Software/MicroManager(マイクロマネージャ)

From ZaberWiki

Micro-Manager は、顕微鏡自動化用の使いやすいインターフェイスを提供する無料のオープンソースソフトウェアパッケージです。 さまざまな顕微鏡、カメラ、モーションコントロールデバイスをサポートし、ImageJ 画像処理ソフトウェアで動作します。

目次

- 1 適合性
- 2 構成(セットアップ)
  - 2.1 Zaber デバイス
  - 2.2 マイクロマネージャ
    - 2.2.1 2 軸システムの設定
    - 2.2.2 短軸システムの設定
    - 2.2.3 フィルターホィールの設定
    - 2.2.4 フィルターキューブターレットの設定
    - 2.2.5 イルミネータ(照明器)の設定
- 3 コントロール(制御)
  - 3.1 ステージ制御プラグイン
  - 3.2 マウス操作
  - 3.3 フィルターホィールのプリセット
  - 3.4 マニュアル調整ノブ
- 4 スクリプト作成

適　合　性

現在、Micro-Manager には、Zaber デバイスを制御するための **5 種類のドライバー**があります。**ステージ**、**XYStage**、**フィルターホイール**、**フィルターキューブタレット**、**イルミネーター**です。 これらのドライバーを使用するには、Zaber デバイスにファームウェア v6.06 以降が必要であり、Zaber ASCII プロトコルを使用するように設定する必要があります。 さらに、**XYStage** ドライバーには少なくとも 2 つのデバイスがポートに接続されている必要があり、**フィルターホイール**ドライバーには **X-FWR フィルターホイール**が接続されている必要があり、**フィルターキューブタレット**には **MVR 顕微鏡**のコンポーネントが必要であり、**照明器**には **X-LCA** シリーズ LED コントローラが必要です。

構成（セットアップ）

Zaber デバイス

Zaber スタンドアロンコントローラーを使用している場合は、まずこちらの簡単なセットアップページに従って構成してください。 https://www.zaber.com/products/product_group.php?group=X-MCB2&tab=Easy%20Set-up. 特に、正しい「peripheralid」設定を実行して、コントローラーの設定が最適化されていることを確認してください。 周辺機器 ID の設定方法の詳細については、Zaber のプロトコルマニュアル（ASCII）を参照してください。

コントローラー内蔵の Zaber デバイスをご使用の場合は、以下の簡単セットアップページに従ってください。
https://www.zaber.com/products/product_group.php?group=X-FWR-E&tab=Easy%20Set-up.

Micro-Manager（マイクロマネージャ）

Micro-Manager の最新バージョンはここからダウンロードできます。 https://micro-manager.org/wiki/Micro-Manager.

Micro-Manager が起動すると、先ず、デモ構成ファイルをロードするように求められます。「Yes」をクリックします。 次に、[ツール]タブで[ハードウェア構成ウィザード]を実行し、新しい構成ファイルの作成を選択します。 デバイスリストで「Zaber」フォルダーを見つけてください。

2 軸システムのセットアップ

デバイスリストの Zaber フォルダーから、XYStage | Zaber XY ステージを選んで下さい。

「add（追加）」を選択すると、次の初期化プロパティを入力するよう求められます。s:

Axis Number (X Axis): これは、X 軸に接続されている軸番号を指します。 X 軸が統合コントローラーまたは単軸コントローラーを使用する場合、値 1 を入力します。 X 軸が X-MCB2 や A-MCB2 などの多軸コントローラーに接続されている場合、値は 1 または 2 になります。

