Asyril
Asycube プロバイダ

ユーザーズガイド

Version 1.1.0

May 12, 2017

備考:

[改訂履歴]

バージョン	日付	内容
1.0.0	2016-01-26	初版発行
1.1.0	2017/05/12	raw コマンド送信と実行シーケンスを追加

内容

1. はじめに

この文書は Asyril 製フィーダシステム Asycube 用の CAO プロバイダである、Asycube プロバイダのユーザーズガイドです。

Asycube プロバイダはイーサネット TCP/IP メッセージのみを用いてフィーダと接続し、すべてのローレベル通信を実施します。シリアルインタフェース(RE232/RS485)を用いたフィーダの場合はシリアル-イーサネットコンバータが必要です。

2. プロバイダの概要

2.1. 概要

このプロバイダは Asycube フィーダの基本制御コマンドへの直接アクセスを可能にします。
フィーダの 3 つの主要部分（バルク、プレート、バックライト）は、CaoController::Execute() の呼び出しを用いたコマンドを介して処理（データ入力・呼び出し）されます。
振動設定と調整には、Asyril HMI ソフトウェアを使用してください。

表 2-1 Asyril Asycube プロバイダ

ファイル名	asycubeDIL.dll
ProgID	CaoProv.Asyril.Asycube
レジストリ登録	regsvr32 asycubeDIL.dll
レジストリ解除	regsvr32 /u asycubeDIL.dll

2.2. フィーダに必要なファームウェア要件

このプロバイダは、最低でも以下のファームウェア要件満たす Asycube フィーダを必要とします。

ファームウェア: V 2.2.0

Asycube モデル 50 および 80.ファームウェア V3.0.0

2.3. メソッド＆プロパティ

2.3.1. CaoWorkspace::AddController メソッド

構文

AddController (<CtrlName>, <ProvName >,<ExecMachineName>,<OptionsStr>)

CtrlName : [In] コントローラ名。任意の文字列 （例."Feeder"）

ProvName : [In] プロバイダ名("**CaoProv.Asyril.Asycube**"に固定）

ExecMachineName : [in] プロバイダを実行するコンピュータ名 **使用しない。空欄とする。**

OptionStr : [In] オプション文字列

オプション	意味
Conn=<接続パラメータ>	接続パラメータを設定。 現時点では、このプロバイダで有効な通信プロトコルは TCP のみです。
Timeout=<タイムアウト時間>	TCP ソケット通信のタイムアウトを設定。単位 [ms] ここで設定したタイムアウト時間内に TCP 接続が確立されなかった場合、タイムアウトエラーが生じます。
NodeNo=<アドレス番号>	物理回転スイッチが機器の背面に装備されている RS232/485 フィーダモデルの場合は必須。 NodeNo パラメータは回転スイッチの選択と一致する必要があります。 整数型範囲 [0-15] イーサネット接続のみの Asycube の場合、このパラメータは省略できます。
Mode=<RawComChannel>	ここでは説明しない。Asyril 用パラメータ。
Descriptor=<Descriptor chain>	ここでは説明しない。Asyril 用パラメータ。

例

RS232/485 Asycubes のプログラム例 (製品名 "mezzo", "forte")

```
cao.AddController("Feeder1", "CaoProv.Asyril.Asycube", "",
"conn=TCP:192.168.127.253:4001,Timeout=2000,NodeNo=1")
```

イーサネット専用 Asycube のプログラム例 (product name "largo", "50","80", "240")

```
cao.AddController("Feeder1", "CaoProv.Asyril.Asycube", "",
"conn=TCP:192.168.127.253:4001,Timeout=2000")
```

3. コマンドリファレンス

3.1. CaoController::Execute(“<Command name>”) コマンド

構文 <Command>,<Value>

注意: 振動設定は Asyril HMI インタフェースを用いて調整する必要があります。.

Asycube の文書に記載された標準の方向キーワードを用いた制御コマンドで、容易に制御できます。

下記の図 1 は、基本方向と関連する制御コマンドをまとめたものです。表 1 は全ての制御コマンドを表します。高度な使用方法については、付属 1 の「設定＆管理コマンド一覧」を参照してください。

コマンド構文の注意:**コマンド文字列は大文字・小文字を区別しません。**

スクリプト実行時の注意: **初期設定では、アクションが完了するまで、すべてのコマンドにおいてスクリプト実行(同期呼び出し)がブロックされます。**しかし、コマンドのうちいくつかには”！”記号の使用が許可されており、それを用いることでブロックされることなく同時コマンド実行(同期呼び出し)が可能です。

例 ″!Move.Forward″.

