#ifndef \_EPWM1\_H

#define \_EPWM1\_H

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include <stdint.h>

#ifdef \_\_cplusplus // Provide C++ Compatibility

extern "C" {

#endif

/\*\*

Section: EPWM Module APIs

\*/

/\*\*

@Summary

Initializes the EPWM1

@Description

This routine initializes the EPWM1\_Initialize.

This routine must be called before any other ECCP1 routine is called.

This routine should only be called once during system initialization.

@Preconditions

None

@Param

None

@Returns

None

@Comment

@Example

<code>

uint16\_t dutycycle;

EPWM1\_Initialize();

EPWM1\_LoadDutyValue(dutycycle);

</code>

\*/

void EPWM1\_Initialize(void);

/\*\*

@Summary

Loads 16-bit duty cycle.

@Description

This routine loads the 16 bit duty cycle value.

@Preconditions

EPWM1\_Initialize() function should have been called before calling this function.

@Param

Pass in 16bit duty cycle value.

@Returns

None

@Example

<code>

uint16\_t dutycycle;

EPWM1\_Initialize();

EPWM1\_LoadDutyValue(dutycycle);

</code>

\*/

void EPWM1\_LoadDutyValue(uint16\_t dutyValue);

#ifdef \_\_cplusplus // Provide C++ Compatibility

}

#endif

#endif // \_EPWM1\_H

/\*\*

End of File

\*/

第4个文件结束

第5个文件Epwm2.C

/\*\*

ECCP2 Generated Driver File

@Company

Microchip Technology Inc.

@File Name

eccp2.c

@Summary

This is the generated driver implementation file for the ECCP2 driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This source file provides APIs for ECCP2.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include "epwm2.h"

/\*\*

Section: Macro Declarations

\*/

#define PWM2\_INITIALIZE\_DUTY\_VALUE 0

/\*\*

Section: EPWM Module APIs

\*/

void EPWM2\_Initialize (void)

{

// Set the PWM to the options selected in PIC10 / PIC12 / PIC16 / PIC18 MCUs

// CCP2M P2A: active high; P2B: active high; DC2B 0; P2M halfbridge;

CCP2CON = 0b10001100;

// CCP2ASE operating; PSS2BD low; PSS2AC low; CCP2AS disabled;

//CCP2AS = 0x00;

// P2RSEN automatic\_restart; P2DC 0;

//PWM2CON = 0x80;

// PWM2CON = 0b10000100;

// STR2B P2B\_to\_CCP2M; STR2A P2A\_to\_CCP2M; STR2SYNC start\_at\_begin;

//PSTR2CON = 0x03;

T2CON = 0x00;

// PR2 255;

PR2 =255;

// Clearing IF flag.

PIR1bits.TMR2IF = 0;

// CCPR2H 0;

CCPR2H = 0x00;

// Selecting Timer2

CCPTMRSbits.C2TSEL = 0x0;

// T2CON =0b00001100;

T2CONbits.T2CKPS = 0b00;

T2CONbits.TMR2ON = 1;

}

第5个文件结束

第6个文件Epwm2.H

/\*\*

ECCP2 Generated Driver File

@Company

Microchip Technology Inc.

@File Name

eccp2.c

@Summary

This is the generated driver implementation file for the ECCP2 driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This source file provides APIs for ECCP2.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

#ifndef \_EPWM2\_H

#define \_EPWM2\_H

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include <stdint.h>

#ifdef \_\_cplusplus // Provide C++ Compatibility

extern "C" {

#endif

/\*\*

Section: EPWM Module APIs

\*/

/\*\*

@Summary

Initializes the EPWM2

@Description

This routine initializes the EPWM2\_Initialize.

This routine must be called before any other ECCP2 routine is called.

This routine should only be called once during system initialization.

@Preconditions

None

@Param

None

@Returns

None

@Comment

@Example

<code>

uint16\_t dutycycle;

EPWM2\_Initialize();

EPWM2\_LoadDutyValue(dutycycle);

</code>

\*/

void EPWM2\_Initialize(void);

/\*\*

@Summary

Loads 16-bit duty cycle.

@Description

This routine loads the 16 bit duty cycle value.

@Preconditions

EPWM2\_Initialize() function should have been called before calling this function.

@Param

Pass in 16bit duty cycle value.

