Roboticsware



Automation シリーズ

ロガーアクション詳細マニュアル

RWMN-5-21-921

ようこそ **4**

本マニュアルについて 4
仮想デバイスとは 5

１章　ロガーアクションの概要 **6**

ロガーアクションとは 6
ログの保存形式 6
ロガーアクションを利用する際の基本手順 7
ロギング処理の流れ 8

２章　はじめてのロギング **9**

２－１　一般型ログの設定例（定周期ログ） 9
概要 9
ステップ１　タグの準備 9
ステップ２　アクション設定 11
ステップ３　イベント設定 13
動作確認 15
２－２　イベント型ログの設定例 16
概要 16
ステップ１　タグの準備 16
ステップ２　アクション設定 16
動作確認 20

３章　ロギングの機能解説 **21**

ロギングの詳細フロー 21
DB のメンテナンス（自動削除） 24
通信エラー発生時の動作設定 26
ＰＣの時刻が戻った時の処理 28
PLC の値が一定範囲外の時、欠損値とする方法 29
ログデータのキャッシュ設定とスクリプトの振る舞い 30
イベント型ログへ複数のタグ値を含める 30
イベント型ログにおけるタグ値変化の検出周期 31

４章　ロギングの実践 **32**

４－１　タグイベントによるロギング（SAMPLE01） 32
概要 32
設定手順 32
動作確認 33
このサンプルの応用例 33
４－２　ログの開始・停止を制御する（SAMPLE02） 34
概要 34
設定手順 34
解説 35
このサンプルの応用例 36
４－３　計算結果をロギングする（SAMPLE03） 37
概要 37
設定手順 37
解説 38

４－４　配列タグを使った大量データのログ（SAMPLE04）..... 40
概要..... 40
設定手順..... 40
解説..... 42
このサンプルの応用例..... 42
４－５　イベント型ログにアラームコメントを出力（SAMPLE05）..... 43
概要..... 43
設定手順..... 43
解説..... 44
４－６　ODBC データベースへのログ（SAMPLE06）..... 45
概要..... 45
設定手順..... 45
解説..... 49
４－７　ロギングの処理要求を PLC から受け取る（SAMPLE07）..... 50
概要..... 50
設定..... 50
解説..... 51
４－８　イベント型ログフォーマットで定周期にログをとる（SAMPLE08）..... 52
概要..... 52
設定手順..... 53
解説..... 54
４－９　CSV ログのファイル名を動的に変更する（SAMPLE09）..... 55
概要..... 55
設定手順..... 56
解説..... 58

ようこそ

本マニュアルについて

本書「ロガーアクション詳細マニュアル」は、ロガーアクションの利用に関する詳細な解説書です。ロガーアクションは本パッケージが提供するアクションの一つで、強力かつ柔軟なロギング機能を提供するものです。ユーザーズマニュアルと併せてお読みいただくことを推奨いたします。また、アクション全般に関するリファレンスである「アクションマニュアル」も必要に応じて参照してください。

仮想デバイスとは

本パッケージは現在 100 機種相当との通信に対応しており、その対応機種の 1 つとして、「仮想デバイス」があります。「仮想デバイス」は PLC をシミュレートしたアドレス空間であり、特定の PLC との通信を行わないという点を除き、標準的な PLC デバイスと同等に扱えます。本マニュアルでは、特定の PLC に限定して説明することを避けるため、「仮想デバイス」を用いて説明を進めます。「仮想デバイス」での扱いを理解すれば、その他の標準的な PLC でも同様の手順で設定が行えます。

仮想デバイス一覧

デバイス	アドレス	説明	サイズ	読み	書き
入力リレー	X0000～X9999	10 進数	ビット	○	○
出力リレー	Y0000～Y9999	10 進数	ビット	○	○
内部リレー	M0000～M9999	10 進数	ビット	○	○
特殊リレー	SM0000～SM9999 **(※1)**	10 進数	ビット	○	○
データレジスタ	D0000～D9999	10 進数	ワード	○	○
特殊レジスタ	SD0000～SD9999 **(※2)**	10 進数	ワード	○	○

(※1)特殊リレー（SM）について

SM0000 10ms ごとに On/Off
SM0001 50ms ごとに On/Off。
SM0002 100ms ごとに On/Off
SM0003 500ms ごとに On/Off
SM0004 1000ms ごとに On/Off
SM0005 2000ms ごとに On/Off
SM0006 3000ms ごとに On/Off
SM0007 5000ms ごとに On/Off

(※2)特殊レジスタ（SD）について

SM0000 10ms ごとに On/Off
SD0001 50ms ごとにインクリメント
SD0002 100ms ごとにインクリメント
SD0003 500ms ごとにインクリメント
SD0004 1000ms ごとにインクリメント
SD0005 2000ms ごとにインクリメント
SD0006 3000ms ごとにインクリメント
SD0007 5000ms ごとにインクリメント

SD0008～SD0015 は、それぞれ SD0000～SD0007 の値を用い、0～100 の間でサインカーブを描く。

1章　ロガーアクションの概要

ロガーアクションとは

ロガーアクションは、PLC から収集したデータを CSV や ODBC データベースへ履歴データとして保存するためのアクションです。高速性を重視した処理方式により、ミリ秒オーダーでの高速なロギングが可能です。また、一旦キャッシュを介してロギング処理を行うことにより、より正確なタイムスタンプがログデータに付与されます。

ログの保存形式

ログデータの保存形式には、大きく分けて「一般型ログ」と「イベント型ログ」の２つの形式があります。一般型ログは一定周期、あるいは定刻にログデータをフィールド毎に保存していく方法です。イベント型ログはタグ値の変化や、タグ値がリミット値を超えた時などをトリガとしてログデータを保存していく方法です。
また、上記の２つの形式を混在させて出力することもできます。この方式のことを「混在型ログ」と呼びます。

1．一般型ログ

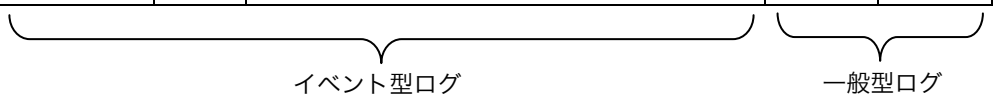
時刻	ms	温度１	温度２	温度３	圧力	バルブ１	バルブ２
2003/1/1 10:12:00	10	10.5	32.8	40.7	5.5	TRUE	FALSE
2003/1/1 10:12:10	10	10.7	32.6	40.6	5.4	FALSE	TRUE
2003/1/1 10:12:20	10	10.8	32.4	40.8	5.3	TRUE	FALSE
・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・

2．イベント型ログ

時刻	タグ	値	コメント
2003/1/1 15:11:12	温度１	15.1	上限値を超えました。装置をチェックして下さい。
2003/1/1 15:21:18	圧力	5	正常値に戻りました。
2003/1/1 15:23:25	バルブ１	1	バルブＯＰＥＮ。
2003/1/1 15:28:12	温度１	15	正常値に戻りました。
・・・	・・・	・・・	・・・

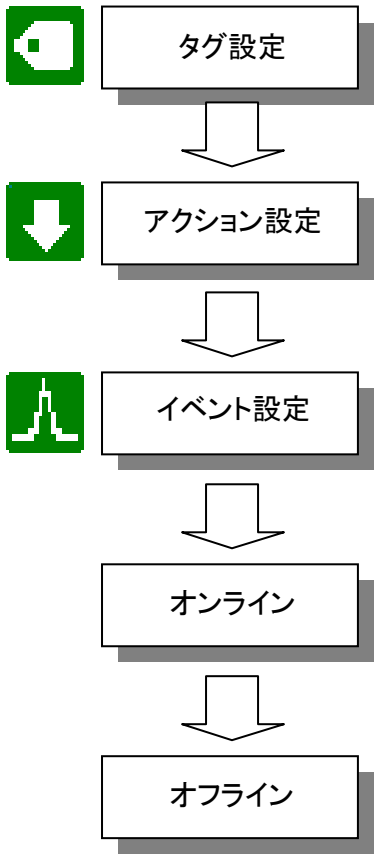
3．混在型ログ

時刻	タグ	値	コメント	バルブ１	バルブ２
2003/1/1 15:11:12	温度１	15.1	上限値を超えました。装置をチェックして下さい。	FALSE	FALSE
2003/1/1 15:21:18	圧力	5	正常値に戻りました。	FALSE	FALSE
2003/1/1 15:23:25	バルブ１	1	バルブＯＰＥＮ。	TRUE	FALSE
2003/1/1 15:28:12	温度１	15	正常値に戻りました。	TRUE	FALSE
・・・	・・・	・・・	・・・	・・・	・・・



ロガーアクションを利用する際の基本手順

以下は、ロガーアクションを利用するための基本的な手順です。



PLC のどのエリアと通信を行うのかを定義します。また、必要に応じ PLC とのハンドシェイク用のタグや計算用のタグを作成します。

ロガーアクションの設定を行います。
データベースの基本設定及びフィールドのレイアウトを定義します。

ロガーアクションを実行するトリガとなるイベントを設定します。

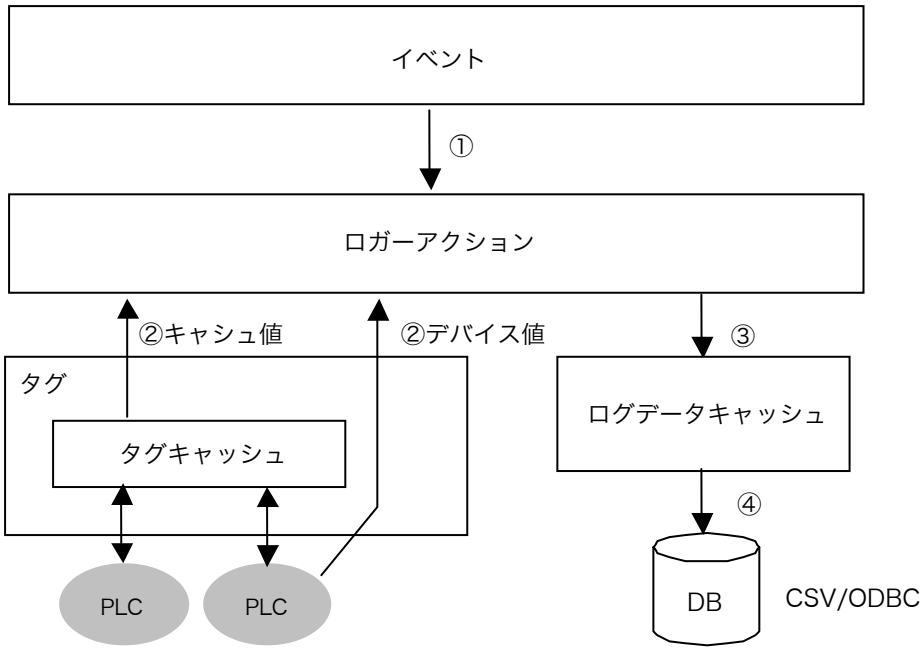
オンラインにするとロギングが始まります。
ログの開始、終了、ログの周期設定などの制御も可能です。

オフラインにするとロギングが終了します。

ロギング処理の流れ

ロガーアクションによるロギング処理は、トリガとなるイベントから実行される事によって行われます。このイベントの発生条件をどのような設定にするのかによって（定周期的なロギングを行うのか、タグ値がある条件を満たす時のみイベントログを出力したいのか等)、ロギングに関する様々な要件を満たすことができます。

下図は、ロギング処理のフローです。



①イベント発生

ロガーアクションはイベントの発生をトリガとして実行されます。ロガーアクション１回の実行で、１レコードがDB へ保存されます。イベントには「定周期イベント」「定刻イベント」「タグイベント」の３種類があります。

②データ収集

ロガーアクションがイベントにより実行されると、必要なデータを収集します。デバイス値によるロギング、又はタグキャッシュ値によるロギングのいずれかを選択できます。

③ログデータのキャシュへの保存

データ収集で集められたログデータは、DB へ保存する前に一旦メモリ上のキャシュに保存されます。その後、随時DB へ保存されます。

④DB への保存

キャッシュ値を DB に保存します。１件毎にディスクに書き出す以外にも、ハードディスクへの負荷を軽減する目的で複数レコード分まとめてハードディスクに書き出す機能（フラッシュ設定）が用意されています。

２章　はじめてのロギング

２－１　一般型ログの設定例（定周期ログ）

概要

ここでは、実際にロガーアクションを使って定周期ロギングを行うための簡単な設定手順について解説します。ここで紹介する手順は、大きく分けて以下の３ステップです。

ステップ１	タグの準備	タグの設定を行います。
ステップ２	アクション設定	ロガーアクションを設定します
ステップ３	イベント設定	トリガとなる定周期イベントを設定します。

この例では、１秒ごとに SD0000 から SD0009 までのタグ値をログデータとして CSV ファイルに保存します。

時刻	SD0000	SD0001	・・・	SD0008	SD0009
2003/1/1 10:12:00	0	0	・・・	0	0
2003/1/1 10:12:01	0	0	・・・	0	0
2003/1/1 10:12:02	0	0	・・・	0	0
・・・	・・・	・・・	・・・	・・・	・・・

ステップ１　タグの準備



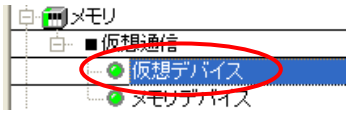
タグ設定

下記の手順に従ってタグの設定を行います。ここでは、仮想デバイスの「SD タグ」を用いて PLC をシミュレートします。SD タグは自動的に値が変化してくれるタグです。

１．ユニットの追加



画面左側のツリーから、「Driver」を右クリックして「ユニット追加」を選択します。選択画面が表示されるので、画面から「メモリ」の「仮想通信」「仮想デバイス」を選択し、「OK」を押してください。



2．フォルダの追加

「U01」を右クリックして、「フォルダ追加」を選択します。



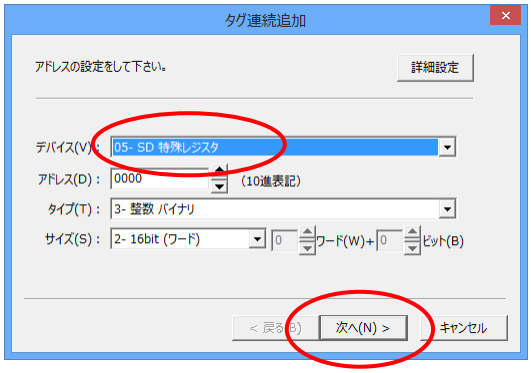
「F01」というフォルダが追加されます。

3．タグの追加

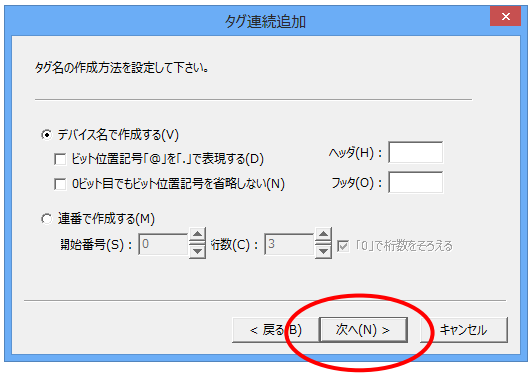
「F01」を右クリックし「タグ連続追加」を選択します。



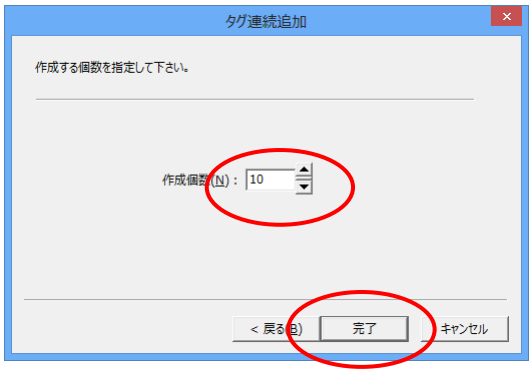
タグ連続追加ダイアログが表示されるので、「SD 特殊レジスタ」を選択して「次へ」を押してください。