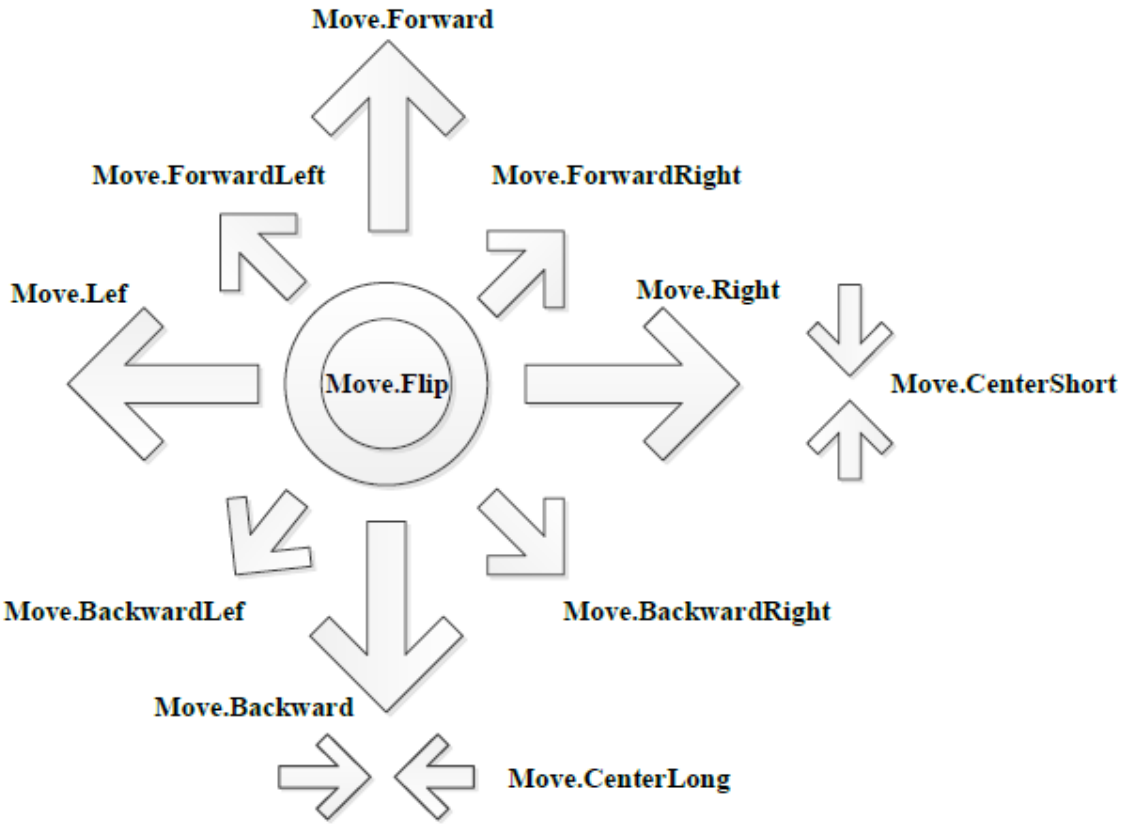

図 1.移動方向と対応するコマンド

3.2. 制御コマンド一覧

表 1.コマンド一覧

<table>
<tr><th>#</th><th>〈コマンド〉</th><th>〈値〉</th><th>説明</th></tr>
<tr><td>1</td><td>″ComChannel.Send″</td><td>TextCmd [文字列]</td><td>Raw テキストコマンドをフィーダに送ります。応答を Raw 文字列として返します。
フィーダ固有のプロトコル書式を使用。
＜例＞
バイブレーションバッチを選択 :
″ComChannel.Send″,″[UV1]″
戻り値 :
″[UV01]″

シーケンスを実行:
″ComChannel.Send″,
″[ES:(NbParts;NbPartsMax;XCenter;YCenter;SeqID]″
戻り値
″[ES:(NbParts;NbPartsMax;XCenter;YCenter;SeqID;Duration)]″