@Returns

None

@Example

<code>

uint16\_t dutycycle;

EPWM2\_Initialize();

EPWM2\_LoadDutyValue(dutycycle);

</code>

\*/

void EPWM2\_LoadDutyValue(uint16\_t dutyValue);

#ifdef \_\_cplusplus // Provide C++ Compatibility

}

#endif

#endif // \_EPWM2\_H

/\*\*

End of File

\*/

第6个文件结束

第7个文件Eusart.C

/\*\*

EUSART Generated Driver File

@Company

Microchip Technology Inc.

@File Name

eusart.c

@Summary

This is the generated driver implementation file for the EUSART driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This header file provides implementations for driver APIs for EUSART.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

/\*\*

Section: Included Files

\*/

#include "eusart.h"

/\*\*

Section: EUSART APIs

\*/

void EUSART\_Initialize(void)

{

// Set the EUSART module to the options selected in the user interface.

// ABDOVF no\_overflow; SCKP Non-Inverted; BRG16 16bit\_generator; WUE disabled; ABDEN disabled;

BAUDCON = 0x08;

// SPEN enabled; RX9 8-bit; CREN disabled; ADDEN disabled; SREN disabled;

RCSTA = 0x80;

// TX9 8-bit; TX9D 0; SENDB sync\_break\_complete; TXEN enabled; SYNC asynchronous; BRGH hi\_speed; CSRC slave;

TXSTA = 0x24;

// Baud Rate = 9600; SPBRGL 103;

SPBRGL = 0x67;

// Baud Rate = 9600; SPBRGH 0;

SPBRGH = 0x00;

}

uint8\_t EUSART\_Read(void)

{

RCSTAbits.SREN = 1;

while(!PIR1bits.RCIF)

{

}

if(1 == RCSTAbits.OERR)

{

// EUSART error - restart

RCSTAbits.SPEN = 0;

RCSTAbits.SPEN = 1;

}

return RCREG;

}

void EUSART\_Write(uint8\_t txData)

{

while(0 == PIR1bits.TXIF)

{

}

TXREG = txData; // Write the data byte to the USART.

}

/\*\*

End of File

\*/

第7个文件结束

第8个文件Eusart.H

/\*\*

EUSART Generated Driver API Header File

@Company

Microchip Technology Inc.

@File Name

eusart.h

@Summary

This is the generated header file for the EUSART driver using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description

This header file provides APIs for driver for EUSART.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 2.00

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

#ifndef \_EUSART\_H

#define \_EUSART\_H

/\*\*

Section: Included Files

\*/

#include <xc.h>

#include <stdbool.h>

#include <stdint.h>

#ifdef \_\_cplusplus // Provide C++ Compatibility

extern "C" {

#endif

/\*\*

Section: Macro Declarations

\*/

#define EUSART\_DataReady (PIR1bits.RCIF)

/\*\*

Section: EUSART APIs

\*/

/\*\*

@Summary

Initialization routine that takes inputs from the EUSART GUI.

@Description

This routine initializes the EUSART driver.

This routine must be called before any other EUSART routine is called.

@Preconditions

None

@Param

None

@Returns

None

@Comment

\*/

void EUSART\_Initialize(void);

/\*\*

@Summary

Read a byte of data from the EUSART.

@Description

This routine reads a byte of data from the EUSART.

@Preconditions

EUSART\_Initialize() function should have been called

before calling this function. The transfer status should be checked to see

if the receiver is not empty before calling this function.

@Param

None

@Returns

A data byte received by the driver.

\*/

uint8\_t EUSART\_Read(void);

/\*\*

@Summary

Writes a byte of data to the EUSART.

@Description

This routine writes a byte of data to the EUSART.

@Preconditions

EUSART\_Initialize() function should have been called

before calling this function. The transfer status should be checked to see

if transmitter is not busy before calling this function.

@Param

txData - Data byte to write to the EUSART

@Returns

None

\*/

void EUSART\_Write(uint8\_t txData);

#ifdef \_\_cplusplus // Provide C++ Compatibility

}

#endif

#endif // \_EUSART\_H

/\*\*

End of File

\*/

第8个文件结束

第9个文件Mcc.C

/\*\*

@Generated PIC10 / PIC12 / PIC16 / PIC18 MCUs Source File

@Company:

Microchip Technology Inc.

