Tiny Tach

Generated by Doxygen 1.8.15

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 bubble_t Struct Reference

`#include <drvr_bubble_display.h>`

Public Attributes

- uint8_t number
- uint8_t location
- uint8_t decimal

3.1.1 Detailed Description

A structure representing a digit to print on the bubble display.

3.1.2 Member Data Documentation

3.1.2.1 decimal

`uint8_t bubble_t::decimal`

Set to 1 to display the decimal point. Set to 0 to show no decimal point.

3.1.2.2 location

`uint8_t bubble_t::location`

The location of the digit. 0 is leftmost, and 3 is rightmost.

3.1.2.3 number

`uint8_t bubble_t::number`

The number to print. 0::9

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

- src/drvr_bubble_display.h

3.2 task_t Struct Reference

Public Attributes

- uint16_t **delta**
- uint32_t **prev_tick**
- void(∗ **func**)(void)

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

- src/app_scheduler.h

Chapter 4

File Documentation

4.1 src/app_scheduler.h File Reference

Classes

- struct task_t

Functions

- void App_Scheduler_Run_Tasks (task_t *tasks)
- void App_Scheduler_Bump_Sys_Tick (void)
- uint32_t App_Scheduler_Get_Sys_Tick (void)

4.1.1 Detailed Description

This is a simple tick time scheduler.

4.1.2 Function Documentation

4.1.2.1 App_Scheduler_Bump_Sys_Tick()

```
void App_Scheduler_Bump_Sys_Tick (
            void  )
```

Bump the sys tick timer. Basically, just put this function in a periodic ISR function.

4.1.2.2 App_Scheduler_Get_Sys_Tick()

```
uint32_t App_Scheduler_Get_Sys_Tick (
            void  )
```

Get the current value of the sys_tick counter.

Since a 32-bit variable reads are non-atomic, the static variable is read twice to make it isn't being modified during a read.

Returns

the 32-bit sys_tick counter value.

4.1.2.3 App_Scheduler_Run_Tasks()

```
void App_Scheduler_Run_Tasks (
            task_t * tasks )
```

Run the scheduler task array. The task array should have a NULL pointer reference for the function in the last entry. This is how the scheduler knows it has reached the end of the task list.

Parameters

in	*const*	task_t *tasks is a pointer to an array of tasks to run.

4.2 src/drvr_bubble_display.h File Reference

Classes

- struct bubble_t

Functions

- void Drvr_Bubble_Display_Init (void)
- void Drvr_Bubble_Display_Print (bubble_t *digit, uint8_t location)
- void Drvr_Bubble_Display_Shutdown (void)

4.2.1 Detailed Description

Tiny Tach uses a single 4 character bubble display, the QDSP 6064. The Display is driven thru a shift register for the segments and GPIO for the cathodes. The display is multiplexed @ ~256Hz The bubble display has 4 cathodes, one for each digit. The cathodes are connected to the uC pins and configured to sink current. Set the the corresponding uC pin LOW to select a cathode to sink current and display data. Set the uC pin HIGH to deselect a cathode.

4.2.2 Function Documentation

4.2.2.1 Drvr_Bubble_Display_Init()

```
void Drvr_Bubble_Display_Init (
            void  )
```

Initialize the Bubble Display driver. This shoud be called from main() prior to using this module.

4.2.2.2 Drvr_Bubble_Display_Print()

```
void Drvr_Bubble_Display_Print (
            bubble_t * digit,
            uint8_t location )
```

Sends a number to the bubble display at the location specified. This function should be called at a regular periodic interval to support multiplexing. of the display.

Parameters

in	*number*	is the numeral to display 0::9.
in	*location*	is the position. 0 is the leftmost character and 3 is the rightmost. i.e. "thousands" and "ones" respectively.
in	*do_decimal*	is set to 1 to display the decimal point on the particular digit. Set to 0 to not display the decimal point.

4.2.2.3 Drvr_Bubble_Display_Shutdown()

```
void Drvr_Bubble_Display_Shutdown (
            void  )
```

Shifts out an empty byte to set the shift reg pins low.

4.3 src/drvr_gpio.h File Reference

Functions

- void Drvr_GPIO_Init (void)
- void Drvr_GPIO_Led_Toggle (void)
- void Drvr_GPIO_Led_Off (void)
- void Drvr_GPIO_Led_On (void)
- void Drvr_Retransmit_Toggle (void)
- uint8_t Drvr_GPIO_Switch_Is_Pressed (void)

4.3.1 Detailed Description

This driver is intended to interface port pins for GPIO. Only basic GPIO such as an LED is handled here.

4.3.2 Function Documentation

4.3.2.1 Drvr_GPIO_Init()

```
void Drvr_GPIO_Init (
             void  )
```

Initialize the GPIO driver. This shoud be called from main() prior to using this module.

4.3.2.2 Drvr_GPIO_Led_Off()

```
void Drvr_GPIO_Led_Off (
             void  )
```

Turn OFF the green board LED. Calling this function when the LED is OFF already will have no effect.

4.3.2.3 Drvr_GPIO_Led_On()

```
void Drvr_GPIO_Led_On (
             void  )
```

Turn ON the green board LED. Calling this function when the LED is ON already will have no effect.

