LabVIEW 概要

（最終改訂　2023/03/27）

LabVIEW とは

LabVIEW とは、通常の言語の関数に相当する Virtual Instruments（以下「VI」と表記）を配線・結合することでプログラムを構成する National Instruments 社（以下「NI」と表記）の計測制御システム開発用のグラフィカルなデータフロー型言語で、本学では 2005 年に LabVIEW7.1 が旧情報工学科の情報工学実験のためにキャンパスライセンスで導入された。LabVIEW には主要な計測器のデバイスドライバが対応し、計測器の設定制御とデータ取得をリアルタイムに行うのが本来の利用形態であるが、特定の計測器が無くとも簡単な A/D 変換器を用いて（前期課題 3「LabVIEW 計測」）、更に PC 単独で（前期課題 4「論理回路設計」）基礎的な学修課題をこなすことができる。

LV_sample.zip には、本稿（拡張子**.pdf**）、本稿掲載図（フォルダ **Figures**：拡張子**.png**）、本稿で例に挙げた 14 例のサンプル VI（サブ VI を含めソースファイル 37 個：拡張子**.vi**、実行ファイル 14 個：拡張子**.exe**、構成ファイル 14 個：拡張子**.ini**）、必要な実行時ライブラリ（フォルダ **data**：拡張子**.dll**）他が含まれており、解凍して実行することを推奨する（p.29 備考参照）。ソースファイル、実行ファイルおよび実行時ライブラリは、2022 年前期まで旧情報科学研究教育センター（現「工手の泉」）にインストールされていた **LabVIEW2017 32b 版**のものである。

LabVIEW では計測制御プログラミングで必要な各データ型の演算をブロックダイアグラム上でアイコン表示（p. 3 図 2 以降ではスペース節約のためプロパティでアイコン表示のチェックを外している）される制御器、表示器と関数部品 VI を結合（配線）して行うが、部品も配線もデータ型によって**ブール値は緑**、**文字はマゼンタ**、**整数は青**、**浮動小数点数は橙**と色分けされ、スカラーと配列では（配列も次元によって）配線の太さを変えている。これにより、入力、出力の部品、配線に同じ色と太さを対応させることで、型の不一致による誤りがすぐに判る様になっている。

1. Mouse_KB.vi

リアルタイムデータ取得の最も単純な例として、ライブラリの VI を呼び出してクリックされたマウスボタンと押下したキーのコードとラベル（註および備考 p.31 参照）、マウスポインタの位置とスクロール量を表示する **Mouse_KB.vi** のブロックダイアグラムを p.2 図 1 左に示す。NI のライブラリ VI（パレットは「接続」→「入力デバイス制御」→「キーボードを初期化」/「マウスを初期化」/「入力データを取得」/「入力デバイスを閉じる」）ではマウスボタンは 4 個（5 ボタン以上のマウスのサイドボタン等には対応しない）、キーは同時に 6 個（これを超える場合はコードの昇順で 6 個）までの情報を取得できる。

註：NI のキー情報取得 VI では **US キーボードのキー**に 0～104 のコードを割付け、ラベルを列挙体定数で対応させている。コードは、TTY 数字 0～9【0～9】、英字キーA～Z【10～35】、ファンクションキーF1～F15【36～50】と連続し、これ以降は最後の 2 個を除き**ラベルの辞書式順序**に従い ADD【51】～ UP【102】、最後は ENTER (num pad)【103】、'(QUOTE)【104】となっている。テンキー数字は NUMPAD_0【76】～ NUMPAD_9【85】である。キーコードが定義されていない PrtSc キー、US キーボードには無いキー（¥）、US キーボードではシフト側にあるキー （^、@、: ）ではキーの押下を検出できない。

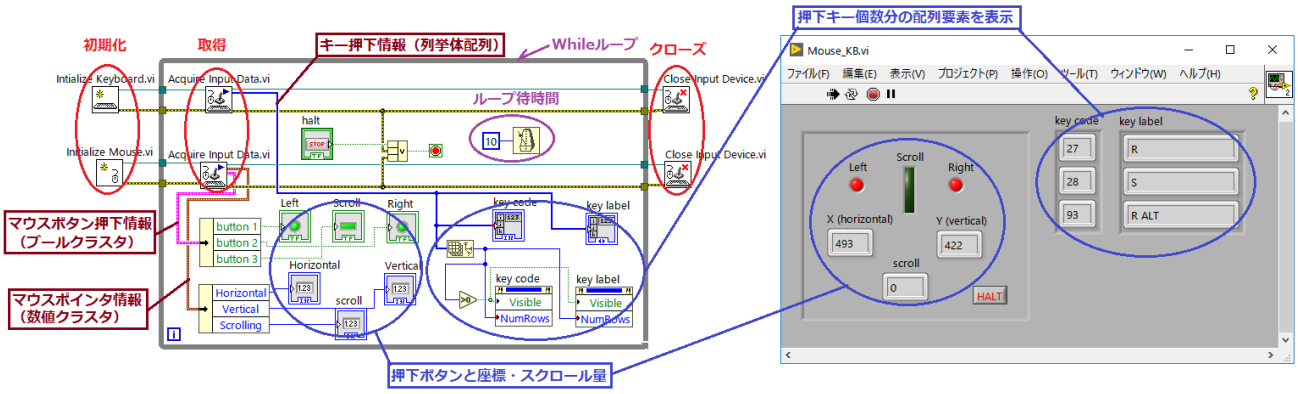

図 1　マウス・キーボード入力 VI のブロックダイアグラム（左）と実行中のフロントパネル（右）

LabVIEW では制御器、表示器を右クリック「作成→プロパティノード→」でプロパティ項目を指定することで**各プロパティを（p.9 図 11 の例の様に他の While ループからでも）実行時に変更（書込み）、参照（読取り）**することが可能である。ブロックダイアグラム（図 1 左）では情報取得と表示の部分を While ループ（ループ待時間を 10ms に設定）の中に入れ、表示器 “key code”、“key label”のプロパティノードで表示（Visible）、行数（NumRows）の**プロパティを実行時に変更**し、押下された個数分（キーコードの昇順で 6 個まで）のキー情報を表示する。図 1 右はソース VI フロントパネルの「実行」ボタン（）をクリックして実行中に、左右のマウスボタンと R, S 両キーを押下しながら Alt+PrtSc でキャプチャしたフロントパネル画面である。

2.　RS_FF.vi　～ フィードバック ～

前期課題 4「論理回路設計」で学修する様に、組合せ回路は単純に各論理ゲートに対応する LabVIEW の関数部品 VI の出力を他の関数部品 VI の入力に配線することにより論理関数をそのまま回路として実現できる。これに対し出力が入力側にフィードバックする順序回路では同じ入力に対しても内部状態（過去の入力履歴による）に出力が依存するため、LabVIEW でゲート図の通りに出力を入力側に直接配線したのでは関数として値が一意に定まらないためにエラーとなる。LabVIEW では、While ループ 1 回分だけ遅延させる（While ループの無い VI では次の実行回に送る）「フィードバックノード」（p. 3 図 2 中の矢印ユニット）を挿入することで順序回路を実現でき、ここでは最も単純な順序回路としてフリップフロップ（高速メモリ SRAM の実体：以下「FF」と表記）を例として取り上げる。

RS 型（SR 型）FF は安定な状態が 2 つある対称な順序回路で、Set、Reset の何れか一方を ON（論理値 1）にして設定した状態が、両方 OFF（論理値 0）で保持され 1 ビットの情報を記憶するもので、Set/Reset の同時 ON は使わない。実際の回路では完全に対称（素子の物理特性が同じ）であることはあり得ず、（素子の特性がごく近い場合に、何れのゲートの出力が 1 となるかが確率的であるとしても）発振することはないが、**理論的には**電源投入時に両方 1 両方 0 の間で振動（発振）する。

LabVIEW の制御器であるフロントパネル上の押しボタン、トグルスイッチ、スライドは何れもマウスボタンにより操作し、1 操作で設定できる値は 1 個である。**RS_FF.vi** は、**Nand ゲート、Nor ゲートによる RS 型（SR 型）FF** に前項で使用した NI のマウス・キー情報取得 VI を用いて、Nand 型では左右のマウスボタン、Nor 型ではキーボードの S（キーコード 28）と R（キーコード 27）の押下を検出してそれぞれ Set と Reset の入力とすることで **Set/Reset 同時 ON/OFF のシミュレーション**を可能にしたものである。

図 2 に **RS_FF.vi** のブロックダイアグラム（FF 本体は図の網掛けの部分で、ブロックダイアグラムの殆どをマウスボタンとキーボード情報識別の入力インタフェースが占める）を示す。

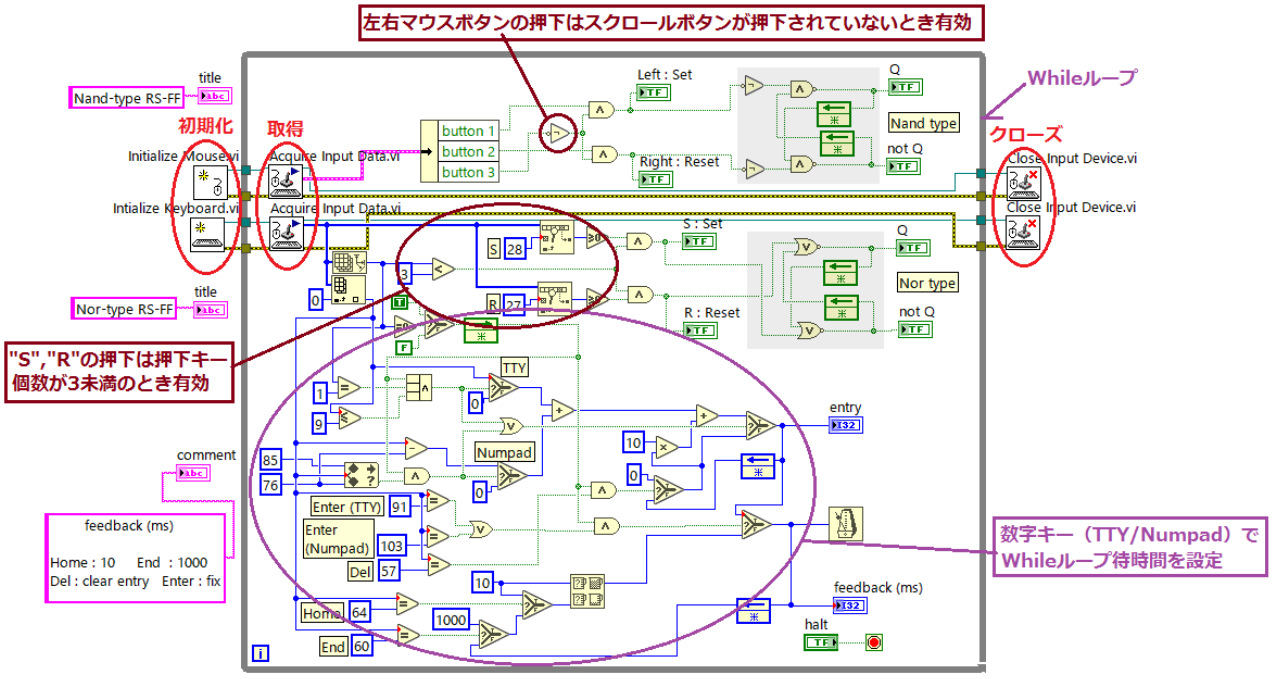

図 2　キーボード、マウスボタンで Set/Reset の同時 ON/OFF が可能な FF のブロックダイアグラム

左右両方のマウスボタンを押すと **Nand** ゲートによる FF の出力 Q、not Q は共に 1 となり、この状態でスクロールボタンを押し（ここで発振する）、押している間に左右のボタンを離すと発振を続ける。キーボードの SR の両方を押すと **Nor** ゲートによる FF の出力 Q、not Q は共に 0 となり、この状態で SR 以外のキーを 2 個（どのキーでもよいが例えば JK：1 個目を押した時に発振する）押している間に SR の両方を離すと発振を続ける（SR 以外に押すキーが 1 個の場合には、SR の内、後で離した方の入力が生き残る）。

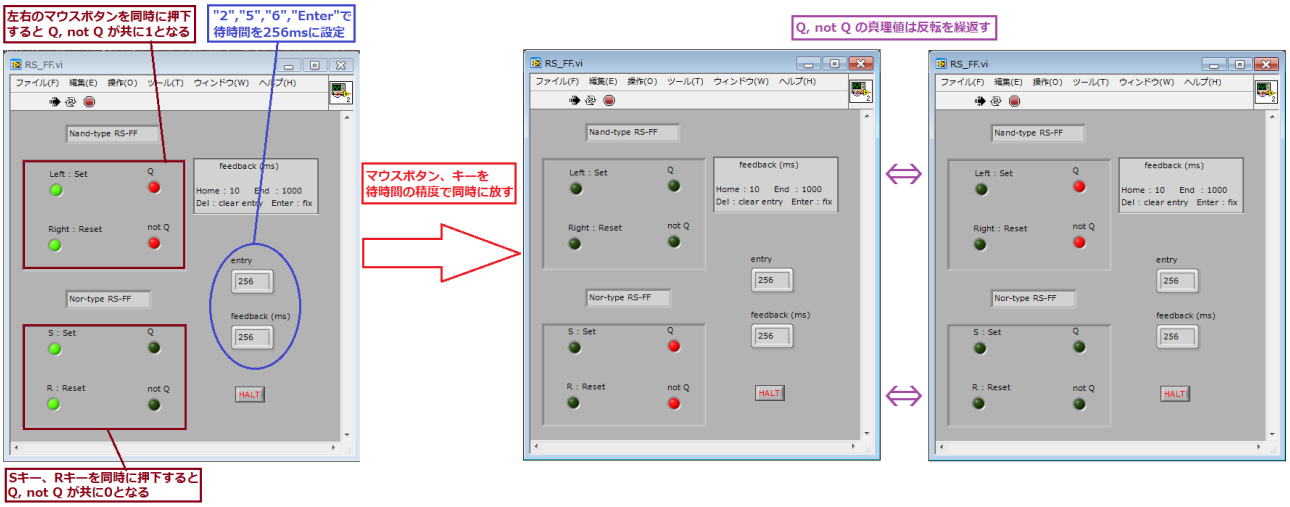

図 3　同時に ON（左）の後、同時に OFF にした発振の様子（中⇔右）

他の入力の補助無しに左右のマウスボタンまたは SR 両キーを同時に無効にすることはできないが、ループ待時間（p.3 図 2 のメトロノームアイコンへの入力で単位は ms：待時間を除いた正味の実行時間を加えたものがクロック周期に相当）を長くすれば同一ループ実行中に両方を離すことは容易である。待時間の設定には Home（10ms に設定）、End（1s に設定）のプリセットの他、数字キー（TTY、Numpad の何れでもよい）と Del（クリア）、Enter（確定：TTY、Numpad の何れでもよい）のキーで値を直接設定する。待時間が長い場合、キーまたはマウスボタンの操作が出力に反映（FF で Q と not Q が異なる値となる）されるまで押し続ける必要があるため、数値入力では入力前に Home を押して 10ms に設定しておくのがよい。p.3 図 3（以降の掲載図の実行時フロントパネル画面は、ソース VI ではなく実行ファイルの動作例を示す）は、待時間を 256ms に設定し、同時 ON から同時 OFF に移行した後に発振する様子である。

3. audio_input_monitor.vi ～ サウンド入力 ～

マウス・キー情報と同様にセンサー、専用の A/D 変換器無しにリアルタイムデータを取得する簡単な例では、内蔵サウンドカードからの入力モニター（本 VI は表示のみ：ファイルに録音する VI は備考掲載の WaveIO 参考プログラム参照）が挙げられる。**audio_input_monitor.vi** は、**Mouse_KB.vi** と同様に初期化・取得（While ループ内）・クローズの流れで Windows 既定のオーディオ録音デバイス（註参照）への信号を表示する VI で、図 4 にブロックダイアグラムを示す。4 個の数値スライドは、それぞれプロパティ「外観」の「デジタル表示器を表示」をチェックして数値ボックスを併設し、パワーレンジ設定用数値スライド（ラベル“dB range”）では範囲設定のためプロパティ「外観」の「追加」をクリックしてノブを 2 個設けている（最大最小関数を介して範囲指定に用いており両ノブの設定値の大小関係は自由である）。

註：実行中にエラーが発生（例：PC の音声入力ジャックに接続が無い場合）すると、エラー出力クラスタ（ラベル“error out”）の status が赤の✖表示に変って停止する。“HALT”ボタンと「実行」ボタン（⇨）を順次クリックすると status が緑の✔に復帰して停止状態となる。

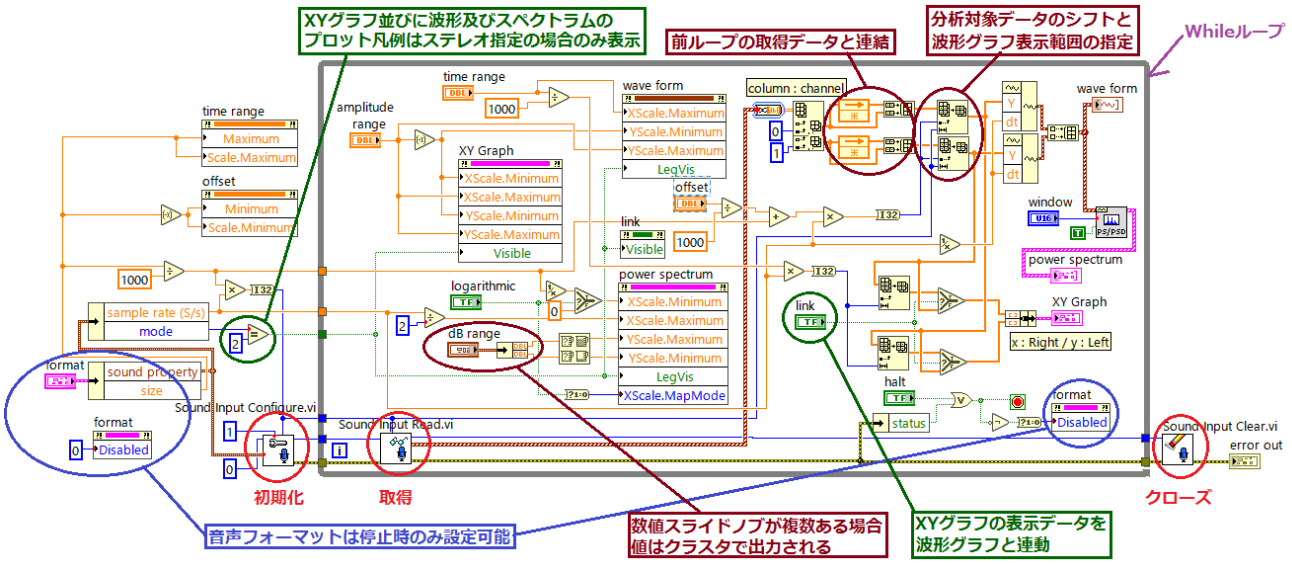

図 4　サウンド入力信号表示 VI のブロックダイアグラム

録音デバイス初期化後の While ループの中では音声フォーマットの制御器クラスタ（ラベル“format”）を操作してもその値が反映されることはない。フロントパネルに表示される設定値と実行環境とが矛盾しない様に、ループ内では“format”制御器をプロパティノードで操作を無効に（無効（Disabled）プロパティの値を 1 に）しており、また実行ファイルを開いた直後に音声フォーマットを設定できる様に、実行ファイルは停止状態で開く設定でビルドしている（フラットシーケンスストラクチャを用いた p.12 の **True_peak_meter.vi** では実行状態で開く設定にできる）。