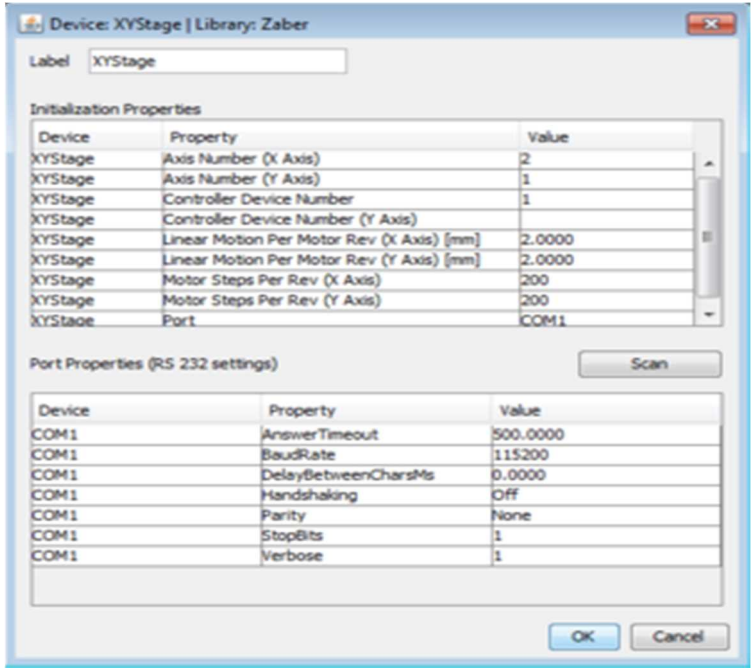
Axis Number (Y Axis): これは、Y 軸に接続されている軸番号を指します。 Y 軸が内蔵コントローラーまたは単軸コントローラーを使用する場合、値 1 を入力します。 Y 軸が X-MCB2 や A-MCB2 などの多軸コントローラーに接続されている場合、値は 1 または 2 になります。

Controller Device Number: X 軸のデバイスアドレス、または両方の軸が X-MCB2 や A-MCB2 などの多軸 Zaber コントローラーに接続されている場合、コントローラーのデバイスアドレス。

Controller Device Number (Y Axis): 軸が別のコントローラー上にあるか、コントローラーが内蔵されている場合、Y 軸のデバイスアドレス。 両方の軸が X-MCB2 や A-MCB2 などの多軸 Zaber コントローラーを使用している場合は、このフィールドを空白のままにします。

Linear Motion Per Motor Rev (X Axis): この値は、web ページのステージの仕様にあります。
Linear Motion Per Motor Rev (Y Axis): この値は、web ページのステージの仕様にあります。
Motor Steps Per Rev (X Axis): この値も、web ページのステージの仕様にあります。（通常 200).
Motor Steps Per Rev (Y Axis): この値も、web ページのステージの仕様にあります。（通常 200).
Port: デバイスが接続されている COM ポート

ポートを選択すると、追加の設定が利用可能になります。 以下の値を使用して下さい。



; XY ステージ構成例

AnswerTimeout: 500.00
BaudRate: 115200（デフォルト（初期値）から変更されていない場合のみ）
DelayBetweenCharsMs: 0.00
Fast USB to Serial: Disable（無効）
Handshaking: Off
Parity: None
StopBits: 1
Verbose: 1

Zaber の 2 軸 X-MCB2 外部コントローラーと ASR 顕微鏡ステージを使用している場合、ホームセンサーとアウェイセンサーを調整することで物理的な可動限界（リミット）が設定できます。 可動限界（リミット）の設定の詳細については、次の手順を参照してください。 https://www.zaber.com/wiki/Manuals/ASR#Setting_Home_and_Away_Sensors. リミットを移動したら、コントローラーの移動範囲も設定する必要があります。t.

「Tool(ツール)」に進み、「Stage position list (ステージ位置リスト)」を選択し、「Set origin(原点設定)」を押して、コントローラーの移動の最小および最大制限を設定します。 リミットを変更した場合、このプロセスを繰り返します。.

単軸システムのセットアップ

デバイスリストの Zaber フォルダーから、[ステージ]Zaber ステージ を選択します。 [add(追加)]を選択すると、次の初期化プロパティ(要素)を入力するよう求められます。s:

Axis Number: X 軸が内蔵コントローラーまたは単軸コントローラーを使用する場合、値 1 を入力します。 X 軸が X-MCB2 や A-MCB2 などの多軸コントローラーに接続されている場合、値は 1 または 2 になります。
Controller Device Number: ステージのデバイスアドレス
Linear Motion Per Motor Rev: この値は、ステージの仕様 Web ページにあります
Motor Steps Per Rev: この値は Web ページのステージ仕様にある値（通常は 200、2 相ステッパーモータの基本ステップ数）
Port: デバイスが接続されている COM ポート.

