PygmyPossum_Firmware

1.1

Generated by Doxygen 1.8.20

Chapter 1

README

The Pygmy Possum

PIR sensor for camera traps

1.0.1 Table of Contents

1.0.2 About The Project

The Pygmy Possum is a battery powered PIR (Passive Infrared) sensor for triggering remote camera traps.

A camera trap is a remotely activated camera that is equipped with a motion sensor or an infrared sensor, or uses a light beam as a trigger. Camera trapping is a method for capturing wild animals on film when researchers are not present, and has been used in ecological research for decades. In addition to applications in hunting and wildlife viewing, research applications include studies of nest ecology, detection of rare species, estimation of population size and species richness, as well as research on habitat use and occupation of human-built structures. [1]

1.0.2.1 How it works..

The Pygmy Possum is based on an 8 bit PIC microcontroller and custom PCB
Firmware is written in C (standard C99). The compiler being used is XC8 v2.30 by Microchip.

The microcontroller will instantly enter a sleep state to minimise current consumtion. The output from the HS-SR501 module is connected to an input pin on the MCU, A rising edge on this input will trigger an interrupt on the MCU, and wake it from sleep. In the interrupt service routine, the Pygmy Possum will read the settings from the DIP switches and send pulses to the output optocoupler. This ensures that the camera circuit is completely isolated from the Pygmy Possum circuit. The optocoupler will short the Tip to the Sheath of the TRS audio Jack. This will activate the shutter release on the connected camera.

1.0.3 Built With

1.0.3.0.1 Hardware:

- `Microchip PIC16F18313` 8-bit MCU
- `HC-SR501 PIR module` low-cost passive infrared sensor board
- `Microchip PICKit4` programmer

1.0.3.0.2 Software:

- `MPLAB X IDE v5.40`

1.0.4 Usage

Set the DIP switches on the front of the board to select a pre-programmed package. Currently there are 8 hard-coded packages that can be selected. Each package write values into the `camParams` Struct for:

- `numOfSnaps` - Number of Shots to take after a Event
- `snapPeriod` - Time Between Shots (ms)
- `minEventPeriod` - Minimum time between Events (s)

The default DIP settings are as follows:

DIP3	DIP2	DIP1	Package Number	numOfSnaps	snapPeriod (ms)	minEvent↩ Period (s)
0	0	0	0	3	50	10
0	0	1	1	3	100	10
0	1	0	2	3	250	10
0	1	1	3	3	500	10
1	0	0	4	3	750	10
1	0	1	5	3	1000	10
1	1	0	6	3	1500	10
1	1	1	7	3	3000	10

1.0.5 License

Distributed under the MIT License. See `LICENSE` for more information.

1.0.6 Contact

- Tom Evison - tom@evisonengineering.com.au
- James Lidsey - james.lidsey93@gmail.com

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 eusart_status_t Union Reference

`#include <eusart.h>`

Data Fields

- struct {
  unsigned perr: 1
  unsigned ferr: 1
  unsigned oerr: 1
  unsigned reserved: 5
  };
- uint8_t status

4.1.1 Detailed Description

Section: Data Type Definitions

Definition at line 75 of file eusart.h.

4.1.2 Field Documentation

4.1.2.1 "@1

`struct { ... }`

4.1.2.2 ferr

```
unsigned ferr
```

Definition at line 78 of file eusart.h.

4.1.2.3 oerr

```
unsigned oerr
```

Definition at line 79 of file eusart.h.

4.1.2.4 perr

```
unsigned perr
```

Definition at line 77 of file eusart.h.

4.1.2.5 reserved

```
unsigned reserved
```

Definition at line 80 of file eusart.h.

4.1.2.6 status

```
uint8_t status
```

Definition at line 82 of file eusart.h.

