Whitepaper über den BlockScan-Modus mit mvBlueFOX3

Endless love

1 Einführung

Im Gegensatz zum Flächenscan-Modus ermöglicht der Blockscan-Modus, dass runde oder rotierende Körper oder lange oder endlose Materialien mit hoher Geschwindigkeit inspiziert werden können (ähnliche wie Zeilenkameras). Hierzu nimmt der BlockScan-Modus einen Area of Interest (AOI) Block, bestehend aus mehreren Zeilen auf. Danach wird eine einstellbare Zahl an AOI-Blöcken zusammen als Gesamtbild übertragen. Dadurch wird der Overhead minimiert, den es anderenfalls bei der Übertagung der AOI-Blöcke als Einzelbilder durch das Transferprotokoll von USB3 bzw. GigE Vision geben würde.

Dieses Dokument zeigt, wie mit dem Blockscan-Modus in Verbindung mit mvBlueFOX3 Industriekameras mit IMX Global Shutter Sensoren von Sony gearbeitet wird. Hierbei wird die Kamera im Blockscan-Modus durch einen Inkrementalgeber getriggert, um Objekte auf einer rotierenden Trommel aufzunehmen.

HINWEIS

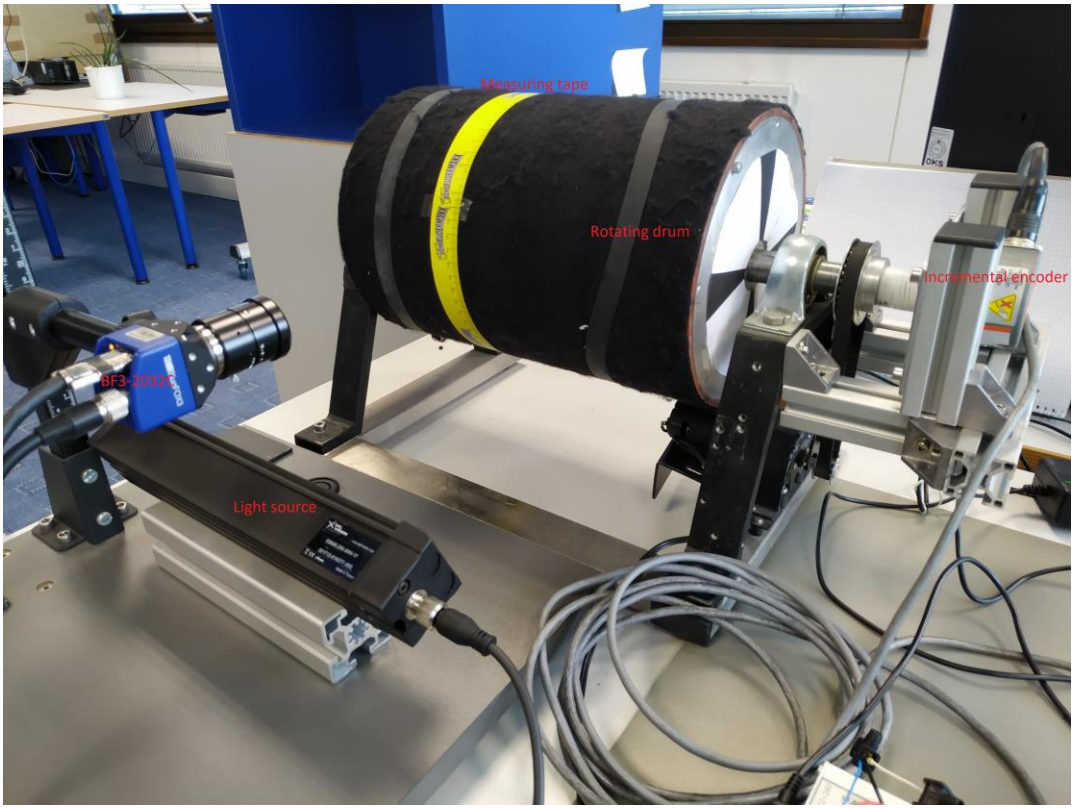
Natürlich funktionieren auch andere Hardware-Trigger (beispielsweise neben einem Förderband) oder Software-Trigger als Quelle. In diesem Fall wird "**EncoderControl**" nicht benötigt und "**TriggerSource**" muss angepasst werden. In diesem Dokument wird jedoch ein Inkrementalgeber als Trigger-Quelle verwendet.