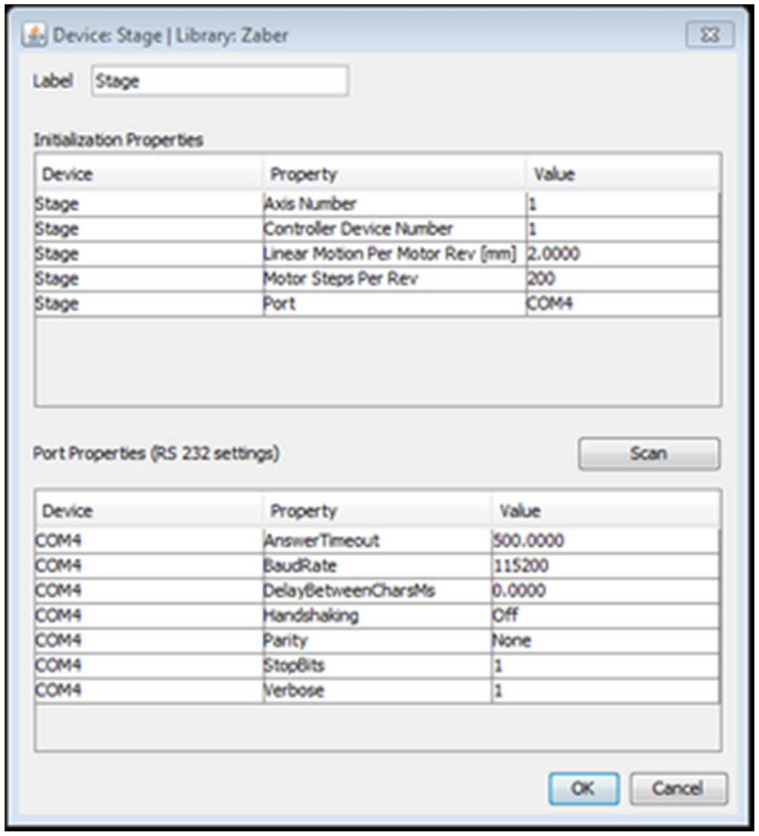
デバイスリストの Zaber フォルダーから、[ステージ]Zaber Stage を選択します。.

[add(追加)]を選択すると、次の初期化プロパティを入力するよう求められます。

Axis Number: X 軸が内蔵コントローラーまたは単軸コントローラーを使用する場合、値 1 を入力します。 X 軸が X-MCB2 や A-MCB2 などの多軸コントローラーに接続されている場合、値は 1 または 2 になります。
Controller Device Number: ステージのデバイスアドレス。
Linear Motion Per Motor Rev: この値は、ステージの仕様 Web ページにあります
Motor Steps Per Rev: T この値は Web ページのステージ仕様にある値(通常 200)
Port: デバイスが接続されている COM ポート.

ポートを選択すると、追加の設定が利用可能になります。 これらの値を使用します。

AnswerTimeout: 500.00
BaudRate: 115200 (デフォルトから変更されていない限り
DelayBetweenCharsMs: 0.00
Fast USB to Serial: Disable(無効)
Handshaking: Off
Parity: None
StopBits: 1
Verbose: 1