The documentation for this union was generated from the following file:

- headers/eusart.h

Chapter 5

File Documentation

5.1 headers/eusart.h File Reference

```
#include <xc.h>
#include <stdbool.h>
#include <stdint.h>
```

Data Structures

- union eusart_status_t

Macros

- #define EUSART_DataReady (EUSART_is_rx_ready())

Functions

- void EUSART_Initialize (void)
- bool EUSART_is_tx_ready (void)
- bool EUSART_is_rx_ready (void)
- bool EUSART_is_tx_done (void)
- eusart_status_t EUSART_get_last_status (void)
- uint8_t EUSART_Read (void)
- void EUSART_Write (uint8_t txData)
- void EUSART_WriteString (uint8_t[ ])
- void EUSART_Transmit_ISR (void)
- void EUSART_Receive_ISR (void)
- void EUSART_RxDataHandler (void)
- void EUSART_SetFramingErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetOverrunErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetTxInterruptHandler (void(∗interruptHandler)(void))
- void EUSART_SetRxInterruptHandler (void(∗interruptHandler)(void))

Variables

- volatile uint8_t eusartTxBufferRemaining
- volatile uint8_t eusartRxCount
- void(* EUSART_TxDefaultInterruptHandler)(void)
- void(* EUSART_RxDefaultInterruptHandler)(void)

5.1.1 Macro Definition Documentation

5.1.1.1 EUSART_DataReady

```
#define EUSART_DataReady (EUSART_is_rx_ready())
```

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 - 1.81.6 Device : PIC16F18313 Driver Version : 2.1.0 The generated drivers are tested against the following: Compiler : XC8 2.30 and above MPLAB : MPLAB X 5.40 Section: Included Files Section: Macro Declarations

Definition at line 69 of file eusart.h.

5.1.2 Function Documentation

5.1.2.1 EUSART_get_last_status()

```
eusart_status_t EUSART_get_last_status (
            void  )
```

@Summary Gets the error status of the last read byte.

@Description This routine gets the error status of the last read byte.

@Preconditions EUSART_Initialize() function should have been called before calling this function. The returned value is only updated after a read is called.

@Param None

@Returns the status of the last read byte

@Example `void main(void) { volatile uint8_t rxData; volatile eusart_status_t rxStatus;`

`Initialize the device SYSTEM_Initialize();`

`Enable the Global Interrupts INTERRUPT_GlobalInterruptEnable();`

`while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData = EUSART_Read(); rxStatus = EUSART_get_last_status(); if(rxStatus.ferr){ LED_0_SetHigh(); } } } }`

Definition at line 98 of file eusart.c.

5.1.2.2 EUSART_Initialize()

```
void EUSART_Initialize (
            void  )
```

@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

Definition at line 43 of file eusart.c.

5.1.2.3 EUSART_is_rx_ready()

```
bool EUSART_is_rx_ready (
            void  )
```

@Summary Checks if the EUSART receiver ready for reading

@Description This routine checks if EUSART receiver has received data and ready to be read

@Preconditions EUSART_Initialize() function should be called before calling this function EUSART receiver should be enabled before calling this function

@Param None

@Returns Status of EUSART receiver TRUE: EUSART receiver is ready for reading FALSE: EUSART receiver is not ready for reading

@Example `void main(void) { volatile uint8_t rxData;`

`Initialize the device SYSTEM_Initialize();`

`while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData = UART1_Read(); if(EUSART_is_tx_ready()) { EUSART_Write(rxData); } } } }`

Definition at line 90 of file eusart.c.

5.1.2.4 EUSART_is_tx_done()

```
bool EUSART_is_tx_done (
            void  )
```

@Summary Checks if EUSART data is transmitted

@Description This function return the status of transmit shift register

@Preconditions EUSART_Initialize() function should be called before calling this function EUSART transmitter should be enabled and EUSART_Write should be called before calling this function

@Param None

@Returns Status of EUSART receiver TRUE: Data completely shifted out if the USART shift register FALSE: Data is not completely shifted out of the shift register

@Example
```
void main(void) { volatile uint8_t rxData;

Initialize the device SYSTEM_Initialize();

while(1) { if(EUSART_is_tx_ready()) { LED_0_SetHigh(); EUSARTWrite(rxData);
} if(EUSART_is_tx_done() { LED_0_SetLow(); } } }
```

Definition at line 94 of file eusart.c.

5.1.2.5 EUSART_is_tx_ready()

```
bool EUSART_is_tx_ready (
            void  )
```

@Summary Checks if the EUSART transmitter is ready to transmit data

@Description This routine checks if EUSART transmitter is ready to accept and transmit data byte

@Preconditions EUSART_Initialize() function should have been called before calling this function. EUSART transmitter should be enabled before calling this function

@Param None

@Returns Status of EUSART transmitter TRUE: EUSART transmitter is ready FALSE: EUSART transmitter is not ready

@Example
```
void main(void) { volatile uint8_t rxData;

Initialize the device SYSTEM_Initialize();

while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData =
UART1_Read(); if(EUSART_is_tx_ready()) { EUSARTWrite(rxData); } } } }
```

Definition at line 86 of file eusart.c.

