---

layout: doc title: PLC Library Usage description: Complete guide to using Event Video Playback function blocks in your TwinCAT PLC projects permalink: /docs/plc-usage/

Overview

The Event Video Playback PLC library provides easy-to-use function blocks for integrating event-driven video recording into your TwinCAT applications. This guide covers the complete API and usage patterns.

Library Reference

Adding the Library

In your TwinCAT PLC project:

1. Right-click on **References**
2. Select **Add Library**
3. Search for "EventVideoPlayback"
4. Select the library and click **OK**

The library will now be available for use in your PLC programs.

Core Function Blocks

FB_EventVideoRecorder

The main function block for recording event-driven videos from TwinCAT Vision images.

Declaration

```
VAR
   fbVideoRecorder : FB_EventVideoRecorder;
END_VAR
```

Input Parameters

Parameter	Type	Description
`sVideoName`	STRING	Unique name for the video file
`bExecute`	BOOL	Rising edge triggers video creation
`tPreEventTime`	TIME	Duration of pre-event video to include (default: 10s)

`tPostEventTime`	TIME	Duration of post-event video to include (default: 5s)
`sImagePath`	STRING	Path to TwinCAT Vision image folder
`bLogToEventLogger`	BOOL	Enable automatic Event Logger integration

Output Parameters

Parameter	Type	Description
`bDone`	BOOL	TRUE when video creation is complete
`bBusy`	BOOL	TRUE while video is being created
`bError`	BOOL	TRUE if an error occurred
`nErrorID`	UDINT	Error code (0 = no error)
`sVideoPath`	STRING	Full path to the created video file

Usage Example

```
PROGRAM MAIN
VAR
   fbVideoRecorder : FB_EventVideoRecorder;
   bMachineError   : BOOL;
   bEventTriggered : BOOL;
   sEventName      : STRING(255);
END_VAR

// Detect machine error event
IF bMachineError AND NOT bEventTriggered THEN
   bEventTriggered := TRUE;
   sEventName := CONCAT('MachineError_', TO_STRING(NOW()));
END_IF

// Record video on event
fbVideoRecorder(
   sVideoName := sEventName,
   bExecute := bEventTriggered,
```

```
    tPreEventTime := T#15S,  // 15 seconds before event
    tPostEventTime := T#10S,  // 10 seconds after event
    sImagePath := 'C:\TwinCAT\Vision\Images',
    bLogToEventLogger := TRUE
);

// Reset trigger when done
IF fbVideoRecorder.bDone OR fbVideoRecorder.bError THEN
    bEventTriggered := FALSE;
END_IF

// Handle errors
IF fbVideoRecorder.bError THEN
    // Log error code: fbVideoRecorder.nErrorID
    // Take corrective action
END_IF
```

Advanced Usage Patterns

Multiple Event Types

Handle different types of events with separate video recordings:

```
PROGRAM MAIN
VAR
    fbVideoQuality    : FB_EventVideoRecorder;
    fbVideoMaintenance : FB_EventVideoRecorder;
    fbVideoSafety     : FB_EventVideoRecorder;

    bQualityIssue     : BOOL;
    bMaintenanceNeeded : BOOL;
    bSafetyEvent      : BOOL;
END_VAR

// Quality issue recording
fbVideoQuality(
    sVideoName := 'QualityIssue_' + TO_STRING(NOW()),
    bExecute := bQualityIssue,
    tPreEventTime := T#20S,
    bLogToEventLogger := TRUE
);

// Maintenance event recording
fbVideoMaintenance(
    sVideoName := 'Maintenance_' + TO_STRING(NOW()),
    bExecute := bMaintenanceNeeded,
    tPreEventTime := T#30S,
    bLogToEventLogger := TRUE
);
```

```
// Safety event recording
fbVideoSafety(
   sVideoName := 'Safety_' + TO_STRING(NOW()),
   bExecute := bSafetyEvent,
   tPreEventTime := T#60S,
   tPostEventTime := T#30S,
   bLogToEventLogger := TRUE
);
```

Conditional Recording

Record videos only under specific conditions:

```
VAR
   fbVideoRecorder : FB_EventVideoRecorder;
   bEventOccurred  : BOOL;
   bRecordEnabled  : BOOL := TRUE;
   nEventSeverity  : UINT;
   nMinSeverity    : UINT := 5;  // Only record severity >= 5
END_VAR

// Only record if enabled and severity threshold met
IF bEventOccurred AND bRecordEnabled AND (nEventSeverity >= nMinSeverity) THEN
   fbVideoRecorder(
       sVideoName := CONCAT('Event_Severity_', TO_STRING(nEventSeverity)),
       bExecute := TRUE,
       bLogToEventLogger := TRUE
   );
END_IF
```