サンプルステージ構成

フィルターホイールのセットアップ

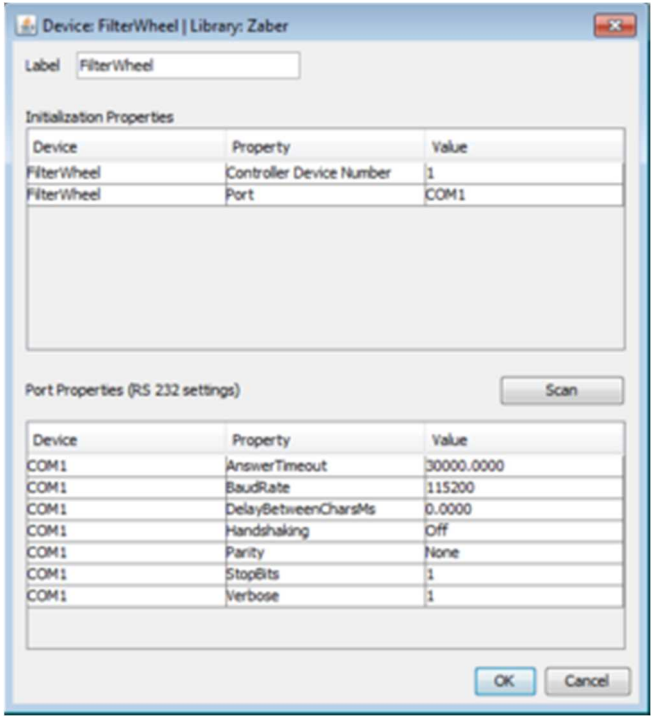
デバイスリストの Zaber フォルダーから、フィルターホイールを選択します。 オプションとして、フィルターホイールが表示されない場合はマイクロマネージャの最新の nightly build に更新して下さい。r.

[add(追加)]を選択すると、次の初期化プロパティを入力するよう求められます。

Controller Device Number: フィルターホイールのデバイスアドレス。
Port: デバイスが接続されている COM ポート。

ポートを選択すると、追加の設定が利用可能になります。 これらの値を使用:

AnswerTimeout: 30000.00
BaudRate: 115200 (デフォルトから変更されていない限り)
DelayBetweenCharsMs: 0.00
Fast USB to Serial: Disable
Handshaking: Off
Parity: None
StopBits: 1
Verbose: 1

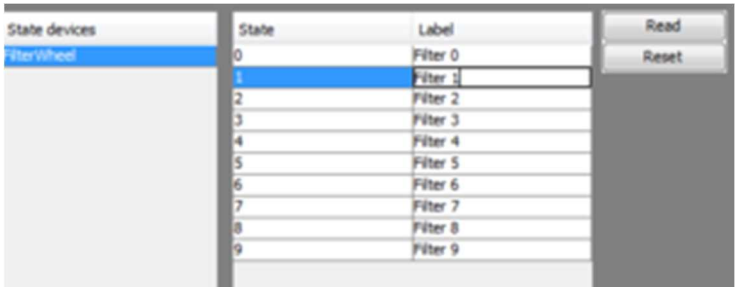


サンプルフィルターホイール構成

Micro-Manager がフィルターホイールの検出プロセスを完了した後、各フィルター位置にラベルを付けるオプションがあります。

フィルターホイールを追加すると、Micro-Manager はフィルターホイール検出ルーチンを自動的に実行します。 これにより、使用可能なフィルターの数が Micro-Manager に通知されます。

デバイスのフィルターホルダーをサイズまたは位置の数が異なるものに変更する場合(たとえば、7 ポジションフィルターホルダーから 10 ポジションフィルターホルダー)、ハードウェア構成ウィザードを使用してフィルターホイール検出コマンドを再実行します。



サンプルフィルターホイールラベル画面

フィルターキューブタレットのセットアップ

フィルターキューブタレットのセットアップは、AnswerTimeout をデフォルト値の 500 に設定したままにしておくことを除いて、フィルターホイールのセットアップと同じです。

