Lepton Main Driver Module

Table of Contents

1. Overview

The `lepton` module is the core driver interface for FLIR Lepton thermal imaging sensors. It provides high-level functions for device initialization, configuration, and control.

1.1. Key Features

- Device initialization and deinitialization
- Video format configuration
- Telemetry control
- Temperature reading (FPA and AUX)
- AGC (Automatic Gain Control) configuration
- ROI (Region of Interest) management for multiple features
- Radiometric measurements (Lepton 3.5)
- Frame capture control
- Color palette conversion

1.2. Supported Sensors

The driver automatically detects and supports:

- **Lepton 3.0** (500-0726-01) - Non-radiometric
- **Lepton 3.5** (500-0771-01) - Radiometric with TLinear

2. Initialization

2.1. Basic Initialization

```
#include "lepton.h"

Lepton_t device;
Lepton_Result_t status;

Lepton_Conf_t config = {
    .CCI = {
        .I2C_Read = my_i2c_read_function,
        .I2C_Write = my_i2c_write_function,
        .Address = LEPTON_I2C_ADDRESS
    },
    .VoSPI = {
        .Host = SPI2_HOST,
        .Interface = {
            .clock_speed_hz = 20000000,  // 20 MHz
            .spics_io_num = GPIO_NUM_10
        },
        .Master = {
            .sclk_io_num = GPIO_NUM_12,
            .miso_io_num = GPIO_NUM_13
        },
        .DMA = SPI_DMA_CH_AUTO
    },
    .VSync = GPIO_NUM_5,
    .Reset = &my_reset_function,
    .PowerDown = &my_powerdown_function
};

if (Lepton_Init(&device, &config, &status) != LEPTON_ERR_OK) {
    ESP_LOGE(TAG, "Failed to initialize Lepton");
}
```

2.2. Configuration Structure

The `Lepton_Conf_t` structure contains:

- **CCI**: I2C communication callbacks and address
- **VoSPI**: SPI configuration (host, pins, clock, DMA)
- **VSync**: GPIO pin for vertical sync interrupt (optional)
- **Reset**: Hardware reset callback (optional)
- **PowerDown**: Power-down control callback (optional)

3. API Functions

3.1. Device Control

3.1.1. Lepton_Init

```
Lepton_Error_t Lepton_Init(
    Lepton_t *p_Device,
    const Lepton_Conf_t *const p_Init,
    Lepton_Result_t *p_Status
);
```

Initialize the Lepton sensor.

Parameters:

- `p_Device`: Pointer to device instance
- `p_Init`: Pointer to configuration structure
- `p_Status`: Optional pointer to receive status information

Returns: `LEPTON_ERR_OK` on success

Initialization Sequence:

1. Hardware reset (if callback provided)
2. Power-up (if callback provided)
3. Wait 950ms for boot
4. Initialize CCI interface
5. Detect sensor model and capabilities
6. Configure default settings

3.1.2. Lepton_Deinit

```
void Lepton_Deinit(Lepton_t *p_Device);
```

Deinitialize the sensor and free resources.

3.1.3. Lepton_HardReset

```
void Lepton_HardReset(Lepton_t *p_Device);
```

Perform hardware reset via external GPIO.

This function is weak and must be implemented by the application.

3.1.4. Lepton_EnablePowerDown

```
void Lepton_EnablePowerDown(Lepton_t *p_Device, bool Enable);
```

Control power-down mode via external GPIO.

This function is weak and must be implemented by the application.

3.2. Video Configuration

3.2.1. Lepton_SetVideoFormat

```
Lepton_Error_t Lepton_SetVideoFormat(
    Lepton_t *p_Device,
    Lepton_VideoFormat_t Format,
    Lepton_Result_t *p_Status
);
```

Set the video output format.

Formats:

- `LEPTON_FORMAT_RAW14`: 14-bit raw thermal data
- `LEPTON_FORMAT_RGB888`: 8-bit RGB (AGC applied)

Changing format triggers VoSPI resynchronization.