5.1.2.6 EUSART_Read()

```
uint8_t EUSART_Read (
            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.

Definition at line 102 of file eusart.c.

5.1.2.7 EUSART_Receive_ISR()

```
void EUSART_Receive_ISR (
            void  )
```

@Summary Maintains the driver's receiver state machine and implements its ISR

@Description This routine is used to maintain the driver's internal receiver state machine.This interrupt service routine is called when the state of the receiver needs to be maintained in a non polled manner.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 160 of file eusart.c.

5.1.2.8 EUSART_RxDataHandler()

```
void EUSART_RxDataHandler (
            void  )
```

@Summary Maintains the driver's receiver state machine

@Description This routine is called by the receive state routine and is used to maintain the driver's internal receiver state machine. It should be called by a custom ISR to maintain normal behavior

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 183 of file eusart.c.

5.1.2.9 EUSART_SetErrorHandler()

```
void EUSART_SetErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Error Handler

@Description This API sets the function to be called upon EUSART error

@Preconditions Initialize the EUSART module before calling this API

@Param Address of function to be set as error handler

@Returns None

Definition at line 215 of file eusart.c.

5.1.2.10 EUSART_SetFramingErrorHandler()

```
void EUSART_SetFramingErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Framing Error Handler

@Description This API sets the function to be called upon EUSART framing error

@Preconditions Initialize the EUSART before calling this API

@Param Address of function to be set as framing error handler

@Returns None

Definition at line 207 of file eusart.c.

5.1.2.11 EUSART_SetOverrunErrorHandler()

```
void EUSART_SetOverrunErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Overrun Error Handler

@Description This API sets the function to be called upon EUSART overrun error

@Preconditions Initialize the EUSART module before calling this API

@Param Address of function to be set as overrun error handler

@Returns None

Definition at line 211 of file eusart.c.

5.1.2.12 EUSART_SetRxInterruptHandler()

```
void EUSART_SetRxInterruptHandler (
             void(*)(void) interruptHandler )
```

@Summary Sets the receive handler function to be called by the interrupt service

@Description Calling this function will set a new custom function that will be called when the receive interrupt needs servicing.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param A pointer to the new function

@Returns None

Definition at line 223 of file eusart.c.

5.1.2.13 EUSART_SetTxInterruptHandler()

```
void EUSART_SetTxInterruptHandler (
             void(*)(void) interruptHandler )
```

@Summary Sets the transmit handler function to be called by the interrupt service

@Description Calling this function will set a new custom function that will be called when the transmit interrupt needs servicing.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param A pointer to the new function

@Returns None

Definition at line 219 of file eusart.c.

5.1.2.14 EUSART_Transmit_ISR()

```
void EUSART_Transmit_ISR (
             void  )
```

Definition at line 146 of file eusart.c.

5.1.2.15 EUSART_Write()

```
void EUSART_Write (
            uint8_t txData )
```

@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

Definition at line 121 of file eusart.c.

5.1.2.16 EUSART_WriteString()

```
void EUSART_WriteString (
            uint8_t [] )
```

@Summary Maintains the driver's transmitter state machine and implements its ISR.

@Description This routine is used to maintain the driver's internal transmitter state machine.This interrupt service routine is called when the state of the transmitter needs to be maintained in a non polled manner.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 138 of file eusart.c.

5.1.3 Variable Documentation

5.1.3.1 EUSART_RxDefaultInterruptHandler

```
void(* EUSART_RxDefaultInterruptHandler) (void)  [extern]
```

Definition at line 33 of file eusart.c.

5.1.3.2 EUSART_TxDefaultInterruptHandler

```
void(* EUSART_TxDefaultInterruptHandler) (void)  [extern]
```

Section: EUSART APIs

Definition at line 32 of file eusart.c.

5.1.3.3 eusartRxCount

```
volatile uint8_t eusartRxCount  [extern]
```

Definition at line 26 of file eusart.c.

5.1.3.4 eusartTxBufferRemaining

```
volatile uint8_t eusartTxBufferRemaining  [extern]
```

Section: Global variables

Definition at line 20 of file eusart.c.

