Code-Control LED Array

Core 64: Interactive Core Memory Badge, V0.1, Andrew Geppert, 2020-01-11

User Application

Use the cores to do something interesting

FreeRTOS

Facilitate running all task threads

**LED\_ARRAY\_HAL.cpp**

was LED\_Array.cpp

For interaction with two abstract memory spaces (1D [64 pixel string] and 2D [8x8 matrix]) which define the LED Array for user to view. Four modes: 1D monochrome, 2D monochrome, 1D color, 2D color.

Enables user application to interact with any abstract LED Array (as a memory buffer) in order to set each real LED as desired.

Interaction is done in one of layouts: Visual (x,y)

logical (bits, bytes, word)

And one of two color modes:

Monochrome

RGB

These functions ask the driver (in this case the FastLED library) to do the detailed work.

**LED\_ARRAY\_HAL.h (with new API)**

LED\_ Array \_Init (create an instance)

LED\_ Array \_Screen\_RAM\_Clear (1D, 2D, or both)

LED\_ Array \_ Monochrome\_Set\_Color (color)

LED\_Array\_

LED\_Array\_Update\_From\_Mono\_RAM ()

LED\_Array\_Update\_From\_Color\_RAM ()

maybe the LED memory buffers.

HAL: Interact with the hardware in an abstract and conceptual way, using high level perspective. Such as visual and logical arrangements.

**FastLED.h**

src/FastLED Library

**Sub-Functions**

Bit Banged SPI

**FastLED\_Config.h**

Define my custom LED array params for the FastLED library.

DRIVER: Control the signals to the hardware and the hardware itself. Control the pins of the microcontroller

**FastLed.c**

Controls the electronics through #include HardwareIOmap.h, Arduino.h. Translates active/inactive to low/high for the IO signals.

BSP: Abstraction of MCU pin reference number to the physical pin on the chip

**Arduino.c and .cpp in the Arduino IDE**

Abstraction of MCU pin to integer pin reference number

**Arduino.h**

HAL

LED\_ARRAY\_HAL.h (defines the API)

LED\_ARRAY\_HAL.cpp (was LED\_Array.cpp).

The application accesses the driver by calling API functions in the driver library.

Set each LEDs on/off or color state in a memory buffers.

Read each LEDs state from the memory buffers.

Clear the memory buffers.

Request a memory buffer to be displayed on the LEDs.

Test functions: cycle through all LEDs in order of 1D and 2D arrangements to confirm expected order of the LEDs.

The driver contains the LED Array memory buffer.

Monochrome 2D 8x8 image as viewed by the user.

Monochrome 1D 64 bit word, with LSB on lower right, MSB on left. First row is low byte of 64 bit word, top row is highest byte.

Color 2D 8x8 image as viewed by the user.

DRIVER

In this case, the FastLED-3.3.2 library in the LIBRARIES folder. Nothing I have to write custom.

This is implemented to talk to the hardware and has many of its own layers and BSPs.

BSP

Combination of files buried in the FastLED Library, and Arduino.