なにも設定せず「次へ」を押します。



「作成個数」に 10 を設定し、「完了」を押してください。





名前	パラメータ	現在値
SD0000	SD0000	***
SD0001	SD0001	***
SD0002	SD0002	***
SD0003	SD0003	***
SD0004	SD0004	***
SD0005	SD0005	***
SD0006	SD0006	***
SD0007	SD0007	***
SD0008	SD0008	***
SD0009	SD0009	***

この操作により、SD0000-SD0009 までの数値タグが１０個作成されます。

ステップ２　アクション設定

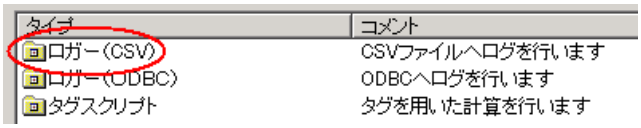
ロガーアクションの設定

1．ロガーアクションの追加

画面左側のツリーから「Action」を右クリックして、メニューから「追加」「アクション」を選択します。



表示されたアクションの追加ダイアログから、「ロガー（CSV)」を選択して「OK」を押します。



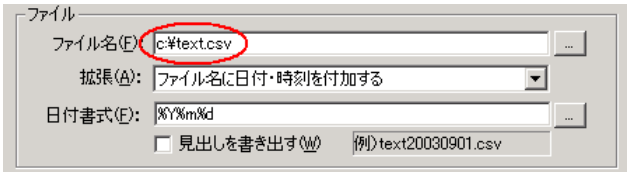
「A01」アクションが追加されました。

2．ログファイルの設定

「A01」を右クリックして「プロパティ」を選択します。



表示されたダイアログから、下記のように設定して「OK」を押してください。



・「ファイル名」を「c:¥text.csv」に設定

3．時刻フィールドの追加

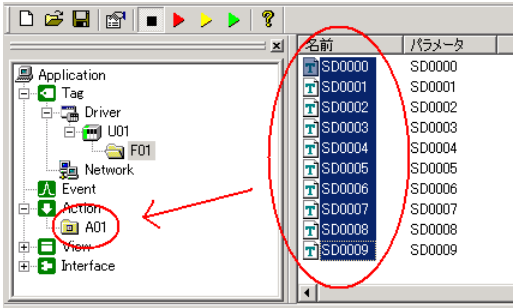
「A01」を右クリックし、「追加」「時刻フィールド」を選択します。



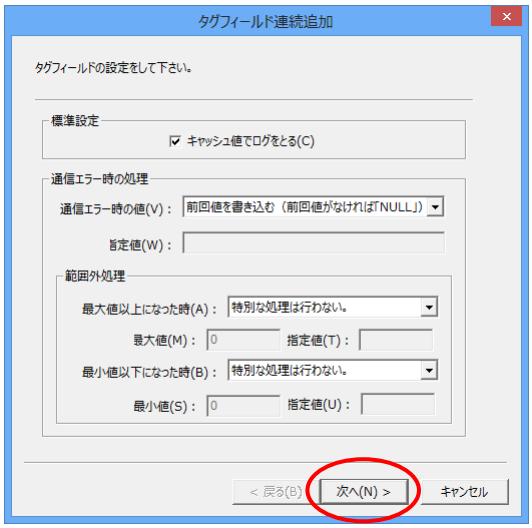
「A01」に時刻フィールド「F01」が追加されました。

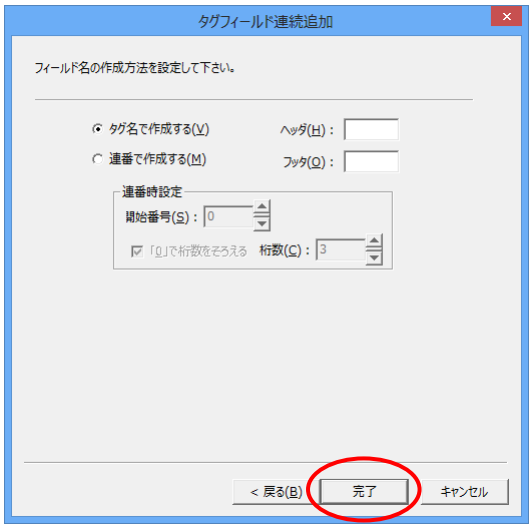
4．タグフィールドの追加

ログデータに含めるタグのフィールドを定義します。ログに含めたいタグ「SD0000」から「SD0009」を選択し、「A01」へドラックアンドドロップします。



タグをドロップすると以下の設定画面が表示されます。ここでは、デフォルトの設定のまま「次へ」ボタンで次へ移動し、「完了」を選択して設定を終了してください。





「U01_F01_SD0000」から「U01_F01_SD0009」までのフィールドが作成されました。

名前	パラメータ
F01	TIME:+FORMAT=%Y/%m/%d %H:%M:%S
U01_F01_SD0000	TAG:+TAG=U01.F01.SD0000
U01_F01_SD0001	TAG:+TAG=U01.F01.SD0001
U01_F01_SD0002	TAG:+TAG=U01.F01.SD0002
U01_F01_SD0003	TAG:+TAG=U01.F01.SD0003
U01_F01_SD0004	TAG:+TAG=U01.F01.SD0004
U01_F01_SD0005	TAG:+TAG=U01.F01.SD0005
U01_F01_SD0006	TAG:+TAG=U01.F01.SD0006
U01_F01_SD0007	TAG:+TAG=U01.F01.SD0007
U01_F01_SD0008	TAG:+TAG=U01.F01.SD0008
U01_F01_SD0009	TAG:+TAG=U01.F01.SD0009

上記がログファイルの CSV フォーマットになります。先頭に時刻フィールド、続いて SD0000 から SD0009 の値がカンマ区切りで順番に並びます。フィールドの順番を変えたい場合には、フィールドを右クリックしてメニューを表示し、「上へ」「下へ」で移動させて下さい。

ステップ３　イベント設定

定周期イベントの設定

1．定周期イベントの追加
画面左側のツリーから「Event」を右クリックしてメニューを表示し、メニューから「追加」「定周期イベント」を選択してください。

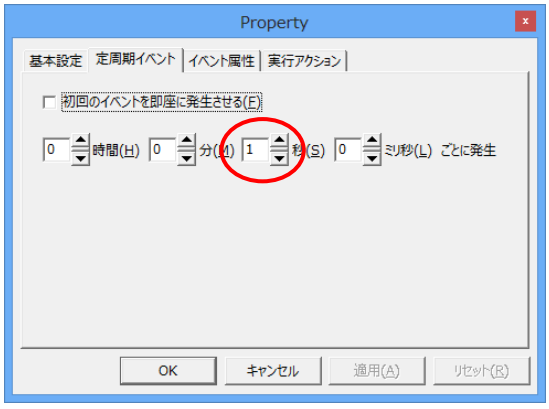


「E01」というイベントが追加されます。

2．イベントの設定
画面から「E01」を右クリックし「プロパティ」を表示してください。

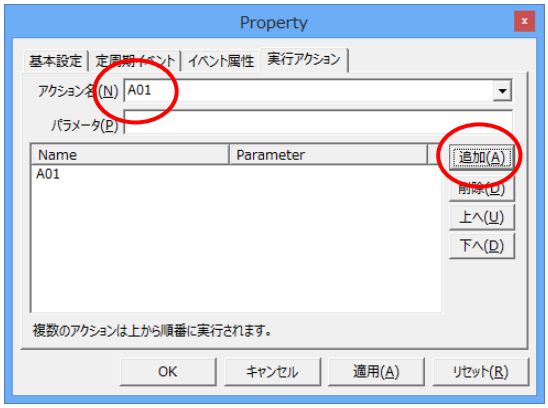


定周期イベント」タブで、１秒が設定されていることを確認してください。



デフォルトの設定は１秒です。

続いて、「実行アクション」タブでこのイベントから実行するアクションを登録します。ここでは、先ほど作成した「A01」を登録して下さい。



これにより、１秒周期でイベントが発生する都度、「A01」のロガーアクションが実行されます。



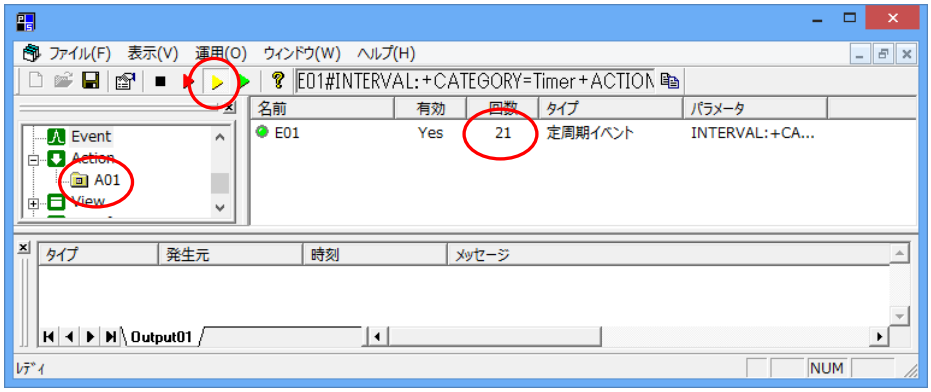
ここで設定した定周期イベント「E01」は常に１秒ごとに発生します。実行アクション設定に「A01」を加えたことにより、１秒毎に「A01」が実行されるようになります。「A01」は実行の都度必要なデータを取得し、設定されたフォーマットに従って１レコード分のデータをロギングします。

動作確認

ステップ１から３までの設定が完了したら、ロギングを実行して動作確認を行います。
ロギングの実行は「オンライン」にする事によって行われます。

オンラインにする

画面上部の「オンラインモニタ」ボタン（黄ボタン）を押してオンラインにしてください。オンラインの状態で画面左側ツリーから「Action」をクリックすると、右側のリストにはアクションが実行された回数が表示されます。この数値が１秒ごとにインクリメントされている事を確認してください。



ログファイルの確認

上記の実行回数が適当な値になったら、「ストップ」ボタン（黒ボタン）を押してオフラインにしてください。オフライン状態に戻ると、ロギング処理は停止します。

ロガーアクションの設定によって指定したログファイル「textyyyymmdd.csv」が所定の場所に作成されていることを確認し、テキストエディタ等でファイルを開いてログデータが正しい内容であることを確認して下さい。



オンラインには３つの状態があります。
赤：オンライン（デバック）‥仮想的に通信します。PLC が手元に無いときに使用します。
黄：オンライン（モニタ）‥‥アプリケーションで現在値の確認や設定が行えます。
緑：オンライン（最速）‥‥‥アプリケーションは消え、アイコンの状態でタスクバーに表示されます。

２－２　イベント型ログの設定例

概要

ここでは、実際にロガーアクションを使ってイベント型ロギングを行うための簡単な設定手順について解説します。イベント型のロギングは、操作履歴やアラーム履歴など PLC の状態に変化があったときのみログをとりたい場合に使用します。

ここで紹介する手順は、大きく分けて以下の３ステップです。

ステップ１	タグの準備	ここでは、「２－１　一般型ログの設定例」で設定したタグをそのまま使用します。
ステップ２	アクション設定	ロガーアクションを設定します
ステップ３	イベント設定	トリガとなるタグイベントを設定します。

この例では、SD0000 から SD0009 までのタグ値の変化を「タグイベント」によって検出し、タグの値が変化する都度、以下のフォーマットで CSV ファイルに保存します。

時刻	タグ名	値
2003/1/1 10:12:00	U01.F01.SD005	81
2003/1/1 10:12:01	U01.F01.SD001	163
2003/1/1 10:12:02	U01.F01.SD004	41
・・・	・・・	・・・

ステップ１　タグの準備

タグ設定

ここでは、「２－１　一般型ログの設定例」で設定したタグをそのまま使用します。詳細なタグの作成手順については前述を参照ください。

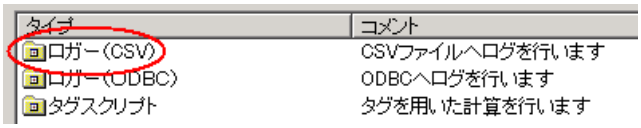
ステップ２　アクション設定

ロガーアクションの設定

１．ロガーアクションの追加
画面左側のツリーから「Action」を右クリックして、メニューから「追加」「アクション」を選択します。



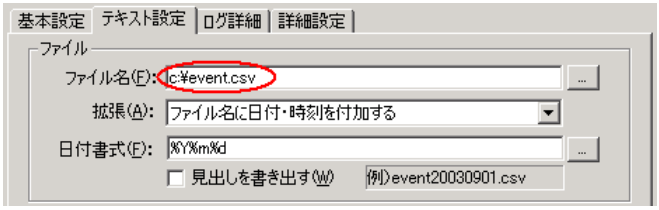
表示されたアクションの追加ダイアログから、「ロガー（CSV）」を選択して「OK」を押します。



新たに「A02」アクションが追加されました。

2．ログファイルの設定

「A02」のプロパティの「テキスト設定」タブを開き、下記のように設定します。



・「ファイル名」を「c:¥event.csv」に設定

3．時刻フィールドの追加

「A02」を右クリックし、「追加」「時刻フィールド」を選択してください（時刻フィールド「F01」が追加されます）。

4．イベントフィールドの追加

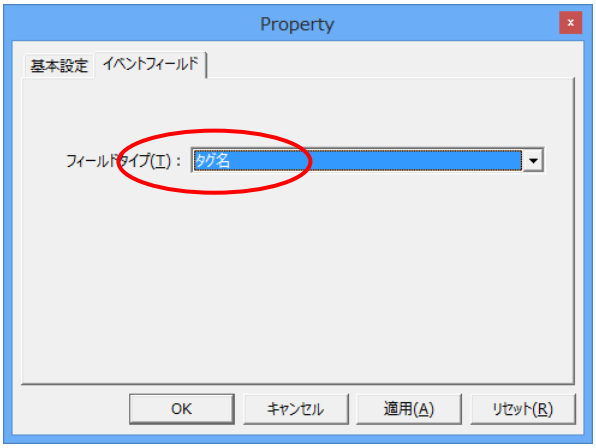
「A02」を右クリックし、「追加」「イベントフィールド」を選択してください。この操作を２回行い、２つのイベントフィールドを追加してください。



この操作により、上記イメージのとおり計３つのフィールドが追加された状態になります（時刻フィールド、イベントフィールド、イベントフィールド）。

5．イベントフィールドの設定（タグ名）

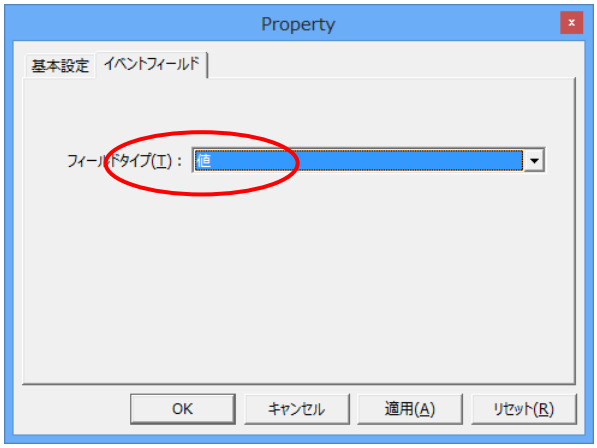
「F02」のプロパティでフィールドタイプを「タグ名」に設定してください。



上記の設定により、このフィールドにはタグ名が出力されます。

6. イベントフィールドの設定（タグ値）

続いて、「F03」のプロパティでフィールドタイプを「値」に設定してください。



上記の設定により、このフィールドにはタグの値が出力されます。

7. イベントにフォルダを追加する

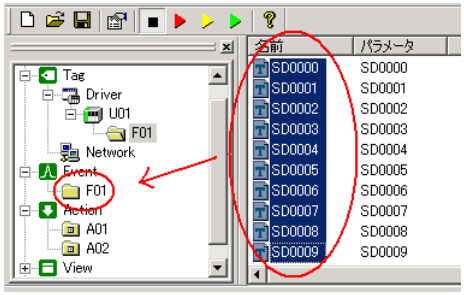
画面左側のツリーから「Event」を右クリックし、メニューから「追加」「フォルダ」を選択します。