3.2.2. Lepton_GetVideoFormat

```
Lepton_Error_t Lepton_GetVideoFormat(
    Lepton_t *p_Device,
    Lepton_VideoFormat_t *p_Format
);
```

Get the current video format.

3.2.3. Lepton_SetVideoSource

```
Lepton_Error_t Lepton_SetVideoSource(
    Lepton_t *p_Device,
    Lepton_VideoSource_t Source,
```

```
    uint16_t Constant,
    Lepton_Result_t *p_Status
);
```

Set the video source for testing and calibration.

Sources:

- `LEPTON_SOURCE_RAW`: Normal thermal imaging
- `LEPTON_SOURCE_COOKED`: Processed thermal data
- `LEPTON_SOURCE_CONSTANT`: Fixed value for testing
- `LEPTON_SOURCE_RAMP`: Ramp pattern for testing

3.2.4. Lepton_FreezeVideo

```
Lepton_Error_t Lepton_FreezeVideo(
    Lepton_t *p_Device,
    bool Freeze,
    Lepton_Result_t *p_Status
);
```

Freeze/unfreeze video output.

3.3. Telemetry

3.3.1. Lepton_EnableTelemetry

```
Lepton_Error_t Lepton_EnableTelemetry(
    Lepton_t *p_Device,
    bool Enable,
    Lepton_Result_t *p_Status
);
```

Enable/disable telemetry data in video stream. Telemetry includes temperature, status, and metadata.

3.4. Temperature Measurement

3.4.1. Lepton_GetTemperature

```
Lepton_Error_t Lepton_GetTemperature(
    Lepton_t *p_Device,
    uint16_t *p_FPA,
    uint16_t *p_AUX,
```

```
    Lepton_Result_t *p_Status
);
```

Read sensor temperatures.

Returns:

- `p_FPA`: Focal Plane Array temperature (centi-Kelvin)
- `p_AUX`: Auxiliary temperature (centi-Kelvin)

Conversion:

```
float celsius = Lepton_KelvinToCelsius(fpa_temp);
```

3.4.2. Lepton_GetPixelTemperature

```
Lepton_Error_t Lepton_GetPixelTemperature(
    Lepton_t *p_Device,
    uint16_t PixelValue,
    float *p_Temperature
);
```

Convert a raw pixel value to temperature in Celsius.

Only available on radiometric Lepton 3.5 with TLinear enabled.

3.5. AGC Configuration

3.5.1. Lepton_EnableAGC

```
Lepton_Error_t Lepton_EnableAGC(
    Lepton_t *p_Device,
    bool Enable,
    Lepton_Result_t *p_Status
);
```

Enable/disable Automatic Gain Control.

- **Enabled**: 8-bit contrast-enhanced output
- **Disabled**: 14-bit raw radiometric data

3.6. ROI Management

The driver supports multiple Region of Interest (ROI) types:

- **Spotmeter ROI**: Single-point temperature measurement
- **Scene ROI**: Scene statistics calculation area
- **AGC ROI**: Area for AGC algorithm
- **Video Focus ROI**: Focus calculation area

3.6.1. Lepton_SetSpotmeterROI / Lepton_GetSpotmeterROI

```
Lepton_Error_t Lepton_SetSpotmeterROI(
    Lepton_t *p_Device,
    Lepton_ROI_t *p_ROI,
    Lepton_Result_t *p_Status
);
```

Configure spotmeter region.

Example:

```
Lepton_ROI_t roi = {
    .Start = {.X = 70, .Y = 50},   // Center of 160x120 image
    .End = {.X = 90, .Y = 70}
};
Lepton_SetSpotmeterROI(&device, &roi, &status);
```

3.6.2. Lepton_GetSpotmeter

```
Lepton_Error_t Lepton_GetSpotmeter(
    Lepton_t *p_Device,
    Lepton_Spotmeter_t *p_Spot,
    Lepton_Result_t *p_Status
);
```

Read spotmeter values (temperature, min, max, population).