@File Name:

mcc.c

@Summary:

This is the mcc.c file generated using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description:

This header file provides implementations for driver APIs for all modules selected in the GUI.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Driver Version : 1.02

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

// Configuration bits: selected in the GUI

// CONFIG1

#pragma config FOSC = INTOSC // Oscillator Selection->INTOSC oscillator: I/O function on CLKIN pin

#pragma config WDTE = OFF // Watchdog Timer Enable->WDT disabled

#pragma config PWRTE = OFF // Power-up Timer Enable->PWRT disabled

#pragma config MCLRE = ON // MCLR Pin Function Select->MCLR/VPP pin function is MCLR

#pragma config CP = ON // Flash Program Memory Code Protection->Program memory code protection is disabled

#pragma config CPD = ON // Data Memory Code Protection->Data memory code protection is disabled

#pragma config BOREN = ON // Brown-out Reset Enable->Brown-out Reset enabled

#pragma config CLKOUTEN = OFF // Clock Out Enable->CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin

#pragma config IESO = ON // Internal/External Switchover->Internal/External Switchover mode is enabled

#pragma config FCMEN = ON // Fail-Safe Clock Monitor Enable->Fail-Safe Clock Monitor is enabled

// CONFIG2

#pragma config WRT = OFF // Flash Memory Self-Write Protection->Write protection off

#pragma config PLLEN = ON // PLL Enable->4x PLL enabled

#pragma config STVREN = ON // Stack Overflow/Underflow Reset Enable->Stack Overflow or Underflow will cause a Reset

#pragma config BORV = LO // Brown-out Reset Voltage Selection->Brown-out Reset Voltage (Vbor), low trip point selected.

#pragma config LVP = ON // Low-Voltage Programming Enable->Low-voltage programming enabled

#include "mcc.h"

void SYSTEM\_Initialize(void)

{

// PIN\_MANAGER\_Initialize();

// OSCILLATOR\_Initialize();

WDT\_Initialize();

EPWM2\_Initialize();

// ADC\_Initialize();

EPWM1\_Initialize();

timer1\_init();

PIE1bits.TMR2IE =1;

PIE1bits.TMR1IE =1;

INTCONbits.IOCIE = 1;

GIE=1; // Global interrupt enable, peripheral interrupt enable

PEIE=1;

}

void timer1\_init(void) //fosc/4=4M/4???????1us

{

T1CON =0b00100001;

//T1CON=0X01; //16?????

TMR1IF=0; //?timer1????

TMR1IE=1; //timer1????

TMR1H=0XFC; //?????TMR1????????

TMR1L=0x26;

}

void OSCILLATOR\_Initialize(void)

{

// SCS FOSC; SPLLEN disabled; IRCF 500KHz\_MF;

OSCCON = 0b01111000;

// TUN 0;

OSCTUNE = 0x00;

}

void WDT\_Initialize(void)

{

// WDTPS 1:65536; SWDTEN OFF;

WDTCON = 0x16;

}

/\*\*

End of File

\*/

第9个文件结束

第10个文件Mcc.H

/\*\*

@Generated PIC10 / PIC12 / PIC16 / PIC18 MCUs Header File

@Company:

Microchip Technology Inc.

@File Name:

mcc.h

@Summary:

This is the mcc.h file generated using PIC10 / PIC12 / PIC16 / PIC18 MCUs

@Description:

This header file provides implementations for driver APIs for all modules selected in the GUI.

Generation Information :

Product Revision : PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.45

Device : PIC16F1825

Version : 1.02

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

\*/

/\*

(c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this

software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER

EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED

WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A

PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION

WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,

INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND

WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS

BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE

FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN

ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,

THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE

TERMS.

\*/

#ifndef MCC\_H

#define MCC\_H

#include <xc.h>

#include "pin\_manager.h"

#include <stdint.h>

#include <stdbool.h>

#include "adc.h"

#include "epwm2.h"

#include "epwm1.h"

#include "eusart.h"

#define \_XTAL\_FREQ 16000000

/\*\*

\* @Param

none

\* @Returns

none

\* @Description

Initializes the device to the default states configured in the

\* MCC GUI

\* @Example

SYSTEM\_Initialize(void);

\*/

void SYSTEM\_Initialize(void);

/\*\*

\* @Param

none

\* @Returns

none

\* @Description

Initializes the oscillator to the default states configured in the

\* MCC GUI

\* @Example

OSCILLATOR\_Initialize(void);

\*/

void timer1\_init(void);

void OSCILLATOR\_Initialize(void);

/\*\*

\* @Param

none

\* @Returns

none

\* @Description

Initializes the WDT module to the default states configured in the

\* MCC GUI

\* @Example

WDT\_Initialize(void);

\*/

void WDT\_Initialize(void);

#endif /\* MCC\_H \*/

/\*\*

End of File

\*/

第10个文件结束

第11个文件Pin\_Manager.C

/\*\*

Generated Pin Manager File

Company:

Microchip Technology Inc.

