k60 Control - libsc

Author: Peter Tse (mcreng)

This acts as a documentation of libsc library.

Battery Meter

Location: `/libsc/battery_meter.h`

Config	Data Type	Description
`voltage_ratio`	`float`	Voltage ratio of voltage divider

Function	Description
`float GetVoltage()`	Get current battery voltage

Bluetooth

Location: `/libsc/k60/jy_mcu_bt_106.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of Bluetooth
`baud_rate`	`BaudRate`	Baud rate of Bluetooth
`rx_isr`	`bool(const Byte*, const size_t)`	Interrupt handler of Bluetooth, incoming data in `const Byte*`.

Function	Description
`bool SendStr(const char*)`	Send string with `char*` type via Bluetooth
`bool SendBuffer(const Byte*, const size_t)`	Send buffer with `Byte*` type and size of buffer with `size_t` via Bluetooth

Button

Location: `/libsc/button.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of button
`is_active_low`	`bool`	Whether the button is active low
`listener`	`void(const uint8_t)`	Interrupt handler on button press, ID of button pressed as parameter
`listener_trigger`	`Trigger`	Set the listener trigger on either rising edge, falling edge or both

Function	Description
`bool IsDown()`	return whether the button is down

Camera

Location: `/libsc/k60/ov7725.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of camera
`w`	`Uint`	Width of image
`h`	`Uint`	Height of image
`fps`	`Fps`	FPS of image
`brightness`	`uint8_t`	Brightness of image
`contrast`	`uint8_t`	Contrast of image

Function	Description
`void Start()`	Start capturing image
`void Stop()`	Stop capturing image
`bool IsAvailable()`	Return whether an image is ready
`const Byte* LockBuffer()`	Return and locks the buffer. Buffer is stored with 8 pixel per byte
`void UnlockBuffer()`	Unlock buffer

Sample code:

```
int main() {
  Ov7725::Config config;
  config.id = 0;
  config.w = 640;
  config.h = 480;
  config.fps = Ov7725Configurator::Config::Fps::kHigh;
  Ov7725 camera(config);

  camera.Start(); // start camera
  const Byte* buf = camera.LockBuffer(); // buf stores image
  camera.UnlockBuffer();
  /* Handles buf */
}
```

Encoder

Location: `/libsc/ab_encoder.h` (AB encoder), `/libsc/dir_encoder.h` (Direction encoder)

Config	Data Type	Description
`id`	`uint8_t`	ID of encoder

Function	Description
`void Update()`	Reset and update encoder count
`int32_t GetCount()`	Get current encoder count

Joystick

Location: `/libsc/joystick.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of joystick
`is_active_low`	`bool`	Whether joystick is active low
`dispatcher`	`void(const uint8_t, const State)`	Interrupt handler for all state of joystick, ID as first parameter, State as second
`listen_triggers[5]`	`Trigger`	The trigger method of all 5 states of joystick

Function	Description
`State GetState()`	Get joystick state

LCD

Location: `/libsc/st7735r.h`

Config	Data Type	Description
`is_revert`	`bool`	Whether to revert the LCD
`is_bgr`	`bool`	Whether using a BGR panel instead of RGB

Function	Description
`void SetRegion(const Rect)`	Select a region in LCD
`void ClearRegion()`	Clear region selection
`void FillColor(const uint16_t)`	Fill the color in the selected region
`void FillGrayscalePixel(const uint8_t*, const size_t)`	Fill an array of grayscale pixel sequentially to the selected region
`void Pixel(const uint16_t*, const size_t)`	Fill an array of coloured pixel sequentially to the selected region
`void FillBits(const uint16_t, const uint16_t, const bool*, const size_t)`	Fill first colour/second colour according to the boolean value of the array, sequentially.
`void FillBits(const uint16_t, const uint16_t, const Byte*, const size_t)`	Fill first colour/second colour according to the bit value of the byte array, sequentially.
`void Clear()`	Clear the screen

LED

Location: `/libsc/led.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of LED
`is_active_low`	`bool`	Whether LED is active low

Function	Description
`void SetEnable(const bool)`	Set LED on/off
`void Switch()`	Toggle the LED

Motor

Location: `/libsc/alternate_motor.h` (Alternate motor); `/libsc/dir_motor.h` (Direction motor)

Config	Data Type	Description
`id`	`uint8_t`	ID of motor

Function	Description
`void SetPower(const uint16_t)`	Set power to motor
`void AddPower(const int16_t)`	Add power to motor, value can be negative
`uint16_t GetPower()`	Return power (not actual power, but the power specified)

MPU

Location: `/libsc/mpu6050.h`

Config	Data Type	Description
`gyro_range`	`Range`	Range of gyroscope
`accel_range`	`Range`	Range of accelerometer
`cal_drift`	`bool`	Whether to calibrate the gyroscope during initialization
`i2c_master_ptr`	`I2cMaster*`	Pointer to I2C pins

Function	Description
`bool Update(bool)`	Update the MPU with integer with/without clamping of values
`bool UpdateF(bool)`	Update the MPU with float with/without clamping of values

The rest are getter functions of

- accelerometer
- gyroscope
- thermometer
- offset

Servo

Location: `/libsc/futaba_s3010.h`

Config	Data Type	Description
`id`	`uint8_t`	ID of servo

Function	Description
`void SetDegree(const uint16_t)`	Set servo degree
`uint16_t GetDegree()`	Return degree (not actual degree, but the degree specified)