2023/06/28 PRISM 原田

MLT Advan ユーザモジュールの概要

ユーザモジュールとは

お客様自身で開発するプログラムです。MLT Advan 上で動作します。MLT Advan に機能を追加できます。

ユーザモジュールの種類（開発手法）

開発手法の違いで次の 2 種類があります。

・DLL((Dynamic Link Library)タイプ

C/C++言語などで記述し、Microsoft Visual Studio などでビルドして作成します。
2006 年より前の製品では、このタイプのユーザモジュールのみ対応していました。

・スクリプトタイプ

スクリプト言語 pScript で記述して作成します。テキストエディタがあれば開発できます。
2006 年以降の製品で対応しています。

※以下に現れる「ユーザモジュール」は、このタイプのユーザモジュールを指すものとします。

ユーザモジュールの種類（実行形態）

実行形態の違いで次の 2 種類があります。

・ナビゲータユーザモジュール

Windows アプリケーション「MLT Advan Navigator」の上（パソコン上）で動作するユーザモジュールです。MLT Advan Hardware を制御してフレーム送信やポート出力、パルス出力、PWM 出力が行えます。また、受信したフレームやポート入力、アナログ入力を条件に行うこともできます。GUI を使用できるのが特徴です。

・ハードウェアユーザモジュール

MLT Advan Hardware 上で動作するユーザモジュールです。ハードウェアのメモリに展開され、組み込みプログラムにより処理され、実行されます。
GUI は使用できませんが、ハードウェア単体（スタンドアローン）で使用できます。
ハードウェアユーザモジュールは、ナビゲータユーザモジュールの機能縮小版です。

何ができるか?

ハードウェアユーザモジュールに絞って、何ができるかの例を記載します。

・フレーム送信（Navigator で登録したフレーム）

Navigator で設定された手動送信、プログラム送信、イベント/定期送信を実行できます。

例: Send: "cmd:04FF,MG,0,7";　　CAN 手動送信の 8 番に登録したフレームを送信

・フレーム送信（ユーザモジュール内で定義したフレーム）

ユーザモジュール内で定義したフレームを送信できます。

```
例: Send: "CAN,1,0,1,23,8,11,22,33,44,55,66,77,88";
例: const frmA = "CAN,1,0,1,23,8,00,00,00,00,00,00,00,00";
    const frmB = "CAN,1,0,1,23,8,11,22,33,44,55,66,77,88";
    Send: frmA;
```

・フレーム送信（編集して送信）

```
例: var frmC = Frame: "CAN,1,0,4,40,4,00,00,00,00";
    frmC.data.@3 = (frmC.data.@3 + 1) & 0xFF;      // データ 1(D1)をインクリメント
    Send: frmC;
```

・ポート出力

```
例: Send: "cmd:0000,TO,0,8,1";          // ポート出力 9 をオン
例: Send: "cmd:0000,TO,0,8,0";          // ポート出力 9 をオフ
例: Send: "cmd:0000,TO,0,8,2";          // ポート出力 9 を反転
```

・フレーム受信時にポート出力

フレーム受信時に何らかの処理を行いたい場合は、OnReceive ハンドラを登録し、その中で処理します。

```
例: OnReceive: "CAN,1,0,3,30,8", {|rx|
        Send: "cmd:0000,TO,0,8,2";          // 受信の度にポート出力 9 が反転する
    }
```

・フレーム受信時にポート出力（条件付き）

例：
```
OnReceive: "CAN,1,0,3,30,8", {|rx|
        (rx.data.@3 == 0x00).if_true: {
            Send: "cmd:0000,T0,0,8,0";    // D1 が"00"ならポート出力 9 を OFF
        }, {
            (rx.data.@3 == 0xFF).if_true: {
                Send: "cmd:0000,T0,0,8,1";// D1 が"FF"ならポート出力 9 を ON
            }
        }
    }
```

・ポート入力でフレーム送信

ポート入力で何らかの処理を行いたい場合は、OnReceive ハンドラを登録し、その中で処理します。

例：
```
OnReceive: "st:Port Status", {|rx|
        (rx.protocol == 0x00).if_true: {          // ポート入力ステータス
            ((rx.data.@1 & 0x01) == 0).if_true: { // ポート入力 1 が Lo になったら
                Send: "CAN,1,0,6,60,8,FF,FF,FF,FF,00,00,00,00";
            }
        }
    }
```

注：ポート入力ステータスは、何れかのポートに変化があったときに現れます。このため、複数のポート入力を使う場合は、対象ボート変化を判定する必要があります。

・1 秒後にポート出力をする

時間経過後に何らかの処理を行いたい場合は、OnIdle ハンドラを登録し、その中で処理します。

例：フレーム受信の 1 秒後にポート出力する
```
    var procTm = -1;
    OnReceive: "CAN,1,0,2,20,8", {|rx|
        procTm = TickCount + 1000;                // 1 秒後
    }
    OnIdle: {
        (procTm != -1 && procTm <= TickCount).if_true: {
            Send: "cmd:0000,T0,0,8,2";
            procTm = -1;
        }
    }
```

・Advan 動作開始時にポート出力をする

動作開始時に何らかの処理を行いたい場合は、OnStart ハンドラを登録し、その中で処理します。

例: OnStart: {

```
        Send: "cmd:0000,T0,0,F,1";          // ポート出力 16 をオン
    }
```

・アナログ入力値に応じて・・・

・パルス出力を・・・

・PWM 出力を・・・

・イベント/定期送信のフレームを変更する

・応答送信のフレームを変更する

・などなど