File Name:

pin\_manager.c

Summary:

This is the Pin Manager file generated using MPLAB(c) Code Configurator

Description:

This header file provides implementations for pin APIs for all pins selected in the GUI.

Generation Information :

Product Revision : MPLAB(c) Code Configurator - 4.26

Device : PIC16F1825

Driver Version : 1.02

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

Copyright (c) 2013 - 2015 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute

Software only when embedded on a Microchip microcontroller or digital signal

controller that is integrated into your product or third party product

(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for

additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,

EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF

MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER

CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR

OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES

INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR

CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF

SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES

(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

\*/

#include <xc.h>

#include "pin\_manager.h"

#include "stdbool.h"

/\* pin 2(OCP): I/O only; RA5 ; input; read as 0 normally

\* pin 3(I-sense):AD pin AN3; RA4;input ;need AD;(calculate for I and power)

\* pin 9(Vcc) :RC1 ;AN5;input;need AD

\*/

void PIN\_MANAGER\_Initialize(void)

{

/\*\*

LATx registers

\*/

LATA = 0x00;

LATC = 0x00;

/\*\*

TRISx registers 1=input

\*/

TRISA = 0x3E; // 0011 1110 ;0x3f

//LATAbits.LATA0 = 1;

TRISC = 0x03;// 0000 0011

/\*\*

ANSELx registers 0 =D

\*/

ANSELC = 0x03; // 0000 0011

ANSELA = 0x12;// 0001 0010; 0X16

/\*\*

WPUx registers

\*/

WPUA = 0x3E; //0x3f

WPUC = 0x3F;

OPTION\_REGbits.nWPUEN = 0;

/\*\*

APFCONx registers

\*/

//APFCON1 = 0x00; //

APFCON0 = 0x00; // 0x04//13 TX 5V

}

void PIN\_MANAGER\_IOC(void)

{

}

/\*\*

End of File

\*/

第11个文件结束

第12个文件Pin\_Manager.H

/\*\*

@Generated Pin Manager Header File

@Company:

Microchip Technology Inc.

@File Name:

pin\_manager.h

@Summary:

This is the Pin Manager file generated using MPLAB(c) Code Configurator

@Description:

This header file provides implementations for pin APIs for all pins selected in the GUI.

Generation Information :

Product Revision : MPLAB(c) Code Configurator - 4.26

Device : PIC16F1825

Version : 1.01

The generated drivers are tested against the following:

Compiler : XC8 1.35

MPLAB : MPLAB X 3.40

Copyright (c) 2013 - 2015 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute

Software only when embedded on a Microchip microcontroller or digital signal

controller that is integrated into your product or third party product

(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for

additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,

EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF

MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER

CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR

OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES

INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR

CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF

SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES

(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

\*/

#ifndef PIN\_MANAGER\_H

#define PIN\_MANAGER\_H

#define INPUT 1

#define OUTPUT 0

#define HIGH 1

#define LOW 0

#define ANALOG 1

#define DIGITAL 0

#define PULL\_UP\_ENABLED 1

#define PULL\_UP\_DISABLED 0

// get/set RA0 procedures

#define RA0\_SetHigh() do { LATAbits.LATA0 = 1; } while(0)

#define RA0\_SetLow() do { LATAbits.LATA0 = 0; } while(0)

#define RA0\_Toggle() do { LATAbits.LATA0 = ~LATAbits.LATA0; } while(0)

#define RA0\_GetValue() PORTAbits.RA0

#define RA0\_SetDigitalInput() do { TRISAbits.TRISA0 = 1; } while(0)

#define RA0\_SetDigitalOutput() do { TRISAbits.TRISA0 = 0; } while(0)

#define RA0\_SetPullup() do { WPUAbits.WPUA0 = 1; } while(0)

