TGRMach - Feature #216

Programar la detección de los contactos del Joystick.

02/08/2016 07:30 PM - Txinto Vaz

Status:	Closed	**Start date:**	02/01/2016
Priority:	Normal	**Due date:**	
Assignee:	Txinto Vaz	**% Done:**	100%
Category:		**Estimated time:**	2.00 hours
Target version:	v0.1_USB	**Spent time:**	4.00 hours

Description

Teniendo en cuenta el cableado de los pines. Desarrollar contenido de detección de contactos de joystick para los siguientes módulos:

Hay que usar la arquitectura de software de gatATAC Osek

- prj_cfg.h → información genérica de configuración.
- prj_dre.h y prj_dre.c → estructuras de datos para guardar la detección de cada uno de los contactos, e inicialización de las mismas.
- prj_pinout.h y prj_pinout.c → asignación de contactos a puertos y configuración de los mismos.
- prj_input.h y prj_input.c → lectura de puertos y relleno de estructuras de datos para reflejar el estado actual de la detección.

Related issues:

Blocked by TGRMach - Feature #215: Cablear contactos de los botones a puertos...	**Closed**	**02/08/2016**

History

#1 - 02/08/2016 07:38 PM - Txinto Vaz

Creamos los siguientes ficheros:

Fichero de configuración genérica: prj_cfg.h

```
#ifndef _PRJ_CFG_H
#define _PRJ_CFG_H

...

/**** Joystick detection section ****/
#define CFG_JOYSTICK_NUMBER_OF_POSITIONS 4
#define CFG_JOYSTICK_NUMBER_OF_BUTTONS 11

#define CFG_JOY_UP_IDX 0
#define CFG_JOY_DOWN_IDX 1
#define CFG_JOY_LEFT_IDX 2
#define CFG_JOY_RIGHT_IDX 3

#define CFG_JOY_GREENBUT_IDX 0
#define CFG_JOY_REDBUT_IDX 1
#define CFG_JOY_BLUEBUT_IDX 2
#define CFG_JOY_YELLOWBUT_IDX 3
#define CFG_JOY_GREYBUT_IDX 4
#define CFG_JOY_BLACKBUT_IDX 5
#define CFG_JOY_WHITELBUT_IDX 6
#define CFG_JOY_WHITERBUT_IDX 7
#define CFG_JOY_MODEBUT_IDX 8
#define CFG_JOY_RESTARTBUT_IDX 9
#define CFG_JOY_PLAYERBUT_IDX 10

...

#endif /* _PRJ_CFG_H */
```

DRE (Data Runtime Environment)

Llamamos DRE a una estructura de datos que centraliza todas las variables globales del sistema.
La parte de detección del DRE se detalla a continuación:

Definición de estructuras datos. prj_dre.h

```
/* Data Runtime Environment is a pool of global variables that can be shared through all the application.  It
acts as a centralized data base for the functionality */

#ifndef _PRJ_DRE_H
#define _PRJ_DRE_H

/* Inclusion of the basic types of the GTTC osek */
#include "gttc_types.h"
/* Configuration of the project */
#include "prj_cfg.h"

typedef struct {
  uint8_t joy_pins[CFG_JOYSTICK_NUMBER_OF_POSITIONS];
  uint8_t but_pins[CFG_JOYSTICK_NUMBER_OF_BUTTONS];
  byte allButtons[CFG_JOYSTICK_NUMBER_OF_BUTTONS];
  byte prevButtons[CFG_JOYSTICK_NUMBER_OF_BUTTONS];
  bool detectedJoy[CFG_JOYSTICK_NUMBER_OF_POSITIONS];
  int angle;
  int prev_angle;
} t_detection;

/* This structure type contains the DRE global variables */
typedef struct {
  t_detection detection;
} t_dre;

/* This declaration make the dre visible to all the project modules */
extern t_dre dre;

/* This function initializes the DRE contents to their default values */
void dreInit(void);

#endif // _PRJ_DRE_H
```