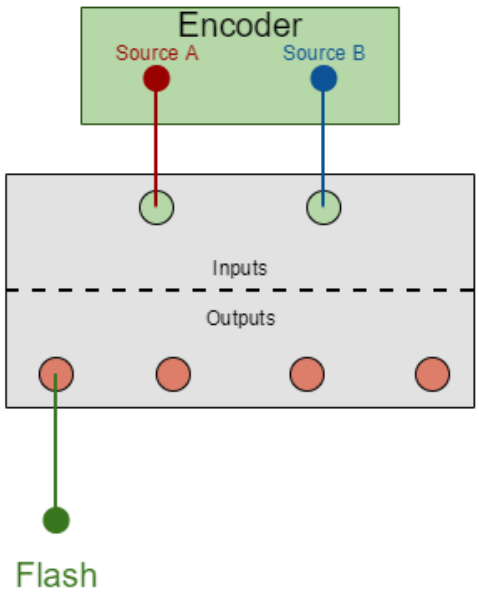
Folgende Punkte werden in diesem Dokument behandelt:

1. Wie wird der Blockscan-Modus in Verbindung mit einem Inkrementalgeber in wxPropView konfiguriert?
2. Wie wird ein Endlosbild auf einem rotierenden Objekt erfasst?
3. Was passiert, wenn ein rotierendes Objekt stehen bleibt und noch Bildblöcke im Bildspeicher vorhanden sind?
4. Was passiert, wenn ein rotierendes Objekt die Richtung ändert und sich dann wieder in die Ursprungsrichtung dreht?

5. Was sind die Zeilenraten für
    a. mvBlueFOX3-2032C/G,
    b. mvBlueFOX3-2051C/G und
    c. mvBlueFOX3-2089C/G?
6. Was sind die Vorteile des Blockscan-Modus gegenüber einer Zeilenkamera?
7. Was sind die Nachteile des Blockscan-Modus im Vergleich zu einer Zeilenkamera?

2 Mechanischer Aufbau





Materialien:

- 1 x rotierende Trommel mit Motor
- 1 x Maßband
- 1 x Inkrementalgeber (1000ppr)
- 1 x Kamera (mvBlueFOX3-2032C)
- 1 x Lichtquelle; 1 x Trigger-Box

Die rotierende Trommel ist mit einem Motor und einem Inkrementalgeber verbunden. Auf der Oberfläche ist ein Maßband angebracht, um die Abstände ablesen zu können *(siehe Bild links oberhalb)*. Der Inkrementalgeber und die Lichtquelle sind mit einer Trigger-Box über die I/Os der Kamera mit dieser verbunden *(siehe Bild rechtsoberhalb)*. (Encoder) Source A → (Camera) Line 4; (Encoder) Source B → (Camera) Line 5.