イルミネーターのセットアップ

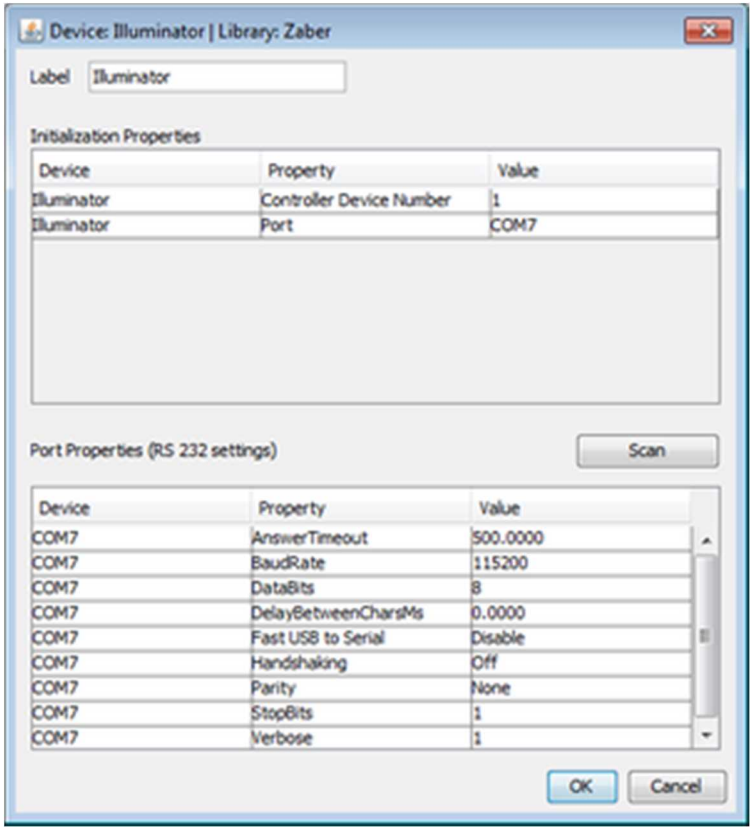
デバイスリストの Zaber フォルダーから、Illuminator を選択します。 オプションとして Illuminator が表示されない場合は、Micro-Manager の最新のナイトリービルド（nightly build）に更新してください。.

[add(追加)]を選択すると、次の初期化プロパティを入力するよう求められます。

Controller Device Number: イルミネーターのデバイスアドレス。
Port: デバイスが接続されている COM ポート

ポートを選択すると、追加の設定が利用可能になります。 これらの値を使用します。

AnswerTimeout: 500.00
BaudRate: 115200（デフォルト値から変更されていない限り）
DelayBetweenCharsMs: 0.00
Fast USB to Serial: Disable（無効）
Handshaking: Off
Parity: None
StopBits: 1
Verbose: 1

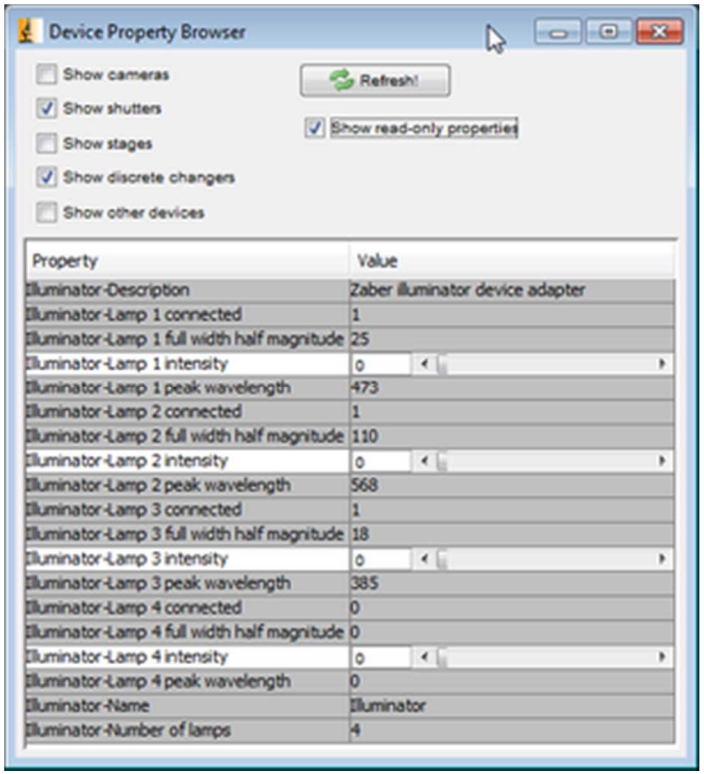


イルミネーターの構成例

構成が完了したら、プリセットとシャッター開閉ボタンを使用してイルミネーターを制御できるようになります。 各 LED には、LED がサポートする最大値の割合として出力光束を制御する強度プロパティがあります。 LED が存在することとそのプロパティを確認するために使用できるいくつかの読み取り専用プロパティもあります。