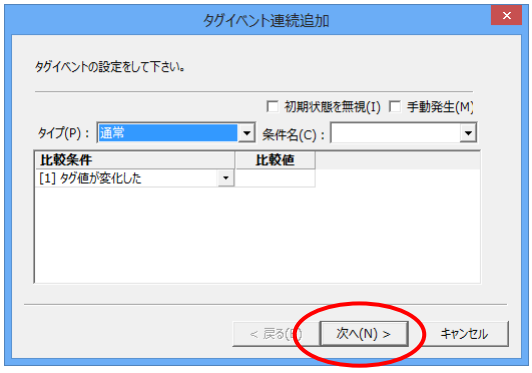
「Event」内に「F01」フォルダが追加されます。

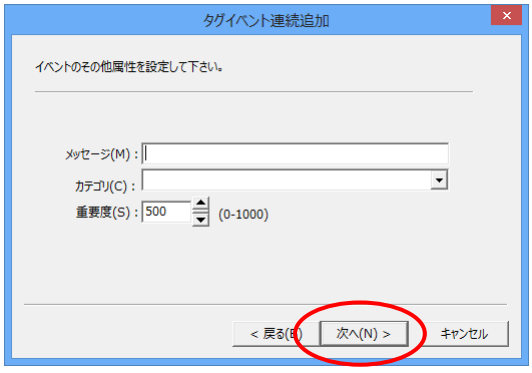
8. タグイベントを追加する

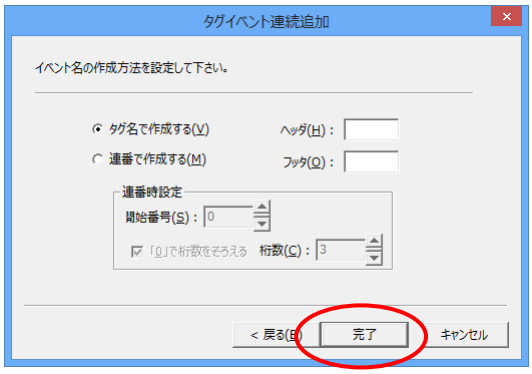
値変化によってロギングの対象としたいタグを、タグイベントとして定義します。対象としたいタグ「SD0000」から「SD0009」を選択し、「F01」フォルダへドラックアンドドロップします。



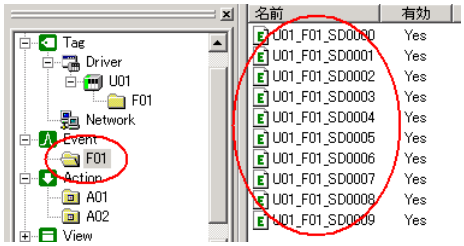
タグをドロップすると以下の設定画面が表示されます。ここでは、デフォルトの設定のまま「次へ」ボタンで画面を進めて、最後に「完了」を選択して設定を終了してください。





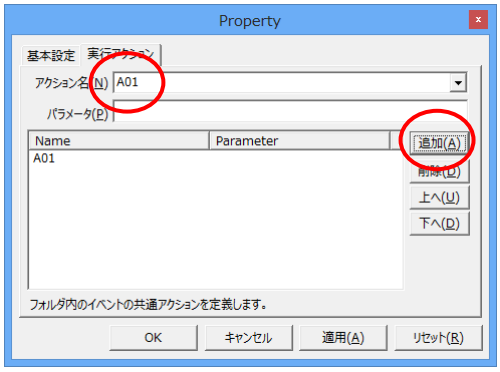


「F01」フォルダ内に１０個のタグイベントが作成されました。



9．実行アクションの登録

「F01」フォルダのプロパティを表示し、「実行アクション」タブから「A02」を追加してください。



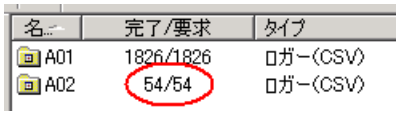
この設定により、F01 に含まれる各タグイベントのうち、いずれか１つでもイベントが発生すると A02 アクションが実行されます。

ヒント

実行アクションは個々のタグイベントに対して登録することもできますが、各イベントをまとめたフォルダに一括で登録することもできます。今回設定した１０個のタグイベントでは、それぞれのタグ値が常時監視され、値に変化があったときにイベントが発生します。各タグイベントはフォルダ「F01」の配下に作成しましたが、「F01」フォルダの実行アクション設定に「A02」を登録したことにより、フォルダ配下のイベントが１つでも発生すると「A02」が実行されるようになります。「A02」が実行されると、必要なデータを取得し、設定されたフォーマットに従って１レコード分のデータをロギングします。

動作確認

「オンラインモニタ」ボタン（黄ボタン）を押してオンラインにしてください。オンライン後に画面左側のツリーから「Action」をクリックすると、右側のリストには実行された回数が表示されます。「A02」の数値がインクリメントされている事を確認してください。

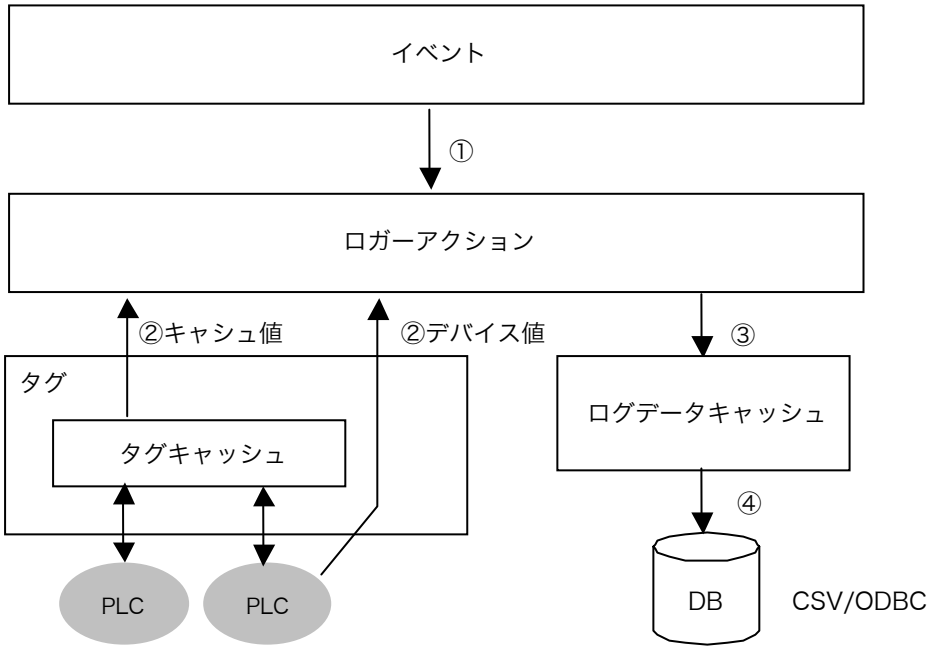


上記の実行回数が適当な値になったら、「ストップ」ボタン（黒ボタン）を押してオフラインに戻し、ログファイルが正しく作成されていることを確認してください。

3章　ロギングの機能解説

ロギングの詳細フロー

ロガーアクションによるロギング処理は、下記の①～④の流れで行われます。



①イベント発生

ロガーアクションはイベントの発生をトリガとして実行されます。ロガーアクション１回の実行で、１レコードがDB へ保存されます。イベントには下記の３種類があります。

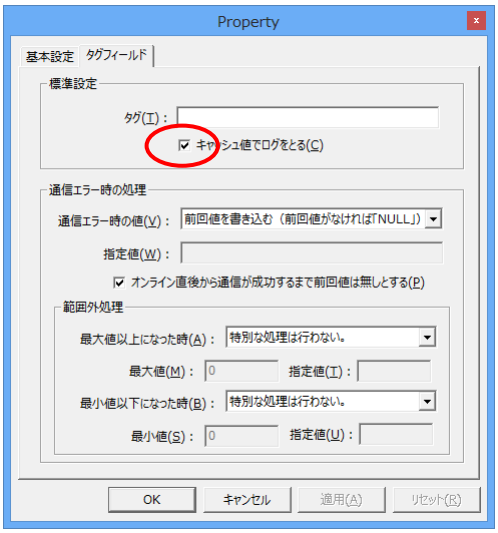
定周期イベント	一定周期で発生させる事ができるイベントです。ミリ秒単位から時間単位まで周期を設定できます。尚、最速指定の設定（設定値を全て０にする）を行うと、できるだけ速い周期でアクションが実行されます。
定刻イベント	決まった時刻に発生させる事ができるイベントです。指定年、毎年、指定月、毎月、月末、指定日、毎日、指定曜日、指定時間、毎時、指定分、毎分、指定秒、毎秒の条件を組み合わせて設定します。
タグイベント	タグ値の評価によって発生させることができるイベントです。値変化、値一致、不一致、大なり、小なりなどの条件を自由に組み合わせて設定することができます。

②データ収集

ロガーアクションがイベントにより実行されると、必要なデータを収集します。タグキャッシュ値によるロギング、又はデバイス値によるロギングのいずれかのタイプを選択することができます。

各タイプの詳細は下記のとおりです。

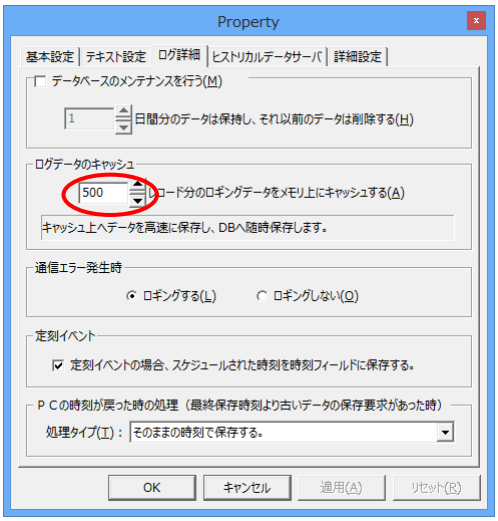
●キャシュ値によるロギング
タグの値は、別途指定された「更新周期」で PLC から定期的に実デバイス値を読み込む事によって定期的に更新が行われます。この更新周期はロガーアクションの設定とは別のものです。キャッシュ値でログをとる設定にすると、実デバイスの値ではなく、アクション実行時点のタグ値がロギングに使用されます。そのため、データ収集のための処理時間は非常に高速です。また、PLC との通信はアクション実行の都度発生しないため、複数のロガーアクションを並列的に実行した場合でも安定したロギングが行えます。但し、ログデータのタイムスタンプはタグの更新周期の範囲で過去のデータになります。
キャッシュ値でログを取る場合は「タグフィールド」プロパティの「キャシュ値でログを取る」にチェックを入れます。



●デバイス値によるロギング
デバイス値によるロギングでは、ロガーアクションが実行される都度、PLC との通信によってデータの収集を行います。よって、最もタイムスタンプが確実なデータを取得できます。単純にロガーアクション 1 つだけの処理しか行わない場合にはこの選択がベストです。しかしながら、複数のロガーアクションで並列的にデバイス値を利用すると、処理タイミングによっては通信要求がバッティングしてしまい、後続の通信要求が待たされる場合があります。例えば、一定周期でログをとっていた場合などは、通信待ちにより定周期性が乱れる可能性があります。特別な理由がない限りは、キャッシュ値によるロギングを推奨します。

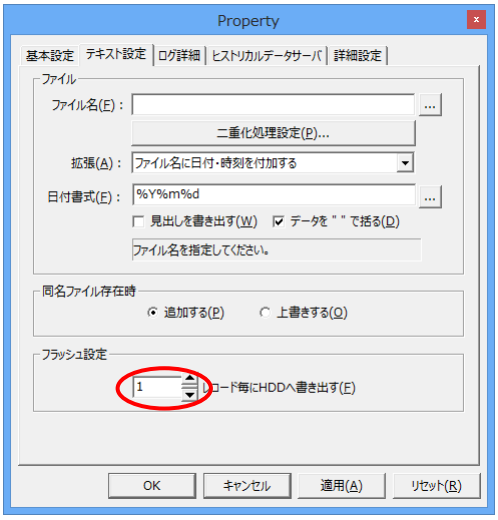
③ログデータのキャシュへの保存

データ収集によって集められたログデータは、DB へ保存する前に一旦メモリ上のキャシュに保存され、その後、随時 DB へ保存されます。
この仕組みにより、DB の負荷に影響されない安定したロギングが可能となります。例えば、イベント型のログをとる場合、大量のデータが同時に変化すると一時的にログの負荷が増えますが、そのような場合でもキャシュを介した仕組みにより安定したロギングが行われます。キャシュに保存する件数の設定はプロパティの「ログ詳細」タブから行うことができます。何らかの理由によりキャシュを利用したくない場合には、０を設定して下さい。尚、キャッシュ上に保存されている件数が設定値を超えた場合には、キャッシュのデータが DB に保存されるまで（キャッシュのデータ件数が設定値以下になるまで）待機します。



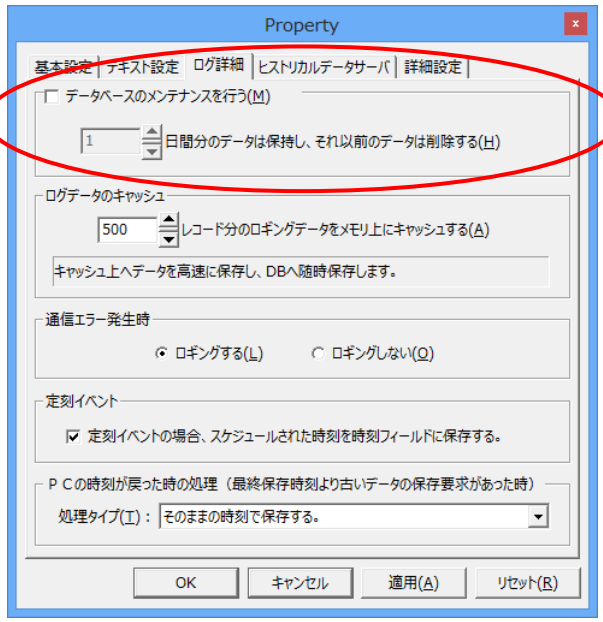
④DB への保存

最後に DB への保存が行われます。
CSV ファイルへのロギングの場合、フラッシュ設定を行うことができます。この設定は、複数のレコードをまとめてハードディスクに書き出すための機能で、長期運用の際などにハードディスクへの負担をできるだけ軽減したいというニーズに対しても有効な機能です。
１レコード毎に書き出しを行いたい場合、フラッシュ設定に「１」を指定します。この設定により、何らかの原因によって不意に PC の電源が切れた場合でも直前のデータまで確実に保存されます。また、ログファイルが外部から参照されるような場合でも、即座に最新のログが反映されます。但し、１レコード毎にハードディスクへの書き込みが行われるため、ハードディスクへの負担が増えます。ハードディスクへの負担を軽減したい場合には、フラッシュの設定値を多めにとります。