3 Abläufe

3.1 Wie wird der Blockscan-Modus in Verbindung mit einem Inkrementalgeber in wxPropView konfiguriert?

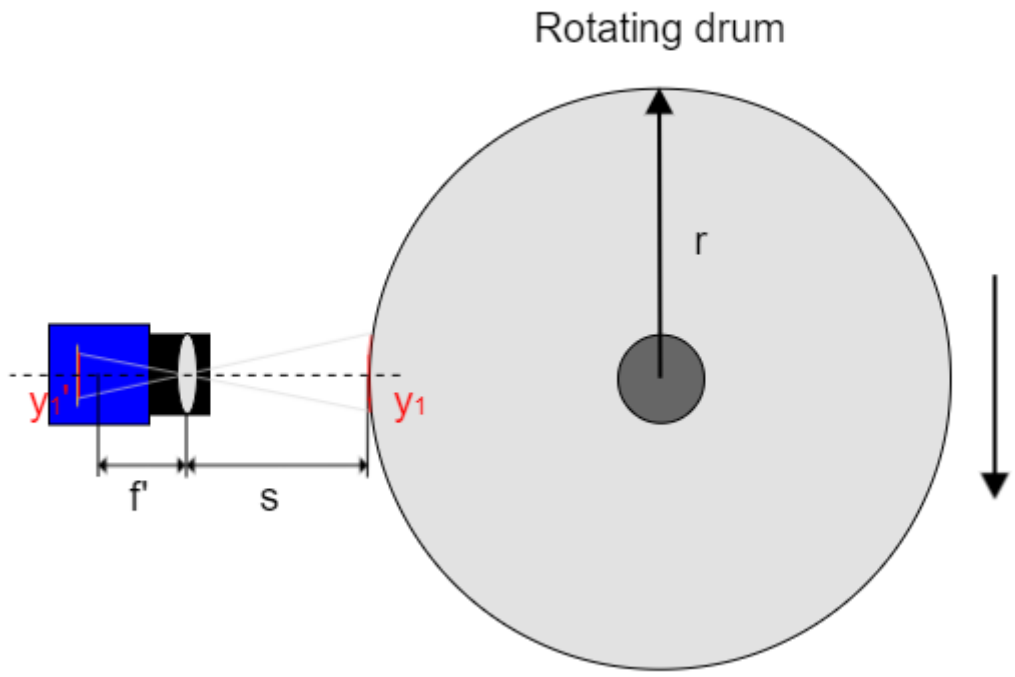
Erforderliche Firmware-Version: > **2.35**

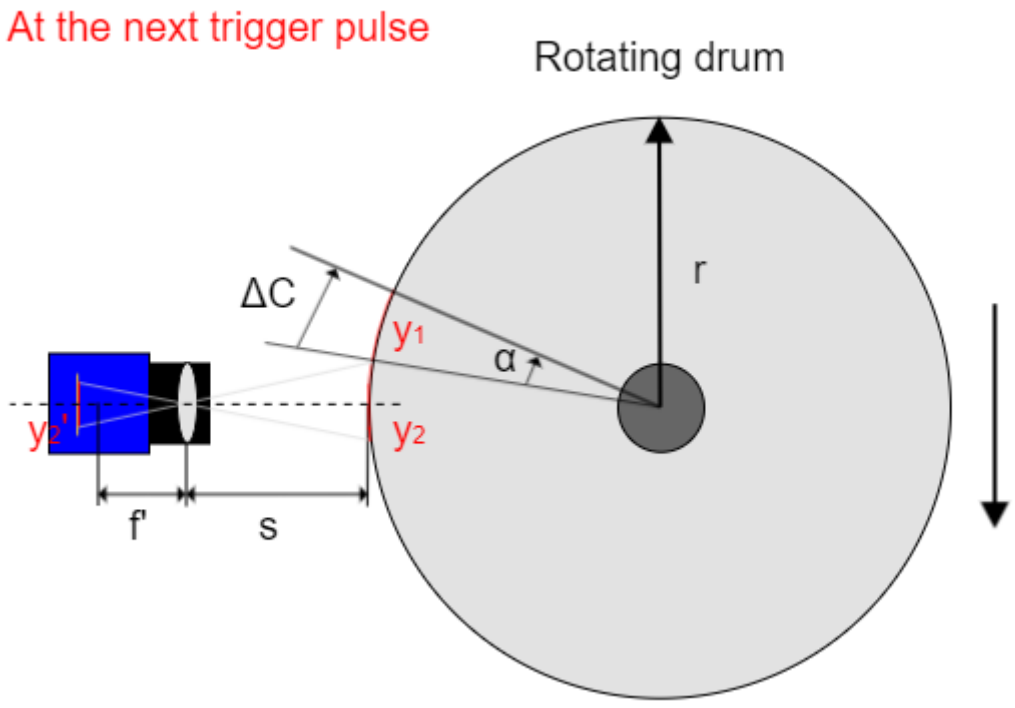
Konfigurationsschritte in wxPropView:

1. Setting → Base → Camera → GenICam → DeviceControl → **DeviceScanType**: mvBlockScan.
2. Setting → Base → Camera → GenICam → ImageFormatControl: configure **OffsetY**, **mvBlockscanLinesPerBlock**(>= 16), **mvBlockscanBlockCount**(>= 2).
3. Setting → Base → Camera → GenICam → ImageFormatControl → **PixelFormat**: BayerRG8 *(für Farbkameras)*.
4. Setting → Base → Camera → GenICam → EncoderControl → **EncoderSourceA**: Line4; **EncoderSourceB**: Line5; **EncoderDivider**: 1; **EncoderOutputMode**: PositionDown/DirectionDown.
5. Setting → Base → Camera → GenICam → AcquisitionControl → **TriggerSelector**: FrameStart; **TriggerMode**: On; **TriggerSource**: Encoder0.
6. Setting → Base → Camera → GenICam → AcquisitionControl → **ExposureTime**.
7. Setting → Base → Camera → GenICam → DigitalIOControl → **LineSelector**: der Ausgang, an welchem die Lichtquelle angeschlossen ist; **LineSource**: ExposureActive.
8. (optional) Setting → Base → Camera → GenICam → DigitalIOControl → **debounce time** für Line4 und Line5 setzen.

3.2 Wie wird ein Endlosbild auf einem rotierenden Objekt erfasst?

Nach einem gewissen zurückgelegten Weg, erzeugt der Inkremetalgeber einen Puls. Jeder Puls (falls Encoder Divider = 1) löst die Aufnahme einer Anzahl an Bildblöcken aus, die dann zu einem Gesamtbild zusammengefasst werden. Die nachfolgende Skizze zeigt den Querschnitt des Inspektionssystems:





Wird eine neuer Trigger-Puls erzeugt, hat sich die Trommel um den Winkel *α* weitergedreht, welche den Umfang *ΔC* abdeckt. Da *r* der Radius der Trommel, *n* der Encoder Divider, welcher die Anzahl der

Inkrementalpulse vorgibt und *p* die Inkrementalpulse (*ppr: pulse per revolution*) sind, kann daraus *ΔC* ermittelt werden:

$\Delta C = 2*\pi*r*n/p$ *(eq-1)*

Damit ein Endlosbild aus den unterschiedlichen Blöcken (y_1', y_2'.....) generiert werden kann, muss die Höhe des Sichtfeldes eines jeden Blockes mit *ΔC* identisch sein:

$y1 = y2 = \Delta C$ *(eq-2)*

Da in unserem Beispiel als Objekt ein Maßband verwendet wird, kann die Höhe des Objekts y*(mm)* und die entsprechende Bildhöhe *y'(Pixel)* leicht abgelesen werden. Dadurch kann der Abbildungsmaßstab des Objektivs *β (Pixel/mm)* auch leicht ermittelt werden:

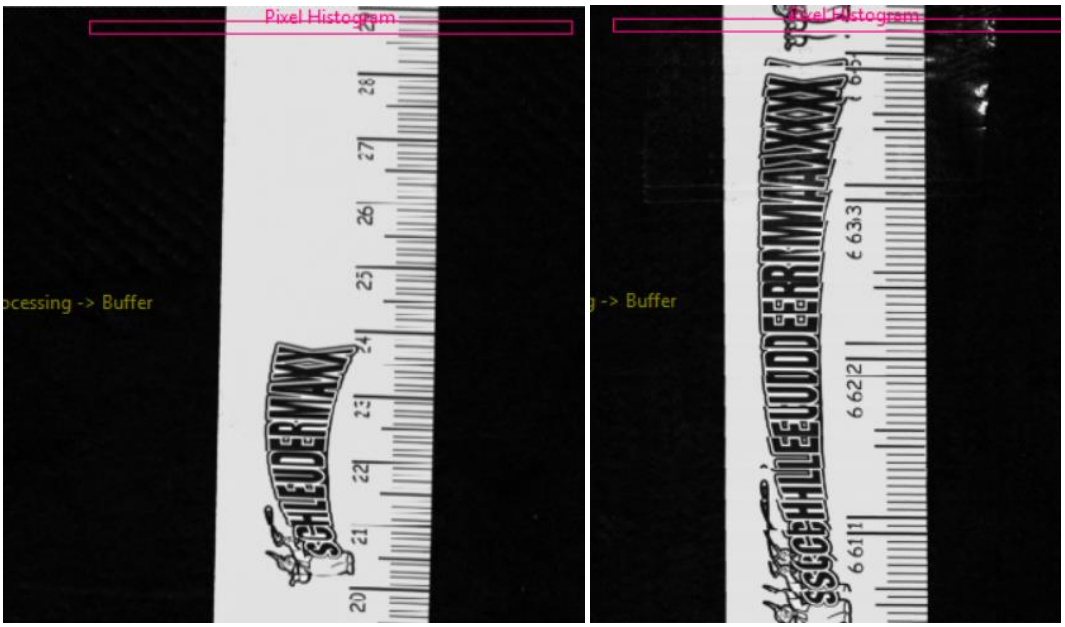
$\beta = y'/y$ *(eq-3)*

Werden alle drei Formel *eq-1, eq-2* und *eq-3* zusammengeführt, kann der benötigte Wert ***lines per block*** (y_1' oder y_2') für Endlosbilder in diesem System extrahiert werden:

$\mathbf{LinesPerBlock = (2*\pi*r*\beta/p)*n}$ *(eq-4)*

Die Berechnung wurde bestätigt. In unserem Aufbau wird ein korrektes Endlosbild mit *EncoderDivider in wxPropView =**3*** und *LinesPerBlock = **20*** erzeugt. Das Testergebnis sieht wie folgt aus:

Falls LinesPerBlock zu niedrig gewählt wird, weißt das erzeugte Bild Lücken auf *(siehe Bild unten links)*. Falls LinesPerBlock zu groß gewählt wird, dann gibt es Überlappungen im Bild *(siehe Bild unten rechts)*.



Falls in anderen Applikationen ein niedriger lines per block (z.B. **16**) erforderlich ist, dann müssen β und n angepasst werden, um ein passendes Endlosbild zu erhalten:

$$\beta = (LinesPerBlock*p/(2*\pi*r))*(1/n) \quad \textit{(eq-5) (abgeleited von eq-4)}$$

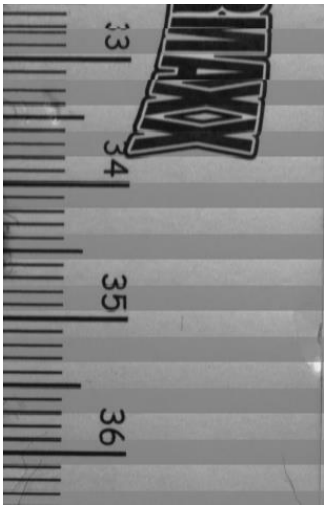
Da β grob durch Gaußsche Formel $\beta = |f'/(f'-S)|$ berechnet werden kann, kann β wiederum durch Anpassung der Brennweite f' und des Arbeitsabstands S variiert werden.

3.3 Drehrichtung

In diesem Beispiel bewegt sich das Objekt, von der Kamera aus gesehen, nach oben. Würde die Kamera auf der anderen Seite der Trommel angebracht werden, dann würde sich das Objekt nach unten bewegen, obwohl sich die Drehrichtung der Trommel nicht geändert hätte. Der Inkrementalgeber würde weiter nach unten zählen und folgendes Bild würde daraus resultieren:

Die Blöcke werden erkennbar in der falschen Reihenfolge zusammengefügt. Dies kann durch Spiegelung der einzelnen Blöcke ausgeglichen werden.

1. Setting → Base → Camera → GenICam → ImageFormatControl → ReverseX
2. Setting → Base → Camera → GenICam → ImageFormatControl → ReverseY



Das resultierende Bild ist invertiert und entspricht nicht der Realität, was für einzelne Anwendungen problematisch sein könnte. Aus diesem Grund empfehlen wir, dass die Kamera so angebracht ist, dass das Objekt sich in Bezug auf den Sensor nach oben bewegt.