取得した指定ブロック長（既定は 100ms）のデータは、オシロスコープの Horizontal Position に相当する位置指定ができる様に、前のループの取得データと連結して 2 ブロック長とし、これを数値スライド（ラベル“offset”、キャプション“time offset (ms)”）でシフト量を指定（既定は 0 で最新データ）して 1 ブロック分を切出している。

図 5 は PC の内蔵サウンドカードではなくループバック可能なオーディオインタフェースを接続して高速スイープ信号（註参照）を再生した **audio_input_monitor.vi** 実行中のフロントパネルの例である。ブロック長は信号周期に合せた 250ms を指定、分析対象の先頭をスイープ開始時点に位置付けている。

周波数が指数関数的にスイープする信号 1 周期のパワースペクトラムは周波数に反比例し、パワー、周波数両対数のスケールでは右下がりの直線（-10dB/decade）になる（窓関数無しに設定した図 5 左）。分析区間が信号周期とは限らない通常用いられる窓関数付（LabVIEW の既定は Hanning）のパワースペクトラムではブロック両端の寄与を下げるため直線からは外れる（図 5 右）。

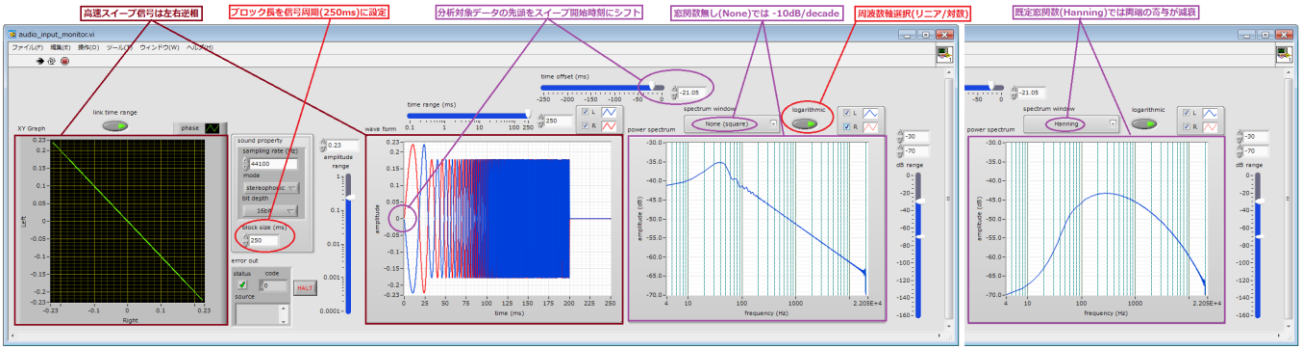

図 5　高速スイープ信号を入力したサウンド入力信号表示 VI の実行例

註：高速スイープ信号（NAB Broadcast & Audio Test CD, Vol 2, Track. 51）はオシロスコープを用いて音響機器の周波数特性データを短時間で得るためのもので、オシロスコープトリガー用に第 1 波のみ 2dB 高く（微分は連続しない）、また機器の極性を問わず低域の応答が不足する場合にもトリガーのソースが得られる様に左右逆相で生成されている。旧情報学実験Ⅱ②「オシロスコープと信号処理[オシロスコープ編]」ではこの信号を MATLAB で生成して使用した。

4.　parametric_plot.vi　～ 数式文字列評価 ～

LabVIEW では、NI ライブラリの数式評価 VI を利用して数値ではなく数式を文字列制御器で入力することができる（パレットは「数学」→「スクリプト&フォーミュラ」→「1D&2D 評価」→「フォーミュラ文字列評価」）。例えばπ/2 の値を設定する場合、数値制御器のボックスに 1.57079632679 と入力するのではなく文字列制御器に（勿論 1.57079632679 と入力しても評価される）`pi(.5)` と打ち込むだけでよい。例として、媒介変数で定義された曲線を XY グラフに表示（点列生成 VI のパレットは「数学」→「スクリプト&フォーミュラ」→「1D&2D 評価」→「X-Y(t)評価」）する **parametric_plot.vi** のブロッ

クダイアグラムを図 6 に示す。

数値制御器（ラベル “index”、キャプション “preset index (-1 for custom)”）でプリセット曲線（While ループの外で文字列定数の 2D 配列で定義した曲線サンプル）と文字列クラスタ（ラベル “CUSTOM” : “index”の値が負のとき表示される）で定義した曲線を XY グラフ（ラベル “xy-graph”、キャプション “parametric plot”）に選択表示する。数式を入力するクラスタ要素は順に x(t)、y(t)、t0（t の始端）、t1（t の終端 : t0、t1 の大小関係は自由）、x 軸最大絶対値（0 で自動スケール）、y 軸最大絶対値（0 で自動スケール）である。

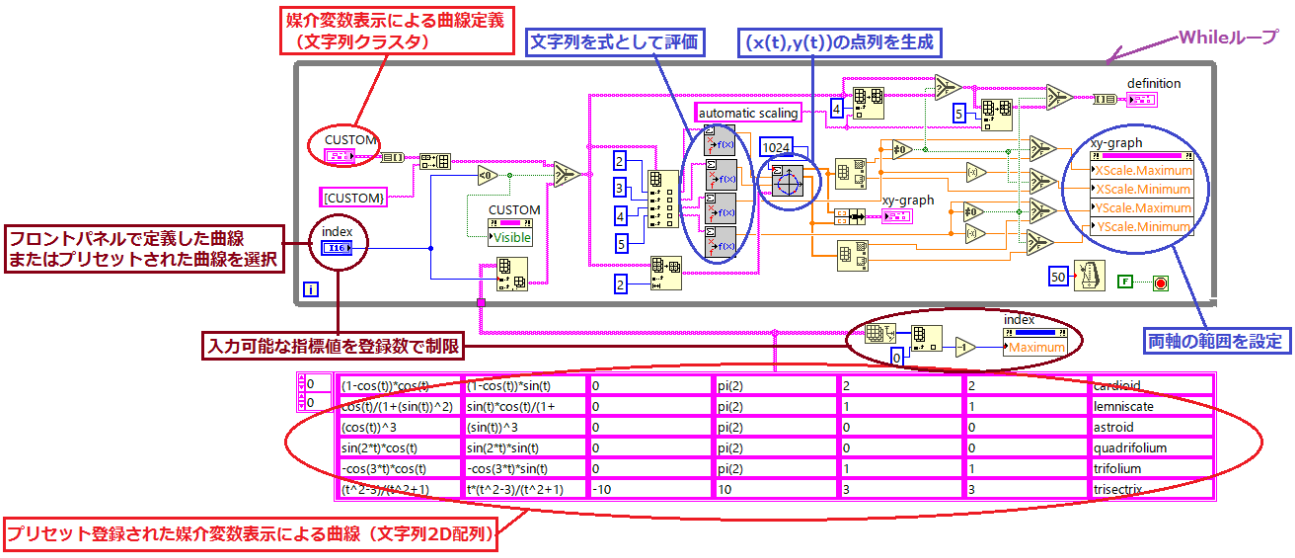

図 6　媒介変数で定義された曲線を表示する VI のブロックダイアグラム

図 7 に、**parametric_plot.vi** の実行例のフロントパネルを示す。

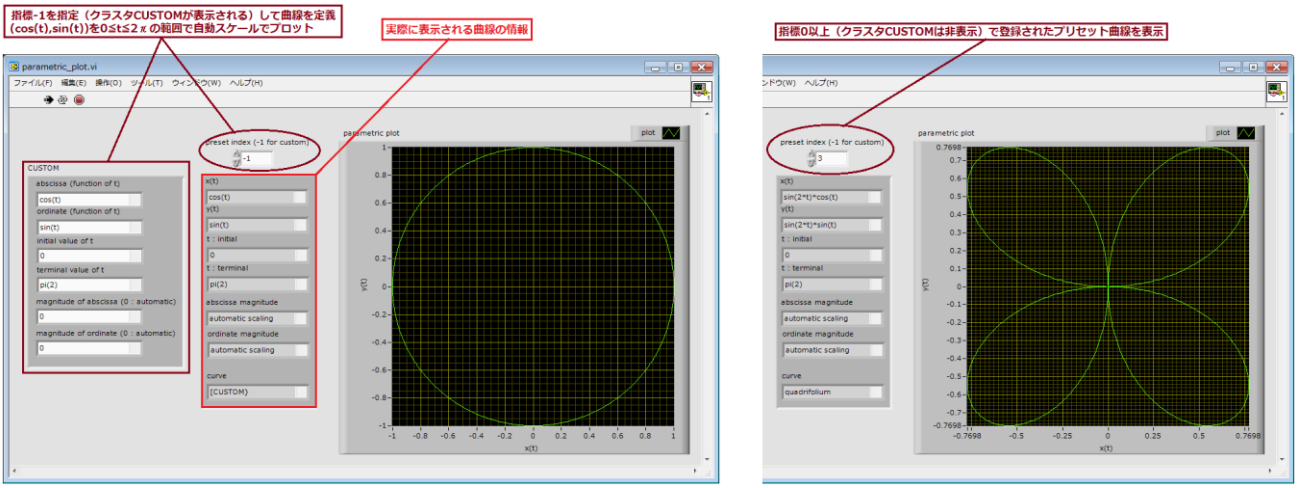

図 7　CUSTOM で指定した曲線（円 : 左）とプリセット登録された曲線（4 葉線 : 右）

5. chalkboard.vi　～ ケーススストラクチャ ～

LabVIEW では While ループ以外に For ループ、フラットシーケンススストラクチャ等のプログラミン

グ上必要な構造が用意され（パレットは「関数」→「プログラミング」→「ストラクチャ」）、たとえばケーススストラクチャを（機械的動作を「放されたらラッチ」に設定したブールボタンを場合分けに）使用することで While ループの中で随時ファイル I/O を行うことができる。

ここまでの例で関数部品 VI のプロパティで設定する項目の値をプロパティノードを用いて実行時に自由に変更できることを見てきたが、プロパティノードでは**制御器・表示器右クリックのプロパティ設定画面には含まれない項目**も指定できる。たとえば画像表示器 2D ピクチャのプロパティで設定する「サイズ」は縁を含んだフロントパネルに占める全体を意味し、実際に画像を表示できる部分は高さ、幅共に 6 画素分少なくなるが、2D ピクチャのプロパティノードを項目「描画画面サイズ」（DrawAreaSize）で作成して「書き込み」に変更することで正味の表示サイズを直接指定することができる（図 8 のブロックダイアグラム左端 While ループ外側茶色囲み部分。ここでは使用していないが 2D ピクチャでは垂直・水平スクロールバー表示指定の項目もある）。また項目「マウス」（Mouse）を指定すると（このプロパティ項目は**性質上読取り専用**で書込みはできない）、マウスボタンの状態とマウスポインタの **2D ピクチャ内での**座標を取得できる（図 8 中央左の緑の囲み部分）。

chalkboard.vi は、ケーススストラクチャの使用例として 2D ピクチャに文字列制御器に入力したテキストと画面上にマウスドラグで入力した図形を重畳して表示する単純な「黒板」VI（表示サイズは 384×256、起動時既定の色は 0x317100 の「緑板」）で、図 8 にブロックダイアグラムを示す（While ループ外側の説明部分にケーススストラクチャの表示外のケースの内容を一部表示している）。複数の色、フォントを同時に使用することはできないが、拡張子.bmp で非圧縮画像の SAVE と LOAD を繰返して表示を重ねることで原理的には可能である。画像ファイルの LOAD/SAVE のブールボタンを押下した後、プロンプトをキャンセルした場合にエラー終了しない様に **VI プロパティの「実行」カテゴリでは、「自動エラー処理を有効」のチェックを外している。**

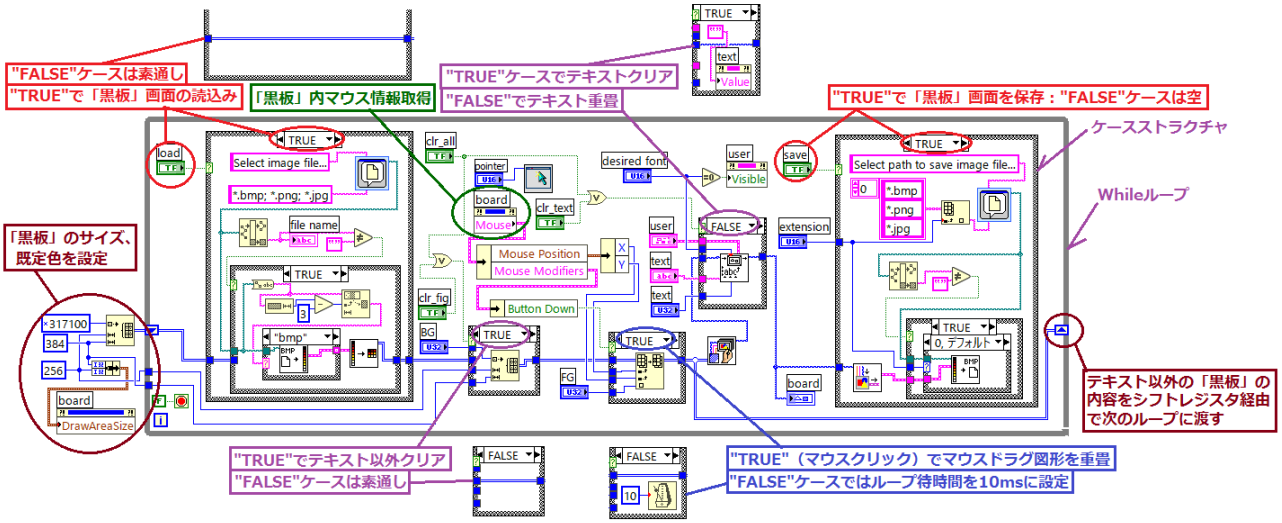

図 8　黒板 VI **chalkboard.vi** のブロックダイアグラム

p.8 図 9 左は起動後（既定のテキストは「工学院大学情報学部 Faculty of Informatics, Kogakuin University」、フォントはメイリオ 24pt.下線付太字）マウスで∞を描いた実行例、図 9 右は、画像ファイル（新 2 号館紹介ビデオクリップのキャプチャ画像 **KogakuinHachioji0.jpg**：**LV_sample.zip** に同梱）を読込み、HGP 教科書体を指定した例（ユーザ指定フォントがインストールされていない場合は LabVIEW 設定の“Application Font”で表示される）である。

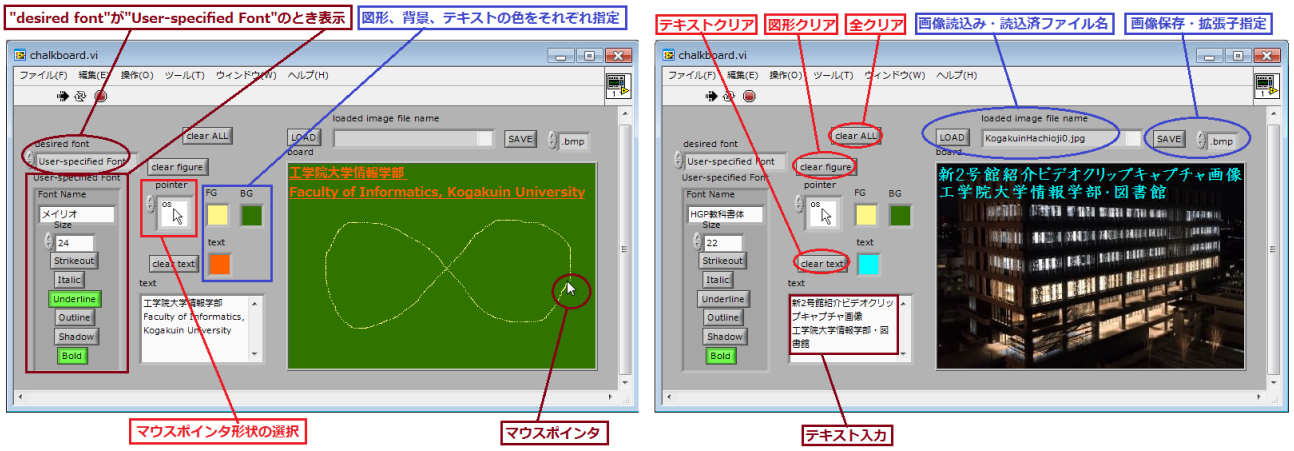

図 9　既定の画面にマウスで∞を描いた例（左）　画像を読込んでフォントを変更した例（右）

6. ring_counter.vi　～ サブ VI ～

LabVIEW では VI の制御器、表示器を入出力端子に割当てたサブ VI（NI のライブラリ VI と同様にブロックダイアグラム上に配置して呼出せる：註参照）を作成してモジュール化することでができる。例としてサブ VI で定義したマスタースレーブ構成の JK 型 FF を使用する 10 進 2 桁のリングカウンタ（例えば 10 進コンピュータ ENIAC のアキュムレータでは 10 進 10 桁の各桁を BCD 表現ではなくリングカウンタで構成した）**ring_counter.vi** を取上げる。

図 10 にサブ VI **JK_FF_w_preset_clear.vi** の内容を示す。実際の TTL IC（入力端子は開放で高電位となるため負論理を採用すればノイズが問題とならない場合論理値 0 は開放でよい）による FF と同様に**初期値設定端子は負論理**としている。制御器（図 10 フロントパネル赤枠）の VI 保存時既定値は、サブ VI として正常に動作させるため全て論理値 0 に設定しており、「連続実行」ボタン（）による単独実行（While ループが無いため、待時間が無く CPU を消費する）では開始時に発振する。Q または not Q の初期値を設定後、“J”、“K”を ON の状態で“clock_in”を操作して**クロックの立下り時**に Q および not Q が反転する様子を見ることができる。

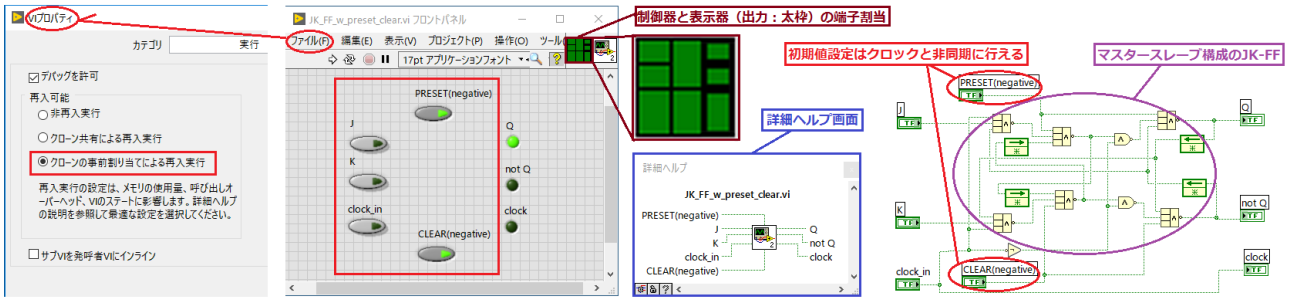

図 10　サブ VI　**JK_FF_w_preset_clear.vi** の定義内容

註：LabVIEW2009 以降ではファイルメニューの VI プロパティの「実行」カテゴリに「再入可能」（Reentrancy）項目が加わり「クローン共有による再入実行」の設定をして**サブ VI 自身からの再帰呼出も可能**になった。既定の「非再入実行」ではサブ VI はメモリ共有で「使い回し」されるため、高速動作が必要な実時間信号処理ではデータ競合を生じ動作が不確実になる。再帰呼出の無いサブ VI を複数の箇所で参照する場合は、**「クローンの事前割り当てによる再入実行」**の設定を推奨する（図 10 左端）。

LabVIEW では**行を時間軸、列をチャンネルとする 2D ブール配列**をデジタル波形に変換（関数パレットで「プログラミング」→「波形」→「デジタル波形」→「デジタル変換」→「ブール配列からデジタルに変換」）して信号の時間変化をデジタル波形グラフ（制御器パレットで「グラフ」→「デジタル波形グラフ」）に表示でき、**ring_counter.vi** では呼出した **JK_FF_w_preset_clear.vi** 個々の FF 値を円形に配置した OK ボタン群（ブール表示器の LED の点灯表示には曲率がありテキストを適切に表示できないため、カウンタ表示には制御器の **OK ボタンを表示器に変更**して使用）と波形グラフによりカウンタの動作状況を表示している（p.10 図 12 参照）。