3.6.3. Scene ROI Functions

- `Lepton_SetSceneROI()` / `Lepton_GetSceneROI()`
- `Lepton_SetAGCROI()` / `Lepton_GetAGCROI()`
- `Lepton_SetVideoFocusROI()` / `Lepton_GetVideoFocusROI()`

3.7. Scene Statistics

3.7.1. Lepton_GetSceneStatistics

```
Lepton_Error_t Lepton_GetSceneStatistics(
    Lepton_t *p_Device,
    Lepton_SceneStatistics_t *p_Statistics,
    Lepton_Result_t *p_Status
);
```

Get scene statistics (min, max, average temperature).

3.8. Radiometric Functions (Lepton 3.5 only)

3.8.1. Lepton_SetEmissivity

```
Lepton_Error_t Lepton_SetEmissivity(
    Lepton_t *p_Device,
    Lepton_Emissivity_t Emissivity,
    Lepton_Result_t *p_Status
);
```

Set material emissivity for accurate temperature measurement.

Common Emissivity Values:

- Human skin: 0.98
- Water: 0.95
- Concrete: 0.92
- Steel (polished): 0.30

3.8.2. Lepton_SetFluxLinearParameters / Lepton_GetFluxLinearParameters

```
Lepton_Error_t Lepton_SetFluxLinearParameters(
    Lepton_t *p_Device,
    Lepton_FluxLinearParams_t *p_Parameters,
    Lepton_Result_t *p_Status
);
```

Configure flux linear calibration parameters.

3.8.3. Lepton_SetTLinearResolution / Lepton_GetTLinearResolution

```
Lepton_Error_t Lepton_SetTLinearResolution(
    Lepton_t *p_Device,
    Lepton_TLinear_Resolution_t Resolution,
```

```
    Lepton_Result_t *p_Status
);
```

Set TLinear temperature resolution mode.

3.9. Frame Capture

3.9.1. Lepton_StartCapture

```
Lepton_Error_t Lepton_StartCapture(
    Lepton_t *p_Device,
    QueueHandle_t p_Queue
);
```

Start frame capture task.

Parameters:

- `p_Queue`: FreeRTOS queue to receive frame buffers

Queue Item Type:

```
Lepton_FrameBuffer_t frame;
if (xQueueReceive(queue, &frame, portMAX_DELAY)) {
    // Process frame.p_Buffer (160x120 pixels)
    // Release when done
}
```

3.9.2. Lepton_StopCapture

```
Lepton_Error_t Lepton_StopCapture(Lepton_t *p_Device);
```

Stop frame capture and free resources.

3.10. Image Processing

3.10.1. Lepton_Raw14ToRGB

```
bool Lepton_Raw14ToRGB(
    Lepton_t *p_Device,
    uint16_t *p_Input,
    uint8_t *p_Output,
    int16_t *p_Min,
    int16_t *p_Max,
```

```
    uint16_t Width,
    uint16_t Height
);
```

Convert 14-bit raw thermal data to RGB using iron palette.

Color Mapping:

- Blue → Cyan → Green → Yellow → Red
- Blue = coldest, Red = hottest

Usage:

```
uint16_t raw[160*120];
uint8_t rgb[160*120*3];
int16_t min, max;

Lepton_Raw14ToRGB(&device, raw, rgb, &min, &max, 160, 120);
ESP_LOGI(TAG, "Temperature range: %d to %d", min, max);
```

3.11. Utility Functions

3.11.1. Lepton_LibVersion

```
std::string Lepton_LibVersion(void);
```

Get library version string (e.g., "1.0.0").

3.11.2. Lepton_GetUptime

```
uint32_t Lepton_GetUptime(
    Lepton_t *p_Device,
    Lepton_Result_t *p_Status
);
```

Get sensor uptime in milliseconds.