5.2 headers/hardware.h File Reference

Macros

- #define DIP3_PIN PORTAbits.RA0
- #define DIP1_PIN PORTAbits.RA1
- #define PIR_PIN PORTAbits.RA5
- #define SHUTTER_PIN LATAbits.LATA2
- #define _XTAL_FREQ 32000000UL
- #define FCY SYS_FREQ/4

Functions

- void Hardware_ConfigureOscillator (void)
- void Hardware_initIO (void)
- void Hardware_initUART (void)
- void Hardware_UARTsendByte (uint8_t)
- void Hardware_UARTsendString (char[ ])

5.2.1 Macro Definition Documentation

5.2.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 32000000UL
```

Definition at line 13 of file hardware.h.

5.2.1.2 DIP1_PIN

```
#define DIP1_PIN PORTAbits.RA1
```

Definition at line 4 of file hardware.h.

5.2.1.3 DIP3_PIN

```
#define DIP3_PIN PORTAbits.RA0
```

Definition at line 3 of file hardware.h.

5.2.1.4 FCY

```
#define FCY SYS_FREQ/4
```

Definition at line 14 of file hardware.h.

5.2.1.5 PIR_PIN

```
#define PIR_PIN PORTAbits.RA5
```

Definition at line 6 of file hardware.h.

5.2.1.6 SHUTTER_PIN

```
#define SHUTTER_PIN LATAbits.LATA2
```

Definition at line 9 of file hardware.h.

5.2.2 Function Documentation

5.2.2.1 Hardware_ConfigureOscillator()

```
void Hardware_ConfigureOscillator (
            void  )
```

Setup oscillator

Author

Thomas Evison

Date

28/10/2020

Definition at line 31 of file hardware.c.

5.2.2.2 Hardware_initIO()

```
void Hardware_initIO (
            void  )
```

Initialize hardware registers to setup IO, Interrupts, etc

Author

Thomas Evison

Date

28/10/2020

Definition at line 45 of file hardware.c.

5.2.2.3 Hardware_initUART()

```
void Hardware_initUART (
            void  )
```

Initialize hardware UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 77 of file hardware.c.

5.2.2.4 Hardware_UARTsendByte()

```
void Hardware_UARTsendByte (
            uint8_t Tx )
```

Send a single byte via UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 105 of file hardware.c.

5.2.2.5 Hardware_UARTsendString()

```
void Hardware_UARTsendString (
            char Tx[] )
```

Send string via hardware UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 115 of file hardware.c.

5.3 headers/pygmy.h File Reference

Functions

- void Pygmy_init (void)
- uint8_t Pygmy_readDipSwitches (void)
- void Pygmy_delay_ms (uint16_t)
- void Pygmy_TriggeredPIR (void)
- void Pygmy_camParamsToString (void)
- bool Pygmy_validCmd (char)
- uint8_t * Pygmy_handleMsg (uint8_t[])

5.3.1 Function Documentation

5.3.1.1 Pygmy_camParamsToString()

```
void Pygmy_camParamsToString (
            void  )
```

5.3.1.2 Pygmy_delay_ms()

```
void Pygmy_delay_ms (
            uint16_t  )
```

5.3.1.3 Pygmy_handleMsg()

```
uint8_t* Pygmy_handleMsg (
            uint8_t [] )
```

5.3.1.4 Pygmy_init()

```
void Pygmy_init (
            void  )
```

5.3.1.5 Pygmy_readDipSwitches()

```
uint8_t Pygmy_readDipSwitches (
            void  )
```

5.3.1.6 Pygmy_TriggeredPIR()

```
void Pygmy_TriggeredPIR (
            void  )
```

5.3.1.7 Pygmy_validCmd()

```
bool Pygmy_validCmd (
            char  )
```

5.4 headers/system.h File Reference

Functions

- void ConfigureOscillator (void)

5.4.1 Function Documentation

5.4.1.1 ConfigureOscillator()

```
void ConfigureOscillator (
            void  )
```

5.5 README.md File Reference

5.6 src/configbits.h File Reference

5.7 src/eusart.c File Reference

Configuration and utilities for PIC hardware UART.