波形ブール値の初期データ（ここでは全て FALSE）は While ループの外で時間レンジ×チャンネル数分を設定し、While ループの中では最も古い行データを捨て最新の行データを連結したもの（これを波形変換して表示）をシフトレジスタ経由で次のループに渡せばよい。**列方向は指標 0 のチャンネルが波形グラフの最上段**にプロットされる（「ブール配列からデジタルに変換」VI のパラメータ“mode”の既定）。ここでは時間レンジは下位カウンタのサイクルの一部が重なる 12 クロック周期（256 ループ 1 クロックで 3072 行：ループ待時間 5ms の設定では現在から約 15s 前までを表示）、チャンネル数は表示が煩雑にならない様に**リングカウンタ動作の**クロックとリングカウンタ FF の 11 チャンネル（上位下位カウンタを切替えて表示：VI 保存時の既定は下位）としている。波形グラフの右には、11 チャンネルの最新の値を LED **クラスタ**（註参照）に波形グラフプロット色（チャンネル個別）またはカラーボックス指定の色（同一）で表示する。

図 11 は **ring_counter.vi** のブロックダイアグラムで **JK_FF_w_preset_clear.vi** を 21 個（内、FF 単体動作確認用の 1 個のみブロックダイアグラム上でラベルを表示）配置している。カウンタ部の 20 個は実際には D 型 FF 動作でシフトレジスタに使用し、**下位桁の 9→0 への転送値が上位桁カウンタのクロック**信号（下位桁カウンタ動作とは異なりデューティ比は 10%）となっている。

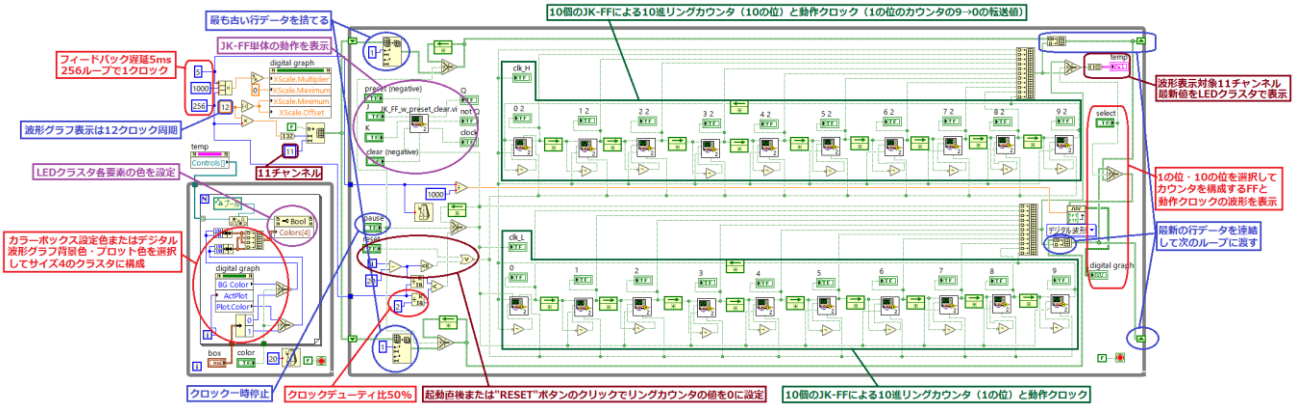

図 11　**ring_counter.vi** のブロックダイアグラム

註：LabVIEW の**配列では要素個別に色その他のプロパティを設定することはできない**。本 VI ではクラスタ要素の色設定を図 11 左下の待時間を 20ms（優先度が低い）とした独立な While ループで次の様に行っている。①デジタル波形グラフのプロット色は右クリックで作成したプロパティノードから読み取る。ビット 0 の色は「プロット領域」→「カラー」→「背景色」（BG Color）、ビット 1 の色は「アクティブプロット」（ActPlot）に For ループ反復端子（i）でプロットチャンネルを書き込み、「プロット」→「プロットカラー」（Plot.Color）で指定チャンネルの色を取得する。②LED クラスタ各要素の参照には**プロパティノードへのリファレンス入力**が必要で、LED クラスタ（ラベル“temp”）を右クリックで項

目「制御器[]」(Controls[]) のプロパティノードを作成し For ループの自動指標トンネルに指標付け使用（既定）で入力する。③クラスタ各要素の色を書き込むプロパティノードはこれまでのサンプル VI の様な制御器・表示器の**右クリックによる作成ではなく、関数パレットから「プログラミング」→「アプリケーション制御」**で「プロパティノード」/「より特定のクラスに変換」/「クラス指定子定数」を各 1 個ブロックダイアグラムに配置する（次項 **screen_pixel.vi** の例もリファレンス入力のあるプロパティノードを使用する）。クラス指定子定数を右クリックして→「VI サーバクラスを選択」→「一般」→「G オブジェクト」→「制御器」→「ブール」を選択し「より特定のクラスに変換」の上の端子に配線し、「より特定のクラスに変換」の入力には For ループ内で指標付けされた「制御器[]」(Controls[]) 要素を配線し、出力をプロパティノードの「リファレンス」端子に入力し（クラスが App から Bool に変る）、プロパティノード右クリック「プロパティを選択」で「カラー[4]」(Colors[4]) を選び書き込みに変更する。④読み取られた色とボックス色とを選択しビット 0、1 の色をサイズ 2 のクラスタ、これを更にサイズ 4 のクラスタに構成してプロパティノードの「カラー[4]」に入力する。

図 12 に **ring_counter.vi** の実行中のフロントパネルを示す。図 12 左は**下位カウンタの 9** が起動後最初に ON となった**直後**で波形の先頭でカウンタ FF が初期化（起動後の 20 ループ期間は RESET ボタンが押されたと同じ）されていること、上位カウンタの動作クロックが ON となっている様子が分る。パネル左上単体の FF は **JK_FF_w_preset_clear.vi** と同様に **ring_counter.vi** のソース VI を開いた時に発振しない値を既定として保存している。図 12 右は単体 FF を反転動作の設定とし、波形表示を上位カウンタの信号に切替えた表示で、カウンタが **79 から 80 に変化する直前**で両カウンタの動作クロックが共に ON になっている。

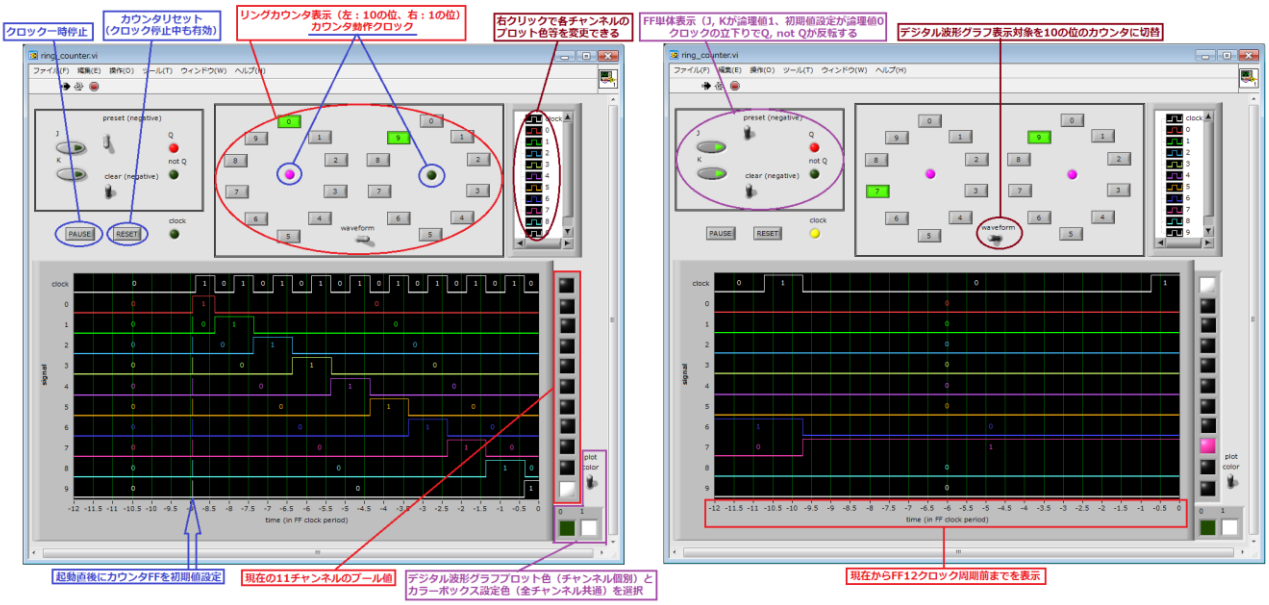

図 12　**ring_counter.vi** のフロントパネル　下位カウンタ波形表示（左）、上位カウンタ波形表示（右）

7. screen_pixel.vi　～ ライブラリ呼び出し 1 ～

LabVIEW では **chalkboard.vi** に見る様に**画像ファイル**の読込み、保存、表示が可能で、画像データを格納した 2D 配列の要素を指定して画像の特定の画素の色情報を取得できるが、NI ライブラリの関数で

はデスクトップ上で座標を指定して画面の色情報を取得することはできない。

screen_pixel.vi は、**Windows ライブラリ関数**の GetPixel を呼び出して（GetPixel の前後に GetDC および ReleaseDC がそれぞれ必要である：註 1 参照）画面座標を数値制御器（スライドまたはボックス：数値の範囲はプロパティノードの項目「表示」→「プライマリワークスペース」（Disp.WkSpace）で得られる長方形領域に制限しており、自動的に隠す指定をしていないタスクバーの領域は指定できない：p.12 図 14 左参照）またはマウスポインタ（タスクバー領域も指定できる：p.12 図 14 右参照）で指定して色情報を表示する VI である。LabVIEW の 32b 画素データは各色を符号無し 8b 整数として上位から 0RGB の順で表現されるが **GetPixel が返すデータは 0BGR の順**であることに注意する。

註 1：関数パレットから「接続」→「ライブラリ&実行可能ファイル」→「ライブラリ関数呼び出しノード」をブロックダイアグラムに配置してダブルクリックし関数、パラメータを設定する。呼び出し規約は stdcall (WINAPI)、項目「ライブラリ名またはパス」は関数名 GetDC および ReleaseDC では user32.dll、GetPixel では gdi32.dll である。ライブラリ関数の詳細は Microsoft のサイトにあり、本 VI で使用する関数は次の 3 本である。

https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-getdc
https://docs.microsoft.com/en-us/windows/desktop/api/wingdi/nf-wingdi-getpixel
https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-releasedc

マウスポインタではなく数値制御器で座標を指定するとき、指定座標の画面内位置の把握を容易にするため **screen_pixel.vi** では指定座標の近傍を**.NET コンテナ**（フロントパネルに制御器パレットで「コンテナ」→「.NET コンテナ」を配置し、右クリックして「.NET コントロールを挿入...」を選び「アセンブリ」で System.Windows.Forms(4.0.0.0)、「コントロール」で **PictureBox** を選択する。このとき境界が消えるので制御器パレットの「装飾体」→「凹フレーム」を重ねている）にビットマップとして表示（指定座標がデスクトップ隅の近くではビットマップの中心ではないため脇の進行状況バーで指定箇所を示す）しており、画面のビットマップコピーのために **.NET Framework ライブラリ**の FromImage メソッドと CopyFromScreen メソッドを呼び出している（註 2 参照）。

註 2：ここではコンストラクタノード 1 個、インボークノード 2 個、プロパティノード 2 個を使用する。コンストラクタノードは関数パレットから「接続」→「.NET」→「コンストラクタノード」をブロックダイアグラムに配置すると「.NET コンストラクタを選択」ウィンドウが開くので「アセンブリ」で System.Drawing(4.0.0.0)、「オブジェクト」で **Bitmap**、「コンストラクタ」で Bitmap(Int32 width, Int32 height)を選択する。2 個のインボークノードは関数パレットから「プログラミング」→「アプリケーション制御」→「インボークノード」をブロックダイアグラムに配置し、それぞれ右クリックで「クラスを選択」→「.NET」→「System.Drawing.Graphics」を選択する。再度右クリックし「メソッドを選択」で 1 個は「[s]FromImage(image image)」、1 個は「CopyFromScreen(Int32 sourceX, Int32 sourceY, Int32 destinationX, Int32 destinationY, Size blockRegionSize)」を選択する。**2 個のプロパティノードは、関数パレットから**「プログラミング」→「アプリケーション制御」→「プロパティノード」をブロックダイアグラムに配置し、「リファレンス」端子に 1 個はコンストラクタノードの「新規リファレンス」、1 個は .NET コンテナ（本 VI ではラベル “neighborhood”）を配線後（プロパティノードのクラスが App から Bitmap、PictureBox に変る）、右クリック「プロパティを選択」でそれぞれ Size、Image を選ぶ。

System.Drawing.Graphics クラスの詳細は次のページにある。

https://docs.microsoft.com/en-us/dotnet/api/system.drawing.graphics?view=netframework-4.7.2

図 13 に **screen_pixel.vi** のブロックダイアグラムを示す。**audio_input_monitor.vi** では数値スライドのプロパティで「デジタル表示器を表示」をチェックして数値ボックスを併設したが、本 VI では座標指定の数値ボックスを水平と垂直でクラスタ化しているため（スライドの併設ではなく独立）スライドとの連動がやや煩雑になっている（右側の While ループ上部）。クラスタ（左側の While ループ下の孤立した“numeric”）の値はプロパティノード（クラスタの要素については右クリックして「リンク先」で選択する）での参照のためブロックダイアグラム上でクラスタ本体は配線されていない。

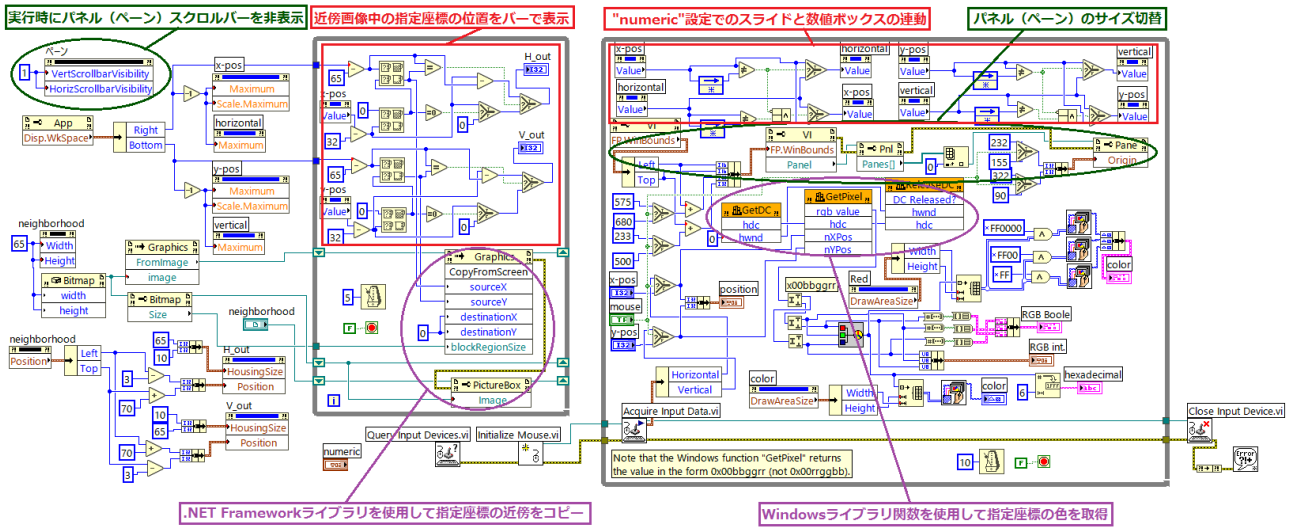

図 13　**screen_pixel.vi** のブロックダイアグラム

screen_pixel.vi 実行中のフロントパネル（2022 年前期までのバーチャルカフェテリアでの実行例）を図 14 に示す。実行中はパネル（ペーン）のスクロールバーが非表示となり、押しボタン SW（ラベル“mouse”、キャプション“numeric / mouse”）で選択した画面座標指定手段に応じて切替わるサイズに固定され、伸縮して変更することもできない。“numeric”設定では垂直スライドのスケール上限は FHD デスクトップの 1079 ではなくタスクバー領域を除いた 1037 となっている（図 14 左）。垂直スライドのノブ上のマウスポインタは近傍画面には表示されない。

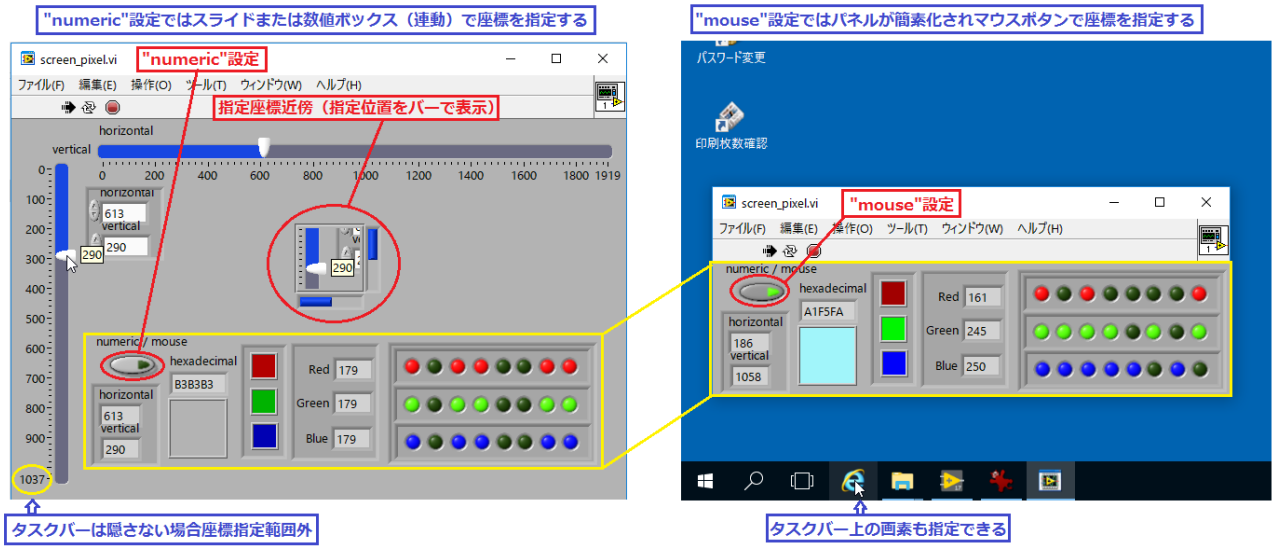

図 14　**screen_pixel.vi** のフロントパネル：”numeric”設定（左）、”mouse”設定（右）

8. True_peak_meter.vi ～ ライブラリ呼び出し 2 ～

LabVIEW ではユーザが開発した関数ライブラリ（拡張子.llb の VI ライブラリおよび拡張子.dll の実行時ライブラリ）を LabVIEW インストール先の **user.lib サブフォルダ**に置いて、関数パレットの「ユーザライブラリ」に作成される「paletteMenu」サブパレットを開いて NI ライブラリの関数 VI と全く同様に参照できる。

PC の管理権限が無い場合にはライブラリをソース VI と同じ場所に置くことでライブラリを使用した既存ソース VI の実行、ソースから実行ファイルのビルドができる。ライブラリ中の VI を参照する新規 VI の作成は、①拡張子.llb の VI ライブラリファイルをダブルクリックして LLB マネージャを開き、②LLB マネージャウィンドウで参照する VI をダブルクリックして開き、③当該 VI のフロントパネル右上のアイコンを新規 VI のブロックダイアグラムにドラグ&ドロップすればよい。

