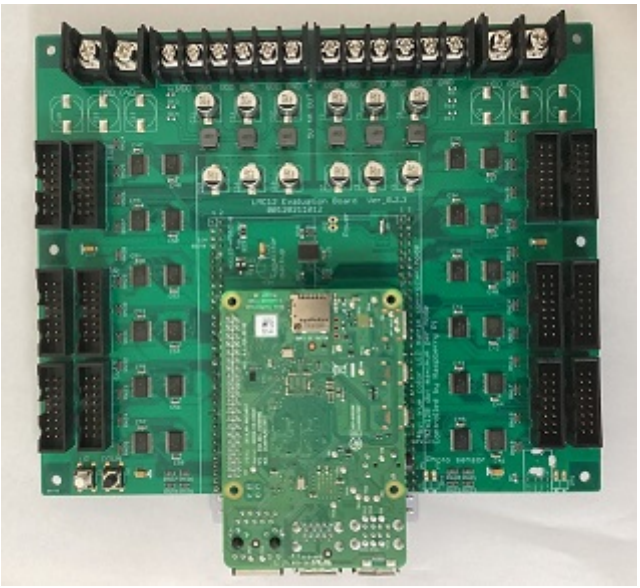
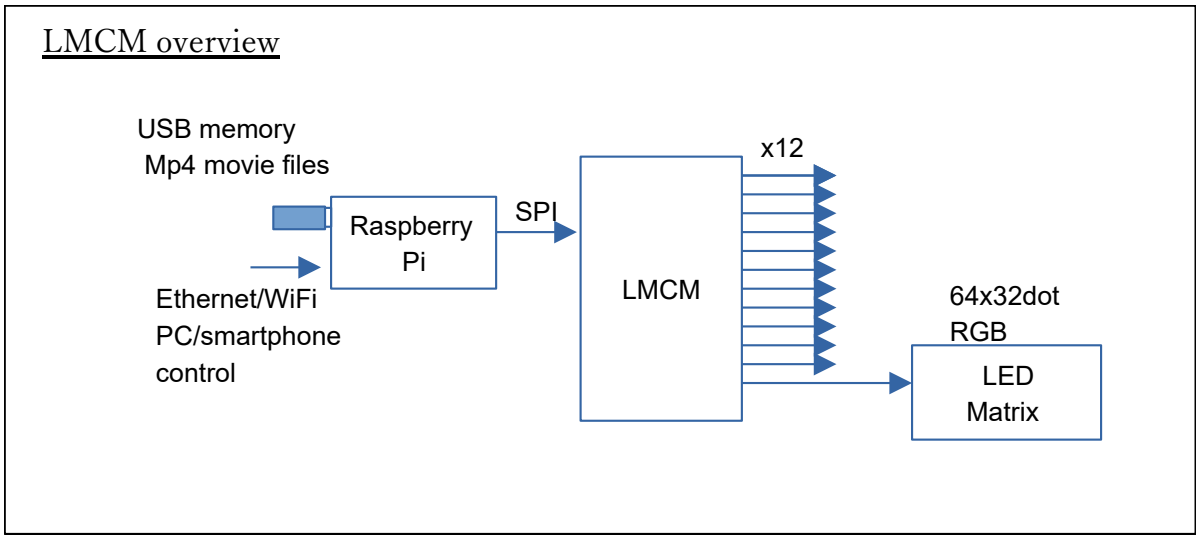
LMCM(LED matrix controller multinode)

LMCMとは



　LMCMは、RaspberryPiを中核にして、かつ安価なLEDマトリックスを大量に使用して、自然な色彩表現の動画表示を可能にするシステムです。画面サイズはネットワークを用いて無制限に拡大が可能です。



　表示する画像には、基本的にはUSBメモリ上のMP4などの動画ファイルを使用します。解像度は2K程度までであれば特に制限はなく、LEDマトリックスの画面サイズに合わせて自動補正します。USBメモリはRaspberryPiのUSBポートに挿入すればデフォルトで自動再生します。

　それ以外に、WindowsPCからの画面のリアルタイムキャプチャ機能も別に用意しています。

LMCMシステムの構成

LEDマトリックス

　市販のLEDマトリックスは、大量のRGBのLEDを縦横に並べたもので、HUB-75と呼ばれる信号インターフェース等で画像データを受け取り、そのデータに応じてLEDを点灯します。
　安価なLEDマトリックスでは、シフトレジスタ定電流ICを使用しており、RGBごとに点灯のONOFFの制御のみを行いますが、制御次第でフルカラー階調表現も可能です。

LMCM

　LMCMは、当社が開発したLEDマトリックス制御ボードです。RaspberryPiを上に乗せるための接続コネクタと、１２本のLEDマトリックスへの信号コネクタが装着されています。それぞれのHUB-75信号コネクタには最大で64x32ドットのフルカラーLEDマトリックスを接続できます。全体としては192x128ドットの完全な24bitRGBフルカラー制御が可能です。
　LMCMは、RaspberyrPiから60fpsごとにSPI信号経由で送られてきたビットマップ画像データを、LEDマトリックスで表示します。

RaspberryPi

　Raspberry Piは市販の安価なLinux制御ボードです。LMCMの上に乗せる形で直接接続します。
　LMCMシステムでは、このRaspberryPi上で動作する専用コマンドラインアプリを開発しました。RaspberryPiのファイルシステム上にあるMP4などの動画ファイル、あるいはUSBメモリに内蔵された動画ファイルを、リアルタイムデコードしてLMCMで表示します。さらにWindowsPCの画像キャプチャをそのままリアルタイム表示する機能もあります。
　LMCMの制御機能とは別に、Pythonで記述されたCommandServerを作成しました。PCやスマホからの表示制御をEthernet,WiFiで受信して、LEDMultiControlを必要に応じて呼び出すことにより、LMCMのファイル再生、停止などの制御などを行うことができます。スマホからの任意のテキストデータをその都度表示することも可能です。