DB のメンテナンス（自動削除）

ログデータのメンテナンス（自動削除）に関する設定は、ロガーアクションのプロパティから行います。「ログ詳細」タブの「データベースのメンテナンスを行う」をチェックすると、きめられた期間でログデータのレコードを自動的に削除します。この期間は日単位で指定することが可能です。



尚、ログデータの削除は下記のタイミングで行われます。

●日付が変更されるとき

実運用時など、システムが連続的にオンラインにされている状態の場合、ログデータのメンテナンスは日付が切り替わるタイミングで行われます。

●オンラインにしたとき

オフラインからオンラインに切り替えた時点で、メンテナンスが必要かどうかを判断し、必要であればログのメンテナンスを行います。オンライン／オフラインの切り替えを１日の中で数回繰り返し実施した場合でも、メンテナンスが行われるのは１日に一回のみです。

CSV のメンテナンス処理

CSV ファイルへのロギングの場合、メンテナンスの処理内容はファイル名の拡張方法によって異なります。

●「なし」の場合

日付ごとにファイルが分けられていない場合、メンテナンスが実行されるとログファイル内のレコードを検索し、削除対象となるレコードを削除した一時ファイルを仮作成します。すべて無事に保存できたことを確認すると、元ファイルを削除して一時ファイルと置き換えます。この時、メンテナンス対象のログファイルが外部アプリケーションなどからオープンされていると、ファイルの削除ができない事によりメンテナンス作業は中断され、次回のメンテナンスのタイミングでリトライされます。



外部アプリケーションなどでログファイルをオープンしている場合には、メンテナンス処理が中断されて次回のメンテナンス時まで処理が持ち越される可能性がありますのでご注意ください。

●「ファイル名に日付・時刻を付加する」の場合

日付ごとにファイルを分けている場合、メンテナンス処理は対象となる日付のファイルを削除します。ファイルを削除するのみの処理なので、高速な処理が行われます。尚、削除を行うか否かの判定では、下記の日付書式に従ってファイル名から日付を抜き出して判断します。

「%Y%m%d%H%M%S」「%Y%m%d%H%M」「%Y%m%d%H」「%Y%m%d」「%Y%m」「%Y」

上記のいずれにも当てはまらない場合には、下記の「ファイル名にタグ値を付加する」の場合と同様の処理が行われます。

●「ファイル名にタグ値を付加する」の場合

指定されたファイルのパスに存在する全てのファイルを検索し、ファイル名で指定した「¥パス¥ファイル名.拡張子」の文字がすべて含まれており、かつ「ファイルの更新日時」（OS の属性）が指定した期間よりも古い日付のファイルを削除します。ファイルを削除するのみの処理なので、高速な処理が行われます。



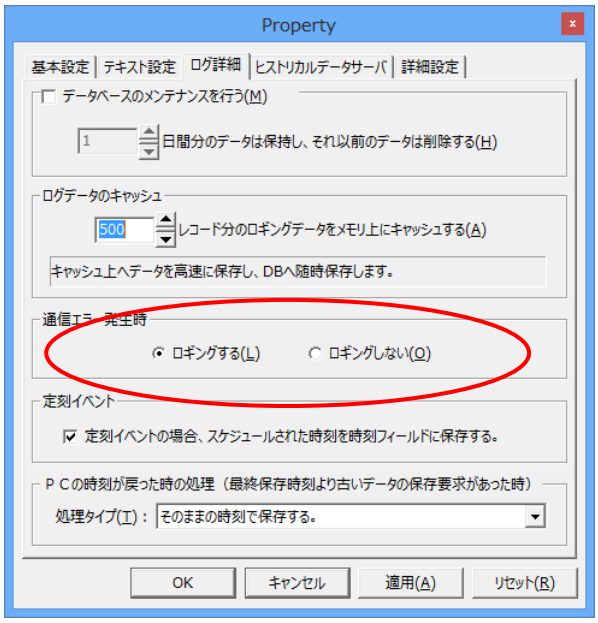
メンテナンス対象のログファイルが格納されているフォルダ内に、ログファイル以外のファイルが混在している場合には、それらのファイル名に注意してください。ファイル名や作成日付が削除条件と一致した場合、予期せぬファイルが削除される可能性があります。従って、ログファイルを格納するフォルダはログ専用のフォルダとして作成し、システムやアプリケーションフォルダ等とは分けた構成とすることを推奨します。

ODBC のメンテナンス処理

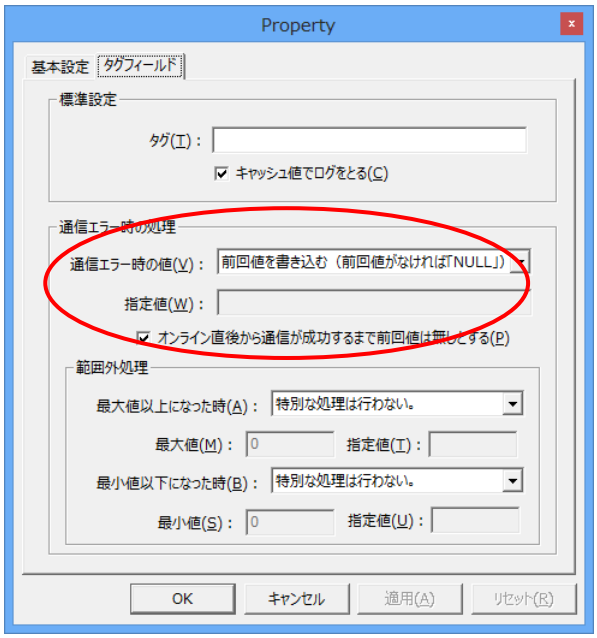
メンテナンスが実行されると、SQL コマンドの発行によって削除対象レコードが削除されます。尚、Oracle の場合、日付の書式で**'yyyy-mm-dd'**が使用できる設定にして下さい。

通信エラー発生時の動作設定

ロガーアクションプロパティの「通信エラー発生時」の設定で、通信が失敗したときの動作を定義できます。



ロガーアクションは、アクションに定義されているタグフィールドの内、１つでも通信ができないタグがあった場合、通信エラーが発生していると認識します。上記の設定で「ロギングしない」を選択した場合、通信エラーが発生した場合にはログ出力を行いません。従って、データベースには通信が成功したデータのみがロギングされます。ここで「ロギングする」を選択した場合、ロギング処理の振る舞いは各タグフィールド個別の設定に依存します。タグフィールド個別の設定は、下記の「タグフィールド」の「エラー時の処理」で定義します。



●前回値を書き込む（前回値がなければ「NULL」）

通信エラーが発生した場合、前回取得に成功した値をロギングに使用します。前回値が存在しなければ NULL を書き込みます（ここで、前回値が存在しない状態とは、オンライン後１度も通信に成功していない状態を指します）。

●前回値を書き込む（前回値がなければ指定値）

通信エラーが発生した場合、前回取得に成功した値をロギングに使用します。前回値が存在しなければ指定した値を書き込みます（ここで、前回値が存在しない状態とは、オンライン後１度も通信に成功していない状態を指します）。

●「NULL」を書き込む

通信エラーが発生した場合、NULL を書き込みます。

●指定値を書き込む

通信エラーが発生した場合、指定値を書き込みます。

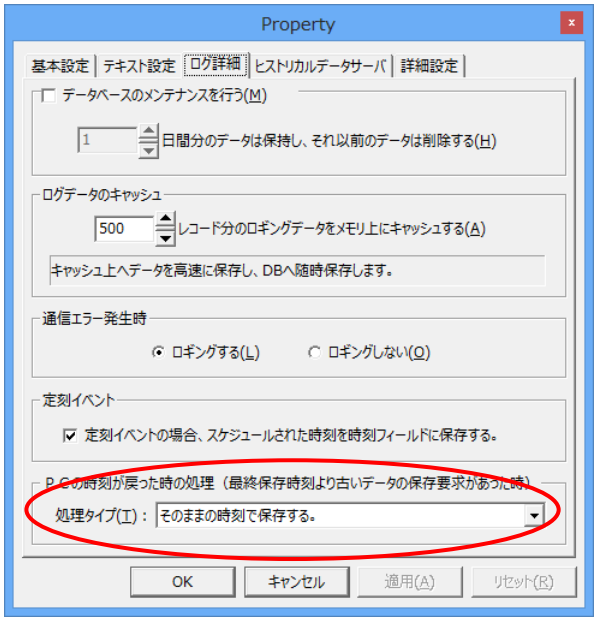


CSV の場合、「NULL」とは空文字のことです。ODBC の場合、データベースが NULL 値に対応していれば NULL 値を書き込みます。NULL 値に対応していないデータベースの場合には、ブール型の場合は FALSE、数値型の場合は 0（ゼロ）、文字型の場合は空文字を書き込みます。

ＰＣの時刻が戻った時の処理

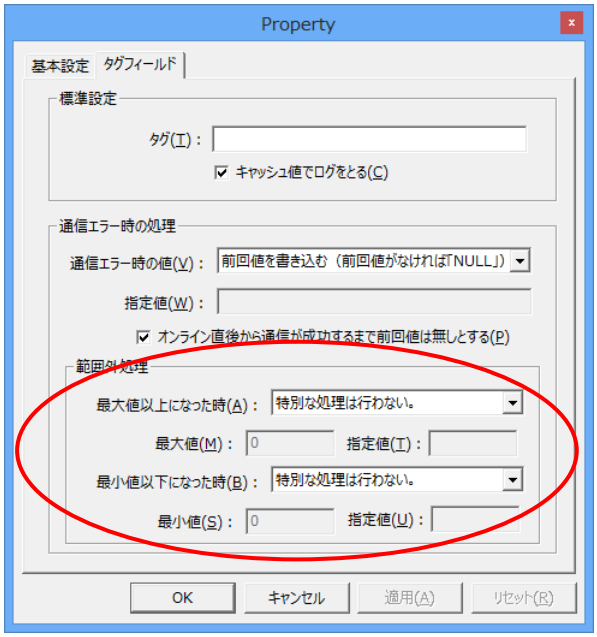
時刻合わせなどで、PC の時刻が戻ってしまったとき、最終的に保存した時刻より過去のデータを保存すると問題が発生する場合があります。例えば、トレンドグラフや日報月報年報アクションでは予期せぬ結果を得てしまう可能性があります。そのため、PC の時刻を戻すことにより、最終的に保存したデータの時刻より過去のデータを保存しようとするとき、下記の３つの処理から選択できます。

・そのままの時刻で保存する
・最終保存時刻に修正して保存する
・データを保存しない



トレンドグラフあるいは日報月報年報アクションを使用する場合は、「最終保存時刻に修正して保存する」（場合によっては「データを保存しない」）を推奨いたします。

PLC の値が一定範囲外の時、欠損値とする方法



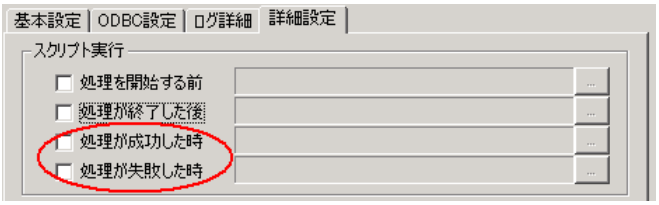
PLC の値が一定範囲外の時、欠損値とするにはタグフィールドの「範囲外処理」の設定を行います。

例えば、PLC との通信は正常に行われているが、実際の機器がオンラインになっていないため、本来とりたくないデータ（0 など）がロギングされることを避けたいときなど、タグフィールドの「範囲外処理」をうまく利用すると、欠損値としてデータを保存することができます（日報などで平均値を求める場合、機器がオンラインになっていないときの状態を無視して平均等の集計をすることができます。「指定値を書き込む」、最小値「0」、指定値「空白」とすると、PLC の値が 0 の時、CSV ファイルに値が書き込まれません）。

ログデータのキャッシュ設定とスクリプトの振る舞い

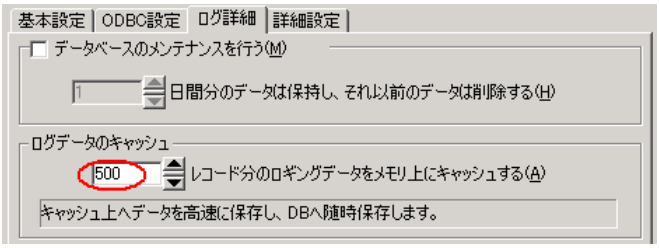
ロガーアクションでは、以下の処理タイミングにおいてスクリプトを実行することができます。

(スクリプト実行の設定)



ここで、「処理が成功した時」と「処理が失敗した時」とは、ログデータがデータベースに正しく書き込まれたか否かを表すものです。この動作は、ログデータのキャッシュの設定内容によって以下のような動作となります。

(キャッシュの設定)



●キャッシュの指定が 0 の時

データベースへのロギングが失敗した場合、「処理が失敗した時」のスクリプトが実行されます。

●キャッシュの指定が 1 以上の時

ログデータはキャッシュに保存されるため、この場合の処理は必ず成功になります（「処理が成功した時」のスクリプトが実行されます)。

イベント型ログへ複数のタグ値を含める

イベント型ログには、イベント発生元のタグに関する情報以外にも、他のタグの値を同じレコードに保存することができます。この方法は、イベント型ロギングを行うロガーアクションにタグフィールドを追加し、ログに含めたいタグをそのフィールドに設定することによって行います。

下記のイベント型ログフォーマットでは、「D0001」と「M0001」がタグフィールドによって出力されています。

例)

時刻	タグ名	値	D0001	M0001
2003/1/1 10:12:00	U01.F01.SD005	81	2	True
2003/1/1 10:12:01	U01.F01.SD001	163	5	True
2003/1/1 10:12:02	U01.F01.SD004	41	8	False
・・・	・・・	・・・	・・・	・・・

イベント型ログにおけるタグ値変化の検出周期

イベント型ログのロガーアクションの実行トリガは、タグイベントによって発生させます。ここで、タグイベントによるタグ値の監視周期は、タグフォルダに対する値の更新周期の設定に依存します。

タグの更新周期はデフォフォルトでは 1 秒ですが、時間、分、秒、ミリ秒の全てに 0（ゼロ）を設定すると、可能な限り速い周期で値を監視します。タグの件数にも左右されますが、Ethernet で PLC と接続されている場合の最速は 15ms から 30ms 程度、RS-232C の場合では最速 30ms から 50ms 程度を目安として下さい。



４章　ロギングの実践

４－１　タグイベントによるロギング（Sample01）

概要

ここでは、タグイベントをトリガとしてロギングを行うための設定方法について解説します。

PLC デバイスの信号値や、VisualBasic で開発した外部アプリケーションなどから受け取ったトリガによってロギングを行いたい場面は多くあります。このようなロギングを行う場合、タグイベントをトリガに使うのがベストです。タグイベントによって、タグの値が ON/OFF したとき、あるいは上下限を超えたときをトリガとしてロギングを行うことができます。

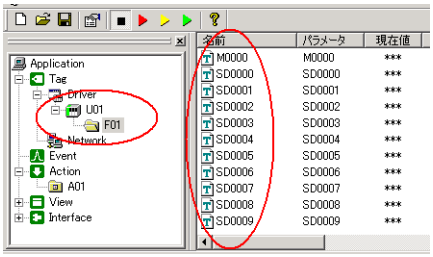
このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample01¥logger_sample01.txt

