

Scanning Modes Explained

JoeScan scanners can operate in different scanning modes. This document explains the differences and provides application examples and requirements.

- [Connecting to a Scanner](#)
- [On-Demand Scanning](#)
 - [Advantages of Using On-Demand Scanning](#)
 - [Disadvantages of using On-Demand Scanning](#)
- [Sync-Mode Scanning](#)
 - [EncoderSyncMode](#)
 - [Advantages of EncoderSyncMode](#)
 - [Disadvantages of EncoderSyncMode](#)
 - [PulseSyncMode](#)
 - [Advantages of PulseSyncMode](#)
 - [Disadvantages of PulseSyncMode](#)
 - [TimeSyncMode](#)

Connecting to a Scanner

For all modes, the control computer must utilize the correct API call to establish a TCP/IP connection to the scan head.

Action	.NET API	C/C++ API
Connect to scan head	Scanner.Connect()	jsOpenConnection() jsOpenConnectionBase() jsConnectionOpenInt()

On-Demand Scanning

Once connected to the control computer via one of the above calls in the API library, a scan head stays in On-Demand Mode unless another mode is requested. For On-Demand scanning, the triggering of scans is initiated by the controlling computer. Using the API calls in the table below, a scan is triggered and the resulting data is returned as a [profile](#). Because the trigger comes from the computer, the request is sent over the ethernet connection. The scanner API library handles all the communication.

Action	.NET API	C/C++ API
Request single profile	Scanner.GetProfile()	jsGetProfile()

Advantages of Using On-Demand Scanning

- simple to implement
- full control over when a scan is triggered
- sequentially triggering scans on multiple heads will never lead to laser cross-talk
- no additional wiring needed beyond power and ethernet

Disadvantages of using On-Demand Scanning

- synchronous/blocking operation (i.e., the scanning software has to wait for the scan to complete and return data before continuing)
- slow scanning: Because acquisition of data is sequential with data transfer and processing, high scan rates cannot be achieved
- difficult to accurately time: If scans in a specific position are required, the scan software has to be carefully written to time scan triggers. Even then, network latency can lead to delays.
- difficult to integrate multiple heads: If multiple heads are to be triggered simultaneously, avoiding cross talk is difficult. Triggering multiple heads sequentially is easy, but slow.

If an encoder is connected to the scan head, the resulting profile will contain the value of the encoder at the time of exposure in the Location field ([Profile.Location](#), [tagProfile::location](#)).



On-Demand scanning is typically not used for full-speed production systems, but can be very useful for calibration or monitoring applications where a high scan rate is not required. Prototyping and testing applications can use this mode.

Sync-Mode Scanning

Sync-Mode scanning is an umbrella term for three scanning modes: EncoderSyncMode, PulseSyncMode and TimeSyncMode.

They have in common that the **triggering of individual scans is controlled by the scan head itself**, not the control computer. When and where a scan happens is dependent on the mode:

Sync-Mode	Scan is triggered when:
EncoderSyncMode	a scanned object has traveled a specific distance / incremental encoder count has increased by a number of pulses (as set in the parameter file with the EncoderPulseInterval and EncoderScanInterval).
PulseSyncMode	a pulse was received on the StartScan input line.
TimeSyncMode	a specific time interval has elapsed according to the internal scan head clock.

Buffering

Because scans happen independently of the controlling computer, the resulting profiles must be buffered in the scan head until they can be read out. The scanning and read-out processes are independent of each other, so that data transfers to the controlling computer and scans happen simultaneously.

The JS-25 models have an internal buffer with room for 500 profiles. Even though the scan heads can buffer up to 500 profiles, it is generally not recommended to rely on that fact. Transferring profiles over the network takes time, and in a fast scanning system the buffer can overrun even when a transfer was already requested. In a well-designed system, the readout should be as fast or faster than the scanning itself.

EncoderSyncMode

In EncoderSyncMode, a scan head **triggers a scan based on traveled distance**, as measured by an encoder. Our scanners allow you to specify the travel distance in physical units (i.e., inches or millimeters), as opposed to encoder units. The parameters [EncoderPulseInterval](#) and [EncoderScanInterval](#) let the scan head convert the raw encoder values into a travel distance. By specifying how much physical distance is equivalent to a single phase of the quadrature signal in [EncoderPulseInterval](#), the scan interval (e.g., every inch, every five millimeters) can be conveniently specified in [EncoderScanInterval](#).

 The specifications of a typical incremental rotary encoder will include the number of pulses per revolution. The JS-25 counts every phase of the quadrature signal, so for most encoders, the Encoder Count in the head will be 4 times the pulses specified with the encoder. You can easily check in [JS Diag / Laser View](#) by noting the encoder count difference for a full revolution.