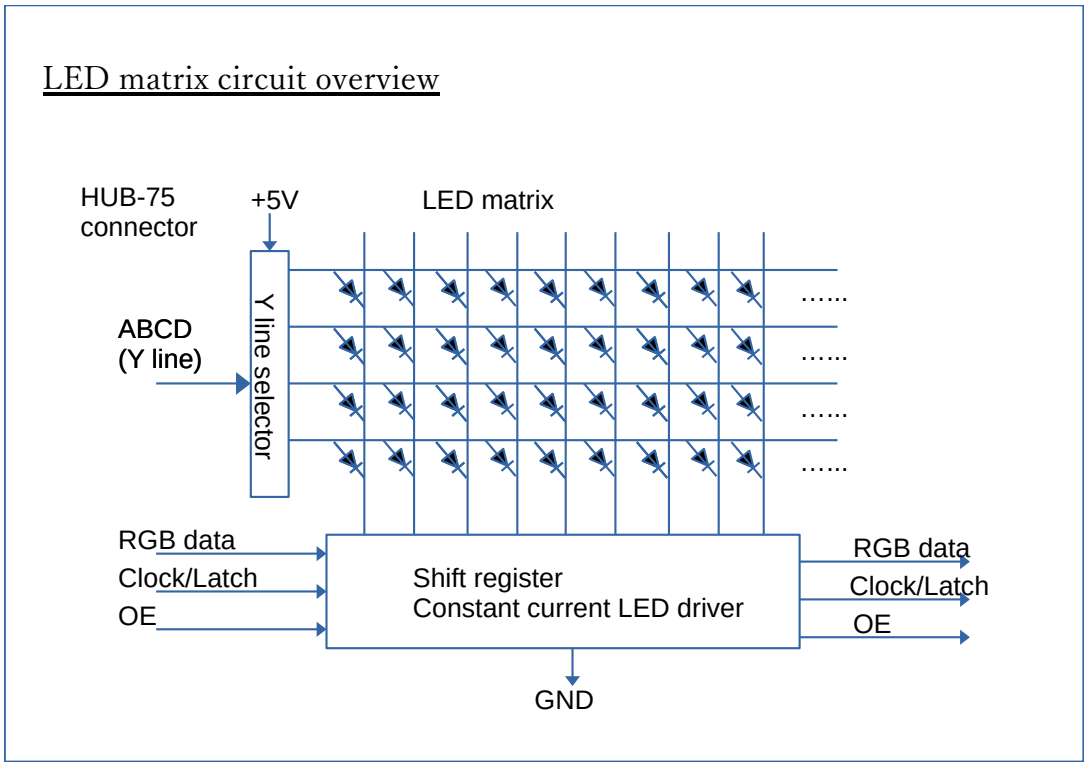
LMCMマルチコントロール

　LMCM＋RaspberryPiを複数使用して、あたかも１つの大画面であるかのように同期して連携表示することが可能です。
　複数のRaspberryPiはLANケーブルで接続します。それぞれのLMCMは大画面上で自由に配置することができます。マルチコントロールLMCMは単一LMCMと同じくビットマップ画像出力、MP4などの動画再生、キャプチャ動作などに対応します。

LEDマトリックス

LEDマトリックスの構造

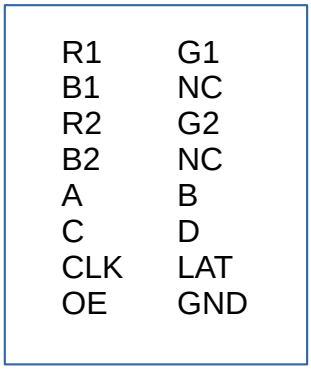
市販のLEDマトリックスは、RGBのLEDが大量に敷き詰められていて、それぞれ個別に点灯制御ができます。大量に並んだ部品がLEDです。左にあるのがYライン（スキャンライン）選択回路、下にあるのがシフトレジスタ定電流LEDドライバICとなります。



市販の安価なLEDマトリックスは、基本的にはLEDの点灯のONOFFだけを行います。主に単色の案内板などを想定した構成です。

LEDマトリックス自体がPWM制御の階調表現を行うことができる製品もありますが、高価で高性能なICを使用しているため、LEDマトリックスを大量に使用するサイネージではコスト高になります。

LED matrix HUB-75 interface



　このHUB-75インターフェースでは、LEDドライバICにRGBデータを時間ごとに１つづつずらして順に送り込みます。データを１つずらすごとにクロック信号のON/OFFを繰り返します。そして、転送が終わった時点でLatch信号を短時間だけONにして記憶させます。

　記憶されたRGBデータを用いてLEDを点灯するにはOE信号をONにします。

　LEDマトリックスはLEDドライバーICを複数のラインで共有するため、それぞれのラインを見えない速度で時間差でずらして表示します。そのための回路がYライン選択回路で、このYラインの選択をA,B,C,D端子で行います。

LMCMの使用方法

LEDMultiControlWin(Windows上のコンソールアプリ）

　LMCMをPCから直接接続して制御するには、LMCMとPCをUSB typeCケーブルで接続します。そのために、"LEDMultiControlWin.exe”をWindows用のコンソールアプリとして作成しました。

　これを用いて、以下のコマンドでWindowsの画面の一部を切り出してリアルタイム表示できます。例ではX軸が画面の50%の位置、Ｙ軸が画面の50%の位置からWidth10%、Height30%で表示を行います。