Windows MME ドライバを使用する LabVIEW のサウンド入出力 VI は Windows Vista 以降で導入された**ループバック入力、WASAPI ドライバ**に対応していないが、幸い Christian Zeitnitz 氏が開発した非商用ではフリーの WaveIO ライブラリ（註 1 参照）を利用することでループバック入力、WASAPI、ASIO 各ドライバに対応できる。

註 1：WaveIO の VI ライブラリは **LV_sample.zip** に**同梱していない**ので（実行時ライブラリは **data フォルダ**に格納されており、**True_peak_meter.exe** はそのまま実行できる）本 VI の**ソースを開く**場合は以下の場所から waveio_108.zip を取得して解凍後 WaveIO.llb および WaveIO.dll（本学で使用している LabVIEW は 32b 版であり WaveIO_x64.dll ではない）をソース VI と同じ場所に置く。
Zeitnitz 氏のページ：https://www.zeitnitz.eu/scms/waveio
WaveIO 最新版：https://www.zeitnitz.eu/scms/getfile?issrc&name=waveio/waveio_108.zip
同解説（waveio_108.zip に同梱）：https://www.zeitnitz.eu/scms/getfile?name=waveio/waveio_108.pdf

WaveIO ライブラリを使用する **True_peak_meter.vi** は、サウンド入力データの標本ピークまたは 4 倍オーバーサンプリングによる標本間ピーク（註 2 参照）を表示するもので、**入力方式 “Windows loopback” を選べばループバック入力が可能なオーディオインタフェース無しに**再生中の音声出力レベルを監視できる。

註 2：標本が表現可能な最大絶対値（0dBFS：例えば 16b 深度では±32767 に相当する。極性反転できない-32768 は使わない）以下であってもサンプルホールドと LPF 処理をした出力は**標本間で 0dBFS 相当のアナログ電圧を超え** DAC が過大入力で歪む可能性がある。標本間ピークを同様に dB スケールで dBTP（dB True Peak）と表すとき、EBU（欧州放送連合）では放送用プログラム制作時のレベル管理を **4 倍オーバーサンプリング**した真ピークレベルメーターにより行い（4 倍オーバーサンプリングでは真ピーク値とは理論上 dB 値で最大 $20\log_{10}\{\sec(\pi/8)\} \fallingdotseq 0.688$ の誤差を生じるため）ピークを**-1dBTP** に抑えることを推奨している。

レベル表示は低レベル、高レベル、0dB 超でそれぞれステップと色を変えた各 16 要素の 3 個の LED 配列と数値ボックスをクラスタ化し、配列の点灯要素計算のためにサブ VI（**JK_FF_w_preset_clear.vi** と同様に**「クローンの事前割り当てによる再入実行」**を設定している）として dB_LED.vi（p.14 図 15 参照）を呼び出している。

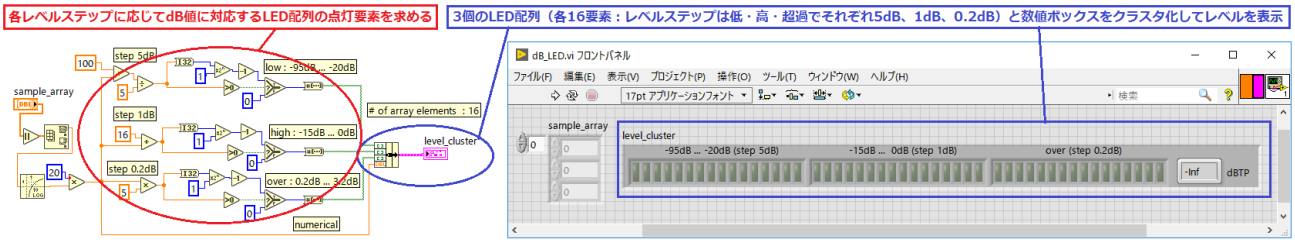

図 15　レベル表示処理サブ VI dB_LED.vi のブロックダイアグラム（左）、フロントパネル（右）

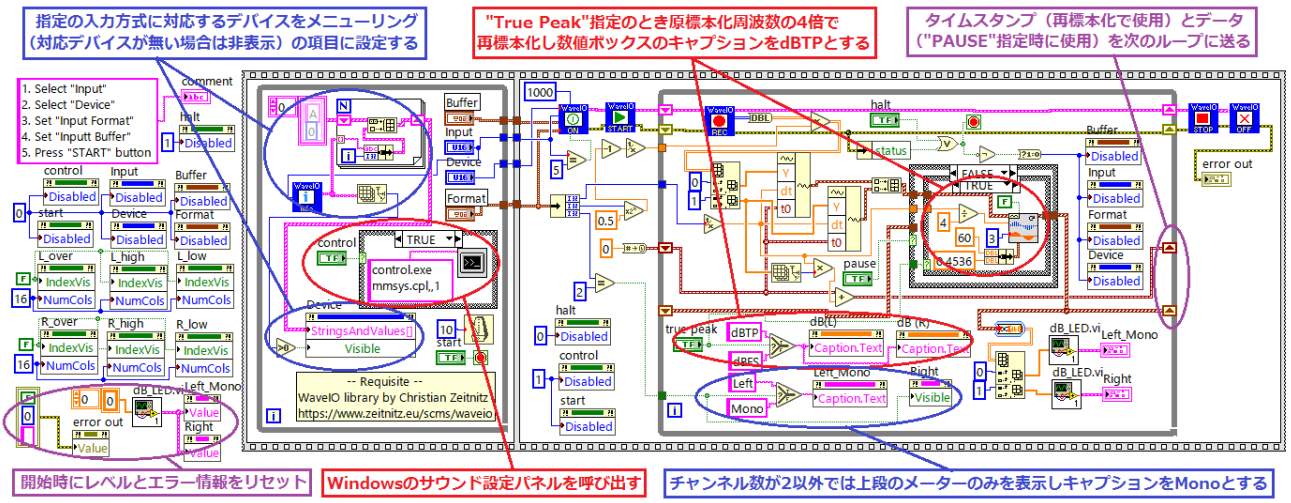

図 16　**True_peak_meter.vi** のブロックダイアグラム

4 倍オーバーサンプリングで使用する FIR フィルタ（関数パレットで「信号処理」→「波形調節」→「複数波形のリサンプル（連続）」の補間モード 3（FIR フィルタ））の既定では音声標本化周波数とブロック長に対して処理が間に合わないため「アンチエイリアス(dB)」を既定の 120 から本 VI の目的には十分な 60 に変更している（図 16 右中段赤枠参照）。Windows のサウンド設定パネルを呼び出すため、関数パレットから「接続」→「ライブラリ&実行可能ファイル」→「システム実行」を使用している。

図 17 に **True_peak_meter.vi** の実行例を示す。入力方式は"WASAPI exclusive"、デバイスは外付 USB DAC、WASAPI 排他モードで再生中の音楽ファイル（マスタリング時のノーマライズ処理の結果、標本間ピークは 0dBTP を超えている）のレベルを標本間ピークで停止表示したフロントパネルである。

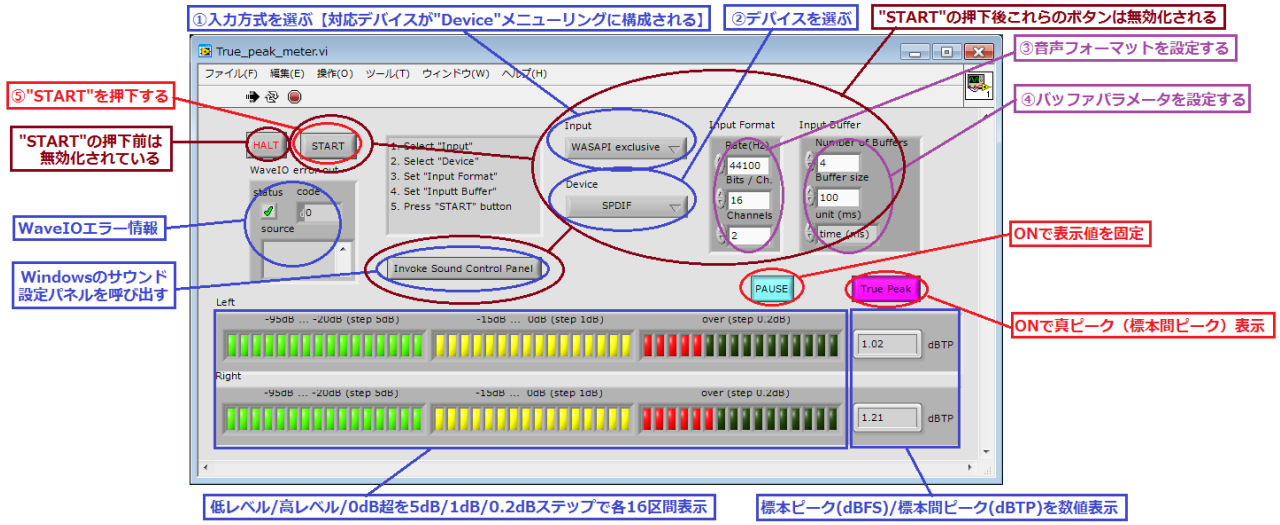

図 17　**True_peak_meter.vi** の実行中のフロントパネル

9. DP_property.vi ～ 型変換 VI・制御器 Refnum ～

通常の型変換では、変換先のデータ長と性質の範囲内で**変換元の値が保存され内部表現は当然別**のもの（例えば DBL 型の**-1** を I32 型に変換すると 16 進表記で **BFF0000000000000** が **FFFFFFFF**）となるが、これに対し LabVIEW の**型変換 VI**（パレットは「関数」→「プログラミング」→「数値」→「データ操作」→「型変換」：型指定ゲートを通るとパイプ断面が矩形から円形に変るアイコン）は**ビット内容不変で解釈される型のみを変更**する。これにより DBL から U64 に変換後「数値をブール配列に変換」でサイズ 64 の 1D ブール配列（指標 0 が元の LSB）とすれば DBL 型データのビットパタンを容易に確認できる。この逆変換では「ブール配列を数値に変換」の既定の出力が U32 であるため（サイズ 64 の 1D ブール配列を入力した結果は指標 0～31 を用いた元々のデータの下位 4B 分となる）、「ブール配列を数値に変換」を右クリックしてプロパティで出力を U64 に変更する必要があることに注意する。

DP_property.vi は、ブールボタンクラスタまたは 16 進数メニューリング配列による**ビットパタン指定**と数式文字列（変数を含む場合は数値制御器で引数を設定）による **DBL 値指定**の両モードで DBL 型と 1D ブール配列の相互変換を行い **IEEE754 規格の浮動小数点表現を対話的に確認**するデモ VI である。ビットパタン指定モード（起動時既定 p.18 図 21）では各ビットを個別に設定して個々のビットの持つ意味、DBL 値指定モード（p.18 図 22）では例えば数式文字列に**"x"**を指定して変数入力のスライドまたは数値ボックス（10 進数値表現の他、**"-Inf"**、**"NaN"**も入力でき、**同じ非数でも数式文字列の"0/0"の評価結果と「定数」"NaN"のビットパタンが異なる**ことが分る：備考 p.32 図 41 下段参照）に入力して数値に対応するビットパタンを容易に確認できる。

LabVIEW では、制御器・表示器の**リファレンス**（ブロックダイアグラムで当該制御器・表示器を右クリック「作成→リファレンス」で作成し配置）をサブ VI の入力パラメータの**制御器 Refnum**（フロントパネルに制御器パレットで「Refnum」→「制御器 Refnum」を選び配置）に渡すことで当該制御器・表示器の**プロパティをサブ VI から制御**できる（制御器 Refnum は、「LabVIEW による乱数参考プログラム」、「LabVIEW による手回し計算機参考プログラム」、「LabVIEW による多倍長演算参考プログラム」で使用されている）。**DP_property.vi** はブールボタンクラスタ要素の表示位置を手作業により調整していた「LabVIEW による浮動小数点表現参考プログラム」（実行ファイルのみ公開：最終改訂 2018/12/12）の **LV_IEEE754DP_compact** に制御器 Refnum を使用した初期化サブ VI **init.vi** 等を追加したものである。制御器 Refnum を使用する**初期化**、数式文字列チェックのサブ VI **init.vi**、**f_check.vi** のブロックダイアグラムを図 18 に示す。制御器 Refnum および呼出のリファレンス（p.17 図 20 メイン VI ブロックダイアグラム左上端）以外の部品は、**ring_counter.vi**、**True_peak_meter.vi** で既出である。

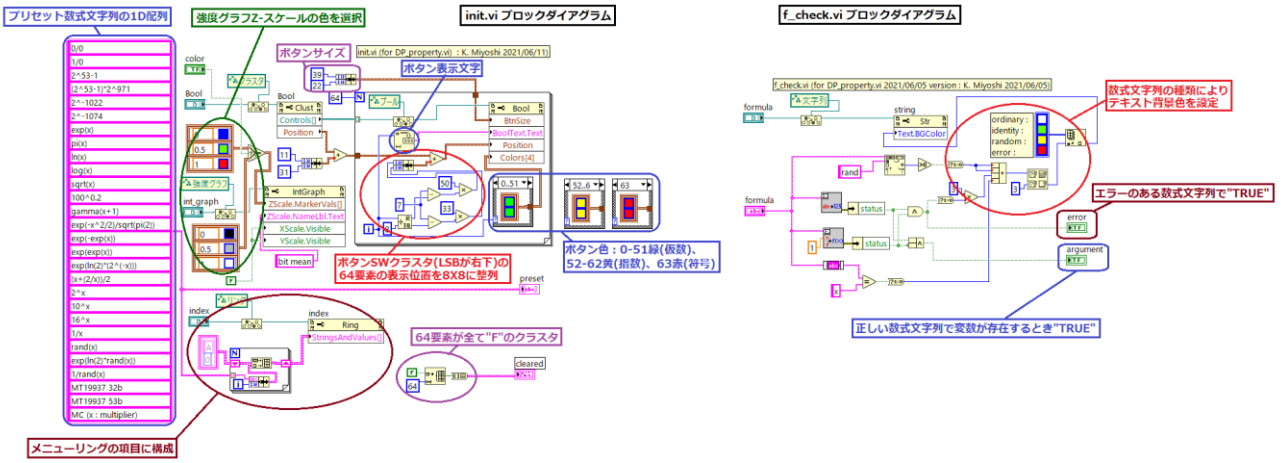

図 18　制御器 Refnum を使用するサブ VI のブロックダイアグラム：**init.vi**（左）、**f_check.vi**（右）

プリセット数式文字列 27 個の内容は以下の通りである。

- **0/0 ： 非数（NaN）となる例**
- **1/0 ： 正の無限大（Inf）となる例**
- **2^53-1 ： 整数として正確に表現される最大の数**
- **(2^53-1)*2^971 ： 表現できる最大の数**
- **2^-1022 ： 53 ビット精度で表現できる最小の正数**
- **2^-1074 ： 表現できる最小の正数**
- **exp(x) ： 指数関数**
- **pi(x) ： π の倍数**
- **ln(x) ： 自然対数**
- **log(x) ： 常用対数**
- **sqrt(x) ： 正の平方根**
- **100^0.2 ： 星の 1 等級の光度比（4dB）**
- **gamma(x+1) ： 非負の整数のとき階乗【起動時の数式文字列の既定】**
- **exp(-x^2/2)/sqrt(pi(2)) ： 正規分布の確率密度**
- **exp(-exp(x)) ： 二重指数関数**
- **exp(exp(x)) ： 二重指数関数**
- **exp(ln(2)*(2^(-x))) ： 2 の繰返し平方根**
- **(x+(2/x))/2 ： Newton-Raphson 法による 2 の平方根【feedback ボタン用の数式例】**
- **2^x ： 2 の冪乗**
- **10^x ： 10 の冪乗**
- **16^x ： 16 の冪乗**
- **1/x ： 逆数【例：整数 x が 2 の冪乗ではないとき 1/x が循環小数となることを確かめる】**
- **rand(x) ： NI ライブラリの(0,1)の一様乱数【x はダミー】**
- **exp(ln(2)*rand(x)) ： 一様乱数の指数関数(1,2)【x はダミー】**
- **1/rand(x) ： 一様乱数の逆数(1,∞)【x はダミー】**
- **MT19937 32b ： MT19937 乱数 32b**
- **MT19937 53b ： MT19937 乱数 53b**
- **MC (x : multiplier) ： 乗算合同法乱数【"argument"で乗数、"seed"で初期値を与える】**

最後の 3 個は NI の数式文字列評価の VI ではなく別処理により求めており（これらの項目が呼出された文字列制御器はテキスト背景色がマゼンタに変り無効化される）、項目 25,26 の **MT19937** では 4 個のサブ VI（内容については「LabVIEW による乱数参考プログラム」を参照されたい）を使用している。
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_random.pdf

DBL 値指定モードでフロントパネルを右に拡大すると各ビットの平均表示の画面が現れ(p.18 図 22)、区間(0, 1)の一様乱数（項目 25-27 に項目 22 の NI 乱数を加えた 4 種類）では、IEEE754 内部表現の他に 2^{53} 倍した整数のビット平均を表示できる（p.19 図 23)。

init.vi、**f_check.vi**、**MT19937** 関連以外の 4 個のサブ VI とメイン VI のブロックダイアグラムをそれぞれ図 19、図 20 に示す。