右の画像は、4 チャンネルイルミネータが提供するすべてのプロパティを示しています。「ランプの数」プロパティは、デバイスが駆動できる LED の数を示します。 1 から始まる各 LED には、ランプが存在するかどうか、およびそのピーク波長と FWHM（全幅半値幅）が何であるかを示すプロパティがあります。 FWHM は、ランプの出力スペクトルの広さの尺度です。 ピーク波長と FWHM はナノメートルで測定されます。 白色 LED は両方の波長特性についてゼロを報告しますが、接続されていると表示されることに注意してください。

LED が過熱した場合、LED は自動的に遮断され、次回 Micro-Manager が Illuminator にコマンドを送信したときにエラーが発生します。 シャッターを閉じて、LED が冷めるまでしばらく待つ必要があります。 温度が下がると、自動的に再び使用可能になります。



サンプル照明のプロパティ

ステージ制御

ステージの位置を制御するには、いくつかの方法があります。

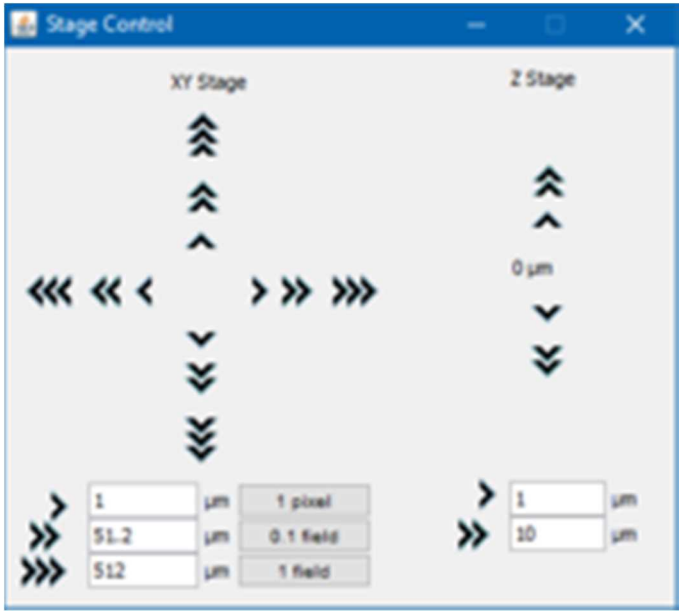
- ステージコントロール通電
- マウスの操作（スナップ/ライブウィンドウでクリック及びドラッグ）
- フィルターホイールプリセット
- コントローラーの手動調整ノブを使用する

Tool> Device/ Property Browser ダイアログボックスを使用して、各デバイスの速度と加速を調整できます。

ステージ制御プラグイン

The stage control plugin can be found under *Plugins* > *Device Control* > *Stage Control*.のところにステージコントロールプラグインがあります。これは XY ステージ(**XY stage**)ドライバー(左側)或いはステージ(**stage**)(右側)を使って操作できる、公開されたコントロールツールです。

微動、中動、粗動の距離をマッピングできます。.



