VoSPI Interface Module

Table of Contents

1. Overview

The `vospi` (Video over SPI) module implements the high-speed SPI interface for receiving thermal image data from the Lepton sensor. It handles packet synchronization, frame assembly, and telemetry extraction.

1.1. Key Features

- DMA-accelerated SPI transfers
- Multi-segment frame synchronization
- Automatic resync on errors
- Multiple frame buffer support
- Telemetry data extraction
- RAW14 and RGB888 format support
- Efficient memory management

2. VoSPI Protocol

2.1. Packet Structure

Each VoSPI packet contains:

Offset	Size	Content
0-1	2 bytes	Header (ID field + segment/packet numbers)
2-3	2 bytes	CRC-16
4-end	160/240 bytes	Payload (80 pixels × 2 or 3 bytes)

2.2. Packet Header Format

```
Bits 15-12: ID field (0x0 = normal, 0xF = discard)
Bits 11-10: Reserved
Bits  9-8:  Segment number (0-3 for Lepton 3.x)
Bits  7-0:  Packet number (0-59 for image, 0-3 for telemetry)
```

2.3. Frame Structure (Lepton 3.x)

A complete frame consists of:

- **4 segments** (Segment 0-3)
- **60 packets per segment** (120 total lines)
- **Optional telemetry**: 4 additional packets per segment

Total packets per frame:

- Without telemetry: 4 × 60 = 240 packets
- With telemetry: 4 × 64 = 256 packets

2.4. Synchronization

VoSPI requires careful synchronization:

1. **Initial Sync**: Wait for Packet 0 of Segment 0
2. **Segment Sync**: Verify ascending segment numbers
3. **Packet Sync**: Verify ascending packet numbers within segment
4. **Resync**: Restart if sync is lost (invalid headers or sequence errors)

3. Frame Buffer Management

3.1. Multi-Buffer System

The module uses rotating buffers:

- Default: 2 frame buffers (configurable via `CONFIG_LEPTON_VOSPI_FRAME_BUFFERS`)
- Capture writes to one buffer while application reads another
- Prevents frame drops during processing

3.2. Buffer Allocation

Buffers are allocated in PSRAM for large frames:

```
Image buffer size = Width x Height x BytesPerPixel
                  = 160 x 120 x 2 = 38,400 bytes (RAW14)
                  = 160 x 120 x 3 = 57,600 bytes (RGB888)

Telemetry buffer size = 4 lines x 160 words x 2 bytes = 1,280 bytes
```

4. Configuration

4.1. Kconfig Options

Option	Description	Default
`CONFIG_LEPTON_VOSPI_FRAME_BUFFERS`	Number of frame buffers	2
`CONFIG_LEPTON_VOSPI_USE_PSRAM`	Allocate buffers in PSRAM	Enabled
`CONFIG_LEPTON_VOSPI_SYNC_TIMEOUT_MS`	Sync timeout in milliseconds	5000

4.2. SPI Configuration

Recommended SPI settings:

- **Clock speed**: 20 MHz (max for Lepton 3.x)
- **Mode**: SPI_MODE_3 (CPOL=1, CPHA=1)
- **Bit order**: MSB first
- **DMA**: Enabled (SPI_DMA_CH_AUTO)

5. Key Functions

5.1. Initialization

```
Lepton_Error_t VoSPI_Init(VoSPI_t *p_Interface);
```

Initialize SPI bus and allocate frame buffers.

5.2. Capture Control

```
Lepton_Error_t VoSPI_StartCapture(VoSPI_t *p_Interface);
Lepton_Error_t VoSPI_StopCapture(VoSPI_t *p_Interface);
```

Start/stop continuous frame capture.

5.3. Frame Capture

```
Lepton_Error_t VoSPI_CaptureImage(VoSPI_t *p_Interface, uint8_t *p_BufferIndex);
```

Capture a single complete frame.

Returns:

- `LEPTON_ERR_OK`: Frame captured successfully

- `LEPTON_ERR_FAIL`: Sync error (frame incomplete/corrupt)
- `LEPTON_ERR_NOT_FINISHED`: Resync in progress

Output:

- `p_BufferIndex`: Index of buffer containing captured frame

5.4. Resynchronization

```
Lepton_Error_t VoSPI_Resync(VoSPI_t *p_Interface);
```

Force resynchronization (called automatically on errors).

6. Packet Processing

6.1. Packet Reading

```
static esp_err_t VoSPI_ReadPacket(
    VoSPI_t *p_Interface,
    uint16_t *p_Header,
    uint8_t **p_PacketData
);
```

Read a single VoSPI packet via SPI.