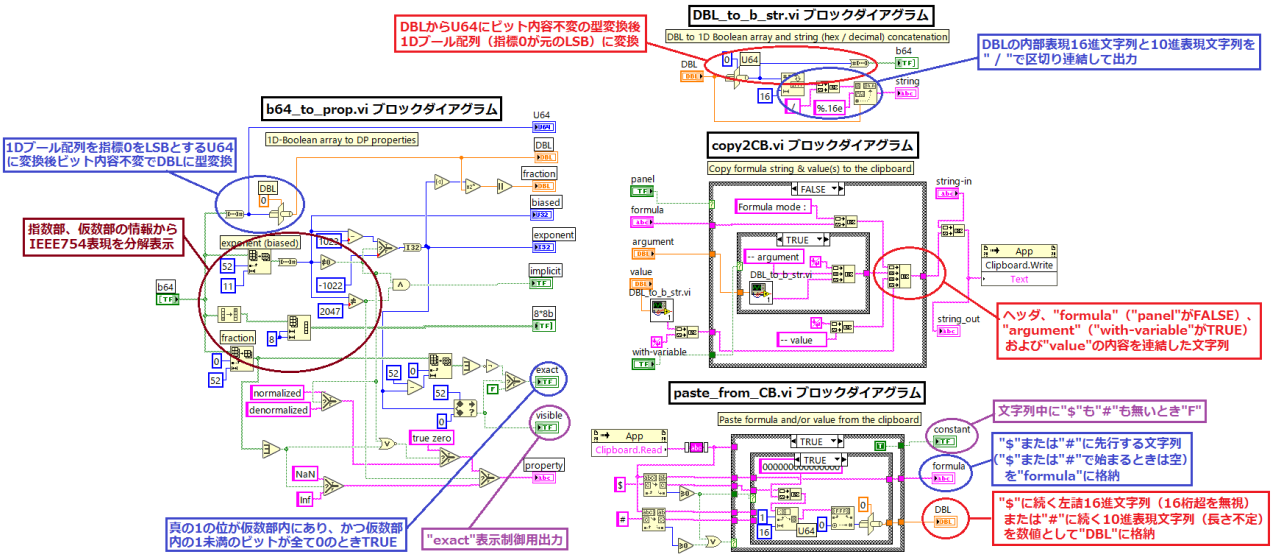

図 19　**MT19937** 関連以外のその他のサブ VI のブロックダイアグラム

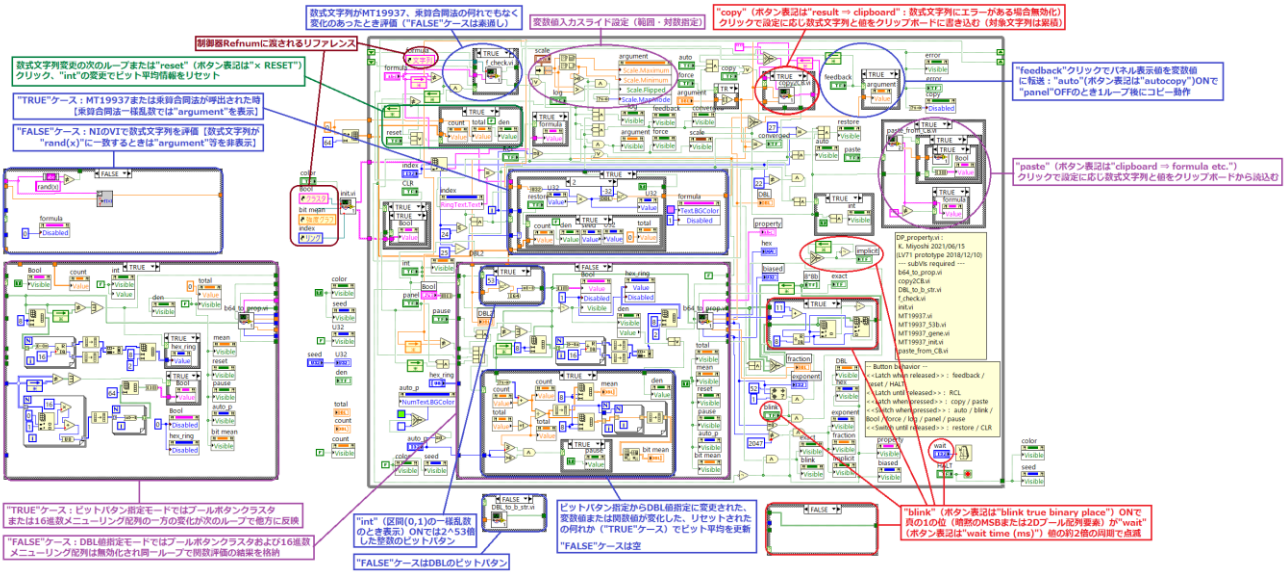

図 20　**DP_property.vi** のブロックダイアグラム

p.18 図 21 に **DP_property.vi** の実行ファイル起動時のフロントパネル（既定は SW “panel” が ON のビットパタン指定モードでπの DBL 値を表示している）を示す。ボタン面に簡単な記号が表示された “RCL”、“feedback”等は制御器パレットの通常の「モダン」パレットではなく「シルバー」パレットのものを使用している。**audio_input_monitor.vi** と同様に**実行ファイルは停止状態で開く**設定でビルドしており、停止時に強度グラフ Z-スケールの色（既定は赤を 1 青を 0 とするカラースケール）および乗算合同法の初期値 “seed”（既定は 3）を設定する。「LabVIEW による浮動小数点表現参考プログラム」では、乗数は 3 以上の奇数、初期値は正の奇数にそれぞれ強制されるが、乗数（使用される**"argument"**の起動時既定は 6）、初期値の少なくとも一方が偶数であると高々32 回で 0 となる過程を（ループ待時間を長くして）見ることができる。

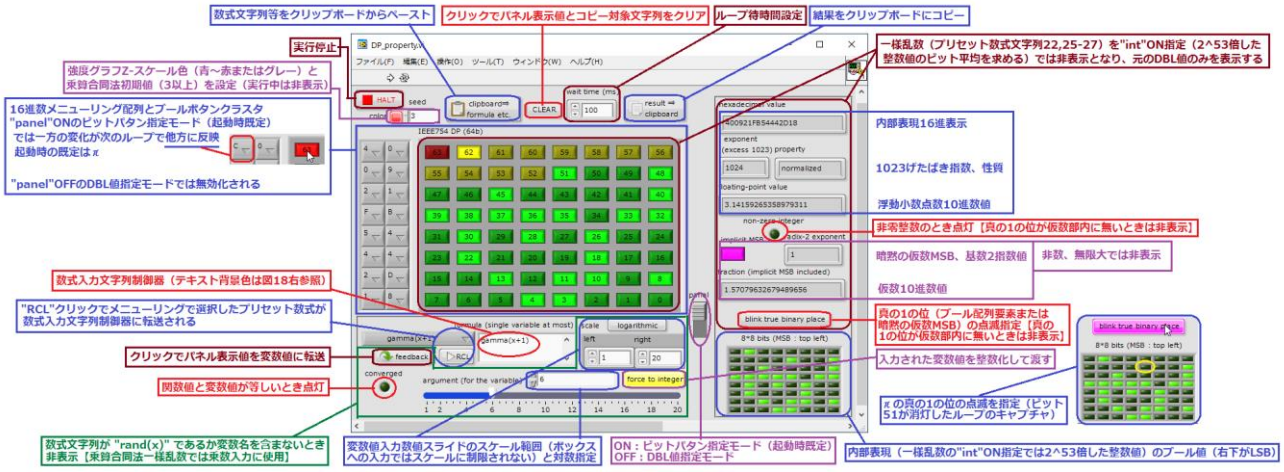

図 21　**DP_property.vi** 実行ファイル起動時のフロントパネル：(停止状態で開く)

図 22 は DBL 値指定モード（“panel”OFF：ビットパタン指定制御器のブールボタンクラスタおよび 16 進数メニューリング配列は無効化される）の既定（数式文字列 **”gamma(x+1)”** 、“force to integer” ボタン ON：このとき 6!の値は 2 進 10 進変換誤差により **719.999999999999999** と表示されるが 1 未満のビット 42 以下が全て 0 で実際の内部表現は正しく整数を表していることを示している）の状態で、①数値スライドのノブを左端の 1 に移動、②“×RESET”ボタン（ラベルは“reset”）をクリック（ビット平均情報がリセットされる）、③数値スライドのノブを右端の 20 まで緩やかに移動（“force to integer” ボタンが ON では数値ボックス増分ボタン 19 回クリックと同等）の順に操作が終った状態のキャプチャである。関数値または実際に関数に渡される変数値が前のループから変化した時に標本が追加され、フロントパネル右側の数値表示器 2D 配列および強度グラフに 1!から 20!までの 20 個の DBL 表現各ビットの平均値が表示される。

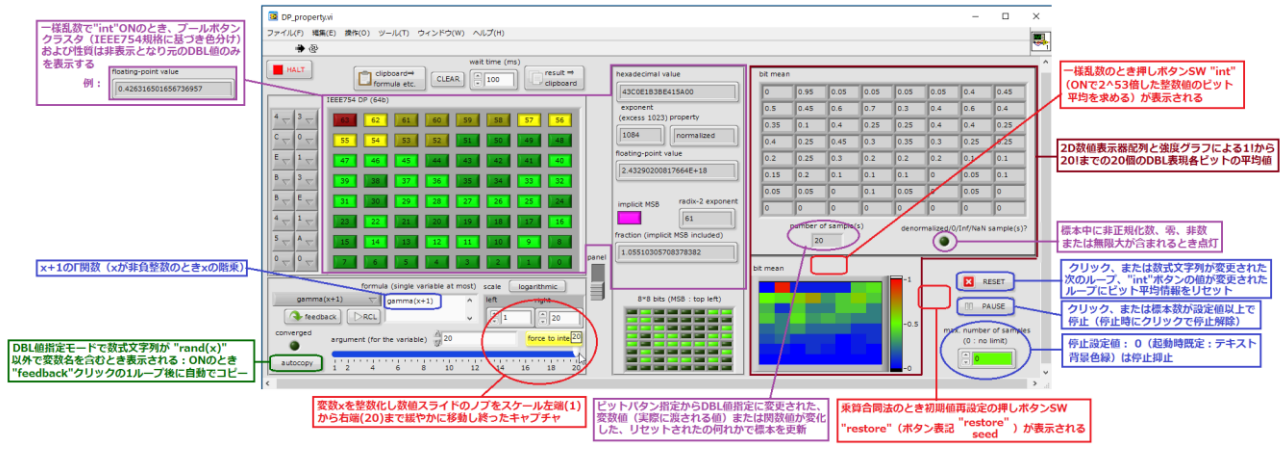

図 22　**DP_property.vi** 実行中のフロントパネル：DBL 値指定モード（“panel”が OFF）

数式文字列が **rand(x)** を含みエラーが無いかまたはプリセット項目 25-27 を呼出した場合は、ループ毎に標本が自動で追加される。p.19 図 23 は、区間(0, 1)の 4 種類の一様乱数について、停止設定値（既定の 0 は停止抑止）を 1000 として実行したフロントパネル右側部分を並べて比較したものである。

区間(0, 1)の一様乱数による IEEE754 表現のビット平均（図上段）は仮数正規化により下位ビットに 0 が多くなり特に最下位ビットの期待値は **0.5** ではなく **0.25** である。押しボタン SW “int”（区間(0, 1)の一様乱数の場合のみ表示される）を ON にして 2^{53} 倍した整数のビット平均では 11MSBs が 0 となるが、**MT19937** では（32b 版では 21LSBs も 0 となる）全ビットが期待値 **0.5** に近いこと、乗算合同法で

$a \equiv 5 \pmod 8$である乗数 a に対し最下位 2b が初期値の最下位 2b と同じであることが分る（図下段）。

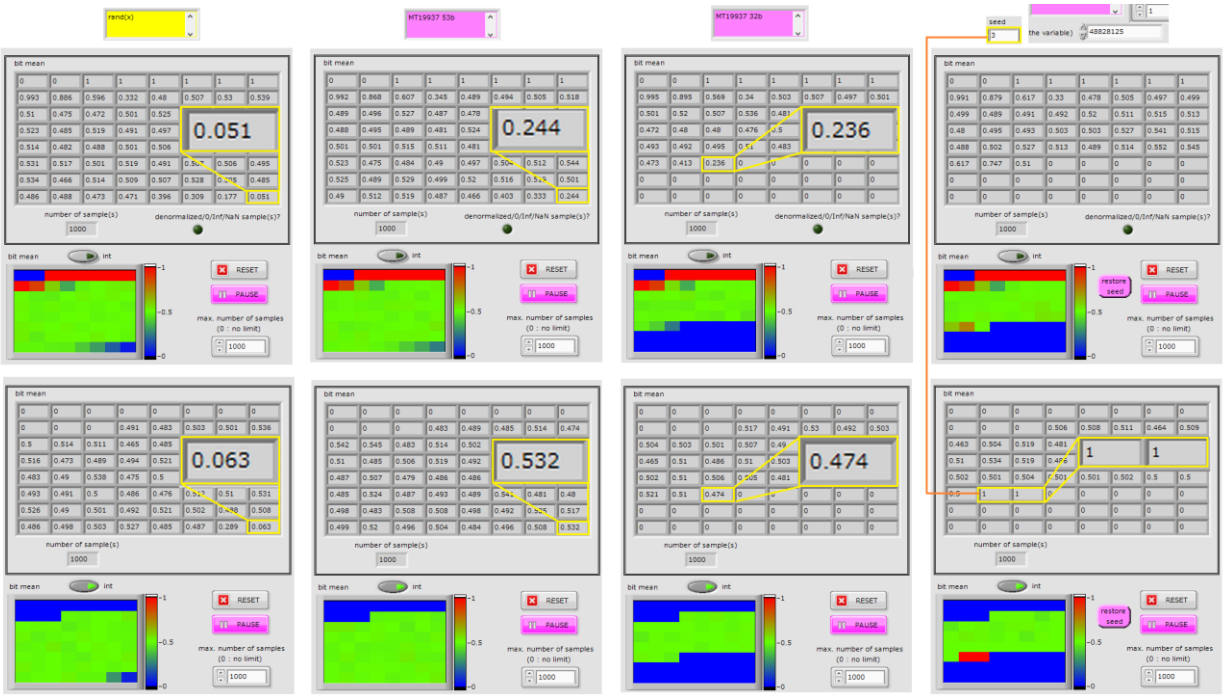

図 23　一様乱数の比較：左から NI ライブラリ、MT19937 53b、MT19937 32b、乗算合同法（乗数 5^{11}）

コピー・ペーストでクリップボードとの間で転送される内容は p.17 図 19 のサブ VI ブロックダイアグラムに示す通りで、次の手順により保存した **sqrt2.txt** をコピー例として **LV_sample.zip** に同梱している。①起動後 “panel”と“force to integer”を OFF、“autocopy”を ON【起動時にコピー対象文字列は空で“CLEAR”のクリックは不要】、②メニューリングで **(x+(2/x))/2** を選択して “RCL”をクリック、③スライドを右端にセット【**x** の初期値を 20 とする】、④"result ⇒ clipboard"をクリック【“autocopy”ON 時に “feedback”クリックで自動コピーされる内容は **feedback 後の演算結果**であるため開始前の初期値に対する結果をコピー】、⑤ “converged”が点灯するまで “feedback”を 9 回クリック、⑥テキストエディタで「貼り付け」、ファイルに保存。

ペーストできる文字列は、以下に例を掲げる 0 個または 1 個の数式文字列とこれに続く（空白 **0x20** を挟んでよい）0 個または 1 個の数値文字列である。数式文字列があるときは何れのモードでも“formula”に読込まれる。数値文字列は$に続く**左詰** 16 進数字（**小文字は大文字に変換**され 16 個に満たない場合は下位に 0 を補い 16 個を超える部分は無視される）または#に続く **LabVIEW の数値入力で許される**文字列（LabVIEW では**”1.2 e 34”**の様に**数値文字列中に空白を入れることはできない**）で$または#のみの場合は 0 となる。数値の読込先は、“panel” ON（ビットパタン指定モード）でパネル表示、OFF（DBL 値指定モード）で “formula”の引数（数式が変数名を含まないときは表示されないが格納はされており、変数名を含む数式文字列を入力したときに表示・使用される）である。

```
x-(getman(x)*2^getexp(x))
20*log(8/pi(2*x+1)^2)
gamma(x)^2$3fe
si(x)*2#1e16
$7FEFFFFFFFFFFFF
#1.2345e-67
```

10. MersennePrime_list.vi　～ HTTP ～

LabVIEW では、ブロックダイアグラムに関数パレットから「データ通信」→「プロトコル」→「HTTP クライアント」の「ハンドルを開く」/「GET」/「ハンドルを閉じる」の 3 個を選んで配線することでウェブ上の HTML データを取得できる（図 24 左上青ブロック）。これにより**表示形式を変えず内容が随時更新**されるページから、編集・加工した最新の情報をフロントパネルに表示できる。残念ながら LabVIEW はインターネットオプションのプロキシ設定を無視しておりプロキシサーバを通した要求はできない（プロキシ経由の HTTP 要求が可能であった NXG は 2020 年に開発が中止され、2022 年 7 月に最終版の延長サポートも終了している）。

素数 p に対し 2^p-1 をメルセンヌ数、これが素数であるときメルセンヌ素数と呼び、現在までに 51 個が見付かっている。**MersennePrime_list.vi** は分散型コンピューティングによりメルセンヌ素数の探索を行っている GIMPS のページ https://www.mersenne.org/primes/ から得た p のリストを編集して 2^p-1 の常用対数の仮数を円周上にプロットし Benford の法則の一例を示す VI である。プロキシを通しているとき（HTTP 要求のタイムアウトでエラー出力の“status”が真になる）には保存されたデータ（備考リストの「定数配列参考プログラム」に記載の方法で定数化した文字型 1D 配列）を使用する。

ウェブページそのもの（GIMPS のページにはメルセンヌ素数の桁数、発見時期、発見者、手法・機器も掲載されている）は、ブロックダイアグラムの関数パレットから「プログラミング」→「ダイアログ&ユーザインタフェース」→「ヘルプ」→「デフォルトブラウザで URL を開く」を配置し、上記 URL を入力して**別ウィンドウ**（註参照）で表示することができる（図 24 左上紫ブロック）。

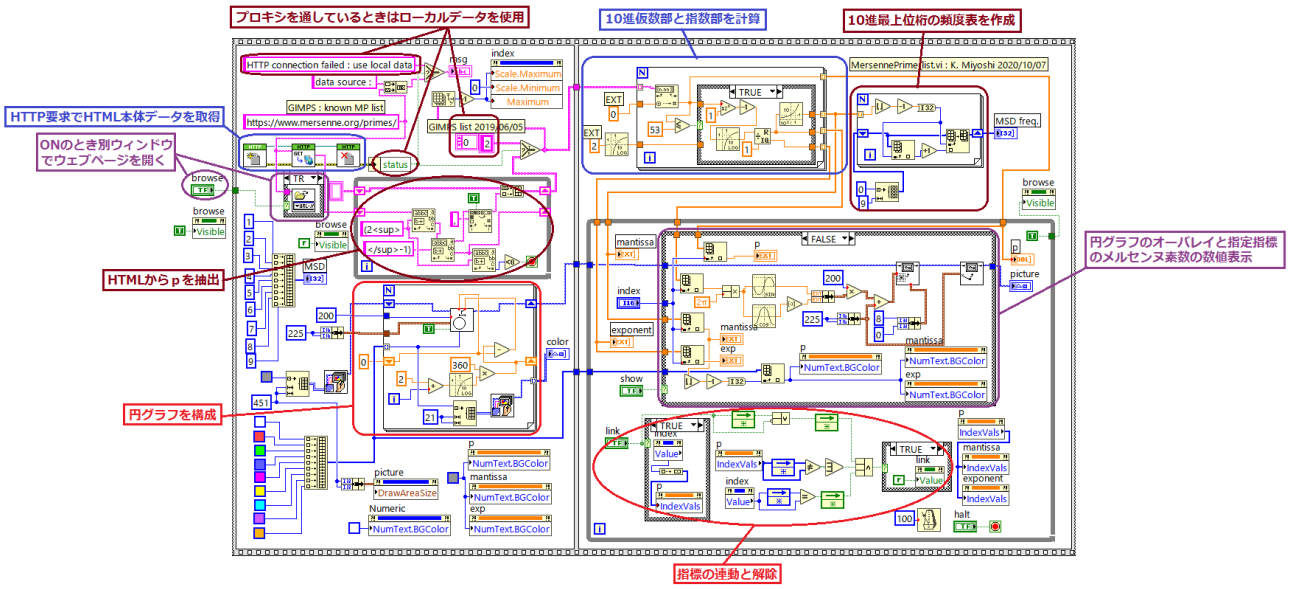

図 24　**MersennePrime_list.vi** のブロックダイアグラム

註：別ウィンドウではなくフロントパネルに表示するには、フロントパネルの制御器パレット既定の「モダン」から「コンテナ」→「ActiveX コンテナ」を選び配置後、右クリックで「ActiveX オブジェクトを挿入...」メニューの「Microsoft Web Browser」を選ぶ（ラベルが“WebBrowser”に変る）。ブロックダイアグラムの関数パレットから「接続」→「ActiveX」→「インボークノード（AvtiveX）」を配置し、コンテナをリファレンス端子に配線し（クラスが“Automation”から“IWebBrowser2”に変る）右ク

リックして「メソッドを選択」で「Navigate」を選ぶ。残念ながら、Internet Explorer と Java の仕様により実行時にスクリプトエラーウィンドウが表示され（無害なコード 0 であるが LabVIEW 側では消すことができない)、何度も「はい(Y)」の応答を求められる。

図 25 に **MersennePrime_list.vi** 実行ファイル起動時のフロントパネルを示す。本 VI も**実行ファイルは停止状態で開く**設定でビルドしている。

