Code-Control LED Array

Core 64: Interactive Core Memory Badge, V0.1, Andrew Geppert, 2020-01-23

User Application

Use the cores to do something interesting

FreeRTOS

Facilitate running all task threads

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

LED\_Array\_Init (create an instance)

LED\_Array\_Memory\_Clear (all buffers)

LED\_Array\_Monochrome\_Set\_Color (color)

LED\_Array\_Test\_Count\_Binary

LED\_Array\_Test\_Pixel\_String

LED\_Array\_Test\_Pixel\_Matrix

LED\_Array\_Display\_Binary ()

LED\_Array\_Write\_Binary()

TODO:

Convert old write/display to new API

Cleanout unused functions

**LED\_Array\_HAL.cpp**

Interaction with the abstract memory buffers which define the LED Array for the user to view:

BINARY [64 bit data word, monochrome]

STRING [1D 64 pixel string, monochrome]

MATRIX\_MONO [2D 8x8 pixel matrix, monochrome]

MATRIX\_COLOR [2D 8x8 pixel matrix, color]

When the buffer is requested for display on the LED Array, appropriate translations are applied for correct orientation of the view to the user.

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, 1D string, 2D matrix (x,y)

Binary, logical (bits, bytes, word)

And one of two color modes:

Monochrome

HSV

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

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.

**DEFINITIONS**

**THREE MONOCHROME SCREEN MEMORY BUFFERS**

**LED SCREEN MEMORY MONOCHROME 2D IMAGE: The LED array as viewed by the user with badge in hand, or looking at C source code array definitions, for the case of symbols drawn on the screen.**

**Was LEDArrayMonochromeMemory [y][x] : Monochrome version of the LED Screen RAM. Easy to visualize screen memory in the form of C Syntax array. Just make sure to use the proper y, x order because of the C syntax. Upper left is 0,0 and lower right is 7,7.**

**LED SCREEN MEMORY MONOCHROME 1D PIXEL STRING: A one-dimensional array addressable 0 to 63. Upper left is 0, lower right is 63. Ordered from left to right, then top to bottom.**

**LED SCREEN MEMORY MONOCHROME 1D DATA: The LED array as viewed by the user, representing 64 bits of data as 8 rows of 8 bits (or 8 rows of bytes) with the least significant bit on the right, and the least significant byte on the bottom. In other words, the whole data value is 64 bits and bit 0 is in the lower right, and bit 63 in the upper left. This is opposite of the LED SCREEN MEMORY MONOCHROME 1D STRING.**

**DISPLAY ONE OF THE MONOCHROME SCREEN MEMORY BUFFERS USING CORRESPONDING FUNCTIONS**

**DISPLAY LED SCREEN MONOCHROME 2D IMAGE: Copies the associated screen memory into the appropriate position of the leds[63] array and sends it to the LED array.**

**DISPLAY LED SCREEN MONOCHROME 1D PIXEL STRING: Copies the associated screen memory into the appropriate position of the leds[63] array and sends it to the LED array.**

**DISPLAY LED SCREEN MONOCHROME 1D DATA: Copies the associated screen memory into the appropriate position of the leds[63] array and sends it to the LED array.**

**LED SCREEN ARRAY FASTLED: A virtual two-dimensional array addressable as in the form of y, x with a range of 0 to 7 for both values. This is part of the FastLED example library (/Examples/FastLED/XYMAtrix), and this arrangement is converted to leds[63] with awareness of the serpentine electrical arrangement of the string of LEDs using the functions XY() or YX().**

**LED ARRAY PIXEL: A single pixel**

**leds[63]: The one-dimensional array that holds color values for the FastLED library to send to the string of LEDs. The lowest LED Array data level that is easily user accessible. This arrangement follows the serpentine electrical arrangement of the LEDs on the Pimoroni Unicorn Hat.**

**VisualLED2DMatrix, User LED Matrix [x,y] : The orientation of the LED array for the user.**

**\*\*\* To convert Array to Matrix use function YX(y,x). \*\*\***

**CSyntaxCORELEDArray, C Syntax LED Array [y,x] : The orientation of the array which aligns with the C language array definition. Note the y position (row) is written first, followed by x position (column). The core matrix wires are labeled this way.**

**CSyntaxBINARYArray, Binary LED Array [y,x] : The orientation of the array which aligns with LSB in the lower right.**

**WriteOneBitToMonochromeLEDArrayMemory(uint8\_t bit, bool value) : Update a single bit of LED screen Memory. Bit position is defined**

**LEDArrayMonochromeUpdate() : Read 64 bits of monochrome LED Screen memory and update the LED Array.**

**FastLED2DMatrix, FastLED (Unicorn Hat) LED Matrix [x,y] : The ordering of the LEDs defined in the FastLED library.**