書式についての詳細はお使いの Asycube の「Programming Manual」を参照してください。

このコマンドはスクリプトをブロックしません。特定の時間長の振動が呼び出されたときは、振動時間の長さを手動で抽出し、その振動が終わるのを待たなければなりません。</td></tr>
<tr><td>2</td><td>″[!]Feed.〈FeedDirection〉″</td><td>時間 [ms]
[1]</td><td>バルク搭載の Asycube に対してのみ使用します。
〈FeedDirection〉キーワードで指定された方向に、一定の時間(ms)、バルクを振動させます。
<table><tr><th>#</th><th><FeedDirection></th></tr><tr><td>1</td><td>"Forward"</td></tr><tr><td>2</td><td>"Backward"</td></tr></table>
例. ″Feed.Forward″</td></tr>
<tr><td>3</td><td>″Feed.ReadState″</td><td>–</td><td>現在のバルク状態を読み取ります。4 段階の整数型の値が返されます。
<table><tr><th>状態値</th><th>振動</th></tr><tr><td>0</td><td>無効</td></tr><tr><td>1</td><td>停止中</td></tr><tr><td>3</td><td>振動中</td></tr><tr><td>5</td><td>オーバーヒート</td></tr></table>詳細は「Programming manual」を参照してください。</td></tr>
<tr><td>4</td><td>″Feed.Stop″</td><td>–</td><td>現在のバルク振動を直ちに停止します。</td></tr>
</table>

<table>
<tr><td>5</td><td>″[!]Move.<<i>MoveDirection</i>>″</td><td>時間 [ms]
[1]</td><td><MoveDirection>キーワードで指定された方向に、一定の時間(ms)、プラットフォームを振動させます。

<table><tr><th>#</th><th><<i>MoveDirection</i>></th></tr><tr><td>1</td><td><i>″Forward″</i></td></tr><tr><td>2</td><td><i>″Left″</i></td></tr><tr><td>3</td><td><i>″Right″</i></td></tr><tr><td>4</td><td><i>″ForwardLeft″</i></td></tr><tr><td>5</td><td><i>″ForwardRight″</i></td></tr><tr><td>6</td><td><i>″Backward″</i></td></tr><tr><td>7</td><td><i>″BackwardLeft″</i></td></tr><tr><td>8</td><td><i>″BackwardRight″</i></td></tr><tr><td>9</td><td><i>″Flip″</i></td></tr><tr><td>10</td><td><i>″CenterShort″</i></td></tr><tr><td>11</td><td><i>″CenterLong″</i></td></tr></table>例. <i>″Move.Forward″</i></td></tr>
<tr><td>6</td><td>″Move.ReadState″</td><td>–</td><td>現在のプラットフォームの状態を読み取ります。
4 段階の整数型の値が返されます。

値 : 振動
0 : 無効
1 : 停止中
3 : 振動中
5 : オーバーヒート

詳細は「Programming manual」を参照してください。</td></tr>
<tr><td>7</td><td>″Move.Stop″</td><td>–</td><td>現在のプラットフォームの振動を直ちに停止します。</td></tr>
<tr><td>8</td><td>″Backlight.Flash.Set″</td><td>時間 [ms]</td><td>バックライトのフラッシュ時間を[ms]で設定します。
備考: サイクル毎にフラッシュ時間を設定する必要はありません。この値は RAM に保存されます。

より良いタイミングで実行するには、フラッシュ時間を変更する必要がある時のみ、Flash.Set を使用してください。</td></tr>
<tr><td>9</td><td>″[!]Backlight.Flash″</td><td>–</td><td>予め″Backlight.Flash.Set」コマンドで設定した時間長だけ、バックライトフラッシュを起動します。</td></tr>
<tr><td>10</td><td>″Backlight.Set″</td><td>[True/False]</td><td>バックライトを永久的にオン(又はオフ)します。

備考: バックライトが長時間にわたりオンのままの場合、オーバーヒートを防ぐ目的で、自動的にオフになります。
バックライトを再度有効にするには、
″ClearAlarms」コマンドを参照してください。
詳細は「Programming manual」を参照してください。</td></tr>
</table>

<table>
<tr><td>11</td><td>″Backlight.ReadState″</td><td>–</td><td>バックライトの起動状態を返します。
戻り値は現在の状態を表す整数型です。
0 : → バックライトはオフ
1: → バックライトはオン</td></tr>
<tr><td>12</td><td>″Outputs[<OutNo>].Analog[<AnalogNo>].Set″</td><td>出力レベル [%]</td><td>出力を搭載した Asycube に対してのみ使用します。

出力開始時に適用されるアナログ出力レベル(番号 1 又は 2) を設定します。
Output[<OutNo>].Start コマンドを参照してください。
詳細は「Programming manual」を参照してください。</td></tr>
<tr><td>13</td><td>″[!]Outputs[<OutNo>].Start″</td><td>時間 [ms]
[1]</td><td>出力を搭載した Asycube に対してのみ使用します。