#define RA0\_ResetPullup() do { WPUAbits.WPUA0 = 0; } while(0)

#define RA0\_SetAnalogMode() do { ANSELAbits.ANSA0 = 1; } while(0)

#define RA0\_SetDigitalMode()do { ANSELAbits.ANSA0 = 0; } while(0)

// get/set channel\_AN3 aliases

#define channel\_AN3\_TRIS TRISAbits.TRISA4

#define channel\_AN3\_LAT LATAbits.LATA4

#define channel\_AN3\_PORT PORTAbits.RA4

#define channel\_AN3\_WPU WPUAbits.WPUA4

#define channel\_AN3\_ANS ANSELAbits.ANSA4

#define channel\_AN3\_SetHigh() do { LATAbits.LATA4 = 1; } while(0)

#define channel\_AN3\_SetLow() do { LATAbits.LATA4 = 0; } while(0)

#define channel\_AN3\_Toggle() do { LATAbits.LATA4 = ~LATAbits.LATA4; } while(0)

#define channel\_AN3\_GetValue() PORTAbits.RA4

#define channel\_AN3\_SetDigitalInput() do { TRISAbits.TRISA4 = 1; } while(0)

#define channel\_AN3\_SetDigitalOutput() do { TRISAbits.TRISA4 = 0; } while(0)

#define channel\_AN3\_SetPullup() do { WPUAbits.WPUA4 = 1; } while(0)

#define channel\_AN3\_ResetPullup() do { WPUAbits.WPUA4 = 0; } while(0)

#define channel\_AN3\_SetAnalogMode() do { ANSELAbits.ANSA4 = 1; } while(0)

#define channel\_AN3\_SetDigitalMode() do { ANSELAbits.ANSA4 = 0; } while(0)

// get/set channel\_AN4 aliases

#define channel\_AN4\_TRIS TRISCbits.TRISC0

#define channel\_AN4\_LAT LATCbits.LATC0

#define channel\_AN4\_PORT PORTCbits.RC0

#define channel\_AN4\_WPU WPUCbits.WPUC0

#define channel\_AN4\_ANS ANSELCbits.ANSC0

#define channel\_AN4\_SetHigh() do { LATCbits.LATC0 = 1; } while(0)

#define channel\_AN4\_SetLow() do { LATCbits.LATC0 = 0; } while(0)

#define channel\_AN4\_Toggle() do { LATCbits.LATC0 = ~LATCbits.LATC0; } while(0)

#define channel\_AN4\_GetValue() PORTCbits.RC0

#define channel\_AN4\_SetDigitalInput() do { TRISCbits.TRISC0 = 1; } while(0)

#define channel\_AN4\_SetDigitalOutput() do { TRISCbits.TRISC0 = 0; } while(0)

#define channel\_AN4\_SetPullup() do { WPUCbits.WPUC0 = 1; } while(0)

#define channel\_AN4\_ResetPullup() do { WPUCbits.WPUC0 = 0; } while(0)

#define channel\_AN4\_SetAnalogMode() do { ANSELCbits.ANSC0 = 1; } while(0)

#define channel\_AN4\_SetDigitalMode() do { ANSELCbits.ANSC0 = 0; } while(0)

// get/set channel\_AN5 aliases

#define channel\_AN5\_TRIS TRISCbits.TRISC1

#define channel\_AN5\_LAT LATCbits.LATC1

#define channel\_AN5\_PORT PORTCbits.RC1

#define channel\_AN5\_WPU WPUCbits.WPUC1

#define channel\_AN5\_ANS ANSELCbits.ANSC1

#define channel\_AN5\_SetHigh() do { LATCbits.LATC1 = 1; } while(0)

#define channel\_AN5\_SetLow() do { LATCbits.LATC1 = 0; } while(0)

#define channel\_AN5\_Toggle() do { LATCbits.LATC1 = ~LATCbits.LATC1; } while(0)

#define channel\_AN5\_GetValue() PORTCbits.RC1

#define channel\_AN5\_SetDigitalInput() do { TRISCbits.TRISC1 = 1; } while(0)

#define channel\_AN5\_SetDigitalOutput() do { TRISCbits.TRISC1 = 0; } while(0)

#define channel\_AN5\_SetPullup() do { WPUCbits.WPUC1 = 1; } while(0)

#define channel\_AN5\_ResetPullup() do { WPUCbits.WPUC1 = 0; } while(0)

