# Lepton Frame Capture Module

## Table of Contents


1. Overview . . . 2
   1.1. Key Features . . . 2
2. Architecture . . . 2
   2.1. Frame Buffer Flow . . . 2
3. Configuration . . . 2
   3.1. Task Configuration . . . 3
   3.2. VSync Configuration . . . 3
4. API Functions . . . 3
   4.1. Lepton_StartCapture . . . 3
   4.2. Lepton_StopCapture . . . 4
5. Frame Buffer Structure . . . 4
   5.1. Lepton_FrameBuffer_t . . . 4
6. Capture Task Implementation . . . 5
   6.1. Task Loop . . . 5
   6.2. Error Recovery . . . 5
   6.3. Sync Error Example . . . 5
7. VSync Interrupt . . . 6
   7.1. Purpose . . . 6
   7.2. ISR Implementation . . . 6
   7.3. Without VSync . . . 6
8. Frame Queue Management . . . 6
   8.1. Queue Behavior . . . 7
   8.2. Queue Size Recommendations . . . 7
9. Performance Optimization . . . 7
   9.1. Task Priority . . . 7
   9.2. Core Affinity . . . 7
   9.3. IRAM Placement . . . 8
10. Watchdog Integration . . . 8
11. Telemetry Support . . . 8
12. Usage Example . . . 9
   12.1. Complete Capture Setup . . . 9
   12.2. Monitoring Performance . . . 10
13. Troubleshooting . . . 10
   13.1. High Sync Errors . . . 10
   13.2. Missed Frames . . . 10
   13.3. Memory Issues . . . 11

14. See Also . . . . . 11
15. License . . . . . 11

# 1. Overview

The `lepton_capture` module implements the frame capture task and synchronization logic for the Lepton thermal camera. It handles continuous frame acquisition, error recovery, and frame buffer management.

## 1.1. Key Features

- FreeRTOS task-based frame capture
- Optional VSync interrupt synchronization
- Automatic sync error recovery
- Frame buffer queue management
- Telemetry data extraction
- Watchdog integration
- Configurable task parameters

# 2. Architecture

The capture system uses a dedicated FreeRTOS task that continuously:

1. Waits for VSync interrupt (optional)
2. Captures frames via VoSPI
3. Handles synchronization errors
4. Queues frame buffers for processing
5. Manages multiple frame buffers

## 2.1. Frame Buffer Flow

```
VSync IRQ → Capture Task → VoSPI Transfer → Frame Buffer → Queue → Application
```

# 3. Configuration

The module can be configured via Kconfig options:

## 3.1. Task Configuration

| Option | Description | Default |
|---|---|---|
| `CONFIG_LEPTON_CAPTURE_TASK_STACK` | Stack size in bytes | 4096 |
| `CONFIG_LEPTON_CAPTURE_TASK_PRIORITY` | FreeRTOS task priority | 16 |
| `CONFIG_LEPTON_CAPTURE_TASK_CORE` | CPU core affinity (0 or 1) | 1 |
| `CONFIG_LEPTON_CAPTURE_TASK_CORE_AFFINITY` | Enable core pinning | Defined |

## 3.2. VSync Configuration

| Option | Description | Default |
|---|---|---|
| `CONFIG_LEPTON_GPIO_USE_VSYNC` | Enable VSync interrupt | Enabled |
| `CONFIG_LEPTON_VSYNC_PLACE_IRAM` | Place ISR in IRAM for faster execution | Enabled |

# 4. API Functions

## 4.1. Lepton_StartCapture

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

Start the frame capture task.

**Parameters:**

- `p_Device`: Pointer to Lepton device instance
- `p_Queue`: FreeRTOS queue handle to receive frame buffers

**Returns:**

- `LEPTON_ERR_OK`: Success
- `LEPTON_ERR_INVALID_ARG`: Invalid parameters
- `LEPTON_ERR_NO_MEM`: Task or ISR creation failed

**Initialization Steps:**

1. Create capture task
2. Register VSync interrupt handler (if enabled)
3. Start VoSPI capture

4. Mark device as capturing

**Example:**

```
// Create queue for 3 frames
QueueHandle_t frame_queue = xQueueCreate(3, sizeof(Lepton_FrameBuffer_t));

if (Lepton_StartCapture(&device, frame_queue) != LEPTON_ERR_OK) {
    ESP_LOGE(TAG, "Failed to start capture");
}
```

## 4.2. Lepton_StopCapture

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

Stop frame capture and clean up resources.