標準出力番号1又は2を、指定された時間(ms)だけ開始します。

備考: 設定に応じて、2 つのデジタル出力と、2つのアナログ出力が有効化されます。
詳細は「Programming manual」を参照してください。</td></tr>
<tr><td>14</td><td>″Outputs.ReadState″</td><td>–</td><td>現在のプラットフォームの状態を読み取ります。
3 段階の整数型の値が返されます。
<table><tr><th>状態値</th><th>出力</th></tr><tr><td>0</td><td>無効</td></tr><tr><td>1</td><td>停止中</td></tr><tr><td>3</td><td>起動中</td></tr></table>
詳細は「Programming manual」を参照してください。</td></tr>
<tr><td>15</td><td>″Outputs.Stop″</td><td>–</td><td>出力を搭載した Asycube に対してのみ使用します。有効化された全ての出力を停止します。
備考: 継続的に有効化されている出力を停止するのにも使用します。</td></tr>
<tr><td>16</td><td>″Din[<DinNo>].Read″</td><td>–</td><td>出力を搭載した Asycube に対してのみ使用します。

2 つのデジタル出力のうち一つの状態を読み取ります。整数型を返します。
0 → False,　1 → True</td></tr>
</table>

17	″ReadWarning″	–	警告レジスタの値を返します。. 戻り値が「0」の場合, 警告が無いことを意味します。 戻り値はフィーダのモデルによって異なります。 注 1: 返されるコードは、整数で表記された長さ8のレジスタです。この値は最大8個の独立したアラームを表します。詳細は「Programming manual」を参照してください。 「ClearWarnings」コマンドを参照してください。 注 2: Largo の製品と併用する場合、ビット4とビット5に、2つのデジタル入力の状態が返されます。
18	″ClearWarnings″	–	警告レジスタをクリアします。
19	″ReadAlarms″	–	アラームレジスタの値を返します。 戻り値が「0」の場合はアラームが無いことを意味します。 アラームは、フィーダをアラーム状態にします。リセットするには、″ClearAlarms」コマンドを参照してください。 備考: 返されるコードは、整数値で表記された長さ8のレジスタです。この値は最大8個の独立したアラームを表します。 詳細は「Programming manual」を参照してください。
20	″ClearAlarms″	–	アラームレジスタをクリアします。

[1] 時間の引数は以下の方法で解釈されます。
a) 整数値 > 0 → 引数として与えられた値が適用されます。
b) 整数値 = 0 → 振動が継続します (無限時間).「Move.Stop」 コマンドを参照してください。
c) 空文字列 ″″ → 対応するバッチに保存されている時間長が適用されます。
例. feeder.Execute(″Move.Forward″,150) .150[ms]の間、振動します。
feeder.Execute(″Move.Forward″,0) .″Move.Stop″が実行されるまで振動を続けます。
feeder.Execute(″Move.Forward″,″″) .バッチ「a」に保存された時間の長さの間、振動します。

[!] コマンド名の先頭に[!]が書かれたコマンドは、同期呼び出しをします。

3.3. エラーメッセージ

以下の表に、プロバイダで定義されたエラーメッセージを記します。

表 2.エラーメッセージ

#	Id	文字列 表記	メッセージ	状況	よくある原因
#1	1	NODE_ADDR_OUT_OF_RANGE	"Node address out of range"	cao.AddController(…) をコールしたとき。	- AddController の「NodeNo」引数が範囲外 [0-15]。
#2	2	TCP_CONNEXION_TIMEOUT	"TCP connection timeout"	cao.AddController(…) をコールしたとき。 AddController の引数で指定されたタイムアウトの時間内に TCP 接続ができなかった。	- イーサネットが接続されていない (物理的に) TCP/IP ネットワークパラメータが一致しない。 タイムアウトパラメータが小さすぎる。(標準的な接続では、約 10[ms]が必要です。)
#3	3	TCP_SEND_ERROR	"TCP send error"	内部通信を使用する何らかのプロバイダの Execute をコールしたとき。 Socket->send()呼び出しによって生じる。	- TCP ソケットエラー
#4	4	RESPONSE_TIMEOUT	"Response timeout"	通信を使用する何らかのプロバイダの Execute ファンクションをコールしたとき。	- イーサネット接続遮断 - TCP 接続遮断 -「NodeNo」の値が正しくないため、機器がコマンドを受け入れない。 - 機器が応答していない (電源オフ等)。
#5	5	CMD_NOT_FOUND	"Command not found"	プロバイダが "Execute」ファンクションを呼び出したとき。 AsycubeModel->Execute()コマンドで発生。	有効でないコマンド文字列。 注意.コマンド文字列は大文字小文字を区別しません。
#6	6	INVALID_CMD_ARGUMENT	"Invalid command argument "	プロバイダが "Execute」ファンクションを呼び出したとき。 ReadRegister 又は WriteRegister コマンドにより発生。	レジスタの数が偶数でない。
#7	7	INVALID_BATCH_IDENTIFIER	"Invalid command identifier "	プロバイダが "Execute」ファンクションを呼び出したとき。バッチ識別子引数をとるコマンドを使用したとき。	バッチ文字が有効な識別子でない。 有効なバッチ識別子[A…Z]