ステージコントロールインターフェース

マウス動作

[ツール(tool)]> (Mouse moves Stagez)マウスでステージを移動]を選択すると、スナップ/ライブ画面で、XY ステージをマウスで移動できます。 クリックしたままにすると、ステージはマウスの位置に従います。 また、ウィンドウ内の任意の場所をダブルクリックすると、ステージが移動してそのポイントを中心にウィンドウが中央に移動します。

この方法を使用する前に、ピクセルサイズを調整する必要があります。 これを実行:

1. 顕微鏡とカメラが設定され、表示するオブジェクトに焦点が合っていることを確認します。
2. [ツール]> [ピクセルサイズ調整]を選択し、[新規]をクリックして新しい調整設定を作成します。
3. キャリブレーション、特にカメラの解像度に影響するプロパティを入力します。
4. ピクセルサイズのサイズを µm で推定し、µm/ピクセルで入力し、[OK]をクリックしてキャリブレーションウィンドウを終了します。

5. [ツール]で[マウス移動ステージ]を選択し、ImageJ ウィンドウからハンドツールを選択します.
6. スナップ/ライブウィンドウで、画像上の小さな明確なポイントを見つけ、クリックしてウィンドウ全体にドラッグします。 ポイントがマウスカーソル/ハンドツールの前にくる場合は、μm/ピクセル設定を減らす必要があります。 カーソルの後ろで終了する場合は、設定を増やす必要があります。 ポイントがマウスカーソルの真下に留まるまで繰り返します。

フィルターホイールプリセット

フィルターホイールデバイスを追加した後、[構成設定]の下にそのためのエントリがあるはずです。 プリセットの下に、各フィルターホイールのラベルまたは番号のエントリを含むドロップダウンメニューがあります。 いずれかを選択して、対応する位置に移動します。

手動調整ノブ

Zaber デバイスは、手動制御ノブを使用して制御することもできます。 ノブを 1 秒間押し続けると、速度モードと変位モードが切り替わります。 ステージの移動中にノブを押すと、ステージが停止します。

スクリプティング

Micro-Manager には、[ツール]> [スクリプトパネル]からアクセスできる組み込みのスクリプトインターフェイスがあります。 Micro-Manager のプログラミング機能については、ガイドをご覧ください: Programming Guide .

以下は、XYStage ドライバーを使用して Zaber の ASR 2 軸ステージを制御するためのサンプルスクリプトです。

```
mmc.unloadAllDevices();
mmc.loadDevice("port", "SerialManager", "COM1");
mmc.setProperty("port", "BaudRate", "115200");

mmc.loadDevice("asr", "Zaber", "XYStage");
mmc.setProperty("asr", "Port", "port");
```

```
// この例では、2 つの周辺軸が接続された単一のコントローラーがある
// と想定し、軸の番号付けにはデフォルト設定を使用します。
// この行は、コントローラーのアドレスを設定する方法を示しています。
mmc.setProperty("asr", "Controller Device Number", "1");
mmc.initializeAllDevices();

// ホームポジションに移動します。
//また、このコマンドにより、デバイスは動きの限界を見つけるため、
// デバイスはその全範囲を移動します。
// このスクリプトを実行する前に、ステージのパスから障害物を取り除きます！
mmc.home("asr");

// 正方形を描きます。 setXYPosition 呼び出しは、移動するたびに
// include waitForDevice 呼び出しがブロックされないことに注意してください。
mmc.setXYPosition("asr", 0, 0);
mmc.waitForDevice("asr");

mmc.setXYPosition("asr", 0, 10000);
mmc.waitForDevice("asr");

mmc.setXYPosition("asr", 10000, 10000);
mmc.waitForDevice("asr");

mmc.setXYPosition("asr", 10000, 0);
mmc.waitForDevice("asr");

mmc.setXYPosition("asr", 0, 0);
mmc.waitForDevice("asr");
```

```
mmc.unloadAllDevices();
```