3.11.3. Lepton_isCapturing

```
bool Lepton_isCapturing(Lepton_t *p_Device);
```

Check if capture is active.

3.11.4. Lepton_GetSyncErrors

```
int32_t Lepton_GetSyncErrors(Lepton_t *p_Device);
```

Get VoSPI synchronization error count.

3.11.5. Lepton_GetFrameCounter

```
int32_t Lepton_GetFrameCounter(Lepton_t *p_Device);
```

Get number of successfully captured frames.

3.11.6. Lepton_KelvinToCelsius

```
float Lepton_KelvinToCelsius(uint32_t Kelvin);
```

Convert centi-Kelvin to Celsius.

Example:

```
uint16_t fpa_temp;
Lepton_GetTemperature(&device, &fpa_temp, NULL, NULL);
float celsius = Lepton_KelvinToCelsius(fpa_temp);
```

4. Error Handling

All functions return `Lepton_Error_t`:

- `LEPTON_ERR_OK`: Success
- `LEPTON_ERR_INVALID_ARG`: Invalid parameter
- `LEPTON_ERR_TIMEOUT`: Communication timeout
- `LEPTON_ERR_NOT_INITIALIZED`: Device not initialized
- `LEPTON_ERR_NOT_SUPPORTED`: Feature not supported
- `LEPTON_ERR_NO_MEM`: Memory allocation failed

The optional `Lepton_Result_t` provides detailed CCI status codes.

5. Usage Example

5.1. Complete Initialization and Capture

```
#include "lepton.h"

static Lepton_t lepton_device;
static QueueHandle_t frame_queue;

void app_main(void) {
    // Create frame queue
    frame_queue = xQueueCreate(3, sizeof(Lepton_FrameBuffer_t));

    // Configure Lepton
    Lepton_Conf_t config = {
        .CCI = {
            .I2C_Read = i2c_read_callback,
            .I2C_Write = i2c_write_callback,
            .Address = LEPTON_I2C_ADDRESS
        },
        .VoSPI = {
            .Host = SPI2_HOST,
            .Interface = {
                .clock_speed_hz = 20000000,
                .spics_io_num = GPIO_NUM_10
            },
            .Master = {
                .sclk_io_num = GPIO_NUM_12,
                .miso_io_num = GPIO_NUM_13
            },
            .DMA = SPI_DMA_CH_AUTO
        },
        .VSync = GPIO_NUM_5
    };

    // Initialize
    Lepton_Result_t status;
    if (Lepton_Init(&lepton_device, &config, &status) != LEPTON_ERR_OK) {
        ESP_LOGE(TAG, "Initialization failed");
        return;
    }

    // Configure for radiometric mode
    Lepton_EnableAGC(&lepton_device, false, &status);
    Lepton_SetEmissivity(&lepton_device, 95, &status);  // 0.95 for most surfaces

    // Start capture
    Lepton_StartCapture(&lepton_device, frame_queue);

    // Process frames
    Lepton_FrameBuffer_t frame;
    while (true) {
```

```
        if (xQueueReceive(frame_queue, &frame, portMAX_DELAY)) {
            // Convert to RGB
            uint8_t rgb[160*120*3];
            int16_t min, max;
            Lepton_Raw14ToRGB(&lepton_device, frame.p_Buffer,
                              rgb, &min, &max, 160, 120);

            ESP_LOGI(TAG, "Frame captured, temp range: %.1f to %.1f °C",
                     Lepton_KelvinToCelsius(min),
                     Lepton_KelvinToCelsius(max));
        }
    }
}
```

6. See Also

- CCI Commands - High-level CCI interface
- Frame Capture - Capture implementation details
- CCI Low-Level - Low-level I2C communication
- VoSPI - SPI video interface

7. License

Website: https://www.kampis-elektroecke.de