この例では、仮想デバイスの M0000（ビット）の ON 時に１０点のアナログ値（SD0000～SD0009）のログを取るサンプルを紹介します。

設定手順

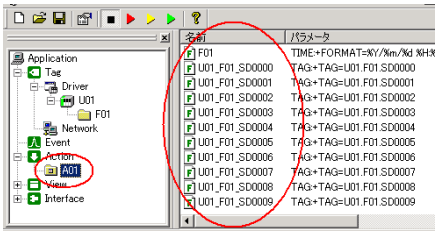
タグ設定

トリガのためのタグ「M0000」及び、
ロギング対象とするタグ「SD0000」～「SD0009」を設定します。



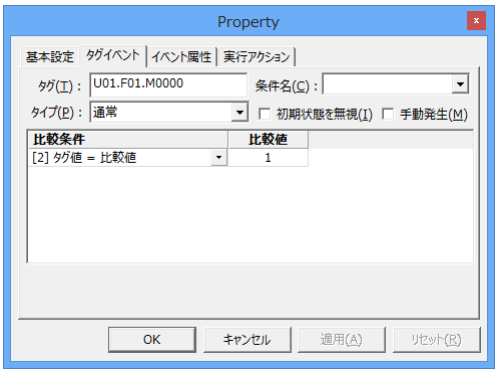
アクション設定

ロガーアクション「A01」を追加し、以下のフィールドを設定します。
・時刻フィールド
・タグフィールド
　（※「SD0000」～「SD0009」を「A01」へドラックアンドドロップ）



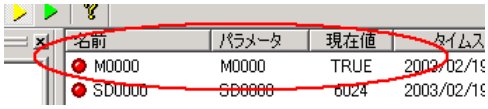
イベント設定

「M0000」タグでタグイベントを作成します。
タグリストの「M0000」をイベントへドラックアンドドロップします。ビットが ON 時イベントを発生させたいため、比較条件：「タグ値 = 比較値」比較値：「1」と設定します。
このイベントの実行アクションに、先の手順で作成した「A01」アクションを設定します。

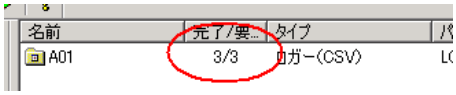


動作確認

オンライン後、「U01.F01.M0000」タグに TRUE／FALSE を交互に数回書き込んでみて下さい（タグの書き込みを行うためには、「M0000」を右クリックしてメニューを表示し、「タグ操作」「書き込み」を選択します）。



「A01」のカウントが「TRUE」になる度に、カウントがインクリメントされることを確認して下さい。また、ログデータが正しくファイルに出力されていることを確認してください。



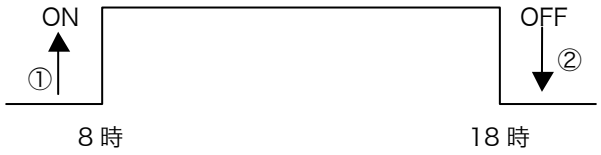
このサンプルの応用例

タグイベントによってタグの値変化をトリガに利用できることは非常に重要です。例えば、PLC と通信するタグをトリガとして設定すれば、PLC のロジックを介して受け取った信号でトリガを作成することができます。あるいは、仮想デバイスやメモリデバイスを用いてアプリケーション内部の処理に応じてトリガを設けることもできます。また、IPLink などのインターフェースを介してタグの値を書き換えることにより、VisualBasic などで開発した外部アプリケーションからロギングのトリガを得ることもできます（IPLink などのインターフェースに関する詳細は、「インターフェース基本マニュアル」を参照してください）。

４－２　ログの開始・停止を制御する（Sample02）

概要

ここでは、タグイベントによってロギングの開始と停止を制御する設定方法について解説します。例として、月曜から金曜日の８時から 18 時までの間のみ、１秒周期のログを取ります。また、サンプルの応用例としてロギングのサンプリングレートをダイナミックに切り替える方法も併せて紹介します。



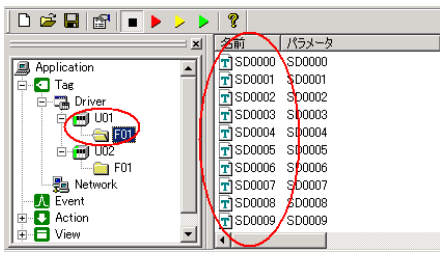
このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample02¥logger_sample02.txt

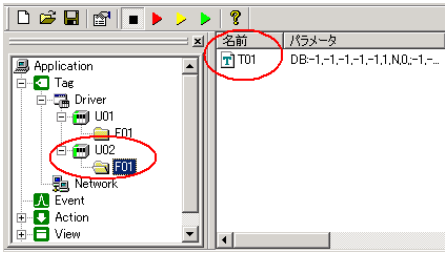
尚、ログファイルのフォーマットは「４－１　タグイベントによるログ」のサンプルと同等とします。

設定手順

タグ設定

ログの開始・停止のトリガのためのメモリデバイスタグ（時刻ビットタグ）「U02.F01.T01」及びロギング対象とする仮想デバイスタグ「U01.F01.SD0000」～「U01.F01.SD0009」を設定します。





メモリデバイスの時刻ビットタグには、月曜から金曜日まで、８時０分０秒から 17 時 59 分 59 秒を設定します。

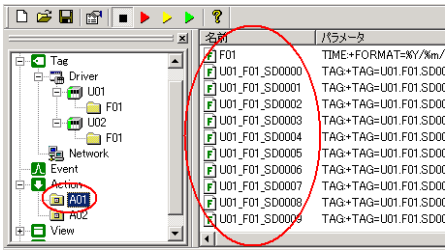


アクション設定

ロガーアクション「A01」及びスクリプト Ver1 アクション「A02」を追加します。

A01：４－１で定義したフォーマットと同様とします。

A02：スクリプト「//Event/E01!Enable = U02.F01.T01」を設定。



イベント設定

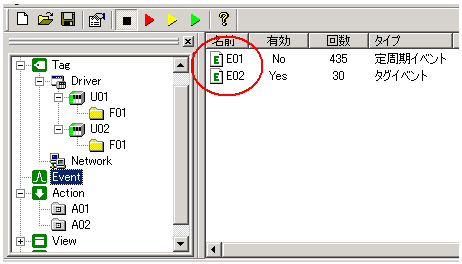
定周期イベント「E01」及びタグイベント「E02」を追加します。

E01:

１秒周期とし、実行アクションに「A01」を設定。初期状態を無効にするため、「無効にする」をチェック。

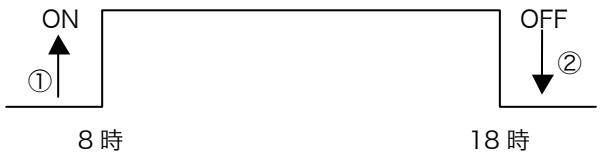
E02：

タグに「U02.F01.T01」、比較条件に「タグ値が変化した」、実行アクションに「A02」を設定する。



解説

メモリデバイスの時刻ビットタグは、指定した時刻や期間のみ ON させることができるビットです。今回の設定では 8 時から 18 時までの間 ON します。



① 8 時に ON

1．U02.F01.T01 が ON になるとタグイベント「E02」が発生し、スクリプト Ver1 アクション「A02」が実行されます。

2．「A02」では「//Event/E01!Enable = U02.F01.T01」の式が実行され、定周期イベント「E01」が有効になり、１秒周期のロギングが開始されます。
この式は「E01」の有効状態を切り替えるためのもので、1 の時有効、0 の時無効になります。「U02.F01.T01」の状態はこの時 ON（１）の状態であるため、「//Event/E01!Enable」には「1」が設定されます。

② 18 時に OFF

1．U01.F01.T01 が OFF になるとタグイベント「E02」が発生し、スクリプト Ver1 アクション「A02」が実行されます。

2．「A02」では「//Event/E01!Enable = U02.F01.T01」の式が実行され、定周期イベント「E01」は無効となり、ロギングが停止します。

E01!Enable は、イベントが持つシステムプロパティの一つです。イベントに関する詳細は「イベントマニュアル」を参照してください。

このサンプルの応用例

今回は時刻ビットタグを用いましたが、PLC と通信するタグを「U02.F01.T01」の代わりに用いれば、PLC からのトリガによってロギングの開始・終了を制御することができます。

また、異常状態を検出した時のみ集中的にログを取りたい場合など、ログのサンプリングレートをダイナミックに切り替えたい場合には、高速用の定周期イベント及び低速用の定周期イベントを２つ用意し、それぞれのイベントの有効・無効をスクリプトアクションで切り替えることで実現できます。

具体的には、下記例のように設定します。
この例の場合、このとき「U02.F01.T01」が ON の時高速で、OFF の時低速でログが取られます。

例）
「EVENT_HI」定周期イベント（高速ログ用）
- 高速用の周期を設定。
- 初期状態を無効にするため、「無効にする」をチェックする。
- 実行アクションに「A01」を設定する。

「EVENT_LO」定周期イベント（低速ログ用）
- 低速用の周期を設定。
- 初期状態を無効にするため、「無効にする」をチェックする。
- 実行アクションに、上記同様「A01」を設定する。

「A02」スクリプトアクション
```
//Event/EVENT_HI!Enable = U02.F01.T01
//Event/EVENT_LO!Enable = Not U02.F01.T01
```

※「Not U02.F01.T01」の Not とは、True/False を反転します。

4－3　計算結果をロギングする（Sample03）

概要

ここでは、PLC から読み込んだデバイス値を演算によって加工してロギングする方法について解説します。

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample03¥logger_sample03.txt

この例では、各２点のデバイス値の平均値を求め、その結果をロギングしています。



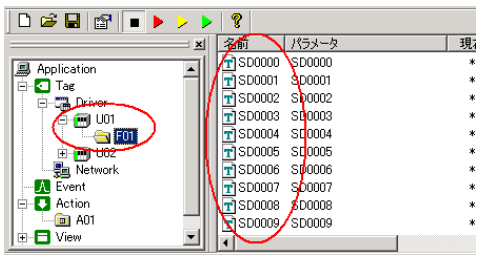
本項目では計算をスクリプトにて行っていますが、簡単な計算であればタグフィルタを使用した方が簡単に処理を行うことができます。
例えば 10 倍や 1/10 の値にしたい場合は「小数点の移動と切り捨てを行う」、工学値変換を行いたい場合は「傾きとオフセットを変換する（工学値変換）」を使用することで対応することが出来ます。
タグフィルタの詳細については「タグマニュアル」を参照してください。

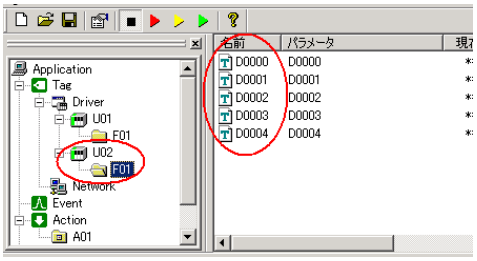
設定手順



タグ設定

データ収集のための仮想デバイスタグ「U01.F01.SD0000」～「U01.F01.SD0009」及び計算結果を格納するための仮想デバイスタグ「U02.F01.D0000」～「U02.F01.D0004」を設定します。

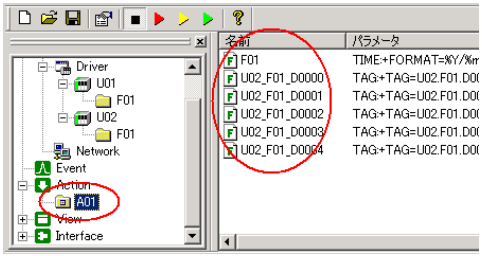






アクション設定

ロガーアクション「A01」を追加します。
時刻フィールドの設定及び、「U02.F01.D0000」～「U02.F01.D0004」のタグフィールドを設定してください（A01 にタグをドラックアンドドロップする）。



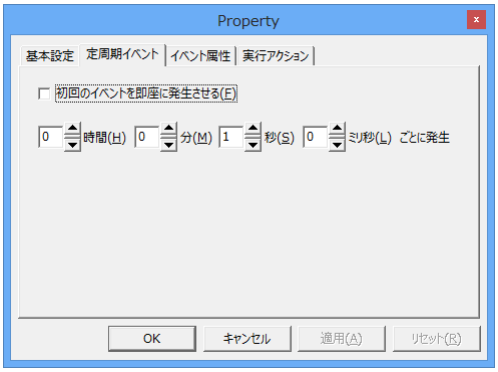
また、「詳細設定」「スクリプト実行-処理を開始する前」をチェックし下記のスクリプトを設定します。

```
U02.F01.D0000 = (U01.F01.SD0000 + U01.F01.SD0001) / 2
U02.F01.D0001 = (U01.F01.SD0002 + U01.F01.SD0003) / 2
U02.F01.D0002 = (U01.F01.SD0004 + U01.F01.SD0005) / 2
U02.F01.D0003 = (U01.F01.SD0006 + U01.F01.SD0007) / 2
U02.F01.D0004 = (U01.F01.SD0008 + U01.F01.SD0009) / 2
```



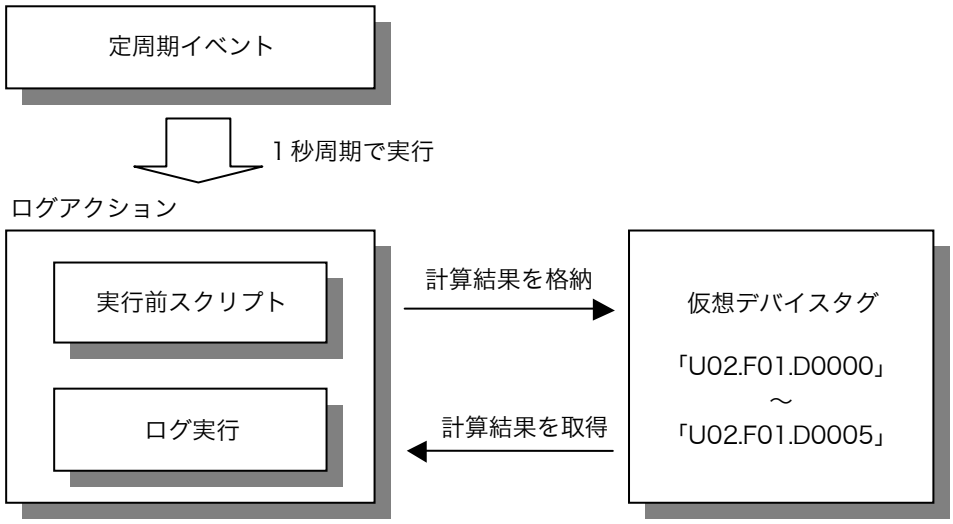
イベント設定

定周期イベント「E01」を追加します。周期は 1 秒周期とし、実行アクションに「A01」を設定します。



解説

この処理のポイントとしては、計算結果を格納するためのタグをあらかじめ用意しておき、実行前スクリプトによって、ロギングが行われる直前のタイミングで平均値を求め、値格納用のタグに代入している点です。ロガーアクションは、計算結果が格納されたタグの値をロギングします。この例では、データ収集のためのタグが「U01.F01.SD0000」～「U01.F01.SD0009」であり、計算結果を格納するためのロギング用タグが「U02.F01.D0000」~「U02.F01.D0005」にあたります。



下記スクリプトは、それぞれ２点のデバイスの平均値を求め、ロギング用のタグに代入しています（※ここでは例として SD タグを使用。実際には PLC と通信するタグを設定します）。

```
U02.F01.D0000 = (U01.F01.SD0000 + U01.F01.SD0001) / 2
U02.F01.D0001 = (U01.F01.SD0002 + U01.F01.SD0003) / 2
U02.F01.D0002 = (U01.F01.SD0004 + U01.F01.SD0005) / 2
U02.F01.D0003 = (U01.F01.SD0006 + U01.F01.SD0007) / 2
U02.F01.D0004 = (U01.F01.SD0008 + U01.F01.SD0009) / 2
```