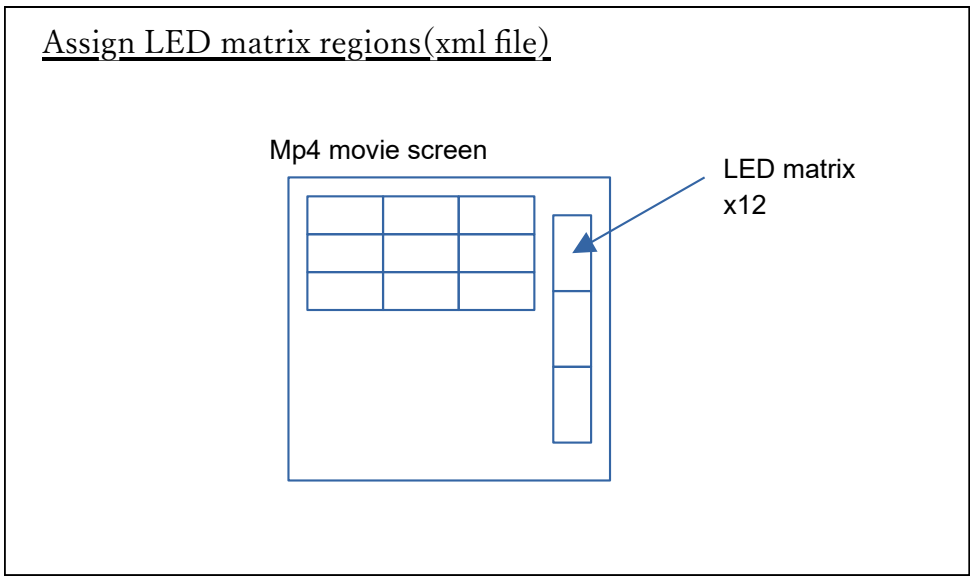
```
(Win)LEDMultiControlWin.exe -ca -tLMC_FC_192x128.ltb -cXR50 -cYR50 -cWR10 -cHR30  -ab255
```

　絶対座標の指定も可能です。以下の例ではX,Y=(400,200)から(128,192)の画面をキャプチャします。この場合、実際の解像度に対して画面の縮小が適用されている場合は縮小後の座標で適用されます。

```
(Win)LEDMultiControlWin.exe -ca -tLMC_FC_192x128.ltb -cX400 -cY200 -cW128 -cHR192  -ab255
```

LEDマトリックスの配置ファイル

　表示したい画像に対してLEDマトリックスを割り当てるには、以下のようなxmlファイルを使用します。



　動画などの画面に対するLEDMatrix端子の配置はMatrixLaneタグで指定します。さらに個別の座標、90度単位の回転などを指定できます。
ファイルの最後で、LEDマトリックスの種類による仕様の個体差を吸収するための設定が可能です。

```
<?xml version="1.0" encoding="UTF-8"?>
<TEMPLATE>
<VERSION>4</VERSION>
<COMMENT>LMCM Single control Horizontal 192x128dots 12lane </COMMENT>
<TILE>
        <ID>1</ID>
        <TileType>1</TileType>
        <MatrixLane>0</MatrixLane>
        <TileDirection>90</TileDirection>
        <TileLocation_X>0</TileLocation_X>
        <TileLocation_Y>64</TileLocation_Y>
        <TileLocation_W>64</TileLocation_W>
        <TileLocation_H>32</TileLocation_H>
        <TileDotPitch>0</TileDotPitch>
        <TileColorType>0</TileColorType>
</TILE>
 ...（省略）...
<LOCAL>
        <XMin>0</XMin>
        <XMax>192</XMax>
        <YMin>0</YMin>
        <YMax>128</YMax>
        <MatrixScanlines>8</MatrixScanlines>
        <MatrixColors>3</MatrixColors>
        <MatrixLayoutMode>1</MatrixLayoutMode>
        <MatrixColorMode>1</MatrixColorMode>
        <MatrixYLineDecoded>0</MatrixYLineDecoded>
        <MatrixEnablePolarity>0</MatrixEnablePolarity>
        <MatrixDataPolarity>0</MatrixDataPolarity>
        <MatrixClockPolarity>0</MatrixClockPolarity>
        <LMCTilesTransferMode>1</LMCTilesTransferMode>
</LOCAL>
</TEMPLATE>
```

LEDMultiControl(RaspberryPi上のアプリ）

　LMCMをRaspberryPiから制御するには、独自開発した"LEDMultiControl"というLinuxコンソールアプリを使用します。このアプリはC++で記述されており、Linuxのコマンドラインで起動できます。
　mp4などの動画デコードはffmpegをDLLライブラリとして使用しています。PC等からLANケーブルを通してリアルタイム転送されたビットマップデータを同期再生することも可能です。この機能を用いてPCの画面キャプチャをLEDマトリックスでリアルタイム再生することが可能です。

ビットマップ静止画、スクロール表示

　ビットマップファイルをそのまま表示します。横スクロール、縦スクロールなどの指定も可能です。

　"LMC_FC_192x128.dat"は配置を示すxmlファイルです。

　pythonスクリプトの補助により、JPG画像や文章をその都度ビットマップに描画して表示することも可能になります。

```
>LEDMultiControl -pLMC_FC_192x128.dat Image.bmp -sc
```

MP4動画表示

　LEDMultiControlに対して、MP4ファイルそのものをパラメータとして与えます。
　画像はLEDマトリックスのサイズに合わせて縮小、トリミングされます。トリミングのポリシーは縦優先、横優先を切り替えることができます。実寸表示で座標指定も可能です。
　音声も同時にデコードしており、RaspberryPi4のオーディオジャックから音声出力されています。

```
>LEDMultiControl -pLMC_FC_192x128.dat -mo SampleMovie.mp4
```

WindowsPCリアルタイムキャプチャ表示

　WindowsPCの”LEDMultiControlWin.exe”を用いて、画面の一部分をRaspberryPiを経由してLMCMのLEDマトリックス画面に縮小表示することが可能です。
　WindowsPCの画面キャプチャアプリを使用する場合は、RaspberryPi側はLEDMultiControlをスレーブモードとして動作させておきます。

```
>LEDMultiControl -s -pLMC_FC_192x128.dat
```

　この状態でWindows上でコマンドを実行します。オプションではキャプチャ領域や輝度も指定しています。

```
(Win)LEDMultiControlWin.exe -ca -m -tLMC_FC_192x128.ltb -cXR50 -cYR50 -cWR10 -cHR30  -ab255
```