#define channel\_AN5\_SetAnalogMode() do { ANSELCbits.ANSC1 = 1; } while(0)

#define channel\_AN5\_SetDigitalMode() do { ANSELCbits.ANSC1 = 0; } while(0)

// get/set RC2 procedures

#define RC2\_SetHigh() do { LATCbits.LATC2 = 1; } while(0)

#define RC2\_SetLow() do { LATCbits.LATC2 = 0; } while(0)

#define RC2\_Toggle() do { LATCbits.LATC2 = ~LATCbits.LATC2; } while(0)

#define RC2\_GetValue() PORTCbits.RC2

#define RC2\_SetDigitalInput() do { TRISCbits.TRISC2 = 1; } while(0)

#define RC2\_SetDigitalOutput() do { TRISCbits.TRISC2 = 0; } while(0)

#define RC2\_SetPullup() do { WPUCbits.WPUC2 = 1; } while(0)

#define RC2\_ResetPullup() do { WPUCbits.WPUC2 = 0; } while(0)

#define RC2\_SetAnalogMode() do { ANSELCbits.ANSC2 = 1; } while(0)

#define RC2\_SetDigitalMode()do { ANSELCbits.ANSC2 = 0; } while(0)

// get/set RC3 procedures

#define RC3\_SetHigh() do { LATCbits.LATC3 = 1; } while(0)

#define RC3\_SetLow() do { LATCbits.LATC3 = 0; } while(0)

#define RC3\_Toggle() do { LATCbits.LATC3 = ~LATCbits.LATC3; } while(0)

#define RC3\_GetValue() PORTCbits.RC3

#define RC3\_SetDigitalInput() do { TRISCbits.TRISC3 = 1; } while(0)

#define RC3\_SetDigitalOutput() do { TRISCbits.TRISC3 = 0; } while(0)

#define RC3\_SetPullup() do { WPUCbits.WPUC3 = 1; } while(0)

#define RC3\_ResetPullup() do { WPUCbits.WPUC3 = 0; } while(0)

#define RC3\_SetAnalogMode() do { ANSELCbits.ANSC3 = 1; } while(0)

#define RC3\_SetDigitalMode()do { ANSELCbits.ANSC3 = 0; } while(0)

// get/set RC4 procedures

#define RC4\_SetHigh() do { LATCbits.LATC4 = 1; } while(0)

#define RC4\_SetLow() do { LATCbits.LATC4 = 0; } while(0)

#define RC4\_Toggle() do { LATCbits.LATC4 = ~LATCbits.LATC4; } while(0)

#define RC4\_GetValue() PORTCbits.RC4

#define RC4\_SetDigitalInput() do { TRISCbits.TRISC4 = 1; } while(0)

#define RC4\_SetDigitalOutput() do { TRISCbits.TRISC4 = 0; } while(0)

#define RC4\_SetPullup() do { WPUCbits.WPUC4 = 1; } while(0)

#define RC4\_ResetPullup() do { WPUCbits.WPUC4 = 0; } while(0)

// get/set RC5 procedures

#define RC5\_SetHigh() do { LATCbits.LATC5 = 1; } while(0)

#define RC5\_SetLow() do { LATCbits.LATC5 = 0; } while(0)

#define RC5\_Toggle() do { LATCbits.LATC5 = ~LATCbits.LATC5; } while(0)

#define RC5\_GetValue() PORTCbits.RC5

#define RC5\_SetDigitalInput() do { TRISCbits.TRISC5 = 1; } while(0)

#define RC5\_SetDigitalOutput() do { TRISCbits.TRISC5 = 0; } while(0)

#define RC5\_SetPullup() do { WPUCbits.WPUC5 = 1; } while(0)

#define RC5\_ResetPullup() do { WPUCbits.WPUC5 = 0; } while(0)

/\*\*

@Param

none

@Returns

none

@Description

GPIO and peripheral I/O initialization

@Example

PIN\_MANAGER\_Initialize();

\*/

void PIN\_MANAGER\_Initialize (void);

/\*\*

\* @Param

none

\* @Returns

none

\* @Description

Interrupt on Change Handling routine

\* @Example

PIN\_MANAGER\_IOC();

\*/

void PIN\_MANAGER\_IOC(void);

#endif // PIN\_MANAGER\_H

/\*\*

End of File

\*/

第12个文件结束