フィルターホイールまたはフィルターキューブタレットドライバーを使用してスクリプトを作成する場合、Micro-Manager はデバイスを状態デバイスと見なします。 これは、各フィルターを表す多数の定義済みの物理状態があることを意味します。 以下は、フィルターホイールのサンプルスクリプトです。

```
mmc.unloadAllDevices();

mmc.loadDevice("port", "SerialManager", "COM1");
mmc.setProperty("port", "BaudRate", "115200");

mmc.loadDevice("wheel", "Zaber", "FilterWheel");
mmc.setProperty("wheel", "Port", "port");
mmc.initializeAllDevices();

int numPositions = mmc.getNumberOfStates("wheel");
gui.message("Filter wheel has " + numPositions.toString() + " positions.");

gui.message("Moving to position 0");
mmc.setState("wheel", 0);
mmc.sleep(250);

for (int i = 1; i < numPositions; ++i)
{
   gui.message("Moving to position " + i.toString());
   mmc.setState("wheel", i);
   mmc.sleep(100);
}
```

```
mmc.sleep(250);

for (int i = numPositions - 2; i >= 0; --i)
{
   gui.message("Moving to position " + i.toString());
   mmc.setState("wheel", i);
   mmc.sleep(100);
}
```

Illuminator のスクリプトを作成するとき、デバイスのプロパティを読み取って、存在する LED の数、波長を決定し、強度プロパティを記述して、各 LED の出力光束を個別に制御できます。 Illuminator は Micro-Manager のシャッターデバイスと見なされるため、シャッター制御機能を使用して LED をオンまたはオフにできます。 LED を検出し、一度に 1 つずつ点灯させるスクリプトの例を次に示します。ただし、可視スペクトルの範囲外にあるものを除きます（偶発的な眼の損傷を防ぐため）。

```
mmc.unloadAllDevices();
mmc.loadDevice("port", "SerialManager", "COM7");
mmc.setProperty("port", "BaudRate", "115200");

// 照明器をセットアップします。
// これは、Zaber デバイスアドレス 1 の X-LCA を想定しています。
mmc.loadDevice("lca", "Zaber", "Illuminator");
mmc.setProperty("lca", "Port", "port");
mmc.setProperty("lca", "Controller Device Number", "1");
mmc.initializeAllDevices();

// ランプを消します。
mmc.setShutterOpen("lca", false);
```

```
// ランプの可能な数を取得します。
numLamps = Integer.parseInt(mmc.getProperty("lca", "Number of lamps"));
int[] lamps = new int[numLamps];
int lampIndex = 0;

gui.message("Device supports up to " + numLamps + " lamps.");

for (int i = 0; i < numLamps; i++)
{
   //目に見える範囲で接続されているすべてのランプを見つけます（目の損傷を避けるため）。
   if (Integer.parseInt(mmc.getProperty("lca", "Lamp " + (i + 1) + " connected")) == 1)
   {
       wavelength = Double.parseDouble(mmc.getProperty("lca", "Lamp " + (i + 1) + " peak wavelength"));
       if ((wavelength >= 400.0) && (wavelength <= 700.0))
       {
           gui.message("Lamp " + (i + 1) + " has a peak wavelength of " + wavelength + " nm and will be tested.");
           lamps[lampIndex] = i + 1;
           lampIndex++;
       }
       else
       {
           gui.message("Lamp " + (i + 1) + " has a peak wavelength of " + wavelength + " nm and will NOT be tested.");
       }

       //また、一度に１つずつ使用するため、すべての強度を０に設定します。
       mmc.setProperty("lca", "Lamp " + (i + 1) + " intensity", 0.0);
   }
   else
```

```
    {
        gui.message("Lamp " + (i + 1) + " is not connected.");
    }
}

// 次に、各ランプを１つずつオンにします。
numLamps = lampIndex;
gui.message("Now testing " + numLamps + " lamps for two seconds each.");

for (int i = 0; i < numLamps; i++)
{
    int lampNumber = lamps[i];
    gui.message("Lamp " + lampNumber);
    mmc.setProperty("lca", "Lamp " + lampNumber + " intensity", 100.0);
    mmc.setShutterOpen("lca", true);
    mmc.sleep(2000);
    mmc.setShutterOpen("lca", false);
    mmc.setProperty("lca", "Lamp " + lampNumber + " intensity", 0.0);
}

// Make sure all lamps are off.
mmc.setShutterOpen("lca", false);

mmc.unloadAllDevices();

gui.message("Done.");
```