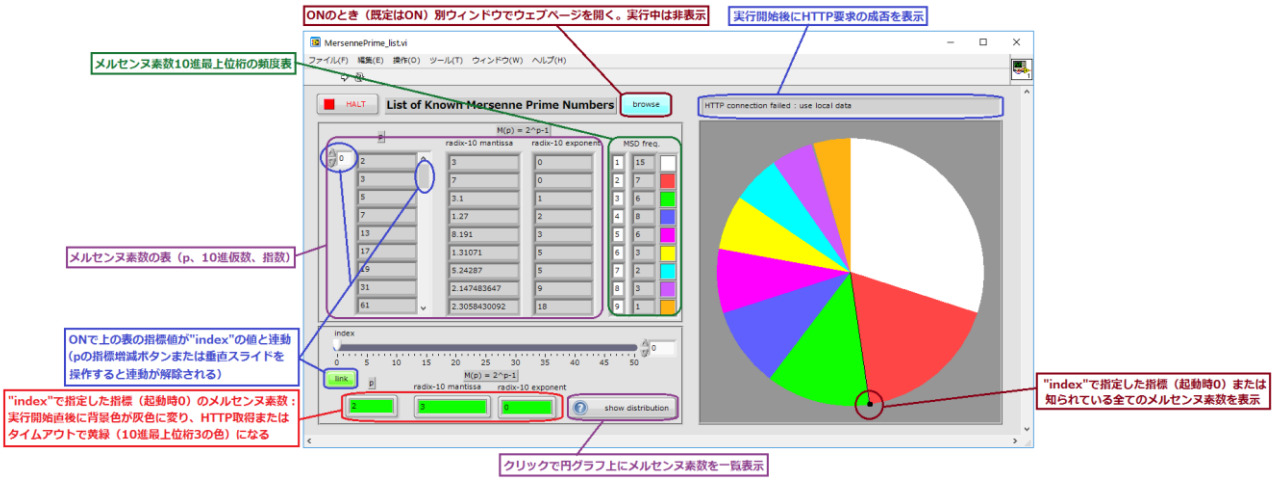

図 25　**MersennePrime_list.vi** 実行ファイル起動時のフロントパネル（停止状態で開く）

図 26 は、実行中のフロントパネル（この図のみ**プロキシを通さない環境で実行**したキャプチャ）の画面で、“show distribution”ボタンをクリックして 51 個の分布を表示(左)、“link”が ON の状態で“index”の水平スライドを操作して 10 進最上位桁が 9 である指標 38（39 番目）のメルセンヌ素数を表示（右）した様子を示す。

既定の設定では実行開始時に別ウィンドウで開く GIMPS のページに見る通り、確率的に 4.58%と数が少ない最上位桁 9 のメルセンヌ素数は、今世紀に入るまで見付からなかった。

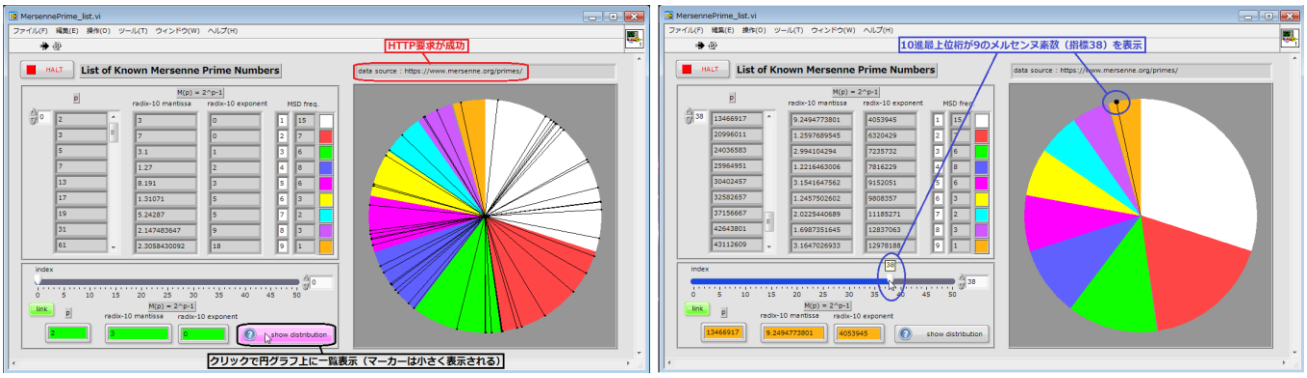

図 26　メルセンヌ素数の一覧表示（左)、指標 38 のメルセンヌ素数の表示（右）

11. EulersEquation.vi　～ 番外 1 ～

複素変数の指数関数 e^z の定義の一つに L. Euler が与えた$\{1+(z/n)\}^n$ の $n\to\infty$ における極限がある。

$z=1$ とおいて自然対数の底 e を求める計算は、収束が $1/n$ のオーダーと遅いため実際に e をこの式で求めてはいけないとされるが、n を 10 の冪、3 の冪と変えて電卓の冪乗計算のアルゴリズムのチェックにも使われる。この式は z が純虚数 iθ の $e^{i\theta} = \cos\theta + i\sin\theta$ に対し、絶対値の収束は $1/n$ であるが偏角は $1/n^2$ で収束する。

複素数 z を $z=\Delta z_1+\Delta z_2+\ldots+\Delta z_k+\ldots+\Delta z_{n-1}+\Delta z_n$ と n 片に分割するとき、$1+\Delta z_k$ の乗積は $\mathrm{Max}\ |\Delta z_k| \to 0$ の極限で同じく e^z に収束するが、複素平面上での経路の選択と分割方法により収束の様子は大きく異なる。**EulersEquation.vi** は、複素平面を実軸・虚軸の 1 単位が 500 画素である 2D ピクチャに描き、このことを確認するものである（註参照）。z が純虚数の場合、半円弧の経路（円弧の中心角が π）では絶対値の誤差は $1/n$ ではなく指数関数的に減少（p.23 図 29）し、また中心角 $2\pi/3$（数式入力の既定値）の円弧では偏角が $1/n^2$ ではなく $1/n^4$ で収束することを確かめられる。

註：本 VI は、LabVIEW2013 版実行ファイルのみを公開している「LabVIEW による Euler 公式 参考プログラム」pp. 9 -11 記載の **LV_Euler0.vi** を LabVIEW2017 版に修正して（“HALT” ボタンを追加し、半円弧経路の等分割で 0⇒i に規格化した経路を表示する既定とし）名称変更したものである。詳しい解説はこちらを参照されたい。http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_Euler.pdf

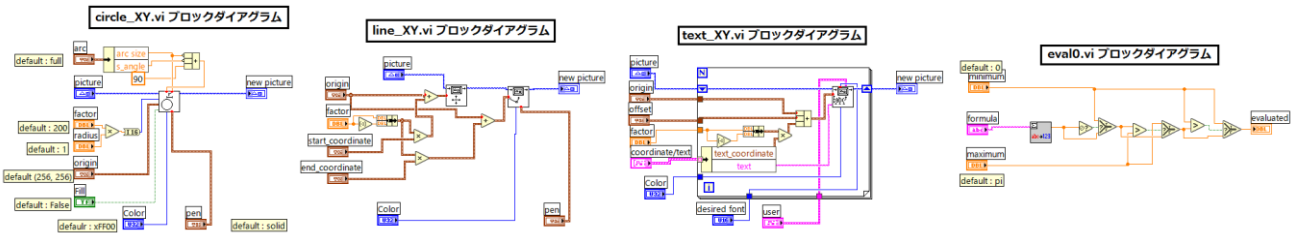

図 27　**EulersEquation.vi** のサブ VI のブロックダイアグラム

図 27 に **EulersEquation.vi** のサブ VI、図 28 に **EulersEquation.vi** のブロックダイアグラムを示す。2D ピクチャの計算はパラメータが変化したときのみ行い（計算終了まで “busy” LED が点灯）、変化の無いときはフィードバックノードを介して前の内容をそのまま使用する。数値制御器の入力値はプロパティのデータエントリの設定で範囲外の値を範囲内に強制できるが、数式文字列による入力にはこの機能がないため、θ および “SET” のクリックで設定される円弧の中心角 ϕ の値は、サブ VI **eval0.vi** によりそれぞれ区間$[0,\pi]$、$[0.001,\pi]$に制限している。

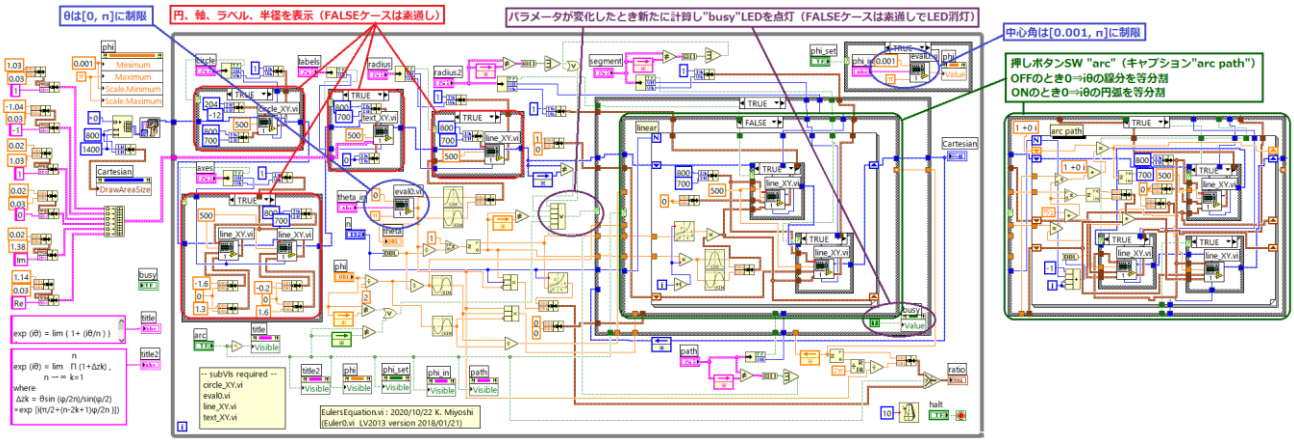

図 28　**EulersEquation.vi** のブロックダイアグラム

図 29 は、半円弧経路の等分割を指定して $e^{i\pi/2} = i$ への収束を確認する例で、2D ピクチャの画素精度では絶対値は分割数が 5（偏角では分割数 33）で収束することが分る。

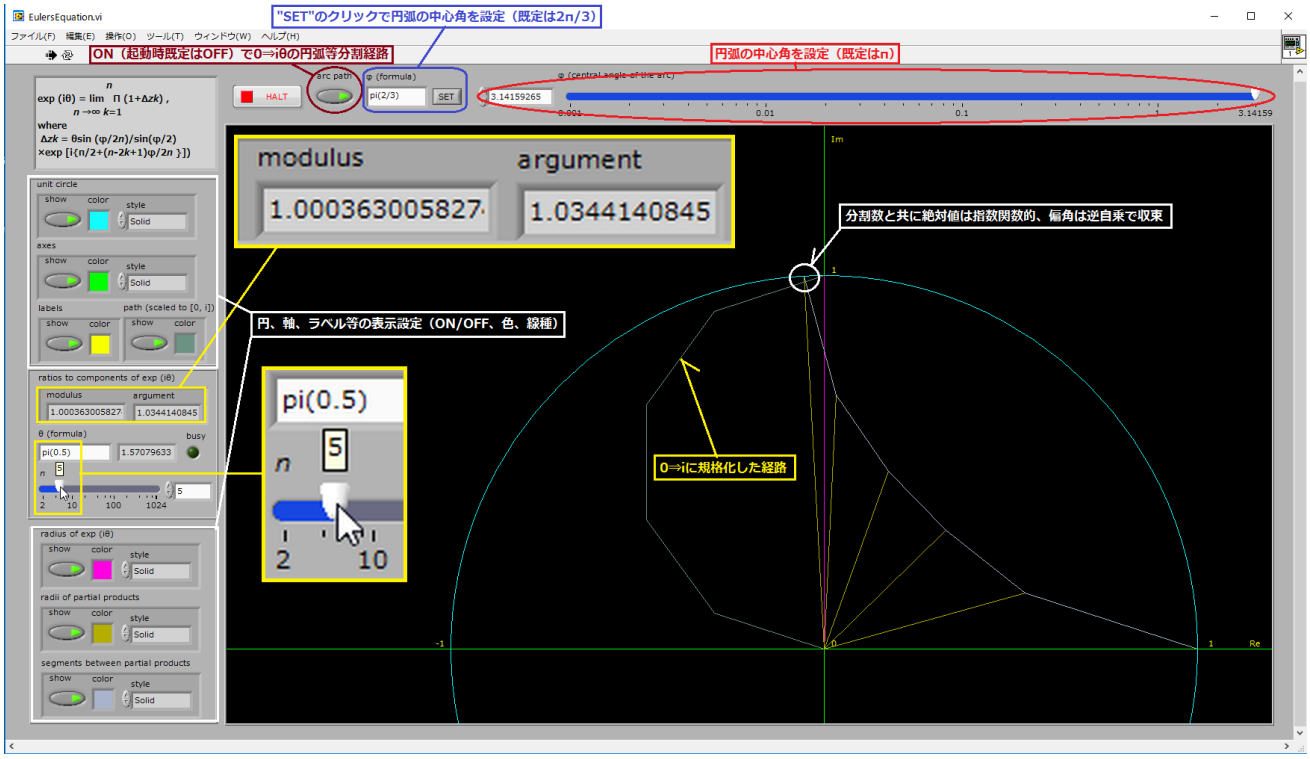

図 29　半円弧経路の等分割では絶対値は急速に収束する：$e^{i\pi/2} = i$ の例（分割数 5）

12. rainbow.vi　～ 番外 2 ～

虹は光の分散により色帯が見られる自然現象であるがプリズムの分光の様に単純ではなく（プリズムの偏角の連想から屈折率の小さい赤を内側に描くのかアニメ、ポスターで虹の色順を逆にした例が後を絶たない）、雨滴内で反射する光線では雨滴に入射した光線と出射する光線のなす角が極値を持つため特定の方向に見えるものである。**rainbow.vi** は単純な幾何光学でこのことを確認するもので、波の性質を考慮せず空中の雨滴を真球と見なせる直径 1～2mm を想定している。図 30 に **rainbow.vi** のサブ VI のブロックダイアグラムを示す。**cossin_deg.vi** は度単位引数を$[0, \pi/4)$に区間還元した余弦・正弦である。

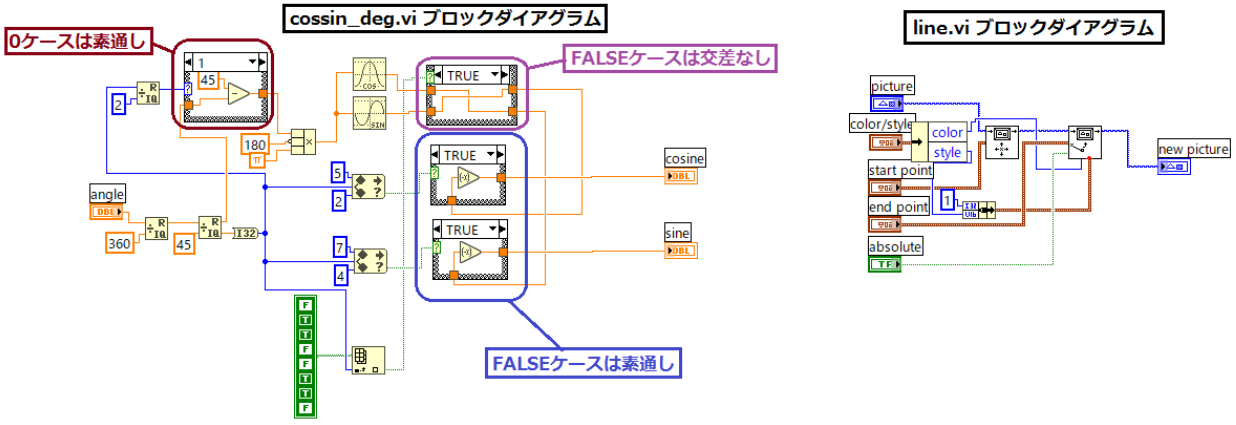

図 30　**rainbow.vi** のサブ VI のブロックダイアグラム

図 31 に **rainbow.vi** のブロックダイアグラムを示す。相対屈折率を設定する数値スライド（ラベル：“n”、キャプション：“refractive index (relative)”）の最大値は CH_2I_2 の空気に対する 1.74 としている。空中の雨滴を想定した押しボタン SW（ラベル：“water”、ボタン表記：“water w.r.t. air”）が ON のとき（相対屈折率の数値スライドは無効化されるがグレーアウトはしない）表示される七色の排他選択押しボタン SW のクラスタ（ラベル：“light” p.25 図 34 右参照）はリファレンス入力のあるプロパティノード（p.9 註参照）を使用して構成している（詳細は「LabVIEW 画像処理参考プログラム」p.14 図 30 参照）。

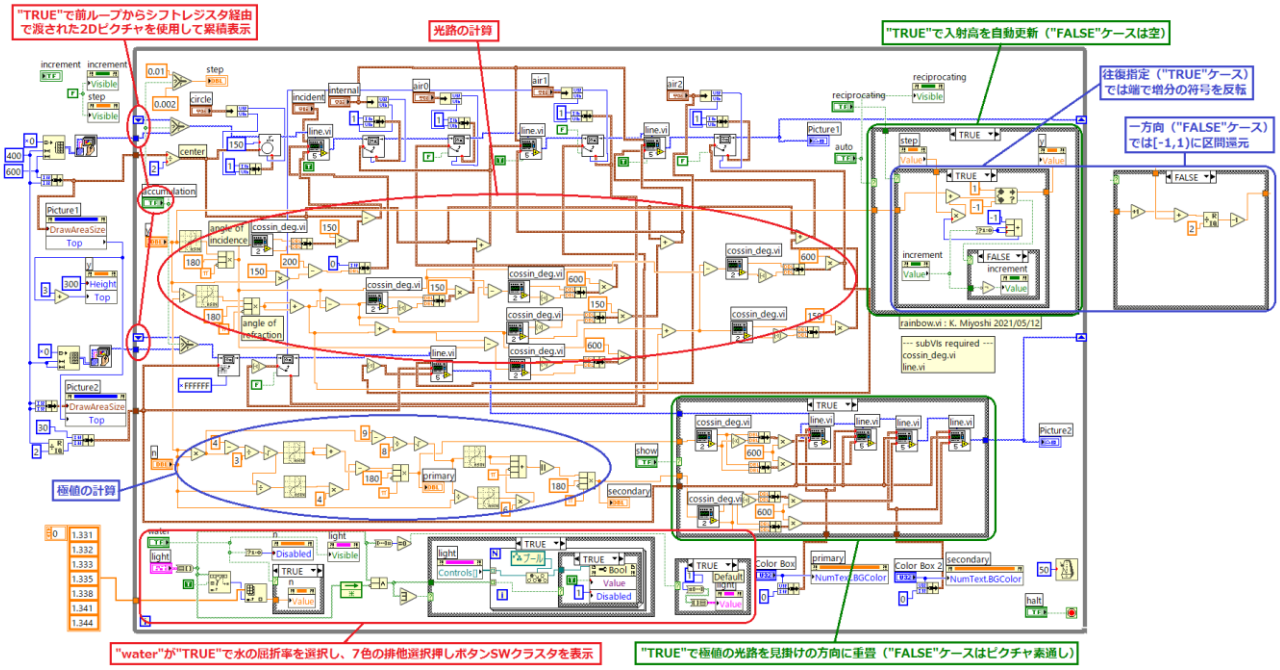

図 31　**rainbow.vi** のブロックダイアグラム

p.25 図 32 に **rainbow.vi** 実行ファイルの起動時のフロントパネルを示す。フロントパネル中段には、光路を含む平面で媒質中の光路（左）と出射光の見掛けの方向（右）を 2D ピクチャで表示し、光線が入射する左側に入射高（衝突係数）設定の数値スライド（ラベル：“y”、キャプション：“incident height”）、フロントパネル上段に描画の色とスタイルを指定する制御器を配置している。フロントパネル下段には左に相対屈折率設定の数値スライドと水を指定する押しボタン SW、右に極値の方向の表示ボタンと数値表示器（極値を度で表示）を配置している。