Sequence Recording

Record a sequence of related events:

```
TYPE E_RecordingState :
(
   IDLE,
   RECORDING,
   WAITING_FOR_NEXT,
   COMPLETE
);
END_TYPE

VAR
   fbVideoRecorder   : FB_EventVideoRecorder;
   eState            : E_RecordingState := IDLE;
   nSequenceCount    : UINT := 0;
   nMaxSequences     : UINT := 5;
   bStartSequence    : BOOL;
END_VAR
```

```
CASE eState OF
   IDLE:
       IF bStartSequence THEN
           nSequenceCount := 0;
           eState := RECORDING;
       END_IF

   RECORDING:
       fbVideoRecorder(
           sVideoName := CONCAT('Sequence_', TO_STRING(nSequenceCount)),
           bExecute := TRUE
       );

       IF fbVideoRecorder.bDone THEN
           nSequenceCount := nSequenceCount + 1;

           IF nSequenceCount >= nMaxSequences THEN
               eState := COMPLETE;
           ELSE
               eState := WAITING_FOR_NEXT;
           END_IF
       END_IF

   WAITING_FOR_NEXT:
       // Wait for condition to trigger next recording
       IF bTriggerNextRecording THEN
           eState := RECORDING;
       END_IF

   COMPLETE:
       // All sequences recorded
       eState := IDLE;
END_CASE
```

Error Handling

Error Codes

Error ID	Description	Resolution
0	No error	N/A
1001	Service not available	Check if EventVideoPlayback service is running

1002	Invalid image path	Verify sImagePath is correct
1003	Insufficient images	Ensure Vision system is capturing images
1004	Video creation failed	Check service logs for details
1005	Disk space insufficient	Free up disk space
1006	Invalid parameters	Check input parameter values

Error Handling Example

```
VAR
   fbVideoRecorder : FB_EventVideoRecorder;
   sErrorMessage   : STRING(255);
END_VAR

IF fbVideoRecorder.bError THEN
   CASE fbVideoRecorder.nErrorID OF
       1001:
           sErrorMessage := 'EventVideoPlayback service is not running';
           // Attempt to restart service or notify maintenance

       1002:
           sErrorMessage := 'Invalid image path configured';
           // Log configuration error

       1003:
           sErrorMessage := 'Not enough images available for video';
           // Check Vision system status

       1004:
           sErrorMessage := 'Video creation failed';
           // Check service logs and disk space

       1005:
           sErrorMessage := 'Insufficient disk space';
           // Trigger cleanup or notification

       ELSE:
           sErrorMessage := CONCAT('Unknown error: ', TO_STRING(fbVideoRecorder.nErrorID));
   END_CASE

   // Log error to Event Logger or HMI
END_IF
```

Best Practices

1. Unique Video Names

Always use unique video names to avoid conflicts:

```
// Good: Include timestamp
sVideoName := CONCAT('Event_', TO_STRING(NOW()));

// Better: Include machine ID and event type
sVideoName := CONCAT('Machine01_QualityIssue_', TO_STRING(NOW()));
```

2. Appropriate Time Windows

Choose pre/post event times based on your needs:

- **Fast events** (< 1 second): Use 5-10 seconds pre-event
- **Slow events** (> 10 seconds): Use 30-60 seconds pre-event
- **Post-event**: Usually 5-10 seconds is sufficient

3. Resource Management

Don't create videos too frequently:

```
VAR
   fbVideoRecorder    : FB_EventVideoRecorder;
   tLastRecording     : TIME;
   tMinInterval       : TIME := T#30S;  // Minimum 30s between recordings
END_VAR

IF (TIME() - tLastRecording) > tMinInterval THEN
   // Allow recording
   tLastRecording := TIME();
END_IF
```

4. Event Logger Integration

Enable Event Logger integration for automatic correlation:

```
fbVideoRecorder(
   sVideoName := sEventName,
   bExecute := bEvent,
   bLogToEventLogger := TRUE  // Automatically creates Event Logger entry
);
```

Performance Considerations

- **Image Buffer**: The service maintains a rolling buffer of images
- **CPU Usage**: Video encoding is performed by the service, not PLC

- **Network Impact**: Minimal - only ADS communication for commands
- **Disk I/O**: Configure video output path on fast storage

Next Steps

- Configure Service Settings for optimal performance
- Add HMI Controls for video playback
- Review Getting Started for installation help