Inicialización de las estructuras de datos: prj_dre.cpp

```
/* Data Runtime Environment is a pool of global variables that can be shared through all the application.  It
acts as a centralized data base for the functionality */

#include "gttc_types.h"
#include "prj_cfg.h"
#include "prj_dre.h"
#include "Arduino.h"
#include "gttc_timer.h"
#include "prj_pinout.h"

t_dre dre;

/* This function initializes the DRE contents to their default values */
void dreInit(void) {

  Serial.println("DRE Initialization");

  dre.detection.angle=-1;
  dre.detection.prev_angle=-1;

/**** Decoding structures for joysticks and buttons */
  dre.detection.joy_pins[CFG_JOY_UP_IDX]=CFG_JOY_UP_PIN;
  dre.detection.joy_pins[CFG_JOY_DOWN_IDX]=CFG_JOY_DOWN_PIN;
  dre.detection.joy_pins[CFG_JOY_LEFT_IDX]=CFG_JOY_LEFT_PIN;
  dre.detection.joy_pins[CFG_JOY_RIGHT_IDX]=CFG_JOY_RIGHT_PIN;

  dre.detection.but_pins[CFG_JOY_GREENBUT_IDX]=CFG_JOY_GREENBUT_PIN;
  dre.detection.but_pins[CFG_JOY_REDBUT_IDX]=CFG_JOY_REDBUT_PIN;
  dre.detection.but_pins[CFG_JOY_BLUEBUT_IDX]=CFG_JOY_BLUEBUT_PIN;
  dre.detection.but_pins[CFG_JOY_YELLOWBUT_IDX]=CFG_JOY_YELLOWBUT_PIN;
  dre.detection.but_pins[CFG_JOY_GREYBUT_IDX]=CFG_JOY_GREYBUT_PIN;
  dre.detection.but_pins[CFG_JOY_BLACKBUT_IDX]=CFG_JOY_BLACKBUT_PIN;
  dre.detection.but_pins[CFG_JOY_WHITELBUT_IDX]=CFG_JOY_WHITELBUT_PIN;
  dre.detection.but_pins[CFG_JOY_WHITERBUT_IDX]=CFG_JOY_WHITERBUT_PIN;
  dre.detection.but_pins[CFG_JOY_MODEBUT_IDX]=CFG_JOY_MODEBUT_PIN;
  dre.detection.but_pins[CFG_JOY_RESTARTBUT_IDX]=CFG_JOY_RESTARTBUT_PIN;
  dre.detection.but_pins[CFG_JOY_PLAYERBUT_IDX]=CFG_JOY_PLAYERBUT_PIN;
```

```
}
```

Ficheros de gestión del pin-out.

Configuración. prj_pinout.h

```
/* ---- This file describes all the pin assignments of the microcontroller --- */

#ifndef _PRJ_PINOUT_H
#define _PRJ_PINOUT_H

/* Inclusion of the basic types of the GTTC osek */
#include "gttc_types.h"

/**** Pin assigment section ****/
#define CFG_POWERGND_PIN 2

#define CFG_JOY_UP_PIN 23
#define CFG_JOY_DOWN_PIN 21
#define CFG_JOY_LEFT_PIN 22
#define CFG_JOY_RIGHT_PIN 20

#define CFG_JOY_GREENBUT_PIN 19
#define CFG_JOY_REDBUT_PIN 18
#define CFG_JOY_BLUEBUT_PIN 17
#define CFG_JOY_YELLOWBUT_PIN 16
#define CFG_JOY_GREYBUT_PIN 15
#define CFG_JOY_BLACKBUT_PIN 14
#define CFG_JOY_WHITELBUT_PIN 12
#define CFG_JOY_WHITERBUT_PIN 11
#define CFG_JOY_MODEBUT_PIN 10
#define CFG_JOY_RESTARTBUT_PIN 9
#define CFG_JOY_PLAYERBUT_PIN 8

/* Prepares the pinout for project use */
void pinoutInit(void);

#endif // _PRJ_PINOUT_H
```

Inicialización. prj_pinout.cpp

```
/* Inclusion of the basic types of the GTTC osek */
#include "gttc_types.h"
/* Configuration of the project */
#include "prj_cfg.h"
/* Inclusion of the DRE of the project */
#include "prj_dre.h"
/* Inclusion of its own header */
#include "prj_pinout.h"

/* Prepares the pinout for project use */
void pinoutInit(void){
  /* Configure the power led DO pin */
  pinMode(CFG_POWERGND_PIN,OUTPUT);

  /* Configure the joystick DI pins */
  for (uint8_t i=0; i<CFG_JOYSTICK_NUMBER_OF_POSITIONS; i++) {
    pinMode(dre.detection.joy_pins[i], INPUT_PULLUP);
  }

  /* Configure the buttons DI pins */
  for (uint8_t i=0; i<CFG_JOYSTICK_NUMBER_OF_BUTTONS; i++) {
    pinMode(dre.detection.but_pins[i], INPUT_PULLUP);
  }
}
```

#2 - 02/08/2016 08:03 PM - Txinto Vaz

- Description updated

#3 - 02/08/2016 08:03 PM - Txinto Vaz

- Blocked by Feature #215: Cablear contactos de los botones a puertos digitales de entrada. added

#4 - 02/08/2016 08:07 PM - Txinto Vaz

Rutinas de detección

Configuración prj_input.h

```
#ifndef _PRJ_INPUT_H
#define _PRJ_INPUT_H

/* Inclusion of the basic types of the GTTC osek */
#include "gttc_types.h"

/* Initialization of the input routines.
WARNING: The pinout initialization is not done here.
It is done in the prj_pinout.c file */
void prjInputInit(void);

/* Routine to read all the inputs in the system
EXCEPTION: The inputs coming through an I/O bus can be acquired in another module.*/
void prjInput(void);

#endif /* _PRJ_INPUT_H */
```