４－４　配列タグを使った大量データのログ（Sample04）

概要

ここでは、配列タグを使って大量データのロギングを行う方法について解説します。

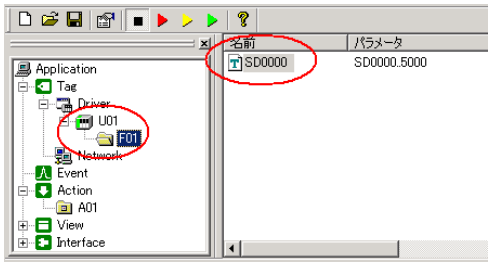
ロギング対象のタグを各１点ずつ用意しても大量データのロギングを行うことは可能ですが、ロガーアクションに登録するタグの点数が多くなりすぎると、設定作業が煩雑なものとなる場合があります。配列タグを利用することによって、１つのタグ登録作業のみで連続したデバイスエリアを簡単にロギングすることができるようになります。

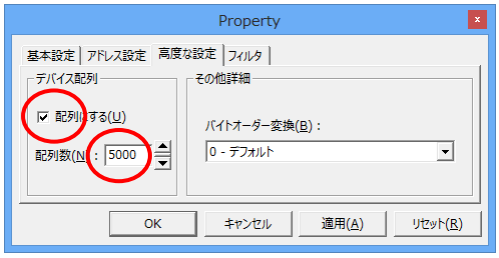
このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample04¥logger_sample04.txt

設定手順

タグ設定

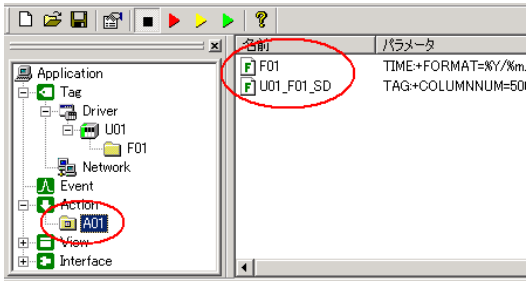
データ収集のための仮想デバイスタグ「U01.F01.SD0000」を設定し、下記プロパティにより配列として定義します（この例では 5000 個の配列とします）。この設定により SD0000～SD04999 の連続したアドレスエリアが１つのタグでまとめて定義されます。

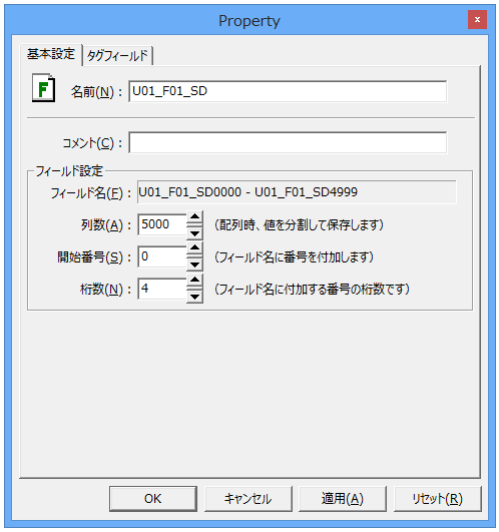




アクション設定

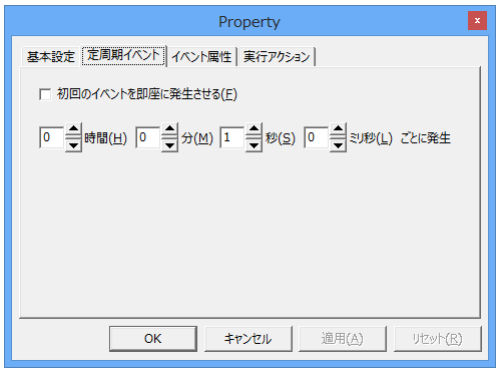
ロガーアクション「A01」を追加します。時刻フィールドを設定した後、先ほど作成した配列タグ「U01.F01.SD0000」をドラックアンドドロップしてタグフィールドを設定して下さい。ここで、タグフィールドのプロパティを下記のように設定します。
名前：「U01_F01_SD」　列数：「5000」　桁数：「４」


名前
パラメータ
F01
TIME:+FORMAT=%Y/%m.
U01_F01_SD
TAG:+COLUMNNUM=50
Application
Tag
Driver
U01
F01
Network
Event
Action
A01
View
Interface


Property
基本設定
タグフィールド
名前(N) :
U01_F01_SD
コメント(C) :
フィールド設定
フィールド名(F) :
U01_F01_SD0000 - U01_F01_SD4999
列数(A) :
5000
(配列時、値を分割して保存します)
開始番号(S) :
0
(フィールド名に番号を付加します)
桁数(N) :
4
(フィールド名に付加する番号の桁数です)
OK
キャンセル
適用(A)
リセット(R)

イベント設定

定周期イベント「E01」を追加します。周期を１秒周期とし、実行アクションに「A01」を設定します。



解説

配列タグを使うと、連続したデバイスを一括で保存することができます。この例では 5000 個の連続したデバイスとして定義しました。CSV ファイルへのロギングにおいてフィールド名はあまり重要な意味を持ちませんが、ODBC 経由でデータベースにロギングを行う場合、ロガーのフィールド名の定義をテーブルレイアウトに一致させる必要があります。この例の場合、フィールド設定画面で名前を「U01_F01_SD」、列数：「5000」、桁数：「４」と設定しました。これによりログデータの各フィールド名は「U01_F01_SD0000」～「U01_F01_SD4999」の連番として出力されます。

このサンプルの応用例

配列タグは、イベント型ログに含めて使用することも可能です。例えば、以下のフォーマットでは各タグの値が変化したことを１行に１タグ単位で保存しています。

時刻	タグ名	値
2003/1/1 10:12:00	U01.F01.SD005	81
2003/1/1 10:12:01	U01.F01.SD001	163
2003/1/1 10:12:02	U01.F01.SD004	41
・・・	・・・	・・・

これを、各タグを配列タグで定義し、フィールド設定の列数を複数列として設定することにより、１行に複数タグをブロック単位でログ出力することができます。この場合、配列タグ内で１つでも値が変化するとロギングが行われます。尚、各タグの配列数がフィールドの列数より多い場合、フィールドの列数を超えた配列要素は切り捨てられます。反対に、配列数がフィールドの列数より少ない場合には、余分なフィールドは NULL で埋められます。

時刻	タグ名	値００	・・・	値９９
2003/1/1 10:12:00	U01.F01.BLOCK1	81	・・・	98
2003/1/1 10:12:01	U01.F01. BLOCK3	163	・・・	54
2003/1/1 10:12:02	U01.F01. BLOCK5	41	・・・	2
・・・	・・・	・・・	・・・	・・・

４－５　イベント型ログにアラームコメントを出力（Sample05）

概要

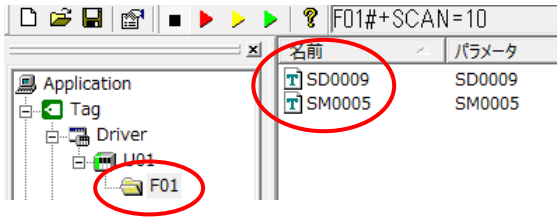
ここでは、イベント型ログの利用により、アラームコメントなどの付属情報をログに出力する方法について解説します。イベント型ログでは、イベントフィールドの設定によって様々な情報をログに含めることができます。

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample05¥logger_sample05.txt

設定手順

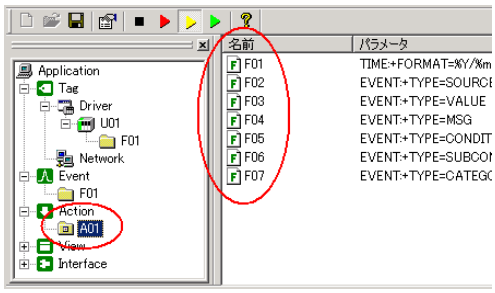
タグ設定

データ収集のための仮想デバイスタグ「U01.F01.SD0009」「U01.F01.SM0005」を設定します。SD0009 は 0 から 100 まで自動的に変化するサイン波で、SM0005 は２秒ごとに自動的に ON/OFF を繰り返します。



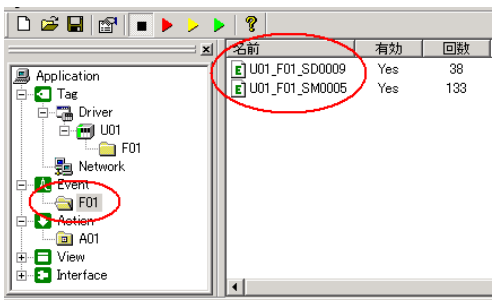
アクション設定

ロガーアクション「A01」を追加します。時刻フィールド及び下記のイベントフィールドを設定して下さい。
F02:フィールドタイプ「タグ名」
F03:フィールドタイプ「値」
F04:フィールドタイプ「メッセージ」
F05:フィールドタイプ「条件名」
F06:フィールドタイプ「サブ条件名」
F07:フィールドタイプ「カテゴリ」



イベント設定

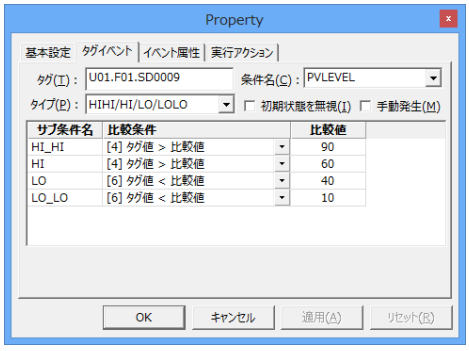
フォルダを作成し、その中にタグイベントを２つ追加します。そして、フォルダの実行アクションを「A01」に設定します。
※イベントの設定内容の詳細は、下記「解説」を参照して下さい。

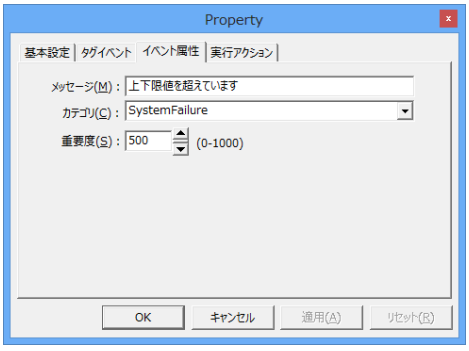


解説

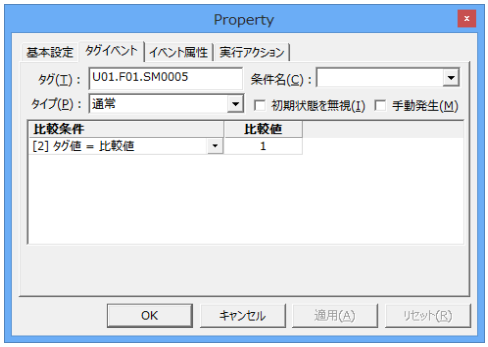
各イベントはメッセージ、カテゴリ、重要度の３つの属性を持っています。また、タグイベントはさらに条件名、サブ条件名の属性を持っています。これらの属性を定義することにより、ログファイルにイベントの属性情報を保存することができます。

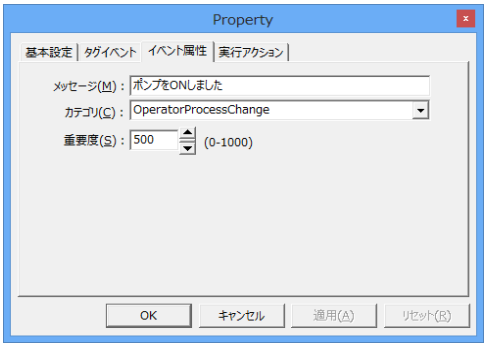
タグイベント「U01.F01.SD0009」





タグイベント「U01.F01.SM0005」





ログ結果

時刻	タグ名	値	メッセージ	条件名	サブ	カテゴリ
2003/2/19 13:38:01	U01.F01.SD0009	99	上下限値を超えています	PVLEVEL	HI_HI	SystemFailure
2003/2/19 13:38:01	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:04	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:06	U01.F01.SD0009	90	上下限値を超えています	PVLEVEL	HI	SystemFailure
2003/2/19 13:38:08	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:12	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:15	U01.F01.SD0009	39	上下限値を超えています	PVLEVEL	LO	SystemFailure
2003/2/19 13:38:16	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:20	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:24	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:28	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:32	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:36	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:40	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:40	U01.F01.SD0009	61	上下限値を超えています	PVLEVEL	HI	SystemFailure
2003/2/19 13:38:44	U01.F01.SM0005	TRUE	ポンプを ON しました			OperatorProcessChange
2003/2/19 13:38:46	U01.F01.SD0009	91	上下限値を超えています	PVLEVEL	HI_HI	SystemFailure

4－6　ODBC データベースへのログ（Sample06）

概要

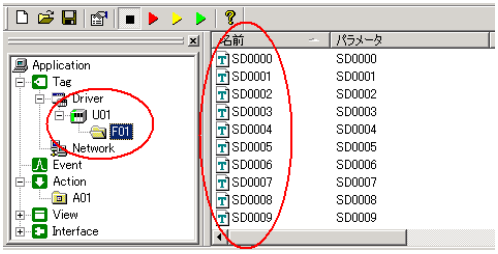
ここでは、ODBC データベースへのロギングについて解説します。例として、Microsoft Access2000 のデータベースへのロギングを行います。

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample06¥logger_sample06.txt
Access2000 MDB ファイル：　¥¥sample¥common¥action¥logger¥sample06¥DB¥dbsample.mdb

設定手順

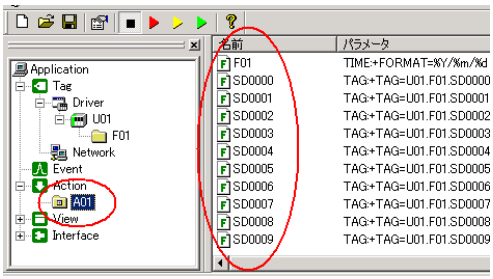
タグ設定

データ収集のためのタグ「SD0000」~「SD0009」を設定します。

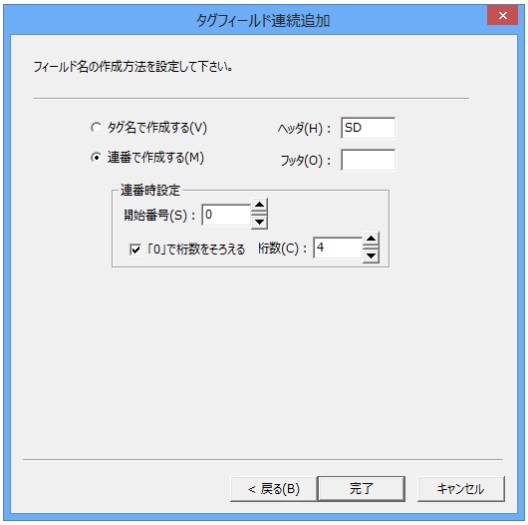


アクション設定

ODBC ロガーアクション「A01」を追加します。時刻フィールド「F01」及びデータ収集のためのタグフィールド「SD0000」~「SD0009」を設定します。

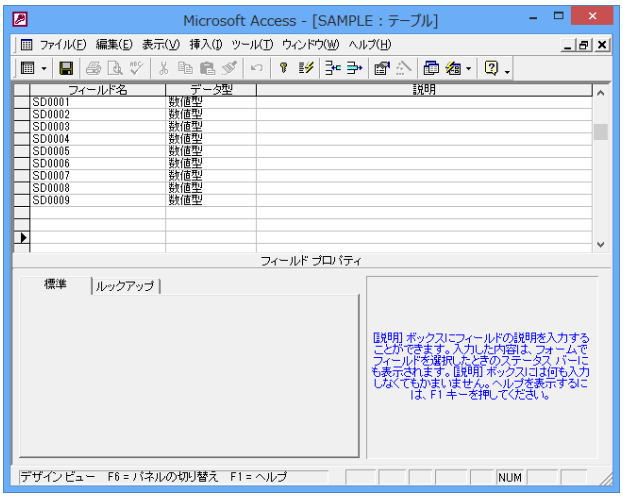


※ログのフィールド名は DB のフィールド名と一致させる必要があります。タグをドラックアンドドロップした後、連続追加ダイアログで、下記のように設定しフィールド名を指定します。