フロントパネル左上の押しボタン SW “auto” が ON のとき（入射高の制御器は無効化されず数値スライドのノブを操作することはできる。併設した数値ボックスの操作は BackSpace キーと Delete キーで空にして入力する）自動更新され、右に表示される押しボタン SW（ラベル：“reciprocating”、形状：）で一方向（既定）、往復（光線の飛躍が無いが入射高 **±1** の光線を極値と見誤る恐れがある）を選ぶ。その下の押しボタン SW “accumulation” が ON のとき累積表示する。

p.25 図 33 に自動更新が一方向（**0.7→1,-1→-0.25**：図左）、往復（**0.7→1→0.24**：図右）でそれぞれ累積表示した例を示す。

p.25 図 34 は入射高を 0 として極値の表示を目立たせた例で、図左は相対屈折率 **1.312** で 2 個の極値が等しくなることを示し、図右は “accumulation” ON で水を指定し 7 色の押しボタン SW を順次押して各色の極値を重畳した例である。

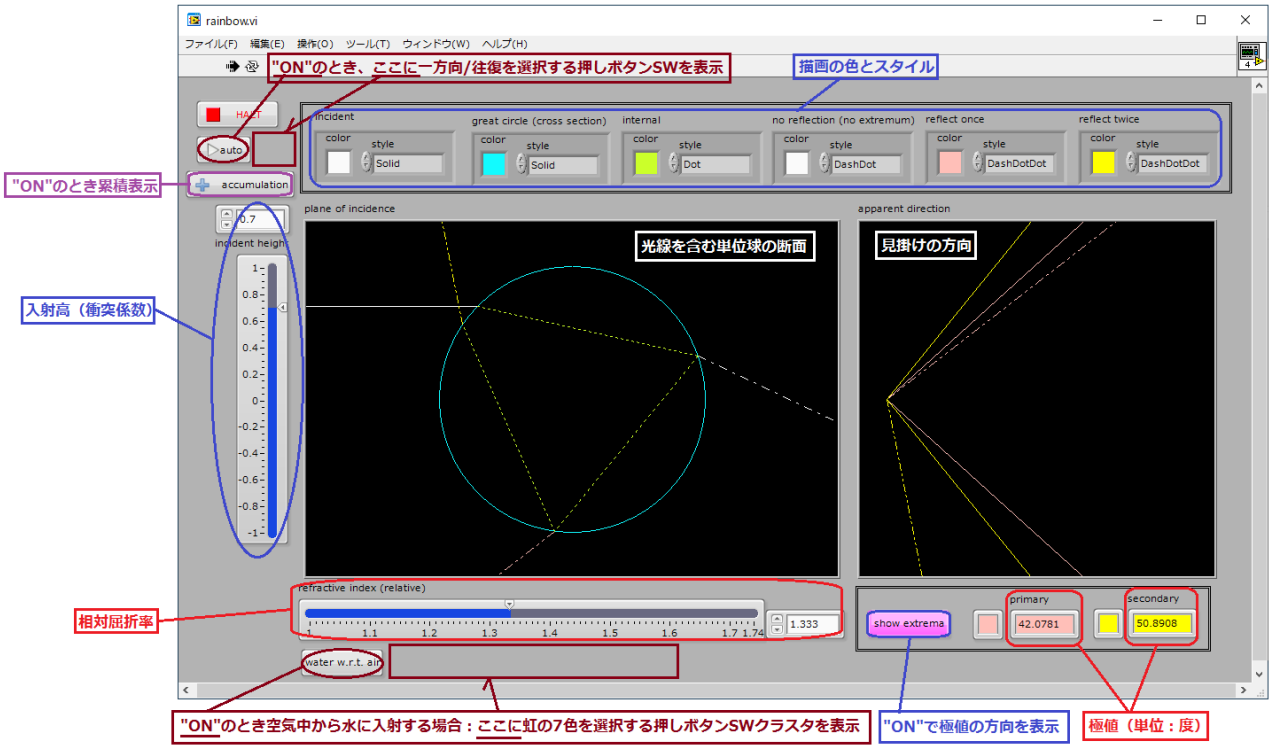

図 32　**rainbow.vi** 実行ファイル起動時のフロントパネル

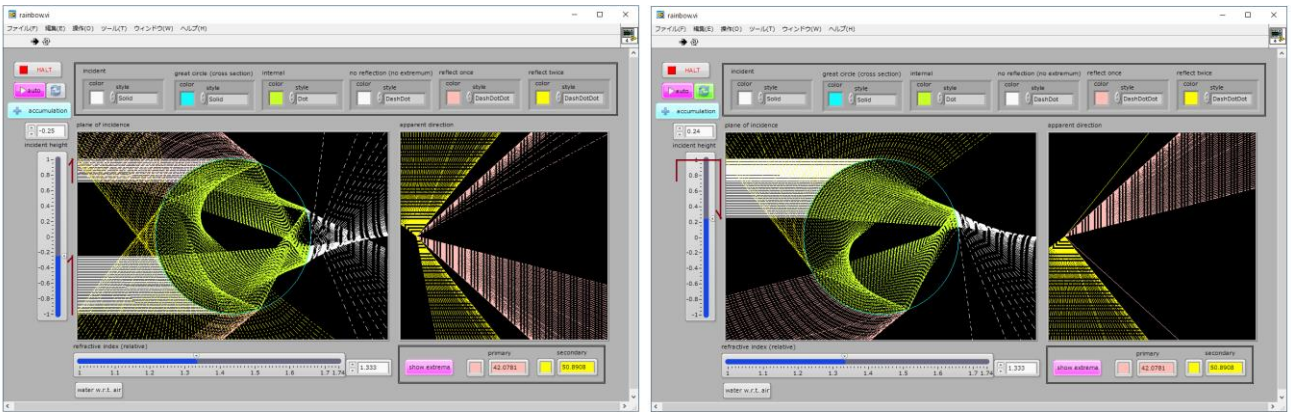

図 33　入射高自動更新の累積表示：一方向（左）、往復（右）

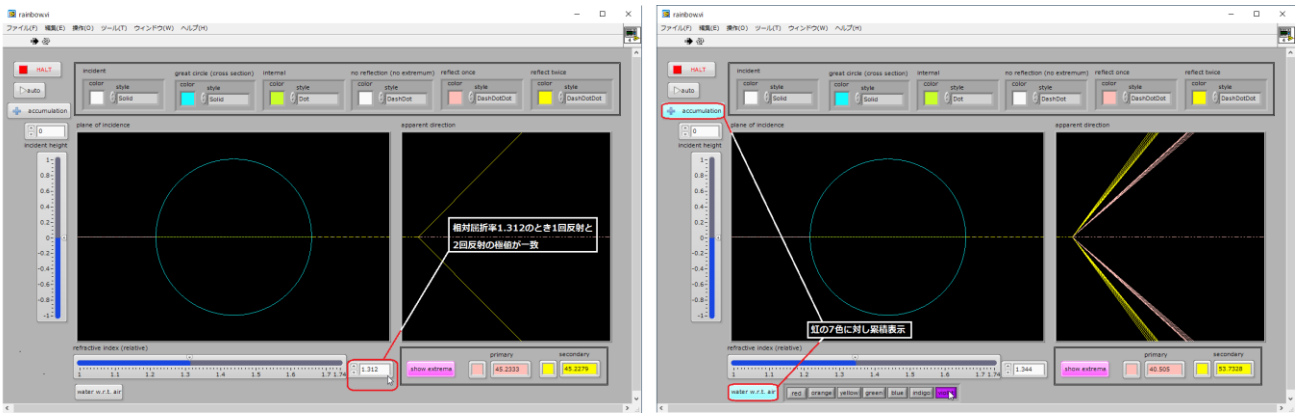

図 34　相対屈折率 1.312 で反射 1 回と 2 回の極値が等しくなる（左）、7 色の極値を重畳（右）

13. ShepardTone.vi　～ 番外 3 ～

ShepardTone.vi は、錯聴の一例**無限音階**と同期して理髪店の回転ポールを表示するデモ VI で「LabVIEW による信号関連参考プログラム」（最終改訂 2018/10/19）pp.10-12 の **LV_Shepard** のサウンド出力を **True_peak_meter.vi** と同様の WaveIO ライブラリに変更し、主 While ループに回転ポール表示の別ループを吸収し回転数（註参照）選択のメニューリングを追加したものである。詳しい解説は、こちらを参照されたい。http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_signal.pdf

註:現在、日本は 90rpm、米国は 60rpm が一般的であるが、1971 公開の米国映画“The Andromeda Strain”（邦題「アンドロメダ」）の理髪店では 30rpm で回転し、占領時代に米軍が接収した銀座松屋百貨店の理髪室も 30rpm で回転した。創業 110 年になる、山岡鉄舟が看板を書いた武蔵野市の理髪店では今も 30rpm で回転している。

図 35 に **ShepardTone.vi** のブロックダイアグラム、図 36 に **ShepardTone.vi** 実行ファイル起動時のフロントパネルを示す。本 VI も**実行ファイルは停止状態で開く**設定でビルドしている。

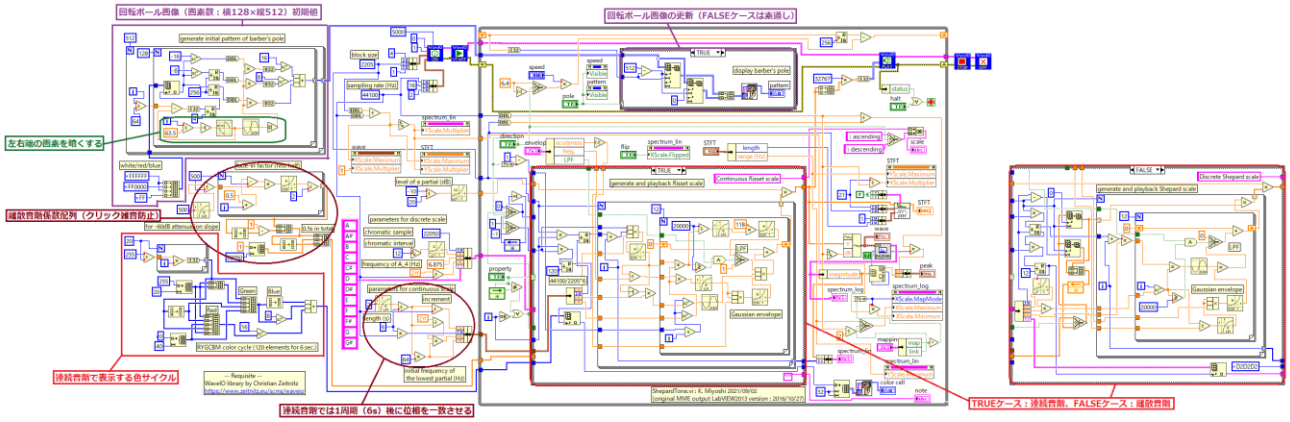

図 35　**ShepardTone.vi** のブロックダイアグラム

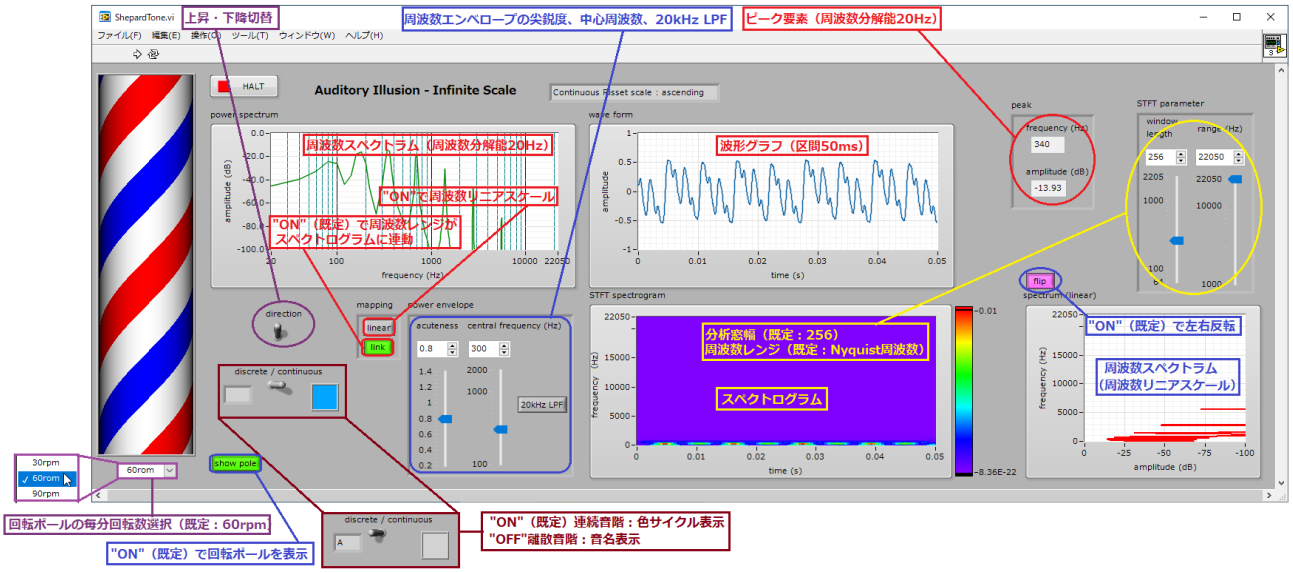

図 36　**ShepardTone.vi** 実行ファイル起動時のフロントパネル（停止状態で開く）

14. num_int.vi　～ 番外 4 ～

無限区間の広義積分または周期関数の 1 周期に亘る積分を数値積分する（無限区間は有限で打切る）場合、関数値の**重みに分点による差を付けない単純な台形則**の精度が圧倒的に良くなる。このことを利用して被積分関数を変数変換により±∞で急速に減衰させ、無限区間での積分に置換える**二重指数関数型数値積分公式（DE 公式）**は、被積分関数が端点で±∞に発散するが積分は収束する広義積分（例は起動時既定の区間（−1, 1）における $1/\sqrt{(1-x^2)}$ の積分：p.28 図 39 参照）でさえも求めることができる。

num_int.vi は被積分関数、積分区間等を数式文字列で与え DE 公式の意味を確認するデモ VI で、実行ファイルのみを公開した「LabVIEW による浮動小数点表現参考プログラム」（最終改訂 2018/12/12）pp. 9-11 に掲載の **LV_num_int.vi**、**LV_DE.vi** を統合して次の修正を行ったものである。詳しい解説は、こちらを参照されたい。http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_IEEE754.pdf

フロントパネル右側（**LV_DE.vi** に対応：ブロックダイアグラムでは下）では、数式文字列のエラーに対して円 LED（キャプション“error”）を入力の文字列制御器に**重ねて**赤色点灯させたものを、当該制御器のテキストカラー背景色を白から赤に変更する方式に改め（註参照）、分点数の数値表示器（ラベル“points”）を追加した。フロントパネル左側（**LV_num_int.vi** に対応：ブロックダイアグラムでは上）では、被積分関数と原始関数の数式文字列に同様のエラーチェックを行い、演算中を示す円 LED（ラベル“busy0”：キャプション“busy”）を追加し、既定の被積分関数を 4 倍した（定積分の値がπとなる）。

註：LabVIEW ではフロントパネルの**同じ場所**に複数のオブジェクトを配置して何れもプロパティで表示を指定するとき、時間的に**後で作成したオブジェクトが優先**する。ブロックダイアグラムをコピーして利用する場合に生成時刻の前後関係は必ずしも保たれないため、ソースプログラムの公開で背景色の変化に改めた。

図 37 に **num_int.vi** が参照する 5 個のサブ VI のブロックダイアグラム、p.28 図 38 に **num_int.vi** のブロックダイアグラムを示す。積分の計算はパラメータが変化したときのみ行い、変化の無いときはフィードバックノードを介して前の内容をそのまま使用するのは p.21 例 11 の **EulersEquation.vi** と同様である。

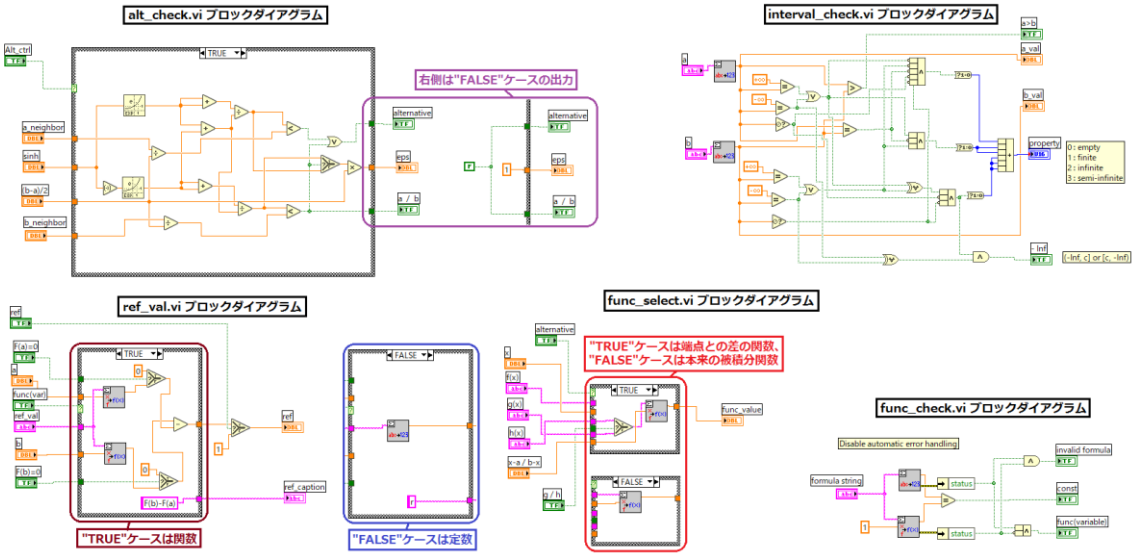

図 37　**num_int.vi** のサブ VI のブロックダイアグラム

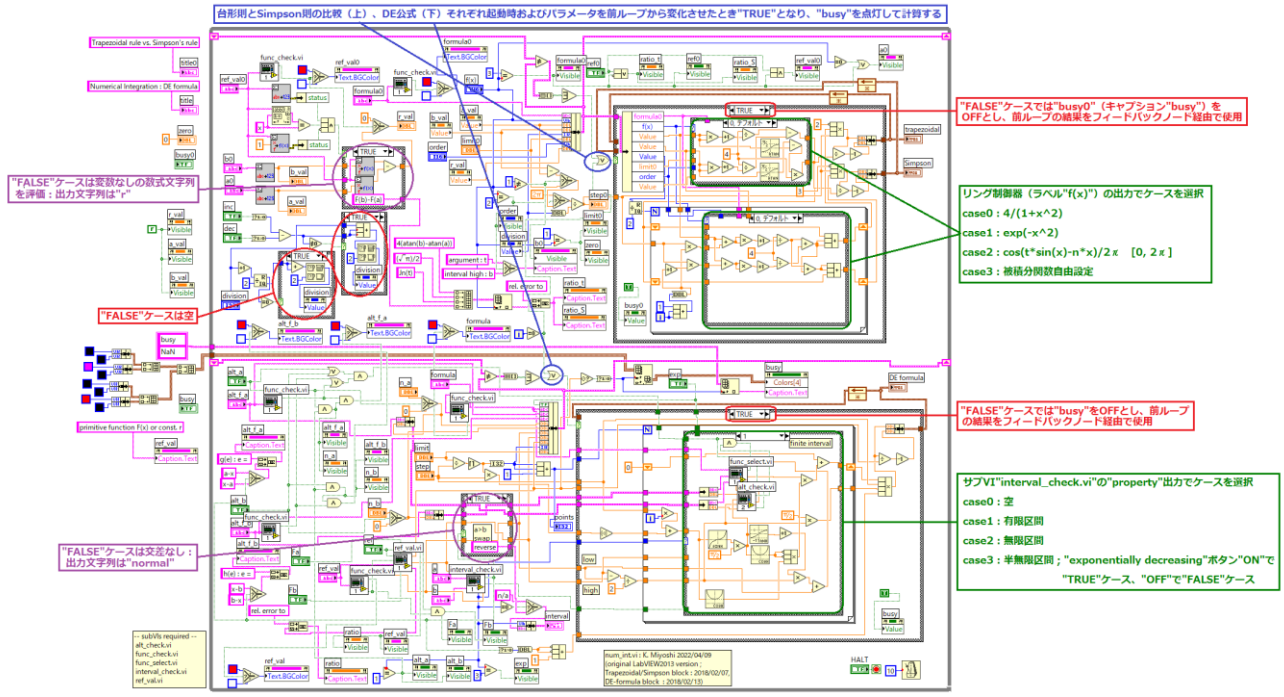

図 38　**num_int.vi** のブロックダイアグラム

図 39 に **num_int.vi** 実行ファイル起動時の既定のフロントパネルを示す。フロントパネル左の積分区間を[0.5, 1.5]に変更（分割数、刻みは不変）すると、台形則の相対誤差が **0.000895509836522868** とほぼ変らないのに対し Simpson 則では**-1.07856707450837E-5** と 2 桁以上悪化する。これが両公式の誤差比較の一般例であり、既定の被積分関数と区間の組合せが特別であることが分る。