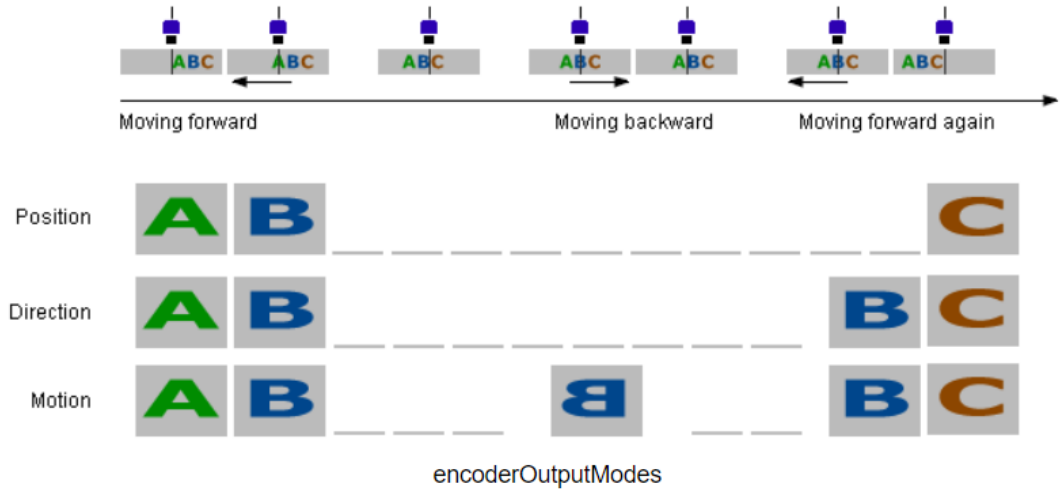
3.4 Was passiert, wenn ein rotierendes Objekt stehen bleibt und noch Bildblöcke im Bildspeicher vorhanden sind?

Während der Erfassung sendet der Sensor kontinuierlich Bildblöcke an den FPGA, welcher die Blöcke zu einem Frame/Bild zusammenfasst, bis der gegebene BlockCount erreicht wird. Der komplette Frame / das komplette Bild wird dann im Bildspeicher des Treibers abgelegt. Wird die Rotation gestoppt, befinden sich Bildblöcke im FPGA. Werden diese Blöcke nicht gelöscht, dann werden diesen weitere Blöcke hinzugefügt, bis der Frame / das Bild vollständig ist. Somit erhalten wir ein Bild aus zwei unterschiedlichen Objekten, was nicht gewünscht ist. Aus diesem Grund sollte **Abort** in wxPropView oder **imageRequestReset** von unserem SDK aufgerufen werden, bevor eine Inspektion mit einem neuen Objekt gestartet werden soll, um den unvollständigen Frame im FPGA zu verwerfen.

Der **Abort** Aufruf benötigt ca. 200us, falls der Host nicht überlastet ist.

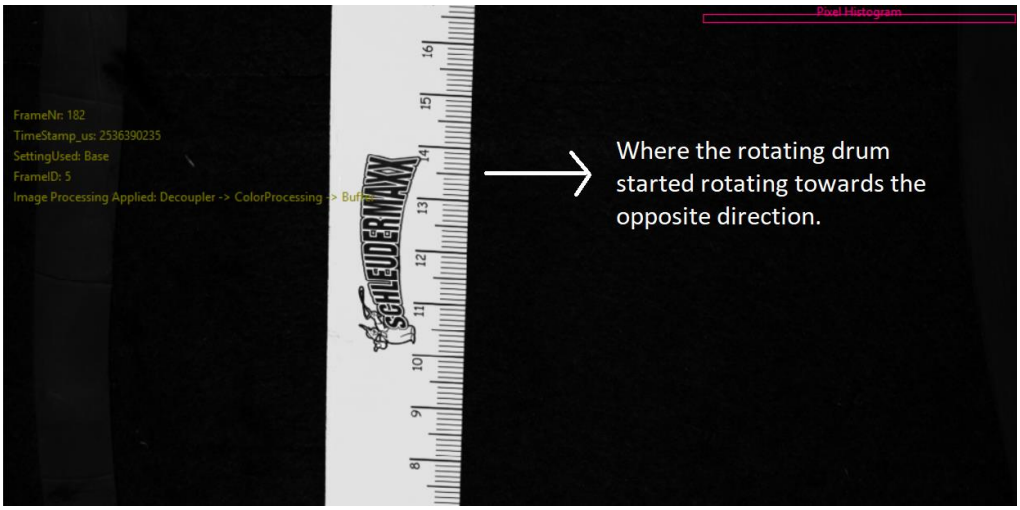
3.5 Was passiert, wenn ein rotierendes Objekt die Richtung ändert und sich dann wieder in die Ursprungsrichtung dreht?

