Astro HUD

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Clock Class Reference

The Clock object calculates and stores the mission time.

`#include <Clock.h>`

Public Member Functions

- Clock ()
  Clock constructor.
- void update ()
- void constraints ()

Public Attributes

- int hour
  Stores hour value of mission clock.
- int min
  Stores minute value of mission clock.
- int sec
  Stores second value of mission clock.
- unsigned long **prevMillis**
- unsigned long **currentMillis**
- unsigned long interval
  Used for hardware interupt. Value depends on hardware clock speed.

2.1.1 Detailed Description

The Clock object calculates and stores the mission time.

A hardware clock is used to create an interupt once a second to increment the sec property of the Clock object. Based on the sec value, the min and hour properties are updated.

2.1.2 Member Function Documentation

2.1.2.1 void Clock::constraints ( )

The constraints method checks whether the sec or min property is equal to 60. If so, the property is set to 0, and the min or hour property is incremented appropriately.

2.1.2.2 void Clock::update ( )

The update method is called by the hardware interupt handler. It increments sec and calculates constraints.

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

- /home/bryan/GIT_projects/Astro-HUD/libraries/Clock/Clock.h

2.2 Sensor Class Reference

Sensor objects store information & provide methods for a given hardware sensor.

`#include <Sensor.h>`

Public Member Functions

- Sensor (String name, int Pin, float Hi_thresh, float Lo_thresh, float Hi_bound, float Lo_bound, float Conv_↩ coef, float Conv_offset)

  Constructor for Sensor object.
- void update ()

  Dynamically updates all sensor properties.
- void check_thresh ()
- void convert ()
- void calc_priority ()

Public Attributes

- String display_name

  The name of the sensor (used on the HUD).
- unsigned short int pin
- unsigned short int sensor_read

  The raw value read off the assigned hardware pin.
- float display_value

  The sensor value converted into approriated units.
- float hi_bound
- float lo_bound
- float hi_thresh
- float lo_thresh
- float conv_coef
- float conv_offset
- float priority
- float priority_offset
- bool display_me
- unsigned short int last_display_x
- unsigned short int last_display_y
- unsigned short int last_display_value_x
- unsigned short int last_display_value_y
- float last_display_value

  Tracks sensor's display_value. Used by front end to clear OLED.

2.2.1 Detailed Description

Sensor objects store information & provide methods for a given hardware sensor.

For each hardware sensor connected to the HUD system a Sensor object must be instantiated. This object is used to handle all actions pertaining to the hardware sensor.

From the point of view of the microcontroller running the HUD system code, a hardware sensor is simply a voltage on one of the GPIO pins. For the voltage to be converted into meaningful data, it is necessary to keep track of information about the hardware sensor and tie that information to the value read on the given GPIO pin. For example, in order for a voltage to represent a given physical parameter, it is necessary to know the conversion formula from volts to the desired physical units.

An instantiated Sensor object keeps track of all necessary sensor properties, such as the sensor's name, units, conersion factors, current and previous values, etc.. It also has methods for performing various sensor related actions, such as reading a voltage off of a pin, converting that value into the specified units, and updating the sensors priority based on external information.

There are also various properties and operations specific to the HUD system that need to be tracked in order to make decisions about how and when to display information on the HUD (see calc_priority, hi_thresh, lo_thresh, and priority). The Sensor object is also used to define and utilize these HUD specific members.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Sensor::Sensor (String *name,* int *Pin,* float *Hi_thresh,* float *Lo_thresh,* float *Hi_bound,* float *Lo_bound,* float *Conv_coef,* float *Conv_offset*)

Constructor for Sensor object.

The parameters of the Sensor constructor include of all the user-specified (via config file) hardware sensor properties which cannot be determined internally. These values are assigned upon instantiation to the corresponding Sensor properties. Any other properties are assigned default values (generally zero).

Note: By convention, parameter names are capitalized, and the corresponding Sensor property goes by the same name, except in all lower-case.

2.2.3 Member Function Documentation

2.2.3.1 void Sensor::calc_priority ( )

Calling calc_priority up calculate priority. The formula is seen here:

2.2.3.2 void Sensor::check_thresh ( )

Calling check_thresh will determine whether the current display_value is outside hi_thresh and lo_thresh. If it is outside the threshold, display_me is set to *True*. Otherwise it is set to *False*.

2.2.3.3 void Sensor::convert ( )

Calling convert will calculate and update the sensor value in its proper units. The formula is:

```
display_value = conv_coef * sensor_read + conv_offset
```

2.2.3.4 void Sensor::update ( )

Dynamically updates all sensor properties.

Calling update will read the assigned pin value, then call convert, check_thresh, and calc_priority.

2.2.4 Member Data Documentation

2.2.4.1 float Sensor::conv_coef

The relationship between the voltage produced by the hardware sensor and the physical parameter represented is assumed to be linear. This is a good approximation in most practical applications. conv_coef is

2.2.4.2 float Sensor::conv_offset

The relationship between the voltage produced by the hardware sensor and the physical parameter represented is assumed to be linear. This is a good approximation in most practical applications. conv_offset is the offset of the linear approximation.

2.2.4.3 bool Sensor::display_me

A bollean used to determine whether or not a sensor should be displayed at any given moment. Sensor values are not always on displayed in order to avoid information fatigue.

2.2.4.4 float Sensor::hi_bound

This value specifies the highest value (in the sensor's units, not volts) the hardware sensor is capable of reaching.

2.2.4.5 float Sensor::hi_thresh

A sensor value (in the sensor's units, not volts) above this will set display_me to *True*. If the sensor value in between hi_thres and lo_thresh, display_me wil lbe set to *False*.

2.2.4.6 unsigned short int Sensor::last_display_value_x

Tracks x-coordinate of display_value on HUD. Used by front end to clear OLED.

2.2.4.7 unsigned short int Sensor::last_display_value_y

Tracks y-coordinate of display_value on HUD. Used by front end to clear OLED.

2.2.4.8 unsigned short int Sensor::last_display_x

Tracks x-coordinate of the senor's name on HUD. Used by front end to clear OLED.

2.2.4.9 unsigned short int Sensor::last_display_y

Tracks y-coordinate of the senor's name on HUD. Used by front end to clear OLED.

2.2.4.10 float Sensor::lo_bound

This value specifies the lowest value (in the sensor's units, not volts) the hardware sensor is capable of reaching.

2.2.4.11 float Sensor::lo_thresh

A sensor value (in the sensor's units, not volts) below this will set display_me to *True*. If the sensor value in between hi_thres and lo_thresh, display_me wil lbe set to *False*.

2.2.4.12 unsigned short int Sensor::pin

A numeric value that represents the pin on the microcontroller which the sensor is connected too. This value will be defined by microcontroller manufacturer.

2.2.4.13 float Sensor::priority

Each sensor is dynamically assigned a priority score for decissions regaurding when and how to display information.

2.2.4.14 float Sensor::priority_offset

Set to zero by default, this property is used to give a sensor an *intrinsic* priorrity. All sensors are genearlly assumed to have the same *intrinsic* priority, and so this is set to zero by default. However, this property can used to "boost" or "subdue" a sensor's priority relative to others.

The documentation for this class was generated from the following files:

- /home/bryan/GIT_projects/Astro-HUD/libraries/Sensor/Sensor.h
- /home/bryan/GIT_projects/Astro-HUD/libraries/Sensor/Sensor.cpp