4. サンプルプログラム

以下は、Denso PacScript で書かれた、パーツ供給及び拡散シーケンスパターンのサンプルプログラムです。
この状態では、Asycube はすでに特定のパーツを供給するように設定されています。全ての振動設定を調整するために、別途、Asyril のユーザインタフェースを使用する必要があります。

この例では、適切な拡散シーケンスを選択するためにパーツ平均位置を用いた、最適化されたフィーディングシーケンスを用いています。
拡散は**2ステップの振動シーケンスを実行することで行われます。**
1) パーツをセンタリングする　2) パーツをフリップする
センタリング方向は、プラットフォームを 9 つに分割した区域と比較したときの、パーツ平均位置によって決まります。タイミングは、プラットフォーム中心へ向かう距離に応じ、また、線形関係を考慮して算出されます。プログラム例を単純なものにするため、このサンプルプログラムでは、この計算は実施されていません。

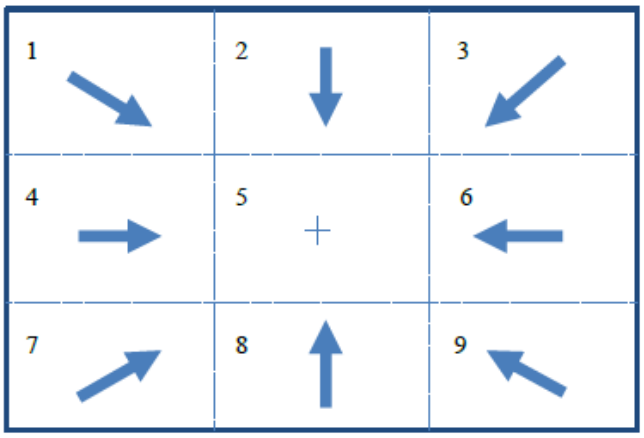

図 2.センタリング方向はパーツ平均位置によります。

ゾーン番号	拡散系列
1	ForwardRight + Flip
2	Right + Flip
3	BackwardRight + Flip
4	Forward + Flip
5	Flip
6	Backward + Flip
7	ForwardLeft + Flip
8	Left + Flip
9	BackwardLeft + Flip