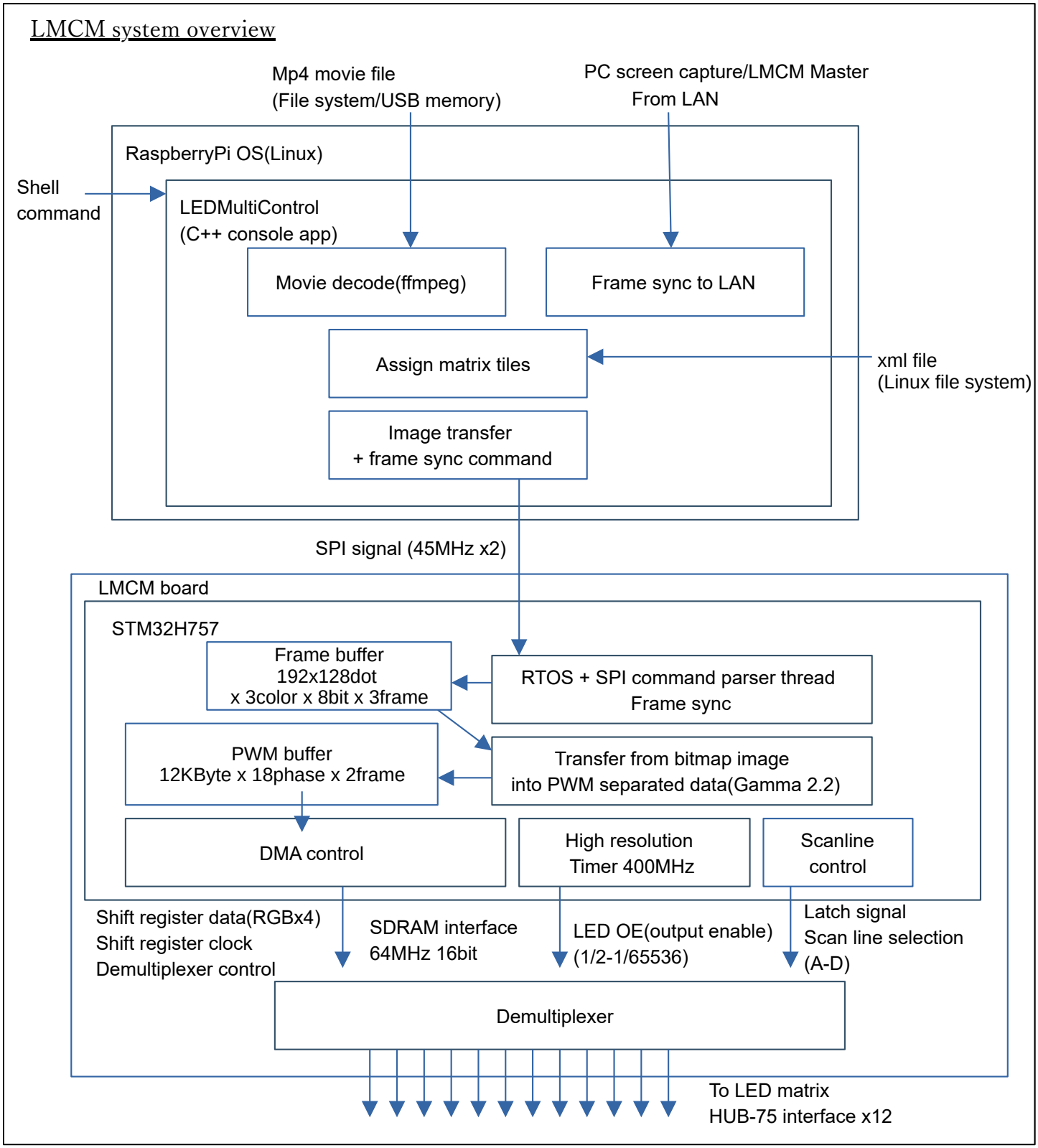
　Windows上の"LEDMultiControlWin.exe”はコマンドプロンプトやスクリプトでも実行できますが、C#で記述された専用Windows GUI操作アプリ"NeCoWin"を別に用意しています。
　キャプチャする画像は、Windowsのキャプチャ領域とLEDマトリックスのサイズに応じて自動縮小されます。

RaspberryPi上のPythonスクリプト制御

　LEDMultiContorlはLinux上のコンソールアプリであり、上位のシェル、Pythonスクリプトなどで自在に自動制御が可能になります。
　LMCシステムは、RaspberryPi上にCommandServerと呼ぶPythonスクリプトを用意しています。このスクリプトは複数画像ファイルの連続再生、ＰＣ，スマホなどからのWiFi制御に対応する機能があります。もちろんユーザーが独自にスクリプトを作成することも可能です。

LMCMシステム全体構成

LMCM+RaspberryPiのシステムは以下の図のように構成されています。



LEDMultiControl(RaspberryPi上)

LEDMultiContorlは、C++で記述されたLinuxコンソールアプリです。主な絹は以下の通りです。

- USBメモリやLinuxファイルシステムからMP4などのデータをロードする
  あるいはLAN経由で到達した画像データを受信する。同時にタイミング同期を行う。
- LEDマトリックスの配置に従って再配置を行う。
  同時にLEDの自動輝度制御も行う。
- SPIインターフェースでLMCMのCPUに対してビットマップ画像と同期信号を送る。

ffmpegデコーダ

ffmpegをDLLとして使用してMP4動画等をデコードします。デコードされたフレームごとにフレームバッファにロードします。ffmpegのフレーム数は不定であるため、転送は60fps未満が基本になります。

音声も同時にデーコーダからオーディオインターフェースに送信しています。

ネットワークインターフェース

スレーブモードのLEDMultiControlは、TCP,UDPポートをそれぞれ1つづつ使用してネットワーク受信を行います。TCPポートは画像データを受信してフレームバッファに順次格納します。UDPポートはUDPのリアルタイム性能を利用してフレーム間のタイミング同期に用いられます。