In EncoderSyncMode, the scan head will also need a hardware signal, StartScan, to be present before profiles are collected. After entering EncoderSyncMode, no profiles will be triggered until the StartScan pin has been pulled to low. From this time on, profiles are collected and buffered until the scanner exits sync mode.

This functionality can be used to avoid scanning the empty belt until a board arrives. For example, the scanners can wait in EncoderSyncMode until a photo eye or PLC pulls the start scan line to low.

If you want to start scanning immediately after entering EncoderSyncMode, use the keyword [UntriggeredSyncScanning](#) in the parameter file.

Action	.NET API	C/C++ API	Comments
Enter EncoderSyncMode	Scanner.EnterEncoderSyncMode()	jsEnterEncoderSyncMode()	
Initiate transfer of buffer contents	Scanner.BeginGetQueuedProfiles()	jsSendMultipleProfileRequest()	functions exist to request <i>n</i> number of profiles at a time
End transfer of buffer contents	Scanner.EndGetQueuedProfiles()	jsReadMultipleProfiles()	
Exit EncoderSyncMode	Scanner.ExitSyncMode()	jsExitSyncMode()	will clean up all profiles in buffer that were not transferred

Effective Network Data Transfer

In all sync modes, the transfer process is split into two calls for efficiency reasons.

- The first call ([Scanner.BeginGetQueuedProfiles\(\)](#)/[jsSendMultipleProfileRequest\(\)](#)) will return immediately (i.e., it doesn't block). The scan head will begin sending data to the control PC immediately over the open TCP/IP connection. This data is buffered by the network hardware and the Operating System.
- The second call then reads the data from the Operating System buffer. This technique ensures that the API doesn't have to wait for the full network roundtrip time when multiple heads are used (i.e., it interleaves requests and responses).

This strategy only makes sense on systems with multiple heads. For single head systems, there is a convenience call ([Scanner.GetQueuedProfiles\(\)](#) / [jsGetMultipleProfiles\(\)](#)) that bundles the call pair and blocks fully until all data has arrived.

Advantages of EncoderSyncMode

- guaranteed location of scan
- high speed of profile acquisition
- operation independent of controlling computer, with some buffering to account for network latency and readout timing
- can utilize Staggering to coordinate large groups of scan heads in order to avoid laser crosstalk

Disadvantages of EncoderSyncMode

- requires an encoder input on all participating heads
- due to buffering and asynchronous operation, the readout of acquired data is more complex
- easy to overdrive the scan heads if chain/belt speed is variable
- more complex setup to ensure scan timing

PulseSyncMode

In PulseSyncMode, a scan head **triggers based on an internal clock in regular intervals**. In addition, a scan head can put out a pulse train on the Start Scan line and trigger a group of scanners that are then all synchronized to the first head. The head controlling the timing is called the **PulseMaster**. There can be only one PulseMaster. All heads in a group must have their Start Scan lines tied together, in parallel. A single head can also be run in PulseSyncMode, no Start Scan line is required in this case.

The acquisition of profiles happens independently of the controlling computer; the profiles are buffered and read out asynchronously, similar to operation in EncoderSyncMode.

Action	.NET API	C/C++ API	Comments
Enter PulseSyncMode	Scanner.EnterPulseSyncMode()	jsEnterPulseSyncMode()	
Initiate transfer of buffer contents	Scanner.BeginGetQueuedProfiles()	jsSendMultipleProfileRequest()	functions exist to request n number of profiles at a time
End transfer of buffer contents	Scanner.EndGetQueuedProfiles()	jsReadMultipleProfiles()	
Exit EncoderSyncMode	Scanner.ExitSyncMode()	jsExitSyncMode()	will clean up all profiles in buffer that were not transferred

Advantages of PulseSyncMode

- guaranteed timing of scan
- high speed of profile acquisition
- operation independent of controlling computer, with some buffering to account for network latency and readout timing
- can utilize Staggering to coordinate large groups of scan heads in order to avoid laser crosstalk

Disadvantages of PulseSyncMode

- due to buffering and asynchronous operation, the readout of acquired data is more complex
- more complex setup to ensure scan timing
- additional wiring to StartScan pin needed if more than one head used

TimeSyncMode

TimeSyncMode is similar to PulseSyncMode in that scans are triggered at fixed time intervals, however, each head uses their own internal clock instead of being triggered externally by the Start Scan signal. In groups of more than one scanner, there is no coordination between heads, and therefore clock drift can lead to laser cross talk.

In practice, TimeSyncMode is rarely used as PulseSyncMode offers the same functionality with the advantage of being usable with more than one head. TimeSyncMode will be deprecated in a future firmware release.