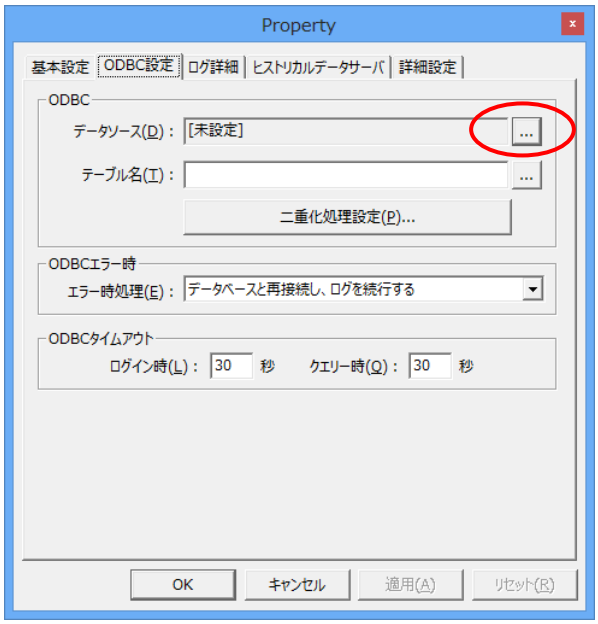
連番で作成する：チェック
ヘッダ：SD
桁数：4

Access でデータベースを作成します。フィールド名はロガーアクションで設定したフィールド名にあわせます。
※Access の設定内容の詳細は、サンプルの MDB ファイルを参照してください。

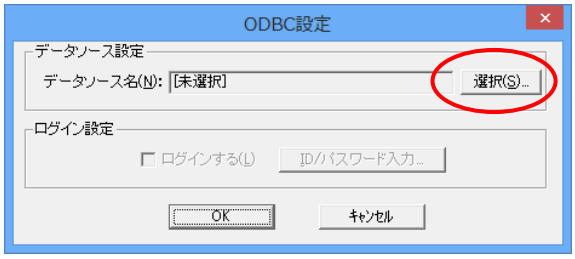


データベースを作成したら「A01」アクションの「ODBC 設定」を行います。先ほど作成した「dbsample.mdb」に ODBC 経由で接続できるように設定します。

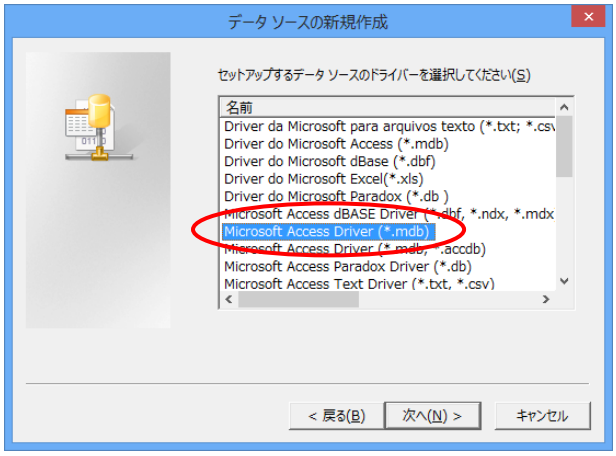
1



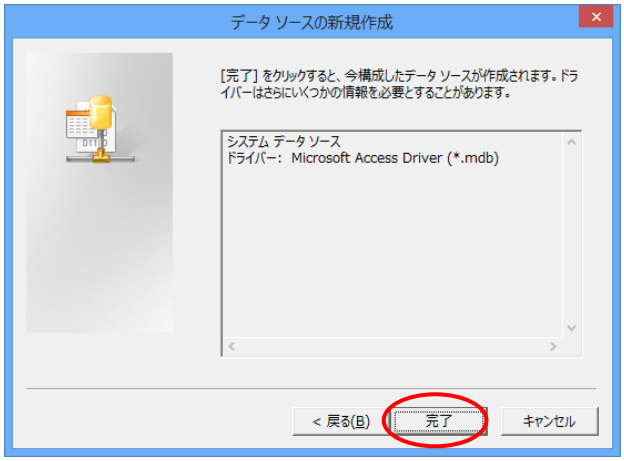
2



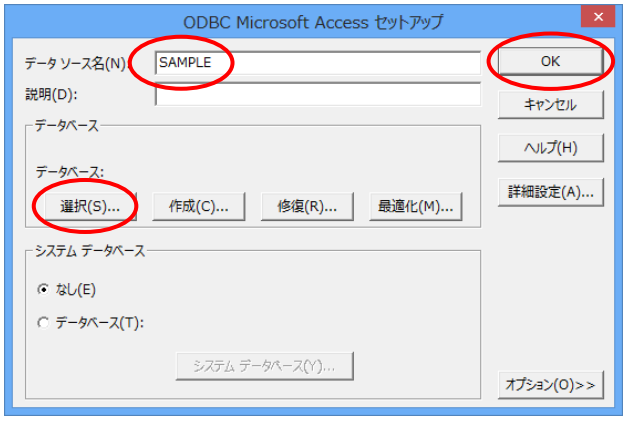
5



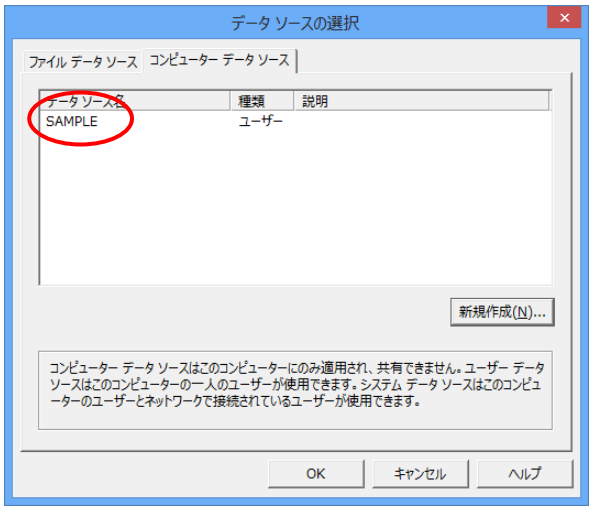
6



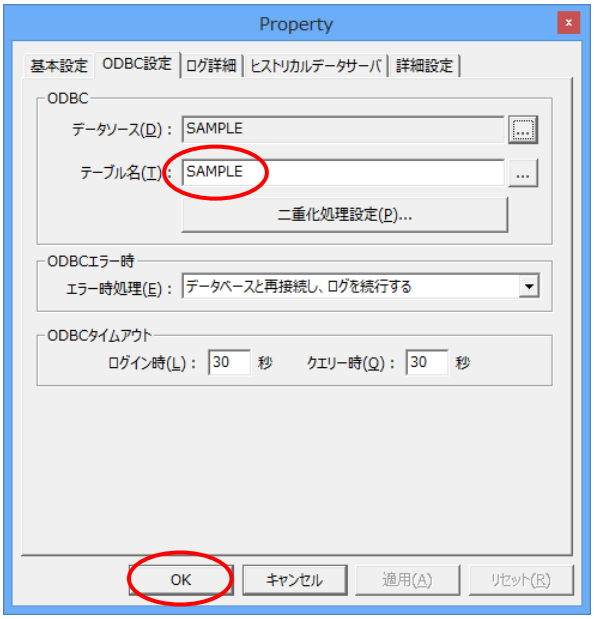
7



8

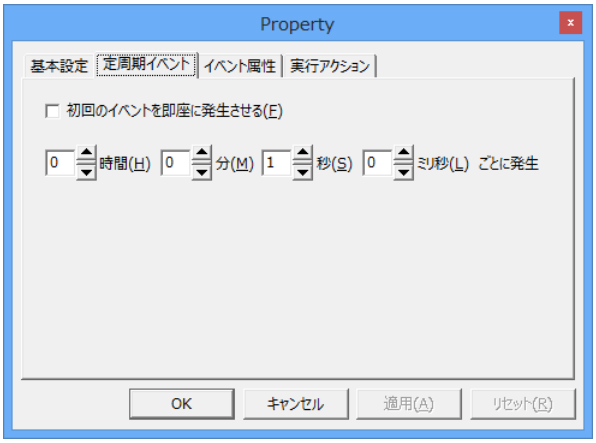






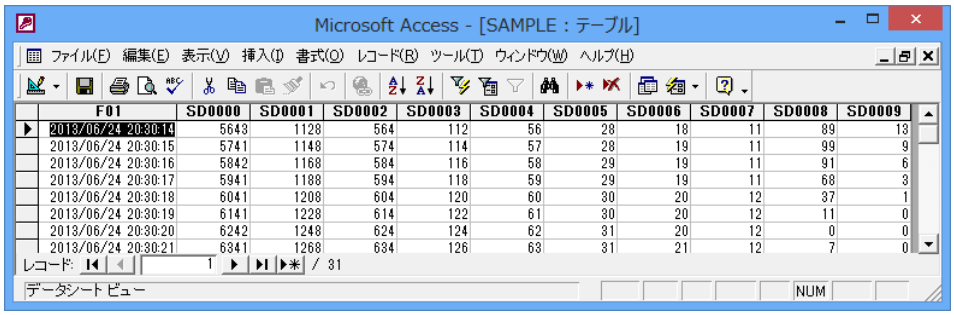
イベント設定

定周期イベント「E01」を追加します。1秒周期で実行アクションを「A01」に設定します。



解説

ODBC データベースへのロギングにおける注意点は、データベースのフィールド名とロガーアクションのフィールド名を一致させる必要がある点です。その他は CSV ファイルへのロギングと同様です。



F01	SD0000	SD0001	SD0002	SD0003	SD0004	SD0005	SD0006	SD0007	SD0008	SD0009
2013/06/24 20:30:14	5643	1128	564	112	56	28	18	11	89	13
2013/06/24 20:30:15	5741	1148	574	114	57	28	19	11	99	9
2013/06/24 20:30:16	5842	1168	584	116	58	29	19	11	91	6
2013/06/24 20:30:17	5941	1188	594	118	59	29	19	11	68	3
2013/06/24 20:30:18	6041	1208	604	120	60	30	20	12	37	1
2013/06/24 20:30:19	6141	1228	614	122	61	30	20	12	11	0
2013/06/24 20:30:20	6242	1248	624	124	62	31	20	12	0	0
2013/06/24 20:30:21	6341	1268	634	126	63	31	21	12	7	0



フィールド名に SQL の予約語（「DATE」など）を使用することはできません。

４－７　ロギングの処理要求を PLC から受け取る（Sample07）

概要

PLC のラダーロジックによって計算された演算値をログデータに保存したい場合、演算が終了したことを表すビット信号を PLC からハンドシェイクビットとして受け取り、そのビットの ON/OFF によってロギングを行うことで演算結果を確実にロギングすることができます。このような場面では、イベントログのトリガを上記のハンドシェイク用ビットタグによって発生させるように設定します。

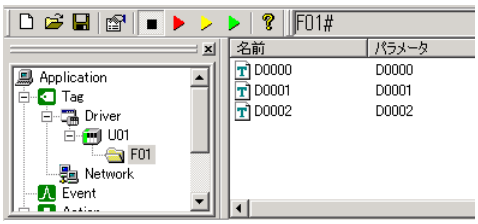
ここでは、三菱 PLC でのラダーシーケンスを例に、ハンドシェイクビットとして「D0002」を用い、「D0000」「D0001」のデータをロギングする方法について解説します。

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample07¥logger_sample07.txt

設定

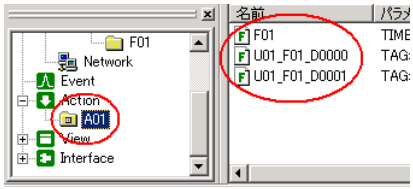
タグ設定

データ収集のためのタグ「U01.F01.D0000」「U01.F01.D0001」及びハンドシェイク用のタグ「U01.F01.D0002」を設定します。



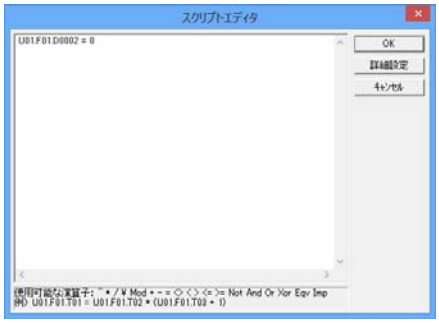
アクション設定

ロガーアクション「A01」を追加します。時刻フィールド及び「D0000」と「D0001」のタグフィールドを作成します。



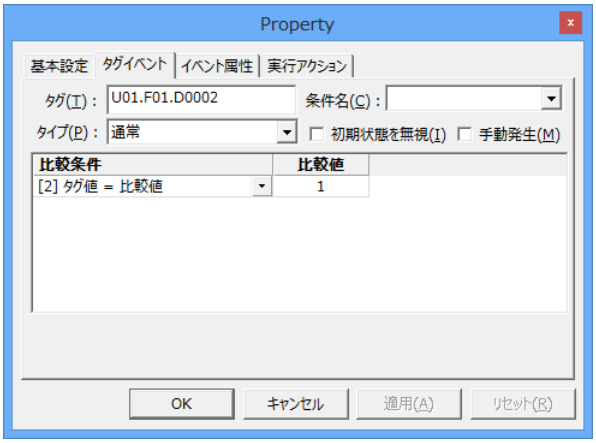
「処理が終了した時」のスクリプトで、ハンドシェイクビットを OFF にする処理を記述します。ここでは、下記のように記述します。

```
U01.F01.D0002 = 0
```



イベント設定

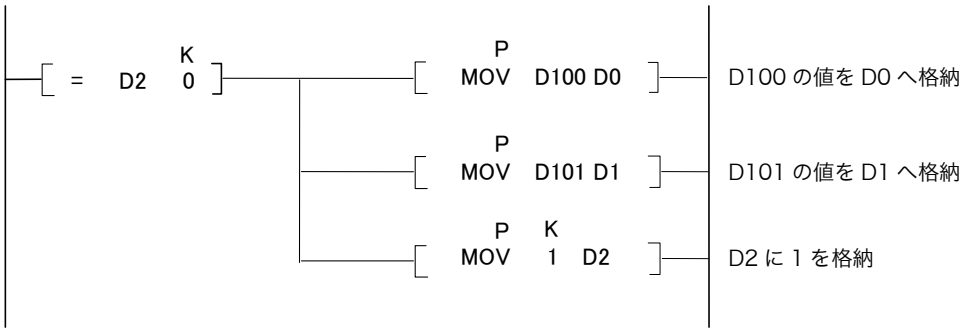
タグイベントを 1 つ追加します。「U01.F01.D0002」が 1 になったときに発生するように設定し、実行アクションに A01 を指定します。



ラダー側では、ロギングを指示したいタイミングで「D0002」を ON にするように設定します。タグイベントは D0002 の ON を検出し、アクションが実行されます。