Unsere Firmware unterstützt 3 Encoder Output Modi: position up/down, direction up/down, motion. Diese 3 Modi werden hier im Detail beschrieben: https://www.matrix-vision.com/manuals/SDK_CPP/classmvIMPACT_1_1acquire_1_1GenICam_1_1EncoderControl.html#a129fb01bf3edfcd0fe6b66197e4284de. Eine grafische Erklärung sieht wie folgt aus:



(aus dem C++ SDK)

Sobald wir position up/down auswählen, wird das Bild angehalten, sobald sich die Rotationsrichtung ändert. Rotiert die Trommel wieder in die ursprüngliche Richtung, dann werden die kommenden Blöcke an die Stelle des Stopps korrekt angehängt. So wird die Erfassung nicht von einem Richtungswechsel betroffen. Unser Test hat das bestätigt:



3.6 Was sind die Zeilenraten für mvBlueFOX3-2032C/G, mvBlueFOX3-2051C/G und mvBlueFOX3-2089C/G?

mvBlueFOX3-	2024C/G	2032C/G	2051C/G	2089G/C
Zeilenrate (kHz) @ 16 Zeilen/Block (kHz)	59,5	56,0	48,5	23,6
Blockrate (kHz) @ 16 Zeilen/Block (kHz)[3]	3,7	3,5	3,1	1,5
Max. Belichtungszeit (us) @ 16 Zeilen/Block (kHz)[1][2]	186	199	228	440

*[1] **Max. Belichtungszeit (us)** @ 16 Zeilen/Block (kHz) kann beobachtet werden, sobald **mvAcquisitionFrameRateLimitMode** auf **mvDeviceMaxSensorThroughput** gesetzt ist.*

*[2] **Max. Belichtungszeit (us)** @ 16 Zeilen/Block (kHz) kann auch mittels **1/BlockRate - t_p** berechnet werden, wobei **t_p** die Zeit zwischen zwei benachbarten Belichtungszeiten ist, wo kein Trigger stattfinden darf. **t_p** für mvBlueFOX3-2024, mvBlueFOX3-2032, mvBlueFOX3-2051 und mvBlueFOX3-2089 beträgt **83us**, **86,5us**, **102us** und **229,5us**.*

*[3] **Blockrate (kHz) @ n Zeilen/Block (kHz) = 1s / (readout time for n lines + readout time for vertical blank lines)**. Für mvBlueFOX3-2024, mvBlueFOX3-2032, mvBlueFOX3-2051 und mvBlueFOX3-2089 ist die Auslesezeit für **vertical blank lines 179us**, **190us**, **220us** und **509us**.*

3.7 Was sind die Vorteile des Blockscan-Modus gegenüber einer Zeilenkamera?

1. Standard-Schnittstelle: USB3 Vision und GigE Vision im Gegensatz zu CoaXPress und Cameralink.
2. Vereinfachter Systemaufbau: Da die Kamera auch als Flächenkamera verwendet werden kann, ist es wesentlich einfacher, den Fokus einzustellen, um ein scharfes Bild zu bekommen.
3. Weniger Blockverluste aufgrund des FPGAs in der Kamera.

4. Weniger Last auf Seiten des Hosts: Blöcke werden in der Kamera gesammelt.

5. Günstiger als Zeilenkameras (bei gleicher Zeilenfrequenz).

3.8 Was sind die Nachteile des Blockscan-Modus im Vergleich zu einer Zeilenkamera?

1. Endlose Bilder benötigen eine exakte Berechnung bei jeder neuen Applikation.

2. Das Bildseitenverhältnis (Höhe zu Breite Verhältnis) muss immer 1:1 sein, um periodische Verzeichnungen in Y-Richtung des Ergebnisbildes zu vermeiden (im Gegensatz zu Zeilenkameras, bei denen jedes Bildseitenverhältnis möglich ist).

3. Aus diesem Grund müssen Optiken und Timings bei jeder Anwendung sorgsam angepasst werden.

4. Min. Zeilen/Block = 16.

5. Der kleinste Erhöhungs-/Erniedrigungsschrittweite des BlockCounts beträgt 4. Der BlockCount kann demnach nur 16, 20, 24, 28, 32 usw. sein.

6. Beide Eingangsleitungen der Kamera sind durch den Inkrementalgeber belegt. Daher kann kein weitere Hardware-Trigger (bspw. eine Lichtschranke) angeschlossen werden.