Standard Operating Procedure (SOP)

Camera capture tool

Document Version: 1.0
Date: January 19, 2026
Purpose: Guide for operating the camera recording tool with synchronisation of electrophysiological data
Author: Cantin Ortiz

1. Overview

This tool captures synchronized video from a FLIR/BlackFly camera with GPIO strobe control, real-time preview, and automated video encoding. All recordings are saved with timing metadata for precise synchronization analysis.

---

2. Safety and Precautions

⚠ **Before starting:**

- Ensure the camera is properly connected via USB 3.0
- Ensure the GPIO cable is also connected
- Verify adequate disk space (approximately 1 GB per minute at 50 Hz)
- Do not disconnect the camera during recording
- Make sure SpinView is closed before starting recording

---

3. Pre-Operation Setup

3.1 System Configuration

1. **Open configuration file** (`config.py`)
2. **Verify settings:**
    - `VENV_PATH`: Path to Python virtual environment (or `None` if system Python has all required packages)
    - `DEFAULT_SAVE_PATH`: Directory where recordings will be saved
    - `DEFAULT_FRAMERATE`: Expected camera frame rate (Hz). Must match SpinView
    - `DEFAULT_LINE`: GPIO line for strobe output (1 or 2). Depends on your setup. Use 1 for Axona, 2 for OpenEphys
    - `CHUNK_DURATION_S`: Video chunk size in seconds (default: 10). Likely no need to edit.
    - `BUFFER_MULTIPLIER`: Memory buffer size (increase if disk is slow). Likely no need to edit.
    - `JPEG_QUALITY`: Image quality 0-100 (default: 85). Likely no need to edit.
3. **Save changes** if any settings were modified

3.2 Camera Configuration (SpinView)

Before using this tool, configure the camera in SpinView:

1. Set desired frame rate (e.g., 50 Hz)
2. Set the resolution as desired
3. Crop the image as wanted
4. **Close SpinView**

---

4. Operating Procedures

4.1 Starting a Recording Session

Method 1: Quick Start (Default Settings)

1. Double-click `start_recording.bat`
2. Wait for camera initialization (3-5 seconds)
3. Live preview window appears showing camera feed
4. Console displays: `Press ENTER to START recording`

Method 2: Custom Parameters

1. Open Command Prompt or PowerShell
2. Navigate to tool directory
3. Run with desired options:

```
start_recording.bat --duration 30 --framerate 50
```

Method 3: If the .bat script does not work

1. Open Command Prompt or PowerShell
2. Navigate to tool directory
3. Run with desired options:

```
python camera_capture_tool\src\main_recorder.py --duration 30 --framerate 50
```

4. You may need to activate a virtual environment first, or select the correct version of python

4.2 Recording Process

1. **Open the video capture tool — preview Phase**
    - Live video preview appears (unless `--nolive` specified)
    - Verify camera is showing correct view
    - Adjust positioning/focus if needed

- Opening the program before starting the ephys recording ensures that the GPIO is set to the correct mode (constant value).

2. **Start Ephys recording**

   - It is crucial for synchronisation that the ephys recording is started *before* the video recording
   - The video recording is considered started when the key "ENTER" is pressed and frames get acquired, not when opening the program.

3. **Start video recording**

   - Press **ENTER** in the console window
   - GPIO strobe activates
   - Console displays: "Acquisition is running"
   - Lag counter shows buffer status: `Lag: X/Y frames | Time: Zs`

4. **During Recording**

   - Monitor lag counter (should stay near 0)
   - If lag increases significantly (>10% of buffer), recording may be too fast for disk
   - If lag gets close to the buffer size (Y), the storage disk is getting overloaded and the recording will likely crash.
   - Simple solutions are to reduce resolution, increase frame compression (lower value for JPEG_QUALITY in config.py), reduce framerate, and check that no other program overloads the storage disk. Also consider using the --sequential mode.

5. **Stop Recording**

   - **Manual stop**: Press **ENTER** in console
   - **Automatic stop**: Wait for specified duration to elapse. Pressing **ENTER** will cause an anticipated ending of the recording.
   - GPIO strobe deactivates
   - Message displays: "Acquisition complete"

4.3 Post-Recording Processing

Automatic steps (no user input required):

1. Frame saving completes
2. Video chunks are encoded (if video generation enabled)
3. Final video is assembled
4. Timing metadata is saved to CSV file
5. Frame files are deleted (unless `--keep-frames` specified)
6. Console displays: "Recording complete!"

Press any key to close the program.