解説

下記は三菱 PLC でのサンプルラダーです。



ラダープログラムが実行されると、「D0000」および「D0001」へデータをセットし、ハンドシェイク用の「D0002」に「1」がセットされます。ロガーをオンラインにすると、「D0002」の値が「1」である事によりタグイベントが発生し、第 1 回目のロギングが行われます。ロギング終了後、ロガーアクションによって「D0002」に「0」が書き込まれます。このラダーの例では、「D0002」が「0」になると、PLC はラダープログラムの［ =D2 K0］の条件により各 MOVP 命令が実行されて「D0002」に再び「1」が設定され、ロギングが繰り返し行われます。この動作はロガーがオフラインになるまで繰り返し実行されます。

４－８　イベント型ログフォーマットで定周期にログをとる（Sample08）

概要

ここでは、イベント型ログフォーマットで定周期にロギングを行うための方法について解説します。この方法によって、以下のようなニーズを解決することができます。

一般型ログのログフォーマットでは、フィールド数はロギング対象とするタグ数に依存します。従って、大量なタグ数でのロギングを行う場合、一般型ログではフィールド数が増えます。

（一般型ログの場合）

時刻	U01.F01.SD0001	U01.F01.SD0002	・・・	U01.F01.SD0999
2003/1/1 10:12:00	81	163	・・・	1243
2003/1/1 10:12:10	124	1635	・・・	6243
・・・	・・・	・・・	・・・	・・・

定周期でロギングを行いたいが、何らかの理由でフィールド数を増やしたくないような場合には、スクリプト Ver1 アクションの利用によって下記のようなイベント型ログフォーマットで定周期にログをとることが可能です。

（イベント型ログフォーマットで定周期にログ出力した場合）

時刻	タグ名	値
2003/1/1 10:12:00	U01.F01.SD0001	81
2003/1/1 10:12:00	U01.F01.SD0002	163
・・・	・・・	・・・
2003/1/1 10:12:00	U01.F01.SD0999	1243
2003/1/1 10:12:10	U01.F01.SD0001	124
2003/1/1 10:12:10	U01.F01.SD0002	1635
・・・	・・・	・・・
2003/1/1 10:12:10	U01.F01.SD0999	6243

この例では、SD0000～SD0999 までを 10 秒ごとに下記のフォーマットで保存します。

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample08¥logger_sample08.txt

設定手順

タグ設定

データ収集のためのタグ「U01.F01.SD0000」～「U01.F01.SD0999」を設定します。

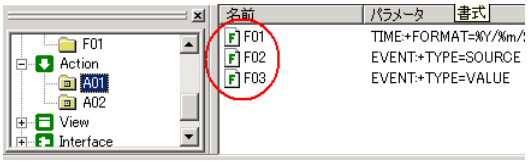


アクション設定

ロガーアクション「A01」を追加します。時刻フィールド「F01」及び下記のイベントフィールドを追加します。

F02:フィールドタイプ「タグ名」

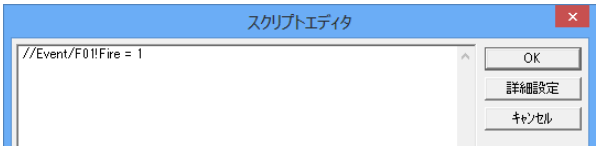
F03:フィールドタイプ「値」



スクリプト Ver1 アクション「A02」を追加します。スクリプトに下記を記述して下さい。

//Event/F01!Fire = 1

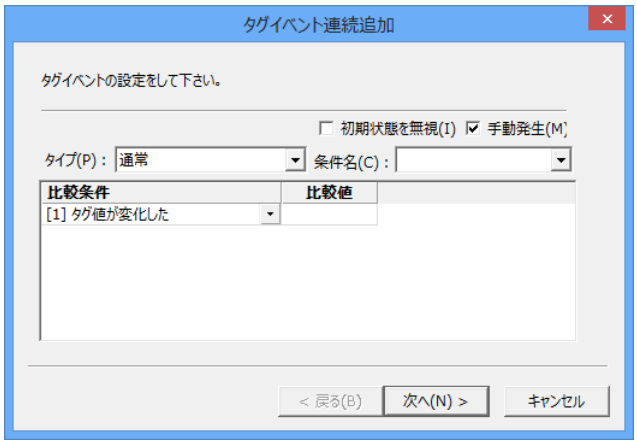
これにより、A02 が実行される都度、F01 のイベントが発生します。



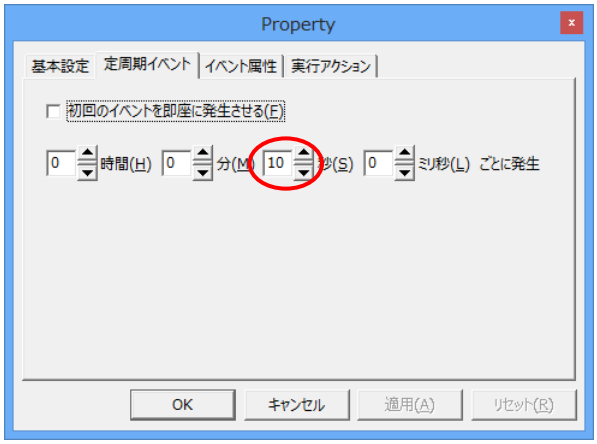
イベント設定

フォルダ「F01」を作成し、実行アクションに「A01」を設定します。

「U01.F01.SD0000」～「U01.F01.SD0999」を選択して、「F01」フォルダにドラックアンドドロップします。タグイベントの連続追加ウイザード（右画面）が表示されるので、「手動発生」をチェックします。この設定により、タグの値が変化してもイベントが発生しなくなります。



定周期イベント「E01」を追加し、実行アクションに「A02」を設定します（※E01 はフォルダ「F01」の中に追加しないこと）。



解説

イベントフォルダ「F01」内に設定した SD0000 から SD0999 までの 1000 個のタグイベントには、すべて「手動発生」がチェックされています。「手動発生」をチェックする事で、タグイベントはタグ値が変化してもイベントが発生しなくなります。一方、スクリプトアクションで設定した下記の式は、フォルダ内のタグイベントを手動で発生させます。

```
//Event/F01!Fire = 1
```

上記スクリプトは、F01 のイベントを発生させます。従って、スクリプトアクションが 10 秒ごとに呼び出されることにより、10 秒周期で「F01」内のすべてのタグイベントが手動で発生します。タグイベントが発生することにより、「F01」で設定した「A01」のロガーアクションが実行され、結果として下記のようなログが行われます。

"2003/01/01 3:19:00","U01.F01.SD0000","886"
"2003/01/01 3:19:00","U01.F01.SD0001","177"
…
"2003/03/17 3:19:00","U01.F01.SD0999","0"
} 2003/01/01 03:19:00 のログ

"2003/01/01 3:19:10","U01.F01.SD0000","1886"
"2003/01/01 3:19:10","U01.F01.SD0001","377"
…
"2003/03/17 3:19:10","U01.F01.SD0999","8"
} 2003/01/01 03:19:10 のログ

…



この例で使用した「E01!Fire」とは、イベントが持つシステムプロパティです。このプロパティを操作する事により、強制的にイベントを発生させることができます。イベントに関する詳細は「イベントマニュアル」を参照してください。

4－9　CSV ログのファイル名を動的に変更する（Sample09）

概要

CSV ファイルでのロギングで、ログファイルのファイル名をランタイム中に目的に応じて変更しながらロギングを行いたい場合があります。

例えば下記のような画面です。

●ワーク、段取り番号ごとにログのファイル名を変更したい
●外部アプリケーションからの設定指示でファイル名を指定したい
●PLC のロジックによってファイル名を指定したい

ここでは、上記のような場面を想定して、ランタイム中にログファイルのファイル名を動的に変更する方法について解説します。この例では、SD0000～SD009 までのデータを、ワーク番号ごとにファイル名を変更して保存します。

具体的には、U02.F01.T01 を文字列タグとして定義し、このタグの内容をファイル名として使用します。通信によって PLC からタグの内容を取得すれば、同様の処理方式によって PLC のラダーによるファイル名の制御が可能となります。また、このタグの内容を VisualBasic や SCADA 等の外部アプリケーションから IPLink や OPC/DDE 経由で書き換える事によっても、ファイル名の制御が可能となります。

尚、この例では、データログのタイミングとして U02.F01.T02（ビットタグ）を定義し、TRUE に変化した時に SD0000 から SD0009 までのデータをロギングします。

WORK01.txt

時刻	SD0000	・・・	SD0009
2003/1/1 10:12:00	0	・・・	11
・・・	・・・	・・・	・・・
2003/1/1 10:12:02	5	・・・	11

WORK02.txt

時刻	SD0000	・・・	SD0009
2003/1/1 10:12:02	12	・・・	22
・・・	・・・	・・・	・・・
2003/1/1 10:12:10	23	・・・	22

WORK03.txt

時刻	SD0000	・・・	SD0009
2003/1/1 10:12:10	14	・・・	33
・・・	・・・	・・・	・・・
2003/1/1 10:12:15	16	・・・	33

このサンプルは以下に格納されています。
設定ファイル：　¥¥sample¥common¥action¥logger¥sample09¥logger_sample09.txt

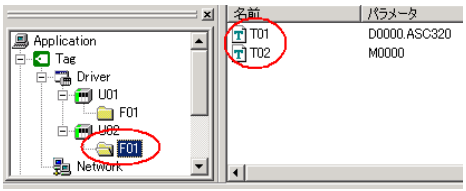
設定手順

タグ設定

データ収集のためのタグ「U01.F01.SD0000」～「U01.F01.SD0009」を設定します。

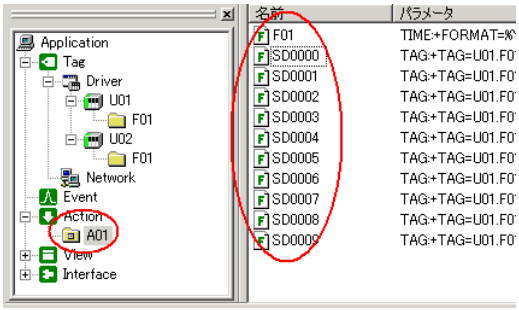


ファイル名のためのタグ「U02.F01.T01」とデータ取得トリガのためのタグ「U02.F01.T02」を設定します。「T01」は文字列タグとして定義してください。



アクション設定

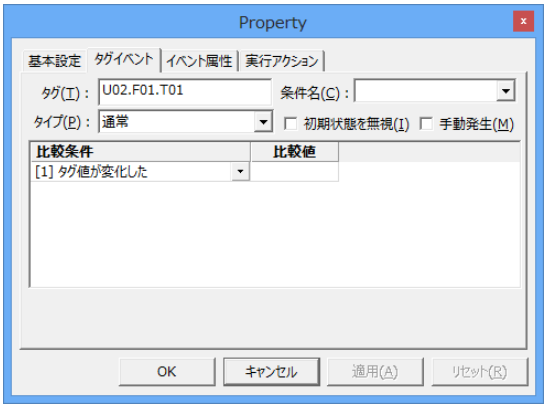
ロガーアクション「A01」を追加し、時刻フィールド「F01」及び「SD0000～SD0009」のタグフィールドを追加します。ここで、ファイル名の「拡張」として「ファイル名にタグ値を付加する」を選択し、「タグ名」として「U02.F01.T01」を指定してください。

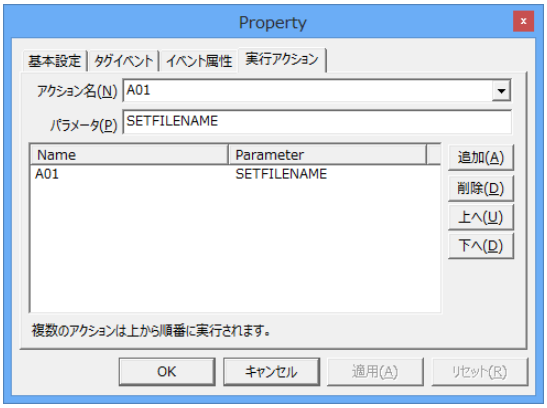


イベント設定

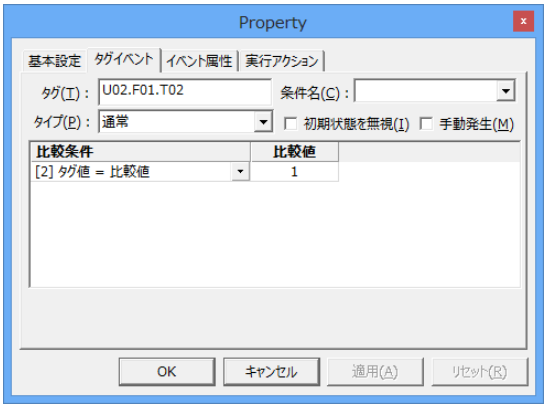
タグイベントとして「U02.F01.T01」を追加します。このイベントはロギング処理のためのものではなく、ログファイル名の変更をアクションに通知するためのものです。

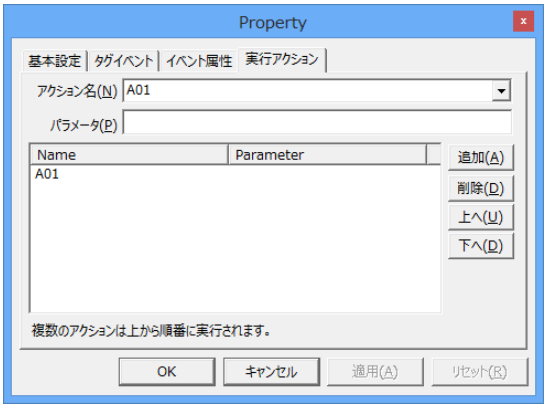
「U02.F01.T01」が変化した時に（すなわち、ファイル名の文字列が変更された時）、アクション名「A01」、パラメータ「SETFILENAME」を実行します。パラメータ「SETFILENAME」を指定することにより、ロガーアクションはファイル名の変更を認識します（尚、この呼び出しではロギングは行われません）。





タグイベント「U02.F01.T02」を追加し、実行アクションに「A01」を指定します。この設定により、「U02.F01.T02」が ON になったときに「A01」が実行されます。





解説

イベントの実行アクションの定義で「SETFILENAME」パラメータを指定すると、そのタイミングでログファイルのファイル名を変更することができます。このサンプルの場合「U02.F01.T01」の文字列が変化したとき、つまりファイル名が変化したときに、ログファイルのファイル名を変更しています。動作を確認するには、適当な文字を「U02.F01.T01」に設定後、「U02.F01.T02」を TRUE にしてみて下さい。ファイル名を変更したい場合は「U02.F01.T01」に異なるファイル名を再設定した後に、再び「U02.F01.T02」を FALSE から TRUE へ変更する事によって、新しいファイル名によってロギングが行われます。



ファイル名が設定されていない状態の場合、ログファイルは保存されません。上記の例では、「U02.F01.T01」が空白文字のまま「A01」「SETFILENAME」を実行したとしても、ファイル名は設定されません。

「ロガーアクション詳細マニュアル」Copyright © 2010 Roboticsware, Inc. All rights reserved.
2010 年 6 月 1 日 発行　株式会社ロボティクスウェア
〒331-0811 埼玉県さいたま市北区吉野町 1-11-1　TEL:048-782-9861 FAX:048-782-9862

■本書は著作権上の保護を受けています。本書の一部または全部について、株式会社ロボティクスウェアの
文章での許諾を得ずに、いかなる方法においても無断で複写・複製することは禁じられています。
■記載されている会社名、製品名、ブランド名は、対応する法人または個人の（登録）商標です。
■このソフトウェアの仕様及び本書に記載されている事項は、将来予告無しに変更することがあります。