LEDマトリックスタイルの配置

１フレームごとに実行されます。XMLファイルに記述されたLEDマトリックスのタイル配置に従い、元画像からLMCMに適合したビットマップ画像へと変換します。LMCMに適合したビットマップ画像とは、縦に１２列のLEDマトリックスが配置されている形となります。

自動輝度制御機能

LEDマトリックスは画像によって消費電力が激しく変動します。とくに完全な白色の場合の電力は巨大すぎるので、この電力の最大値に合わせた出力の大きな電源回路が必要になります。

そのため、LMCMには、全体の電流が指定したしきい値を超えた時に、しきい値以下になるように全体の輝度を落とすという機能があります。この制御はタイル配置の時に同時に行っています。

SPI送信

LMCMの配置に変換されたビットマップ画像をSPI通信でLMCMに送信します。SPIは2ポートを同時に使用可能で、それぞれ43MHzで送信を行っています。フレームの起点タイミングを示す同期信号もタイミングを合わせて送信しています。

USB送信

　LMCMの配置に変換されたビットマップ画像をUSB通信でLMCMに送信します。USB2.0以上が必要です。

LMCM回路基板

LMCMの回路基板の構成

　LMCMの回路基板は主に、STMicroElectronics社のARM CortexM7ベースの組み込みCPUであるSTM32H757と、個別ロジックICから構成されたデマルチプレクサ回路とで構成されています。

　制御CPU(STM32H757)内部

- RaspberryPiからSPI通信でフレームバッファのデータを受けとる。
  疑似PLLによるフレーム同期も行っている。
- フレームバッファを展開してシフトレジスタのPWMデータに変換する。
- DMAを使用してデマルチプレクサ回路にPWMデータを高速転送する。
- LEDマトリックスに対してスキャンライン制御信号等を送る。
  特にOE信号は高精度タイマーを用いて正確な輝度制御を行う。

　デマルチプレクサ回路

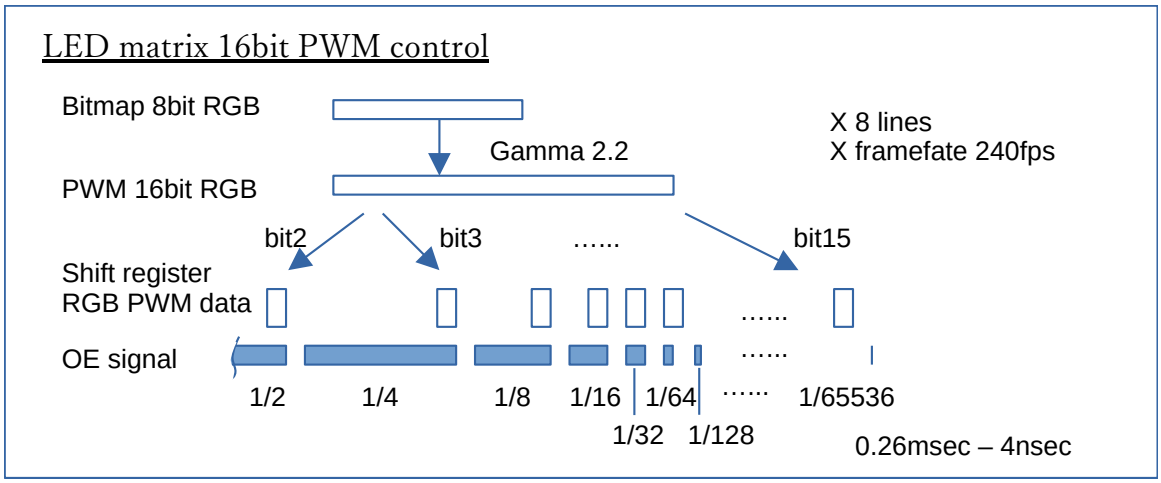
- デマルチプレクサ回路でPWMデータを12本のLEDマトリックス端子に対して分配する。

ソフトウェア疑似PLLによる同期

　LEDのフレームチェンジのタイミングをRaspberyPiのSPI信号に対してソフトウェア疑似PLLによって同期を行っています。同期の目的は、２つのPWMバッファが共に使用中で、次のフレームのロードに失敗するという状態を避けるためです。逆に、同期がなければPWMバッファが３つ以上必要になります。

PWM色調制御

　階調のないLEDマトリックスでの階調表現を可能にするPWM制御について説明します。



LEDマトリックスの点灯制御を行うOE信号を1/65536単位で高精度タイマーで制御します。LMCMで使用している高精度タイマーは400MHzで動作するため、2.5usecの精度でのパルスが出力可能です。

- 8bit×3のRGBデータを、ガンマ補正で16bit×3に拡張する。
  ガンマ値は2.2を採用しているが、ソフトウェアで変更も可能。
- 16bitデータを転送に適したPWMデータに変換する。
- LEDマトリックスのシフトレジスタにPWMデータを送り込む。
- OE信号をOFFにしたところで、LAT信号を送って転送済みのPWMデータに切り替える。
- OE信号をONにすることでLED点灯を行う。
- 点灯中に同時に次のPWMデータを送り込む。
- OE信号をOFFにしたところで、LAT信号を送って転送済みのPWMデータに切り替える。
- 次の区間のOE信号をONにしてLEDを点灯する。次のLED点灯時間は前の半分になる。

1/2、1/4、1/8、1/16、1/32、...の16種類のPWMデータを送り込み、それぞれ1/2、1/4、1/8、...の時間だけ点灯させています。この方法で16回のデータ転送で16ビットの完全な階調表現が可能になります。さらに、LMCMのPWM制御はその誤差を16ビットの量子誤差以下、すなわち65536分の1以下に抑え込むことに成功しています。そのため、LMCMの色彩表現力は液晶等と比較しても優位性があり、体感的に実際の解像度以上の表現力があります。