---

5. Command-Line Options

Option	Values	Default	Description

Option	Values	Default	Description
`--duration`	Number (seconds)	None	Auto-stop after specified time
`--framerate`	Number (Hz)	50	Expected camera frame rate
`--save_path`	Path string	`~/Documents/flea3_recordings`	Recording destination
`--line`	1 or 2	2	GPIO line for strobe output
`--output`	`video`, `images`, `both`	`video`	What to save
`--keep-frames`	Flag	Off	Keep frame files after video generation
`--sequential`	Flag	Off	Disable concurrent video rendering
`--nolive`	Flag	Off	Disable live preview window
`--debug`	Flag	Off	Enable debug messages

Examples:

```
# 30-second recording at 50 Hz, save only video
start_recording.bat --duration 30 --framerate 50

# Continuous recording until manual stop, keep frame files
start_recording.bat --keep-frames

# Save only images, no video encoding
start_recording.bat --duration 10 --output images

# Recording without live preview (better performance)
start_recording.bat --duration 60 --nolive
```

---

6. Output Files

Each recording session creates:

6.1 Directory Structure

```
~/Documents/flea3_recordings/
└── VIDEO_YYYYMMDD-HHMMSS/
    ├── VIDEO_YYYYMMDD-HHMMSS.mp4     (if video output enabled)
    ├── VIDEO_YYYYMMDD-HHMMSS.csv     (timing and metadata)
    └── frames/                        (if --keep-frames specified)
        ├── frame_0000000.jpg
```

```
        ├── frame_0000001.jpg
        └── ...
```

6.2 CSV Metadata File

The `.csv` file contains:

- **Timing data**: Precise timestamps for synchronization
  - `t_first_frame`: First frame capture time
  - `t_last_frame`: Last frame capture time
  - `t_set_line_exposure`: GPIO strobe ON time
  - `t_set_line_constant`: GPIO strobe OFF time
- **Configuration parameters**: All settings used for the recording
  - `duration_s`, `framerate_hz`, `gpio_line`
  - `generate_video`, `keep_frames`, `concurrent_render`
  - `live_video`, `debug_mode`
  - `chunk_duration_s`, `buffer_multiplier`, `jpeg_quality`

Use case: This metadata ensures full reproducibility and enables precise synchronization with external devices.

---

7. Troubleshooting

7.1 Common Issues

Problem	Possible Cause	Solution
"Camera not detected"	Camera disconnected or in use	Check USB connection; close SpinView
"PySpin not found"	Virtual environment not activated	Verify `VENV_PATH` in config.py
High lag during recording	Slow disk I/O	Increase `BUFFER_MULTIPLIER` in config.py
Frame rate error	Mismatch with camera settings	Match `--framerate` to SpinView configuration
Video file missing	Frame rate instability	Check console for frame rate error messages
Batch file label error	Incorrect config.py encoding	Re-save config.py as UTF-8

7.2 Frame Rate Warnings

If console displays:

```
[ERROR] Estimated frame rate (X Hz) differs from expected (Y Hz) by more than 1 Hz
```

Actions:

1. Frames are saved but video won't be generated (safety feature)
2. Check camera configuration in SpinView
3. Verify `--framerate` matches camera settings
4. Review system performance (CPU/disk usage)

7.3 Buffer Overflow

If lag counter approaches buffer size:

```
Lag: 950/1000 frames | Time: 25.3s
```

Actions:

1. Current recording will complete safely (buffer has headroom)
2. For future recordings, increase `BUFFER_MULTIPLIER` in config.py
3. Consider: faster storage device, lower JPEG quality, or slower frame rate

---

8. Data Management

8.1 Storage Requirements

Approximate disk usage per minute of recording:

Frame Rate	Resolution	JPEG Quality	Storage/min
50 Hz	1920×1200	85	~1.2 GB
50 Hz	1920×1200	75	~0.9 GB
100 Hz	1920×1200	85	~2.4 GB

8.2 Archiving Recordings

1. Recordings are saved in timestamped folders: `VIDEO_YYYYMMDD-HHMMSS`
2. Each folder is self-contained with video and metadata
3. Move entire folders to archive storage
4. CSV file enables later verification of recording parameters

---

9. Maintenance

9.1 Regular Checks

- **Weekly**: Verify disk space availability
- **Monthly**: Review saved recordings and archive as needed
- **As needed**: Update config.py settings for new experiments

9.2 Software Updates

When updating the tool:

1. Note current `config.py` settings
2. Pull updates from repository: `git pull`
3. Review `config.py` for new options
4. Test with short recording before full use

---

10. Best Practices

☑ Do:

- Test with short recording (2-5s) before long sessions
- Monitor lag counter during first minute of recording
- Keep backup of important recordings
- Document experimental conditions in lab notebook
- Use `--nolive` for long recordings to reduce CPU load

❌ Don't:

- Disconnect camera during recording
- Run multiple camera applications simultaneously
- Modify files in recording directory during capture
- Ignore frame rate error warnings

---

11. Contact and Support

For technical issues or questions about this tool:

- Check README.md for detailed documentation
- Review git commit history for recent changes
- Contact: [Your contact information]

Revision History

Version	Date	Changes	Author
1.0	2026-01-19	Initial SOP creation	-