4.3.2.4 Drvr_GPIO_Led_Toggle()

```
void Drvr_GPIO_Led_Toggle (
             void  )
```

Toggle the green board LED. Calling this function will turn ON the LED if it is OFF and vice versa.

4.3.2.5 Drvr_GPIO_Switch_Is_Pressed()

```
uint8_t Drvr_GPIO_Switch_Is_Pressed (
             void  )
```

Function to interface the tactile switch onboard. Includes a blocking debounce.

Returns

1 if the switch is pressed, debounced, and released. Otherwise, return 0.

4.3.2.6 Drvr_Retransmit_Toggle()

```
void Drvr_Retransmit_Toggle (
            void  )
```

Toggle the retransmit pin. Calling this function toggles the retransmit putput pin at half of the input frequency. This can be useful if the input pulse is very narrow such as with an encoder index.

4.4 src/drvr_serial.h File Reference

Functions

- void Drvr_Serial_Init (void)
- void Drvr_Serial_Print_String (const char ∗str)

4.4.1 Detailed Description

This driver is for UART Serial. Only TX is configured right now.

4.4.2 Function Documentation

4.4.2.1 Drvr_Serial_Init()

```
void Drvr_Serial_Init (
            void  )
```

Initialize the Serial driver. This shoud be called from main() prior to using this module.

4.4.2.2 Drvr_Serial_Print_String()

```
void Drvr_Serial_Print_String (
            const char * str )
```

Prints a null terminated string to the serial uart

Parameters

in	*const*	char ∗str is a pointer to the string.

4.5 src/drvr_tach.h File Reference

Macros

- #define **CAPTURE_RESULT_READY** 2

Functions

- void Drvr_Tach_Init (void)
- void Drvr_Tach_Reset (void)
- void Drvr_Tach_Rexmit_Off (void)
- void Drvr_Tach_Rexmit_On (void)
- void Drvr_Tach_Sensor_Disable (void)
- void Drvr_Tach_Sensor_Enable (void)
- uint32_t Drvr_Tach_Get_Clk_Cyc (void)
- void Drvr_Tach_Rearm_Input_Capture (void)
- uint8_t Drvr_Tach_Get_Capture_State (void)

4.5.1 Detailed Description

This module contains ISR config, GPIO setup, and calculations related to the tachometer function.

4.5.2 Function Documentation

4.5.2.1 Drvr_Tach_Get_Capture_State()

```
uint8_t Drvr_Tach_Get_Capture_State (
            void  )
```

Getter for the capture state.

Returns

capture_state 0 -> no capture events registered. 1 -> first capture event complete. 2 -> second capture event complete. - ready for calculation.

4.5.2.2 Drvr_Tach_Get_Clk_Cyc()

```
uint32_t Drvr_Tach_Get_Clk_Cyc (
            void  )
```

Get the total clock cycles.

Since a multi-byte variable reads are non-atomic, the static variable is read twice to make it isn't being modified during a read.

Returns

the number of clock cycles between input capture events.

4.5.2.3 Drvr_Tach_Init()

```
void Drvr_Tach_Init (
            void  )
```

Initialize the tach driver. This shoud be called from main() prior to using this module.

4.5.2.4 Drvr_Tach_Rearm_Input_Capture()

```
void Drvr_Tach_Rearm_Input_Capture (
            void  )
```

Rearm the input capture system. This prepares the input capture system for the next input pulse.

4.5.2.5 Drvr_Tach_Reset()

```
void Drvr_Tach_Reset (
            void  )
```

Reinitialize tach counters to zero.

4.5.2.6 Drvr_Tach_Rexmit_Off()

```
void Drvr_Tach_Rexmit_Off (
            void  )
```

Turn OFF the retransmit pin.

4.5.2.7 Drvr_Tach_Rexmit_On()

```
void Drvr_Tach_Rexmit_On (
            void  )
```

Turn ON the retransmit pin

4.5.2.8 Drvr_Tach_Sensor_Disable()

```
void Drvr_Tach_Sensor_Disable (
            void  )
```

Turn OFF the IR emitter pin.

4.5.2.9 Drvr_Tach_Sensor_Enable()

```
void Drvr_Tach_Sensor_Enable (
            void  )
```

Turn ON the IR emitter pin

4.6 src/drvr_watchdog.h File Reference

Functions

- void Drvr_Watchdog_Init (void)
- void Drvr_Watchdog_Off (void)
- void Drvr_Watchdog_Pet (void)

4.6.1 Detailed Description

A driver for the watchdog timer.

4.6.2 Function Documentation

4.6.2.1 Drvr_Watchdog_Init()

```
void Drvr_Watchdog_Init (
             void  )
```

Initialize the watchdog timer. This shoud be called from main() prior to using this module.

4.6.2.2 Drvr_Watchdog_Off()

```
void Drvr_Watchdog_Off (
             void  )
```

Disable the watchdog timer.

4.6.2.3 Drvr_Watchdog_Pet()

```
void Drvr_Watchdog_Pet (
             void  )
```

Service the watchdog timer system to show we are alive still.