**Cleanup Steps:**

1. Signal task to stop
2. Remove VSync interrupt handler
3. Stop VoSPI transfer
4. Delete capture task
5. Clear capturing flag

**Example:**

```
Lepton_StopCapture(&device);
vQueueDelete(frame_queue);
```

# 5. Frame Buffer Structure

## 5.1. Lepton_FrameBuffer_t

```
typedef struct {
    uint16_t *Image_Buffer;         // Pointer to thermal image data
    uint8_t *Telemetry_Buffer;      // Pointer to telemetry (or NULL)
    uint16_t Width;                 // Image width (160)
    uint16_t Height;                // Image height (120)
    uint8_t BytesPerPixel;          // Bytes per pixel (2 for RAW14)
} Lepton_FrameBuffer_t;
```

**Fields:**

- `Image_Buffer`: Thermal image data (160x120 pixels, 14-bit values)
- `Telemetry_Buffer`: Optional telemetry data (if enabled)
- `Width`: Image width (always 160)
- `Height`: Image height (120 for Lepton 3.x)
- `BytesPerPixel`: Always 2 for RAW14 format

# 6. Capture Task Implementation

## 6.1. Task Loop

The capture task runs continuously:

```
while (Device->Internal.VoSPI.isCapturing) {
    1. Wait for VSync interrupt (optional)
    2. Capture frame via VoSPI
    3. Handle errors:
       - LEPTON_ERR_OK: Queue frame
       - LEPTON_ERR_FAIL: Log sync error, retry
       - LEPTON_ERR_NOT_FINISHED: Resync in progress
    4. Reset watchdog timer
}
```

## 6.2. Error Recovery

The task implements intelligent error recovery:

- **Consecutive Errors**: Tracks sync errors in a row
- **Backoff**: Increases delay after errors to reduce CPU load
- **Logging**: Throttled error logging (every 10th consecutive error)
- **Automatic Resync**: VoSPI handles resynchronization

## 6.3. Sync Error Example

```
if (Error == LEPTON_ERR_FAIL) {
    Device->Internal.VoSPI.SyncErrors++;
    ConsecutiveErrors++;

    if (ConsecutiveErrors % 10 == 1) {
        ESP_LOGW(TAG, "Sync error (consecutive: %d, total: %u)",
                 ConsecutiveErrors, Device->Internal.VoSPI.SyncErrors);
    }
```

```
    vTaskDelay(10 / portTICK_PERIOD_MS);  // Backoff
}
```

# 7. VSync Interrupt

## 7.1. Purpose

The VSync signal indicates the start of a new frame. Using it:

- **Reduces CPU usage**: Task sleeps until frame is ready
- **Perfect timing**: Captures frames at sensor rate (~8.6 Hz)
- **Eliminates polling**: No busy-waiting

## 7.2. ISR Implementation

```
static void IRAM_ATTR Lepton_VSync_ISR_Handler(void *p_Args)
{
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;

    // Notify capture task
    xTaskNotifyFromISR((TaskHandle_t)p_Args, 0, eNoAction,
                       &xHigherPriorityTaskWoken);

    if (xHigherPriorityTaskWoken) {
        portYIELD_FROM_ISR();
    }
}
```

When `CONFIG_LEPTON_VSYNC_PLACE_IRAM` is enabled, the ISR is placed in IRAM for fastest response.

## 7.3. Without VSync

If VSync is not available:

- Task runs continuously without waiting
- Higher CPU usage due to polling
- May miss frame timing boundaries

# 8. Frame Queue Management

## 8.1. Queue Behavior

The capture task uses `xQueueOverwrite()` for frame buffers:

- **Latest Frame**: Always contains newest data
- **No Blocking**: Overwrites old frame if not processed
- **Single Item Queue**: Recommended queue size is 1

**Alternative:** Use larger queue with `xQueueSend()` if you need to process every frame.

## 8.2. Queue Size Recommendations

| Queue Size | Use Case | Behavior |
|---|---|---|
| 1 | Real-time display | Always shows latest frame |
| 2-3 | Balanced performance | Some buffering, low latency |
| 5+ | Frame recording | Process every frame, higher latency |

# 9. Performance Optimization

## 9.1. Task Priority

Set priority based on application needs:

- **High priority (15-20)**: Real-time thermal imaging
- **Medium priority (10-14)**: General applications
- **Low priority (<10)**: Background monitoring