実際の制御はピーク電流を分散させること、OEの消灯期間を最小限にするなどの工夫があるのでもう少し複雑なものとなっています。

このPWMデータ変換は特に処理性能が必要であるため、コアコードは高速化のためにARMのアセンブラを使用しています。

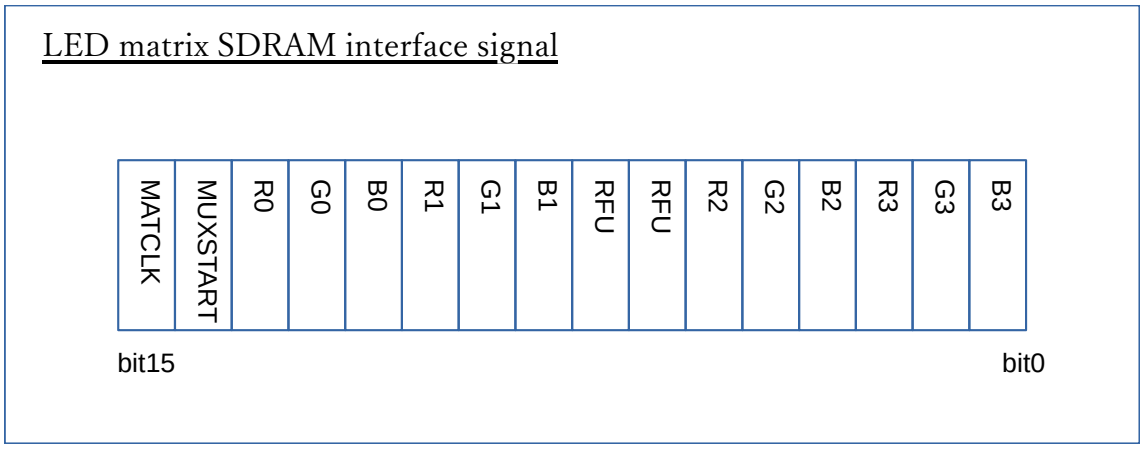
高速スキャンライン制御

LEDマトリックスでは、すべてのLEDを同時に点灯制御するのではなくて、スキャンラインと呼ばれるラインごとにLED制御ICを共有して、高速で切り替えて表示しています。スキャンラインは4から16程度が一般的です。スキャンライン数はLEDマトリックスの種類に応じてソフトウェアで切替が

可能です。
　スキャンラインの切替速度が低いと画像のちらつきを感じるため、少なくとも60fps以上の速度で一画面分のスキャンを終了する必要があります。LMCMは基本的に240fps、高輝度部分では疑似的に480fpsのスキャンライン更新を実現しました。

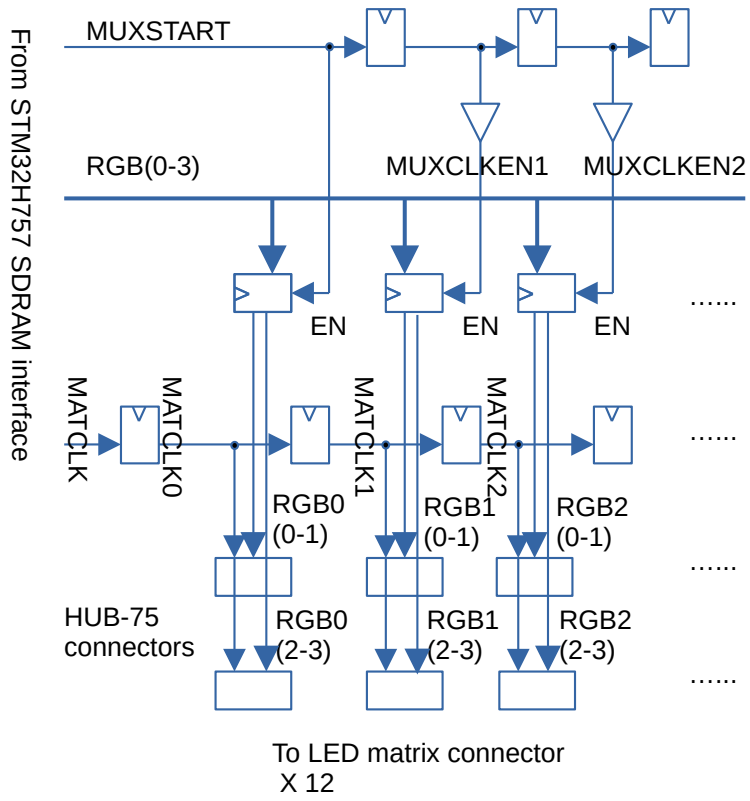
デマルチプレクサ回路

　LMCMは、STM32H757から高速出力されたPWMデータを、12本のLEDマトリックス端子へと分配伝送します。そのために、PWMデータは16ビットのSDRAMインターフェースを用いて64MHzで出力します。この16ビットデータを分配するために、デマルチプレクサと呼ぶ回路を作成しています。
　分配する16ビットデータの内訳は以下の通りです。RGB0,1がLEDマトリックス端子の前半6本、RGB2,3が後半6本分に相当し、それぞれを時間ごとにさらに6分割します。つまりSDRAMへは常に6ワード単位で送信を行います。