```
#include "eusart.h"
#include <string.h>
```

Macros

- #define EUSART_TX_BUFFER_SIZE 8
- #define EUSART_RX_BUFFER_SIZE 8

Functions

- void EUSART_DefaultFramingErrorHandler (void)
- void EUSART_DefaultOverrunErrorHandler (void)
- void EUSART_DefaultErrorHandler (void)
- void EUSART_Initialize (void)
- bool EUSART_is_tx_ready (void)
- bool EUSART_is_rx_ready (void)
- bool EUSART_is_tx_done (void)
- eusart_status_t EUSART_get_last_status (void)
- uint8_t EUSART_Read (void)
- void EUSART_Write (uint8_t txData)
- void EUSART_WriteString (uint8_t txData[ ])
- void EUSART_Transmit_ISR (void)
- void EUSART_Receive_ISR (void)
- void EUSART_RxDataHandler (void)
- void EUSART_SetFramingErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetOverrunErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetErrorHandler (void(∗interruptHandler)(void))
- void EUSART_SetTxInterruptHandler (void(∗interruptHandler)(void))
- void EUSART_SetRxInterruptHandler (void(∗interruptHandler)(void))

Variables

- volatile uint8_t eusartTxHead = 0
- volatile uint8_t eusartTxTail = 0
- volatile uint8_t eusartTxBuffer [EUSART_TX_BUFFER_SIZE]
- volatile uint8_t eusartTxBufferRemaining
- volatile uint8_t eusartRxHead = 0
- volatile uint8_t eusartRxTail = 0
- volatile uint8_t eusartRxBuffer [EUSART_RX_BUFFER_SIZE]
- volatile eusart_status_t eusartRxStatusBuffer [EUSART_RX_BUFFER_SIZE]
- volatile uint8_t eusartRxCount
- volatile eusart_status_t eusartRxLastError
- void(∗ EUSART_TxDefaultInterruptHandler)(void)
- void(∗ EUSART_RxDefaultInterruptHandler)(void)
- void(∗ EUSART_FramingErrorHandler)(void)
- void(∗ EUSART_OverrunErrorHandler)(void)
- void(∗ EUSART_ErrorHandler)(void)

5.7.1 Detailed Description

Configuration and utilities for PIC hardware UART.

Author

Thomas Evison

Date

28/10/2020

5.7.2 Macro Definition Documentation

5.7.2.1 EUSART_RX_BUFFER_SIZE

```
#define EUSART_RX_BUFFER_SIZE 8
```

Definition at line 12 of file eusart.c.

5.7.2.2 EUSART_TX_BUFFER_SIZE

```
#define EUSART_TX_BUFFER_SIZE 8
```

Definition at line 11 of file eusart.c.

5.7.3 Function Documentation

5.7.3.1 EUSART_DefaultErrorHandler()

```
void EUSART_DefaultErrorHandler (
            void  )
```

Definition at line 203 of file eusart.c.

5.7.3.2 EUSART_DefaultFramingErrorHandler()

```
void EUSART_DefaultFramingErrorHandler (
            void  )
```

Definition at line 192 of file eusart.c.

5.7.3.3 EUSART_DefaultOverrunErrorHandler()

```
void EUSART_DefaultOverrunErrorHandler (
            void  )
```

Definition at line 195 of file eusart.c.

5.7.3.4 EUSART_get_last_status()

```
eusart_status_t EUSART_get_last_status (
            void  )
```

@Summary Gets the error status of the last read byte.

@Description This routine gets the error status of the last read byte.

@Preconditions EUSART_Initialize() function should have been called before calling this function. The returned value is only updated after a read is called.

@Param None

@Returns the status of the last read byte

@Example `void main(void) { volatile uint8_t rxData; volatile eusart_status_t rxStatus;`

`Initialize the device SYSTEM_Initialize();`

`Enable the Global Interrupts INTERRUPT_GlobalInterruptEnable();`

`while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData = EUSART_Read(); rxStatus = EUSART_get_last_status(); if(rxStatus.ferr){ LED_0_SetHigh(); } } } }`

Definition at line 98 of file eusart.c.

5.7.3.5 EUSART_Initialize()

```
void EUSART_Initialize (
            void  )
```

@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

Definition at line 43 of file eusart.c.

5.7.3.6 EUSART_is_rx_ready()