フロントパネル右の DE 公式で原被積分関数が発散する例では“alternative”ボタンを“OFF”にすると変数変換後の端点付近で桁落ち、あふれを生じ結果は Inf となる。“limit”をあふれない限界の 3.15 まで下げると相対誤差は DBL 型の精度からは遠い **1.91501925428383E-10** に悪化する。数学公式を用いて計算する場合に計算機のデータ長が有限であることに配慮が必要なことが分る。

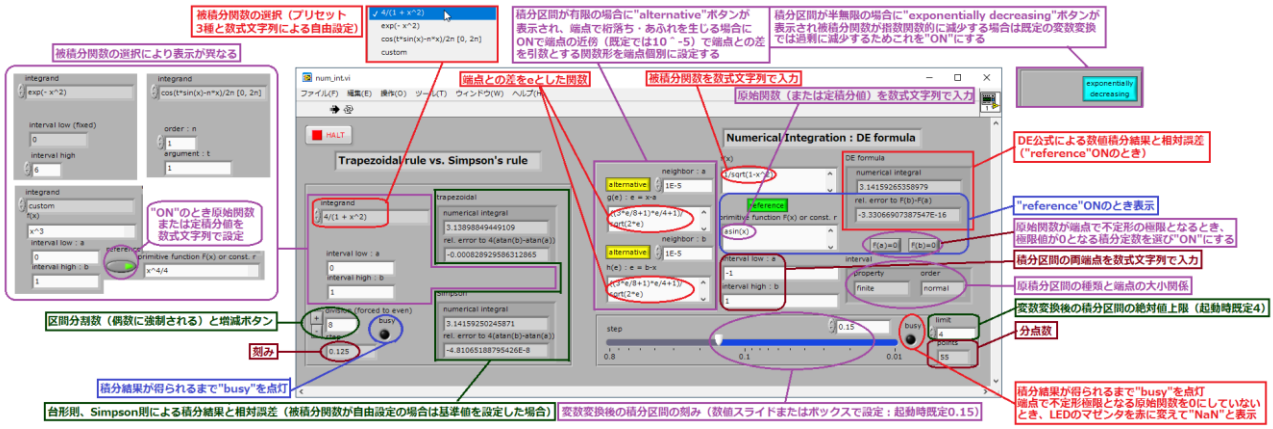

図 39　**num_int.vi** 実行ファイル起動時のフロントパネル

備考

本稿以外の LabVIEW サンプル VI

「情報学実験のページ」https://www.ns.kogakuin.ac.jp/~ct13050/ では LabVIEW 教材と旧課程の情報学実験テキストを公開している。**LabVIEW ソース VI を掲載しているものは**最終改訂の新しい順に次の通りで、**13**、**17** は **LabVIEW2013** 版、**LabVIEW2017** 版、**s** はソース VI（拡張子**.vi**）、**e** は実行ファイル（スタンドアロンアプリケーション：拡張子**.exe**）をそれぞれ意味する。

● 2 の補数参考プログラム **13s**、**13e**【最終改訂 2022/12/19】
https://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_complement.zip

● 楕円軌道参考プログラム **17s**、**17e**【最終改訂 2022/08/16】
https://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_Kepler.zip

● WaveIO 参考プログラム **17s**、**17e**【最終改訂 2021/08/28】pp.6-7 に錯聴のデモ VI がある。
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_WaveIO.zip

● 多倍長演算参考プログラム **17s**、**17e**、**13s**、**13e**【最終改訂 2020/09/27】
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_MP.zip

● 手回し計算機参考プログラム **17s**、**17e**、**13s**、**13e**【最終改訂 2019/11/28】
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_pinwheel.zip

● 画像情報処理参考プログラム **17s**、**17e**【最終改訂 2019/03/31】
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_sample2.zip

● 乱数参考プログラム **17s**、**17e**、**13s**、**13e**【最終改訂 2019/02/14】pp.5-6 に MT19937 乱数使用法の説明がある。
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_random.zip

● 基数変換参考プログラム **17s**、**17e**、**13s**、**13e**【最終改訂 2019/01/14】
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_r_conv.zip

● 信号関連参考プログラム **17s**、**17e**、**13e**【最終改訂 2018/10/19】本稿 p.26 参照
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_signal.zip

● 定数配列参考プログラム **13s**【最終改訂 2017/12/08】LabVIEW/MATLAB の処理結果を定数配列として VI に取込む（ソース VI でのみ可能）プログラム。**LabVIEW2017** 以降の版でも開くことはできる。
http://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LED_scroll.zip

以下のプログラムは互換性等の問題で**実行ファイルのみ**を公開している。

● 浮動小数点表現参考プログラム **17e**、**13e**【最終改訂 2018/12/12】本稿 pp.15-19, 27-28 参照
https://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_IEEE754.zip

● 波形生成参考プログラム **13e**【最終改訂 2018/05/17】WaveIO 参考プログラムに **17s**、**17e** 掲載
https://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_Wave.zip

● Euler 公式参考プログラム **13e**【最終改訂 2018/01/23】本稿 pp.21-23 参照
https://www.ns.kogakuin.ac.jp/~ct13050/johogaku/LV_Euler.zip

旧課程課題テキストを含め信号関連等の教材に登場する efu さんの WaveSpectra/WaveGene のページがリンク切れとなっているが下記の保存サイトからダウンロードできる。
http://web.archive.org/web/20171105052121/http://efu.jp.net/

LabVIEW2017 ヘルプ

LabVIEW2017 では英語版ヘルプが通常の文体の詳細な表現に改められたため、以前の版の日本語版ヘルプに見られた目に余る（恐らく機械翻訳と思われる）誤訳は少なくなったが、機械翻訳とは無関係な例えば図 40 の誤訳例は LabVIEW2020 でも放置されていた（**LabVIEW2022 では改められている**）。

最初の高水準言語 FORTRAN で導入された組込関数 **atan2** は**第 1 引数を y 座標、第 2 引数を x 座標**（2 変数、1 変数の逆正接関数を同時に実装する場合、この方が全体のコードが僅かながら短くなる）とする座標点の偏角を値域 **(－π，π]** で与える 2 変数の 4 象限逆正接関数である。C、Java、MATLAB、VB 等にも同名で用意され関数形は歴史的に **atan2(y,x)** と書かれるが、あろうことか Microsoft Excel では同じ関数名のまま引数の順序を逆にしている。これが原因ではなかろうが、**atan2(y,x)** の日本語ヘルプの記述は VI（図 40 左）、フォーミュラノード関数（図 40 右）の何れも誤っている。

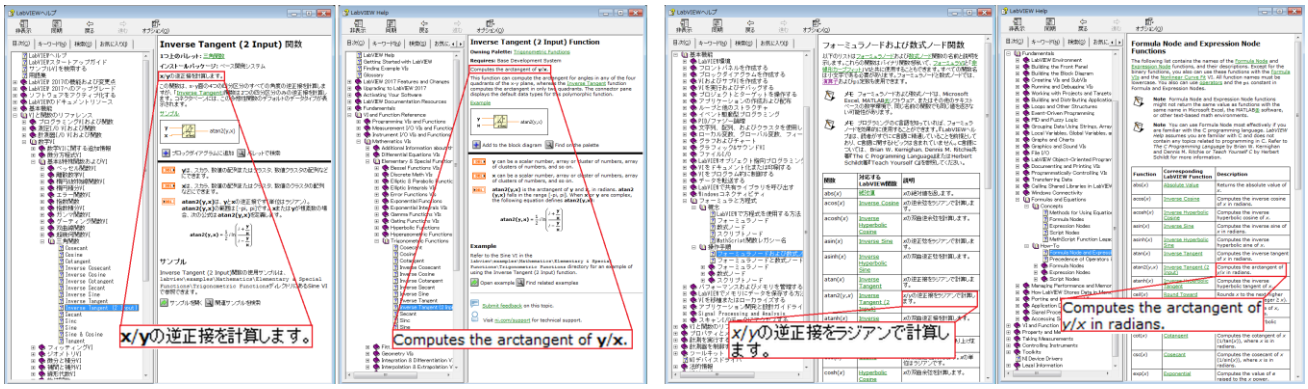

図 40　4 象限逆正接関数の日英ヘルプ対比　関数 VI（左）、フォーミュラノード関数（右）

LabVIEW2017 の日本語版および英語版ヘルプファイルはそれぞれ次の場所にあり、ダウンロードして解凍後、先頭ページの **lvhelp.chm** を開いて利用する。

英語版ファイル：http://www.ni.com/pdf/manuals/371361p.zip

日本語版ファイル：http://www.ni.com/pdf/manuals/371361p_0112.zip

LabVIEW ランタイムエンジン

本学では、より密度の高い学修のため 2022 年度後期より LabVIEW 等の基幹ソフトウェアを BYOD の PC にインストールして行う形態となり、これ以外の個人所有 PC では LabVIEW ソース VI を作成できない。実行ファイルを開くだけであれば **NI User Account を取得**しダウンロードページでログインして保存したランタイムエンジンを PC にインストールすればよい。LabVIEW2017 32b 版で作成された実行ファイルには、**LVRTE2017SP1_f3Patchstd.exe**（ファイルサイズ：363.29 MiB）が対応する。

ダウンロードページ：http://www.ni.com/download/labview-run-time-engine-2017-sp1/7191/en/

LabVIEW2013 32b 版で作成された実行ファイルのランタイムエンジン **LVRTE2013std.exe**（ファイルサイズ：257 MiB）のダウンロードには **NI User Account が不要**である。

http://ftp.ni.com/support/softlib/labview/labview_runtime/2013/Windows/LVRTE2013std.exe

実行時ライブラリとフォント指定について

プログラム例の実行ファイルを開くには **chalkboard.vi**、**ring_counter.vi**、**MersennePrime_list.vi** お

よび **rainbow.vi** の 4 本を除き実行時ライブラリ（拡張子.dll）が必要で、**LV_sample.zip** を解凍して生成される data フォルダを実行ファイルと同じ場所に置いて開く。必要なライブラリは、次の通りである。

Mouse_KB.exe : lvinput.dll
RS_FF.exe : lvinput.dll
audio_input_monitor.exe : lvanlys.dll および lvsound2.dll
parametric_plot.exe : lvanlys.dll
screen_pixel.exe : lvinput.dll
True_peak_meter.exe : lvanlys.dll および WaveIO.dll
DP_property.exe : lvanlys.dll
EulersEquation.vi : lvanlys.dll
ShepardTone.vi : lvanlys.dll および WaveIO.dll
num_int.vi : lvanlys.dll

LabVIEW では VI の作成時にフォントを明示的に指定することもできるが、実行ファイルを作成する場合には（指定フォントがインストールされていない環境では表示できなくなるため）通常は **OS 既定のフォント**を使用している。作成時と実行時のフォントが異なると文字列が枠に収まらない・配列が斜めにずれるなどレイアウトが乱れるため、**実行ファイルと同じ場所に置いた Windows 構成ファイル（拡張子.ini）**でフォントを設定する。同梱の構成ファイルには Yu Gothic UI フォントがインストールされていない Windows 8.1 以前の OS でも実行可能な様に、以下の 4 行が書かれている。

```
[VI ファイル名（拡張子無し）]
appFont="メイリオ" 17
dialogFont="メイリオ" 17
systemFont="メイリオ" 17
```

LV_sample.zip 中の VI のフロントパネルレイアウトは、ソース VI ではなく**実行ファイルを**この構成ファイルの設定で開いた環境に合せている。

LabVIEW のキーコードとラベルについて

LV_sample.zip に同梱のキー一覧 **key_table.txt** は次の様にして NI ライブラリ VI の列挙定数から取得したものである。① **Mouse_KB.vi** のブロックダイアグラムで**「キーボードを初期化」VI（Initialize Keyboard.vi）を配線した**「入力データを取得」VI（Acquire Input Data.vi）をダブルクリックして開く「キーボード取得」VI（keyboardAcquire.vi）のブロックダイアグラムを表示する。② 列挙定数（ラベル “key type”）を**作業用 VI** のブロックダイアグラムに**コピーし、コピーを**右クリックして**制御器に変更**する。③ 制御器となった “key type”を右クリック「作成」でプロパティノードの項目**「文字列[]」**（“Strings[]”）を指定する。④ プロパティ “Strings[]”の出力はキーコードを指標としたキーラベル文字列の 1D 文字列定数配列と同等であり、作業用 VI のブロックダイアグラム上でこれを For ループで処理後テキストファイルに保存する。

数式文字列について

parametric_plot.vi、**DP_property.vi**、**EulersEquation.vi**、**num_int.vi** の 4 本の VI で、数式文字列として解釈できる関数は（数式評価 VI を開き、中で参照しているサブ VI を次々に開いて辿ることで確認はできるが）、**LV_sample.zip** に同梱の **LabVIEWFormulaStrings.pdf** にリストがある（現在リンク切れの http://www.mathmachines.net/Construction/SSSP/LabVIEWFormulaStrings.pdf を保存したもの）。

数式文字列中の演算評価は「数値」パレット中の関数部品を用いた演算とは必ずしも一致しないことに注意する。浮動小数点表現では 0 の表現に正負 2 種類が存在し、DBL 型数値ボックスへの入力で**文字列 “-0”は負号付 0**（MSB のみ 1 で他のビットが 0）と解釈されるのに対し、数式文字列の **“-0”は通常の 0（全ビットが 0）と解釈**される。また浮動小数点数に対して「数値」パレット中の反転（negate : ）は常に MSB を反転するが、**引数 0 を入力した数式 “-x”の結果の MSB は 0 のまま**である。すなわち 0 の符号反転ができないため、例えば **DP_property.vi** で数値ボックスに**"0","-0","Inf","-Inf"**を入力した場合の数式文字列 **“-1/x”**の評価結果は**"-Inf","Inf","0"**（**"-0"**にはならない：図 41 左上）**,"0"**となり、また数式文字列 **“1/(-x)”**の評価結果は**"Inf"**（**"-Inf"**にはならない：図 41 右上）**,"Inf","-0","0"**となる。

数式文字列で結果が非数となる演算（図 41 左下）は関数部品による評価と同じ **FFF8000000000000** を与えるが、数値ボックスに「定数」**"NaN"** を入力した結果（図 41 右下）**7FFFFFFFFFFFFFFF** とは異なる。何れの非数もビット 51 が 1 である quiet NaN である。

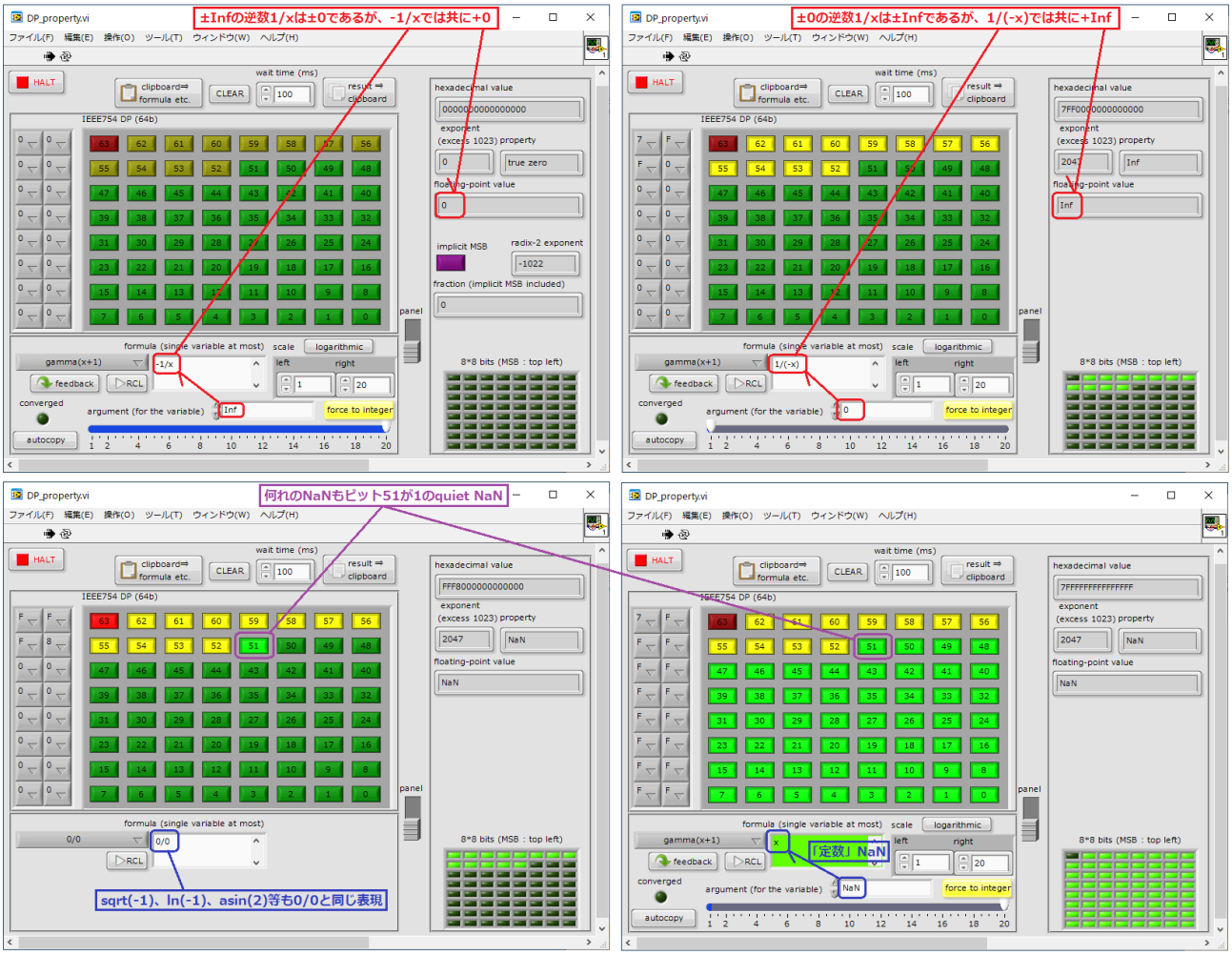

図 41　数式文字列中で 0 は符号反転されない（上）、数式文字列の 0/0 と定数 NaN の表現は異なる（下）