Detección prj_input.h

```
/* This file perform the acquisition tasks of the inputs of the system */

/* Inclusion of the basic types of the GTTC osek */
#include "gttc_types.h"
/* Configuration of the project */
#include "prj_cfg.h"
/* Inclusion of its own header */
#include "prj_input.h"
/* Inclusion of the pinout information */
#include "prj_pinout.h"
/* Inclusion of the DRE */
#include "prj_dre.h"
/* Inclusion of the arduino methods */
#include "Arduino.h"

  /*
    LEGACY CODE THAT DEMONSTRATES HOW TO READ THE DIGITAL INPUTS OF A JOYSTICK AND ENCODE THEM AS THE HEAD PAD
 BUTTON
void prjInputReadHat(void){
  // Read the "hat" controller
  for (uint8_t i=0; i<sizeof(joy_pins) ; i++) {
    if (digitalRead(joy_pins[i])) {
      detectedJoy[i]=false;
    } else {
      detectedJoy[i]=true;
      digitalWrite(CFG_POWERGND_PIN,HIGH);
    }
  }
  if (detectedJoy[CFG_JOY_UP_IDX]){
    if (detectedJoy[CFG_JOY_LEFT_IDX]){
      angle=315;
    } else {
      if (detectedJoy[CFG_JOY_RIGHT_IDX]){
        angle=45;
      } else {
        angle=0;
      }
    }
  } else {
    if (detectedJoy[CFG_JOY_DOWN_IDX]){
      if (detectedJoy[CFG_JOY_LEFT_IDX]){
        angle=225;
      } else {
        if (detectedJoy[CFG_JOY_RIGHT_IDX]){
          angle=135;
        } else {
          angle=180;
        }
      }
    } else {
```

```
      if (detectedJoy[CFG_JOY_LEFT_IDX]){
        angle=270;
      } else {
        if (detectedJoy[CFG_JOY_RIGHT_IDX]){
          angle=90;
        } else {
          angle=-1;
        }
      }
    }
  }
  if (angle!=prev_angle){
    Serial.print("Hat: ");
    Serial.println(angle);
    prev_angle=angle;
  }
  Joystick.hat(angle);
}
*/

/* Initialization of the input routines.
WARNING: The pinout initialization is not done here.
It is done in the prj_pinout.c file */
void prjInputInit(void) {

}

/* Routine to read all the inputs in the system
EXCEPTION: The inputs coming through an I/O bus can be acquired in another module.*/
void prjInput(void) {

/*************** BUTTON SECTION ********************/

  // read digital pins and use them for the buttons
  for (uint8_t i=0; i<CFG_JOYSTICK_NUMBER_OF_BUTTONS ; i++) {
    if (digitalRead(dre.detection.but_pins[i])==HIGH) {
      // when a pin reads high, the button is not pressed
      // the pullup resistor creates the "on" signal
      dre.detection.allButtons[i] = 0;
    } else {
      // when a pin reads low, the button is connecting to ground.
      dre.detection.allButtons[i] = 1;
#ifdef DEBUG_BUT_INPUT
      Serial.print("Detectado botón ");
      Serial.println(i);
#endif
    }
  }

/* LEGACY CODE: SEND THROUGH SERIAL PORT WHEN A BUTTON HAS CHANGED
  // check to see if any button changed since last time
  boolean anyChange = false;
  for (uint8_t i=0; i<sizeof(but_pins); i++) {
    if (allButtons[i] != prevButtons[i]) {
      anyChange = true;
    }
    prevButtons[i] = allButtons[i];
  }

  // if any button changed, print them to the serial monitor
  if (anyChange) {
    Serial.print("Buttons: ");
    for (uint8_t i=0; i<sizeof(but_pins); i++) {
      Serial.print(allButtons[i], DEC);
    }
    Serial.println();
  }
*/

/************ JOYSTICK SECTION ****************/

  for (uint8_t i=0; i<CFG_JOYSTICK_NUMBER_OF_POSITIONS ; i++) {
    if (digitalRead(dre.detection.joy_pins[i])==HIGH) {
      dre.detection.detectedJoy[i]=false;
    } else {
```

```
      dre.detection.detectedJoy[i]=true;
#ifdef DEBUG_JOY_INPUT
      Serial.print("Detectado joy ");
      Serial.println(i);
#endif
    }
  }

/*
    LEGACY CODE THAT DEMONSTRATES HOW TO READ THE DIGITAL INPUTS OF A JOYSTICK AND ENCODE THEM AS THE HEAD PAD
 BUTTON
prjInputReadHat();
*/
}
```

#5 - 02/08/2016 08:08 PM - Txinto Vaz

- Status changed from New to In Progress

- % Done changed from 0 to 100

#6 - 02/08/2016 08:10 PM - Txinto Vaz

- Status changed from In Progress to Resolved

#7 - 02/08/2016 08:10 PM - Txinto Vaz

- Status changed from Resolved to Closed