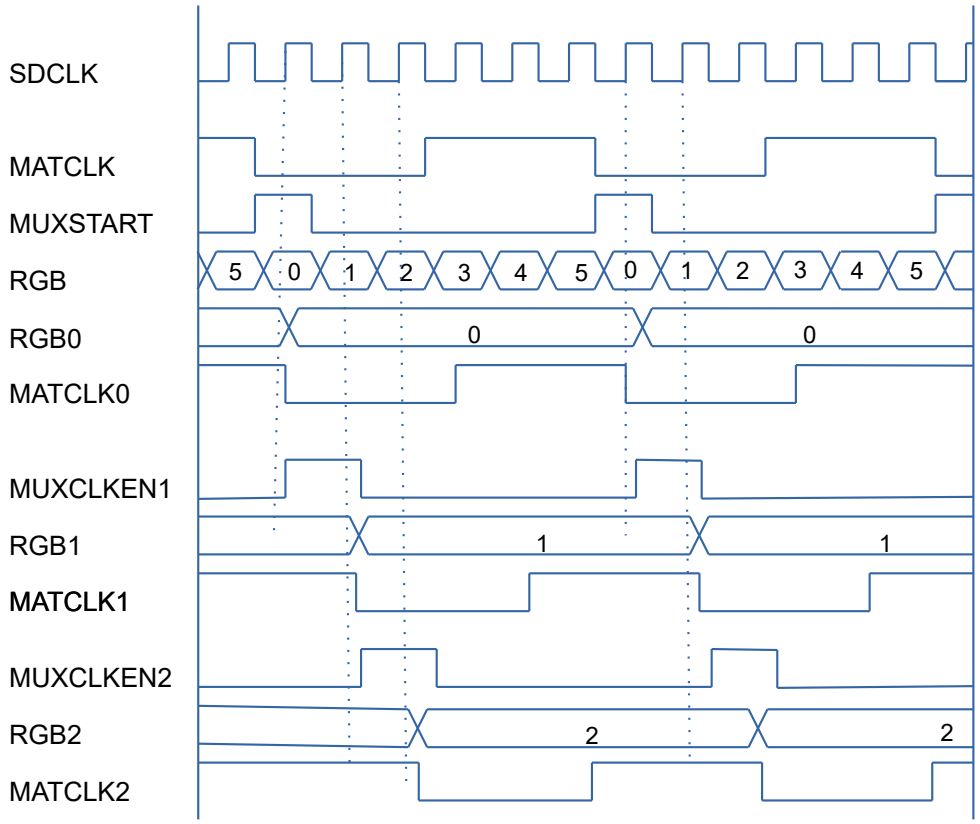
デマルチプレクサ回路の構成は以下のとおりです。図では6本のLEDマトリックス端子だけですが、実際は12本の分配を行っています。

Circuit diagram of RGB signal demultiplexer



デマルチプレクサの回路の信号タイミングは以下のようになります。

Timing chart of RGB data demultiplexer



CPUが生成するMUXSTART信号から、シフトレジスタで１クロックごとにずれたパルス状のMUXCLKENx信号を5本生成します。RGB信号分配用の大量のフリップフロップは、このMUXCLKENx信号がONの時だけRGB信号を受信します。この結果、6クロックごとに6系統のRGBフリップフロップにそれぞれデータが格納されます。

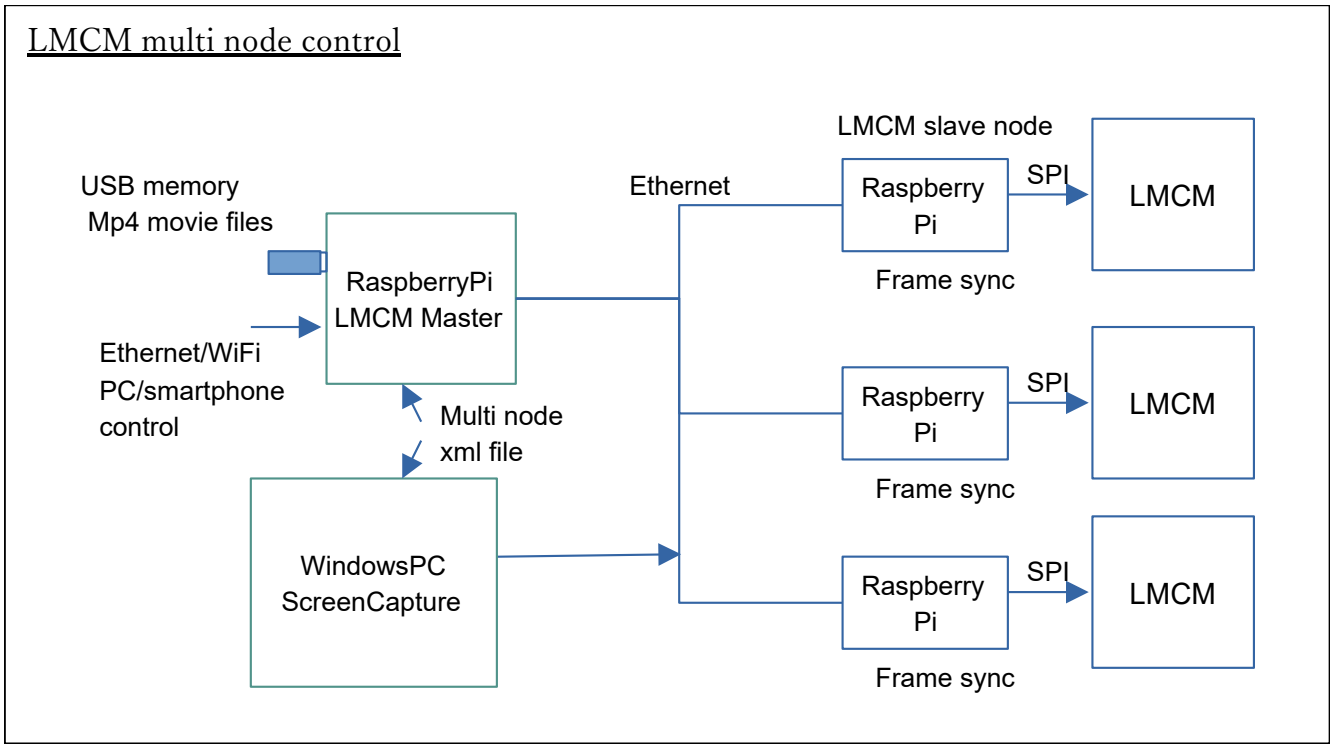
MATCLKはLEDマトリックスのシフトレジスタに直接入力するためのクロックです。MATCLKそのものはCPUが生成しますが、マトリックスのレーンごとに位相をずらす必要があるため、シフトレジスタを用いています。

この結果、それぞれのLEDマトリックスに分配される信号の周波数は10.6MHzとなります。

LMCMマルチノード制御

マルチノード構成

LAN接続によってLMCMシステムを複数連携して１つの画面として表示することが可能です。



LMCMのスレーブに対して、そのつど画像データを分割してLANで伝送して、それぞれ同期表示を行います。つまり、大画面の動画を複数のLMCMが連携して表示することができます。