```
bool EUSART_is_rx_ready (
            void  )
```

@Summary Checks if the EUSART receiver ready for reading

@Description This routine checks if EUSART receiver has received data and ready to be read

@Preconditions EUSART_Initialize() function should be called before calling this function EUSART receiver should be enabled before calling this function

@Param None

@Returns Status of EUSART receiver TRUE: EUSART receiver is ready for reading FALSE: EUSART receiver is not ready for reading

@Example
```
void main(void) { volatile uint8_t rxData;

Initialize the device SYSTEM_Initialize();

while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData =
UART1_Read(); if(EUSART_is_tx_ready()) { EUSART_Write(rxData); } } } }
```

Definition at line 90 of file eusart.c.

5.7.3.7 EUSART_is_tx_done()

```
bool EUSART_is_tx_done (
            void  )
```

@Summary Checks if EUSART data is transmitted

@Description This function return the status of transmit shift register

@Preconditions EUSART_Initialize() function should be called before calling this function EUSART transmitter should be enabled and EUSART_Write should be called before calling this function

@Param None

@Returns Status of EUSART receiver TRUE: Data completely shifted out if the USART shift register FALSE: Data is not completely shifted out of the shift register

@Example
```
void main(void) { volatile uint8_t rxData;

Initialize the device SYSTEM_Initialize();

while(1) { if(EUSART_is_tx_ready()) { LED_0_SetHigh(); EUSARTWrite(rxData);
} if(EUSART_is_tx_done() { LED_0_SetLow(); } } }
```

Definition at line 94 of file eusart.c.

5.7.3.8 EUSART_is_tx_ready()

```
bool EUSART_is_tx_ready (
            void  )
```

@Summary Checks if the EUSART transmitter is ready to transmit data

@Description This routine checks if EUSART transmitter is ready to accept and transmit data byte

@Preconditions EUSART_Initialize() function should have been called before calling this function. EUSART transmitter should be enabled before calling this function

@Param None

@Returns Status of EUSART transmitter TRUE: EUSART transmitter is ready FALSE: EUSART transmitter is not ready

@Example `void main(void) { volatile uint8_t rxData;`

`Initialize the device SYSTEM_Initialize();`

`while(1) { Logic to echo received data if(EUSART_is_rx_ready()) { rxData = UART1_Read(); if(EUSART_is_tx_ready()) { EUSARTWrite(rxData); } } } }`

Definition at line 86 of file eusart.c.

5.7.3.9 EUSART_Read()

```
uint8_t EUSART_Read (
            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.

Definition at line 102 of file eusart.c.

5.7.3.10 EUSART_Receive_ISR()

```
void EUSART_Receive_ISR (
            void  )
```

@Summary Maintains the driver's receiver state machine and implements its ISR

@Description This routine is used to maintain the driver's internal receiver state machine.This interrupt service routine is called when the state of the receiver needs to be maintained in a non polled manner.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 160 of file eusart.c.

5.7.3.11 EUSART_RxDataHandler()

```
void EUSART_RxDataHandler (
            void  )
```

@Summary Maintains the driver's receiver state machine

@Description This routine is called by the receive state routine and is used to maintain the driver's internal receiver state machine. It should be called by a custom ISR to maintain normal behavior

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 183 of file eusart.c.

5.7.3.12 EUSART_SetErrorHandler()

```
void EUSART_SetErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Error Handler

@Description This API sets the function to be called upon EUSART error

@Preconditions Initialize the EUSART module before calling this API

@Param Address of function to be set as error handler

@Returns None

Definition at line 215 of file eusart.c.

5.7.3.13 EUSART_SetFramingErrorHandler()

```
void EUSART_SetFramingErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Framing Error Handler

@Description This API sets the function to be called upon EUSART framing error

@Preconditions Initialize the EUSART before calling this API

@Param Address of function to be set as framing error handler

@Returns None

Definition at line 207 of file eusart.c.

5.7.3.14 EUSART_SetOverrunErrorHandler()

```
void EUSART_SetOverrunErrorHandler (
            void(*)(void) interruptHandler )
```

@Summary Set EUSART Overrun Error Handler

@Description This API sets the function to be called upon EUSART overrun error

@Preconditions Initialize the EUSART module before calling this API

@Param Address of function to be set as overrun error handler

@Returns None

Definition at line 211 of file eusart.c.