## 9.2. Core Affinity

Pin capture task to dedicated core:

```
#define CONFIG_LEPTON_CAPTURE_TASK_CORE 1  // Use core 1
```

Benefits:

- Reduces cache thrashing
- Predictable timing
- Leaves core 0 for other tasks

## 9.3. IRAM Placement

Enable IRAM for VSync ISR:

```
#define CONFIG_LEPTON_VSYNC_PLACE_IRAM  // Place in IRAM
```

Trade-offs:

- **Faster ISR**: No flash cache access
- **Higher IRAM usage**: Uses precious IRAM space

# 10. Watchdog Integration

The capture task integrates with ESP-IDF watchdog:

```
esp_task_wdt_add(NULL);  // Add current task

while (capturing) {
    esp_task_wdt_reset();  // Reset watchdog every loop
    // ... capture frame ...
}
```

This prevents watchdog timeout during normal operation.

# 11. Telemetry Support

When telemetry is enabled, the capture module:

1. Allocates telemetry buffer
2. Extracts telemetry lines from VoSPI stream
3. Provides telemetry pointer in frame buffer

**Accessing Telemetry:**

```
Lepton_FrameBuffer_t frame;
xQueueReceive(queue, &frame, portMAX_DELAY);

if (frame.Telemetry_Buffer != NULL) {
    // Parse telemetry data
    // Format: 4 lines x 160 words
}
```

# 12. Usage Example

## 12.1. Complete Capture Setup

```
#include "lepton.h"

static Lepton_t lepton;
static QueueHandle_t frame_queue;

void capture_task(void *arg) {
    Lepton_FrameBuffer_t frame;

    while (true) {
        if (xQueueReceive(frame_queue, &frame, portMAX_DELAY)) {
            ESP_LOGI(TAG, "Frame: %dx%d, %d bytes/pixel",
                     frame.Width, frame.Height, frame.BytesPerPixel);

            // Process frame.Image_Buffer here
            // 160x120 pixels, 14-bit per pixel

            // Optional: Check telemetry
            if (frame.Telemetry_Buffer != NULL) {
                // Parse telemetry
            }
        }
    }
}

void app_main(void) {
    // Initialize Lepton
    Lepton_Conf_t config = { /* ... */ };
    Lepton_Init(&lepton, &config, NULL);

    // Create queue (size 1 for real-time)
    frame_queue = xQueueCreate(1, sizeof(Lepton_FrameBuffer_t));

    // Start capture
    if (Lepton_StartCapture(&lepton, frame_queue) != LEPTON_ERR_OK) {
        ESP_LOGE(TAG, "Capture failed");
        return;
    }

    // Create processing task
    xTaskCreatePinnedToCore(capture_task, "capture", 4096, NULL, 10, NULL, 0);
}
```

## 12.2. Monitoring Performance

```
void print_stats(Lepton_t *device) {
    int32_t frames = Lepton_GetFrameCounter(device);
    int32_t errors = Lepton_GetSyncErrors(device);

    float error_rate = (errors * 100.0f) / frames;

    ESP_LOGI(TAG, "Frames: %d, Errors: %d (%.2f%%)",
             frames, errors, error_rate);
}
```

# 13. Troubleshooting

## 13.1. High Sync Errors

**Symptoms:** Many `LEPTON_ERR_FAIL` errors

**Causes:**

- Poor SPI connections
- Electrical noise
- Clock speed too high
- Inadequate power supply

**Solutions:**

- Check wiring and shielding
- Reduce SPI clock speed
- Add decoupling capacitors
- Use shorter cables

## 13.2. Missed Frames

**Symptoms:** Frame counter increases slowly

**Causes:**

- Processing too slow
- VSync not connected
- Task priority too low

**Solutions:**

- Increase task priority
- Connect VSync pin
- Optimize processing code
- Use larger queue

## 13.3. Memory Issues

**Symptoms:** `LEPTON_ERR_NO_MEM` on start

**Causes:**

- Insufficient heap memory
- Too many frame buffers
- Stack overflow

**Solutions:**

- Reduce `CONFIG_LEPTON_VOSPI_FRAME_BUFFERS`
- Increase heap size
- Use PSRAM for buffers

# 14. See Also

- Main Driver - Initialization and configuration
- VoSPI - Low-level SPI communication
- CCI Commands - Telemetry control

# 15. License



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