設定XMLファイルにはスレーブに割り当てられたIPアドレスが記録されており、それぞれのIPアドレスに対して伝送するため、RaspberryPiやネットワークの性能の限りノード数を増加できます。

RaspberryPiマスター

RaspberryPiのコンソールアプリの"LEDMultiControl"をマスターモードで起動すると、複数のスレーブへの通信が可能になります。この際にマルチノードの設定ファイルを使用します。以下の例では、
”LMC_MC2_256x192.ltb”
となります。

```
>LEDMultiControl -m -mo -TLMC_MC2_256x192.ltb SampleMovie.mp4
```

WindowsPCマスター

キャプチャ用アプリ"LEDMultiControlWin.exe”をマスターモードで起動します。

```
(Win)LEDMultiControlWin.exe -m -ca -tLMC_MC2_256x192.ltb -cXR50 -cYR50 -cWR10 -cHR30  -ab255
```

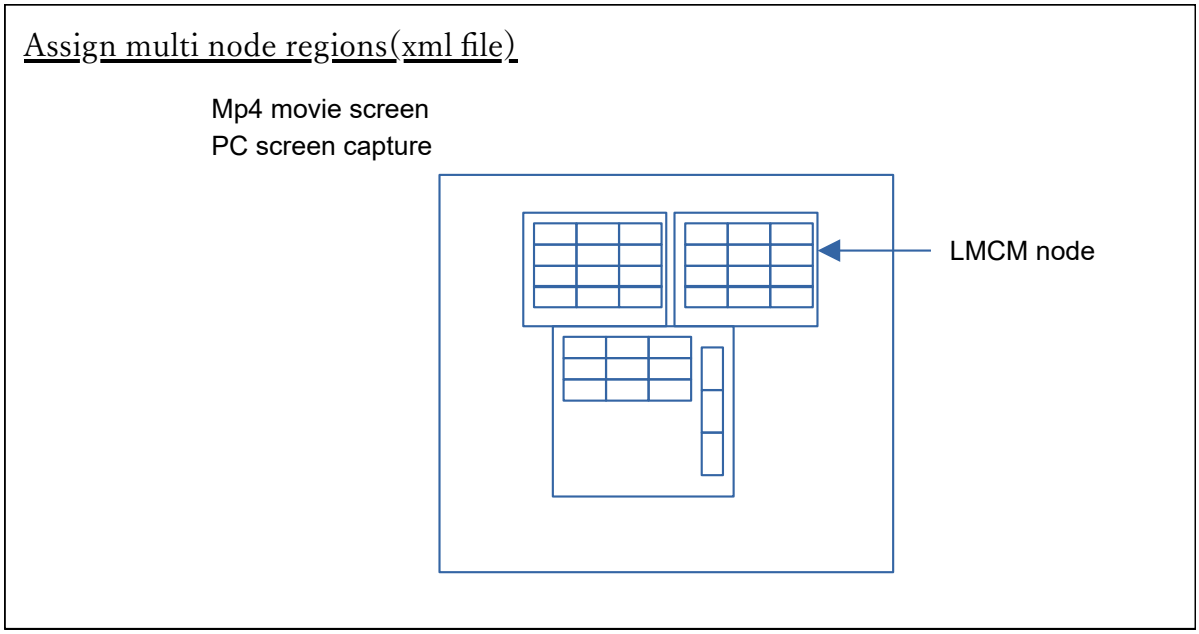
LMCMスレーブ

RaspberyPi上の設定ファイルにスレーブIDを書き込んでおきます。

マスター側のRaspberryPi上のPythonスクリプトは、起動時にネットワーク上のスレーブを自動検索して、スレーブIDに対して発見されたIPアドレスをそのつど連結してテンポラリファイルに格納します（WindowsPCでも同じ機能があります）。

スレーブはマスターから受信したフレームに対してタイミングを同期させて表示します。ネットワーク経由の表示なので転送に遅延が発生することになりますが、表示までのレイテンシフレーム数を指定して統一させることも可能です。

マルチノード設定ファイル例



以下は設定XMLファイルの一例です。大画面に対する座標、90度単位の回転などを指定できます。設定ファイルのIPアドレスは強制指定もできますが、ネットワーク自動スキャンによるIDからの自動割り当ても可能にしました。LinuxあるいはWindowsのPythonスクリプトで起動ごとにXMLファイルを生成しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<TEMPLATE>
<VERSION>4</VERSION>
<COMMENT>Block2x1 Horizontal 256x192 </COMMENT>
<BLOCK>
        <ID>1</ID>
        <BlockType>0</BlockType>
        <BlockDirection>0</BlockDirection>
        <BlockLocation_X>128</BlockLocation_X>
        <BlockLocation_Y>0</BlockLocation_Y>
        <BlockLocation_W>128</BlockLocation_W>
        <BlockLocation_H>192</BlockLocation_H>
        <BlockDotPitch>0</BlockDotPitch>
        <BlockColorType>0</BlockColorType>
        <BlockIP>192.168.43.201</BlockIP>
</BLOCK>

<BLOCK>
        <ID>2</ID>
        <BlockType>0</BlockType>
        <BlockDirection>0</BlockDirection>
        <BlockLocation_X>0</BlockLocation_X>
        <BlockLocation_Y>0</BlockLocation_Y>
        <BlockLocation_W>128</BlockLocation_W>
        <BlockLocation_H>192</BlockLocation_H>
        <BlockDotPitch>0</BlockDotPitch>
        <BlockColorType>0</BlockColorType>
        <BlockIP>192.168.43.202</BlockIP>
</BLOCK>

</TEMPLATE>
```

HDMI、DisplayPort入力

　HDMIやDisplayPortから複数のLMCMへの直接表示は原理的には可能ですが、専用のHDMItoLAN信号変換システムが必要になります。これについては現在検討中です。