5.7.3.15 EUSART_SetRxInterruptHandler()

```
void EUSART_SetRxInterruptHandler (
            void(*)(void) interruptHandler )
```

@Summary Sets the receive handler function to be called by the interrupt service

@Description Calling this function will set a new custom function that will be called when the receive interrupt needs servicing.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param A pointer to the new function

@Returns None

Definition at line 223 of file eusart.c.

5.7.3.16 EUSART_SetTxInterruptHandler()

```
void EUSART_SetTxInterruptHandler (
             void(*)(void) interruptHandler )
```

@Summary Sets the transmit handler function to be called by the interrupt service

@Description Calling this function will set a new custom function that will be called when the transmit interrupt needs servicing.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param A pointer to the new function

@Returns None

Definition at line 219 of file eusart.c.

5.7.3.17 EUSART_Transmit_ISR()

```
void EUSART_Transmit_ISR (
             void  )
```

Definition at line 146 of file eusart.c.

5.7.3.18 EUSART_Write()

```
void EUSART_Write (
             uint8_t txData )
```

@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

Definition at line 121 of file eusart.c.

5.7.3.19 EUSART_WriteString()

```
void EUSART_WriteString (
            uint8_t [] )
```

@Summary Maintains the driver's transmitter state machine and implements its ISR.

@Description This routine is used to maintain the driver's internal transmitter state machine.This interrupt service routine is called when the state of the transmitter needs to be maintained in a non polled manner.

@Preconditions EUSART_Initialize() function should have been called for the ISR to execute correctly.

@Param None

@Returns None

Definition at line 138 of file eusart.c.

5.7.4 Variable Documentation

5.7.4.1 EUSART_ErrorHandler

```
void(* EUSART_ErrorHandler) (void)
```

Definition at line 37 of file eusart.c.

5.7.4.2 EUSART_FramingErrorHandler

```
void(* EUSART_FramingErrorHandler) (void)
```

Definition at line 35 of file eusart.c.

5.7.4.3 EUSART_OverrunErrorHandler

```
void(* EUSART_OverrunErrorHandler) (void)
```

Definition at line 36 of file eusart.c.

5.7.4.4 EUSART_RxDefaultInterruptHandler

```
void(* EUSART_RxDefaultInterruptHandler) (void)
```

Definition at line 33 of file eusart.c.

5.7.4.5 EUSART_TxDefaultInterruptHandler

```
void(* EUSART_TxDefaultInterruptHandler) (void)
```

Section: EUSART APIs

Definition at line 32 of file eusart.c.

5.7.4.6 eusartRxBuffer

```
volatile uint8_t eusartRxBuffer[EUSART_RX_BUFFER_SIZE]
```

Definition at line 24 of file eusart.c.

5.7.4.7 eusartRxCount

```
volatile uint8_t eusartRxCount
```

Definition at line 26 of file eusart.c.

5.7.4.8 eusartRxHead

```
volatile uint8_t eusartRxHead = 0
```

Definition at line 22 of file eusart.c.

5.7.4.9 eusartRxLastError

```
volatile eusart_status_t eusartRxLastError
```

Definition at line 27 of file eusart.c.

5.7.4.10 eusartRxStatusBuffer

```
volatile eusart_status_t eusartRxStatusBuffer[EUSART_RX_BUFFER_SIZE]
```

Definition at line 25 of file eusart.c.

5.7.4.11 eusartRxTail

```
volatile uint8_t eusartRxTail = 0
```

Definition at line 23 of file eusart.c.

5.7.4.12 eusartTxBuffer

```
volatile uint8_t eusartTxBuffer[EUSART_TX_BUFFER_SIZE]
```

Definition at line 19 of file eusart.c.

5.7.4.13 eusartTxBufferRemaining

```
volatile uint8_t eusartTxBufferRemaining
```

Section: Global variables

Definition at line 20 of file eusart.c.

5.7.4.14 eusartTxHead

```
volatile uint8_t eusartTxHead = 0
```

Section: Global Variables

Definition at line 17 of file eusart.c.

5.7.4.15 eusartTxTail

```
volatile uint8_t eusartTxTail = 0
```

Definition at line 18 of file eusart.c.

5.8 src/fifo.c File Reference

5.9 src/fifo.h File Reference

5.10 src/hardware.c File Reference

Takes care of hardware initialisation.

```
#include <xc.h>
#include <stdint.h>
#include <stdbool.h>
#include "system.h"
#include "hardware.h"
#include <string.h>
```

Functions

- void Hardware_ConfigureOscillator ()
- void Hardware_initIO ()
- void Hardware_initUART ()
- void Hardware_UARTsendByte (uint8_t Tx)
- void Hardware_UARTsendString (char Tx[ ])

5.10.1 Detailed Description

Takes care of hardware initialisation.