Steps:

1. Perform SPI transaction (4 + payload bytes)
2. Extract 16-bit header (big-endian)
3. Return pointer to payload data

6.2. Header Parsing

```
uint8_t id = (header >> 12) & 0x0F;        // ID field
uint8_t segment = (header >> 8) & 0x03;    // Segment number
uint8_t packet = header & 0xFF;            // Packet number
```

Valid Headers:

- ID = 0x0: Normal packet
- ID = 0xF: Discard packet (ignore)
- Segment: 0-3

- Packet: 0-59 (image) or 0-3 (telemetry)

6.3. Sync Detection

The module detects sync by:

1. Waiting for ID=0 (valid packet)
2. Checking for Packet 0
3. Verifying Segment 0
4. Ensuring ascending packet/segment numbers

7. Telemetry Handling

When telemetry is enabled:

- Additional 4 packets after each segment
- Packet numbers 0-3 (after 60 image packets)
- Total 4 telemetry lines per segment
- 16 telemetry lines per complete frame

Telemetry is stored in a separate buffer for application access.

8. Performance Considerations

8.1. SPI Transfer Rate

- Packet size: 164 bytes (RAW14) or 244 bytes (RGB888)
- Packets per frame: 240 (no telemetry) or 256 (with telemetry)
- Frame time: ~116ms (~8.6 Hz)

At 20 MHz SPI clock:

- Transfer time per packet: ~65 µs (RAW14)
- Total transfer time per frame: ~15.6ms
- Overhead: ~100ms (processing, sync, gaps)

8.2. Memory Requirements

For 2 frame buffers with telemetry:

- Image buffers: 2 × 38,400 bytes = 76,800 bytes
- Telemetry buffers: 2 × 1,280 bytes = 2,560 bytes

- Packet buffer: 244 bytes (single)
- **Total**: ~80 KB

Recommendation: Use PSRAM for frame buffers.

8.3. CPU Load

- DMA reduces CPU usage during SPI transfers
- Packet parsing is lightweight
- Sync logic adds minimal overhead
- Capture task typically uses <5% CPU at 8.6 Hz

9. Error Recovery

9.1. Sync Errors

Causes:

- SPI communication errors
- Electrical noise
- Missing packets
- Out-of-order packets

Recovery:

1. Increment sync error counter
2. Call `VoSPI_Resync()`
3. Wait for Packet 0, Segment 0
4. Resume normal capture

9.2. Timeout Handling

If no valid frame is received within timeout:

- Return `LEPTON_ERR_TIMEOUT`
- Automatic resync attempt
- Log error for diagnostics

10. Usage Example

```
VoSPI_t vospi = {
```

```
    .Host = SPI2_HOST,
    .Interface = {
        .clock_speed_hz = 20000000,
        .mode = 3,
        .spics_io_num = GPIO_NUM_10,
        .queue_size = 1
    },
    .Master = {
        .sclk_io_num = GPIO_NUM_12,
        .miso_io_num = GPIO_NUM_13,
        .mosi_io_num = -1
    },
    .DMA = SPI_DMA_CH_AUTO,
    .ImageWidth = 160,
    .ImageHeight = 120,
    .BytesPerPixel = 2,
    .useTelemetry = true
};

// Initialize
VoSPI_Init(&vospi);

// Start capture
VoSPI_StartCapture(&vospi);

// Capture frames
while (true) {
    uint8_t buffer_index;
    Lepton_Error_t error = VoSPI_CaptureImage(&vospi, &buffer_index);

    if (error == LEPTON_ERR_OK) {
        // Process frame
        uint16_t *image = vospi.Image_Buffer[buffer_index];
        uint8_t *telemetry = vospi.Telemetry_Buffer[buffer_index];

        ESP_LOGI(TAG, "Frame %d captured", vospi.FrameCounter);
    } else if (error == LEPTON_ERR_FAIL) {
        ESP_LOGW(TAG, "Sync error %d", vospi.SyncErrors);
    }
}
```

11. Troubleshooting

11.1. High Sync Error Rate

- Check SPI wiring and signal quality
- Reduce clock speed
- Add pull-up resistors

- Improve power supply stability

11.2. Frame Drops

- Increase frame buffer count
- Optimize processing code
- Raise capture task priority
- Use faster CPU clock

11.3. Memory Errors

- Enable PSRAM
- Reduce buffer count
- Check heap fragmentation

12. See Also

- Frame Capture - Capture task implementation
- Main Driver - Complete driver API

13. License

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