表 3.パーツの位置に応じた拡散系列

```
!TITLE "ACube_SampleCode"

' Author: Asyril
' このプログラムは Aeycube を試験的に実施するための典型的なコマンドを示します。
' このプロバイダは TCP/IP 通信でのみ動作します。.RS232/485 インタフェースのみを備えた Asycube モデルの場合は、'
TCP/IP <-> RS232/485 コンバータが必要です。
'
'*****************************************************************************************************
‘Asycube はこのスクリプトからでは設定できません。振動設定は Asyril のユーザインタフェースを用いて実施してください。
'*****************************************************************************************************

Sub Main

    Dim feeder as Object
    On Error Goto ErrorMgmt

    ' Asycube コントローラのインタフェースを作成
    ' TCP/IP を用いたフィーダのフィーダオブジェクトを作成

    feeder = cao.AddController("Feeder1", "CaoProv.Asyril.Asycube", "",

    "conn=TCP:192.168.127.254:4001,Timeout=10000") Dim partNbr as Integer = 0
    Dim feedLimit as Integer = 5
    Dim partLocZone as Integer  = 0
    Dim mvCenterDuration as Integer = 0

    '*************************************************************************
     ' ここにパーツの数と集約情報を取得するためのビジョンのダイアログを挿入。
    ' partNbr = VisionSystem.QueryPart(...)
    ' partLocZone = VisionSystem.QueryPartLocZone(...)
    ' mvCenterDuration = VisionSystem.QueryCenterDistanceDelay(...)
    ' バックライトとカメラを同期するには"BackLight.Flash" コマンドを使用。
    ' feeder.Execute("!Backlight.Flash","")を呼び出すか、"synchroBackLight"のデジタル入力を実施
    '*************************************************************************

    '必要であればさらにパーツを供給
    If partNbr < feedLimit Then
        call feeder.Execute("Feed.Forward",1000)
    End if

    ‘ パーツ標準集約ゾーンに応じた適切な振動を選択することでプラットフォーム上にパーツを分散させる

    Select Case partLocZone
        Case 1
            call feeder.Execute("Move.ForwardRight",mvCenterDuration)
        Case 2
            call feeder.Execute("Move.Right",mvCenterDuration)
        Case 3
            call feeder.Execute("Move.BackwardRight",mvCenterDuration)
        Case 4
            call feeder.Execute("Move.Forward",mvCenterDuration)
        Case 5
            'Case 6 は振動無し
            call feeder.Execute("Move.Backward",mvCenterDuration)
        Case 7
            call feeder.Execute("Move.ForwardLeft",mvCenterDuration)
        Case 8
            call feeder.Execute("Move.Left",mvCenterDuration)
        Case 9
            call feeder.Execute("BackwardLeft", mvCenterDuration)
```

```
    End Select

    'フリップを実行し、パーツを拡散させる

    call feeder.Execute("Move.Flip",300)

    ''接続遮断
    Disconnect :
    cao.Controllers.Remove feeder.Index
    feeder = Nothing
    Exit Sub

    ErrorMgmt:
    ' エラー処理
    PrintMsg
    "Error..."Resume
    Disconnect

End Sub
```

5. 付属 1. 設定＆管理コマンド一覧

警告! 誤使用の場合、フィーダに対して矛盾する設定ができるような高度なコマンドを以下に紹介します。

#	<コマンド>	<値>	説明
1	"[!]Bulk.Batch[<batchCode>].Start"	時間 [ms] [1]	<batchCode>引数で与えられたバッチパラメータを用いて、一定時間(ms)だけバルクを振動させます。 <batchCode> argument.[a…z]
2	"[!]Platform.Batch[<batchCode>].Start"	時間 [ms] [1]	<batchCode>引数で与えられたバッチパラメータを用いて、一定時間(ms)だけプラットフォームを振動させます。 <batchCode> argument.[a…z]
3	"Bulk.Batch[<batchCode>].Read"	–	特定のバルクバッチのパラメータを文字列で返します。 文字列は値をセミコロンで区切られた「ベクトル」としてフォーマットされます。 数値はフィーダの種類によります。詳細は「Programming manual」を参照してください。 (Command LB[A..Z])
4	"Bulk.Batch[<batchCode>].Write"	–	特定のバルクバッチのパラメータを文字列で指定します。 文字列引数は値をセミコロンで区切られた「ベクトル」としてフォーマットしなければなりません。 数値はフィーダの種類によります。詳細は「Programming manual」を参照してください。 (Command SB[A..Z])
5	"Platform.Batch[<batchCode>].Read"	–	特定のプラットフォームバッチのパラメータを文字列で返します。 文字列は値をセミコロンで区切られた「ベクトル」としてフォーマットされます。 数値はフィーダの種類によります。詳細は「Programming manual」を参照してください。 (Command LC[A..Z])

6	″Platform.Batch[<batchCode>].Write″	−	特定のプラットフォームバッチのパラメータを文字列で指定します。 文字列引数は値をセミコロンで区切られた「ベクトル」としてフォーマットしなければなりません。 数値はフィーダの種類によります。詳細は「Programming manual」を参照してください。(Command SC[A..Z])
7	″Register[<RegNo>].Read″	−	<RegNo>引数で指定されたレジスタの値を返します。 RegNo は偶数の整数とします。 詳細については「Programming manual」を参照してください。

8	"Register[<RegNo>].Write"	レジスタの値 [整数型]	<RegNo>引数で指定されたレジスタの値を書き込みます。 注意: RegNo は偶数の整数とします。プロバイダは内部では奇数アドレス(RegNo+1)を計算しています。 詳細については「Programming manual」を参照してください。
10	"GetLibVersion"	–	プロバイダのソフトウェアバージョンを文字列として返します。 フォーマットは以下の通り: [libName=AsyucbeDILProv]-[LibVersion=<*version label*>]-[ReleaseDate=<*release date*>]

[!] コマンド名の先頭に[!]が書かれたコマンドは、同期呼び出しをします。