Author

Thomas Evison

Date

28/10/2020

Vdd |1 8| GND/VSS PIR sensor (in) GP5 |2 7| GP0 (in) DIP3 SHUTTER (out)GP4 |3 6| GP1 (in) DIP1

5.10.1.1 (in) GP3 |4 5| GP2 (in) DIP2

5.10.2 Function Documentation

5.10.2.1 Hardware_ConfigureOscillator()

```
void Hardware_ConfigureOscillator (
            void  )
```

Setup oscillator

Author

Thomas Evison

Date

28/10/2020

Definition at line 31 of file hardware.c.

5.10.2.2 Hardware_initIO()

```
void Hardware_initIO (
            void  )
```

Initialize hardware registers to setup IO, Interrupts, etc

Author

Thomas Evison

Date

28/10/2020

Definition at line 45 of file hardware.c.

5.10.2.3 Hardware_initUART()

```
void Hardware_initUART (
            void  )
```

Initialize hardware UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 77 of file hardware.c.

5.10.2.4 Hardware_UARTsendByte()

```
void Hardware_UARTsendByte (
            uint8_t Tx )
```

Send a single byte via UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 105 of file hardware.c.

5.10.2.5 Hardware_UARTsendString()

```
void Hardware_UARTsendString (
            char Tx[] )
```

Send string via hardware UART

Author

Thomas Evison

Date

28/10/2020

Definition at line 115 of file hardware.c.

5.11 src/interrupts.c File Reference

interrupt handler for all interrupt triggers

```
#include <xc.h>
#include <stdint.h>
#include <stdbool.h>
#include "pygmy.h"
#include <string.h>
#include "eusart.h"
```

Functions

- void UART_sendString (char[ ])
- void UART_sendByte (uint8_t)
- void __interrupt () isr(void)

Variables

- volatile uint8_t array [16]

5.11.1 Detailed Description

interrupt handler for all interrupt triggers

Author

Thomas Evison

Date

28/10/2020

5.11.2 Function Documentation

5.11.2.1 __interrupt()

```
void __interrupt ( )
```

Interrupt handler - first checks which flags have been set, then will call corresponding interrupt subroutine

Author

Thomas Evison

Date

28/10/2020

Definition at line 30 of file interrupts.c.

5.11.2.2 UART_sendByte()

```
void UART_sendByte (
            uint8_t  )
```

5.11.2.3 UART_sendString()

```
void UART_sendString (
            char [] )
```

5.11.3 Variable Documentation

5.11.3.1 array

```
volatile uint8_t array[16]  [extern]
```

5.12 src/main.c File Reference

main file to initialize components and initiate main loop

```
#include "configbits.h"
#include <xc.h>
#include <stdint.h>
#include "hardware.h"
#include <stdbool.h>
#include "system.h"
#include "pygmy.h"
#include "eusart.h"
#include <string.h>
```

Macros

- #define USB_POWERED true

Functions

- void menuRun (void)
- void main (void)

5.12.1 Detailed Description

main file to initialize components and initiate main loop

Author

Thomas Evison

Date

28/10/2020

5.12.2 Macro Definition Documentation

5.12.2.1 USB_POWERED

```
#define USB_POWERED true
```

Definition at line 28 of file main.c.

5.12.3 Function Documentation

5.12.3.1 main()

```
void main (
            void  )
```

Main entry point of program

Author

Thomas Evison

Date

28/10/2020

Definition at line 35 of file main.c.

5.12.3.2 menuRun()

```
void menuRun (
            void  )
```

Outputs a brief message to serial

Author

Thomas Evison

Date

28/10/2020

Definition at line 147 of file main.c.

5.13 src/pygmy.c File Reference

Contains all functions and utilities to load settings and control the camera.

```
#include <xc.h>
#include <stdint.h>
#include <stdlib.h>
#include "hardware.h"
#include <stdbool.h>
#include <string.h>
```

5.13.1 Detailed Description

Contains all functions and utilities to load settings and control the camera.

Author

Thomas Evison

Date

28/10/2020

Index