[APNOTE04]

監視時間帯で DINPUT 変化時にアラート出力

ABS-9000 DeviceServer
APNOTE04 Rev A.1.0
2008/10/1

オールブルーシステム (All Blue System)
ウェブページ: www.allbluesystem.com
コンタクト:contact@allbluesystem.com

1　　イントロダクション

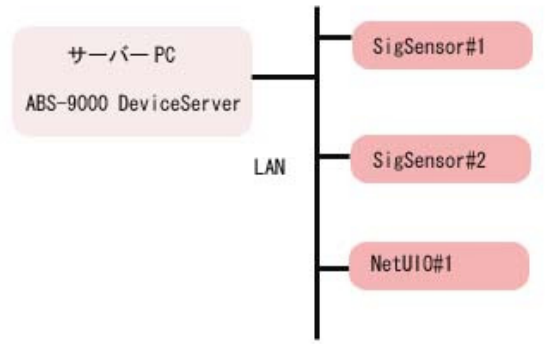
アラームデバイス(SigSensor,NetUIO)の DINPUT の値が変化した時に、アラート出力（ブザー、ランプ点滅、メール通知）を行います。このとき、予め決められた監視時間帯に入っている時だけ、監視を行うシステムについて説明します。

複数のアラームデバイスを同時に監視しますので、いずれかのデバイスの DINPUT 入力が変化した場合に、デバイス名と変化したポートをアラームデバイスのLCD に表示します。 また同時に、電子メールでアラート内容を通知します。

2　　必要な機材・リソース

必要なシステムやデバイス等	説明
ABS-9000 DeviceServerの動作しているPC	スタンダードライセンスもしくはエンハンスライセンスが必要になります。
SigSensorデバイス NetUIOデバイス	DINPUT の入力 I/O にセンサーやスイッチ等が実装されたSigSensor または、NetUIOデバイスが最低１台必要になります。アラームデバイスを複数同時に接続しても使用できます。

3　　システム構成図



4　　システム動作概要

- アラームデバイス(SigSensor,NetUIO) 上のDINPUT入力値が変化すると、ALARM_DINPUT_CHANGEイベントが DeviceServer で発生します。
- ALARM_DINPUT_CHANGE イベントハンドラで、変化のあったデバイス・ポート番号を特定した後に、システムアラートデバイスとして登録されているアラームデバイスに、ブザーとランプ、LCD メッセージを出力します。

また同時に、電子メールでも通知を行います。

- Windows のタスクスケジューラコマンド(ATコマンド)で、予め設定された監視時間帯(監視開始時刻と監視終了時刻) にスクリプトを起動して、ALARM_DINPUT_CHANGE イベントハンドラで上記の検出を行うかどうかのフラグを操作します。

5 設定手順

5.1 デバイス設定

SigSensor デバイスまたは、NetUIOデバイスを LAN 上に設置します。

DeviceServer のアラーム管理プログラムで、設置したアラームデバイスをDeviceServer に登録します。

また、アラーム管理プログラムで、アラームデバイス(SigSensorまたは NetUIO) の下記の項目の詳細設定行います。

アラーム管理プログラムで設定するデバイスの詳細機能設定	
設定が必要な項目	**設定内容**
IPアドレス	デバイスのIP アドレスを設定する
IPネットワークマスク	デバイスを設置した LAN の環境に合わせる
デフォルトゲートウェイアドレス	デバイスを設置した LAN の環境に合わせる
イベント送信先ホストアドレス	DeviceServer の動作しているPC の IPアドレスを設定する
送信先ポート番号	27102
DINPUT を有効にする	チェックを付ける
DINPUT値が変化したときに、サーバーにCSVIFパケット送信	チェックを付ける

デバイス登録と詳細機能設定の方法については “DeviceServerユーザーマニュアル”を参照してください。また、SigSensor、NetUIO デバイスの詳しい使用方法については、”SigSensor_NetUIOユーザーマニュアル” を参照してください。

5.2 スクリプト・イベントハンドラ設定

5.2.1 DINPUT_ALARM_ACTION スクリプト作成

監視時間帯に、DINPUT の値が変化した時に実行するスクリプトを作成します。

システムアラートデバイスとして登録されているアラームデバイスに、ブザーとランプ、LCD メッセージを出力します。また同時に、電子メールでも通知を行います。

ファイル名(DINPUT_ALARM_ACTION.lua) で DeviceServerのスクリプトフォルダに保管します。アラート出力の内容や、メールの宛先部分、メール本文は、適宜環境に合わせて修正してください。

```
file_id = "DINPUT_ALARM_ACTION"
```

```
local device_name = g_params["DeviceName"]
local device_port = g_params["DevicePort"]
-----------------------------------------------------------------------
-- システムアラートとしてフラグが設定されたデバイスに,DINPUT の変化を
-- Red ランプと Beep1 ブザー, LCD メッセージで知らせる
-----------------------------------------------------------------------
local stat,name,type = alarm_sysalert_list()
if not stat then error() end
for key,val in ipairs(name) do
  if (type[key] == "SIGSENSOR") or (type[key] == "ALARMSIGNAL") then
    if (type[key] == "SIGSENSOR") then
      stat = alarm_signal_message(val,"**DINPUT CHANGE*" .. device_name .. " " .. device_port)
      if not stat then error() end
    end
    stat = alarm_signal_set(val,"BlinkingRed",true)
    if not stat then error() end
    stat = alarm_signal_set(val,"Beep1",true)
    if not stat then error() end
  end
end
-----------------------------------------------------------------------
-- メールでDINPUT の変化を通知する
-----------------------------------------------------------------------
error_mail_addr = "エラーメール宛先 <your_mail_address@your_mail_domain.com>"
body = {}
table.insert(body,"アラームデバイスの DINPUT に変化があります")
table.insert(body,"デバイス名  :" .. device_name)
table.insert(body,"DINPUTポート:" .. device_port)
stat = mail_send(error_mail_addr,"","アラームデバイス DINPUT 変化",unpack(body))
if not stat then error() end
```

5.2.2 ALARM_DINPUT_CHANGE イベントハンドラの作成

アラームデバイスのDINPUT値が変化した時に起動される、ALARM_DINPUT_CHANGE イベントハンドラを下記の様に記述します。ファイル名(ALARM_DINPUT_CHANGE.lua) で DeviceServerのスクリプトフォルダに保管します。

```
file_id = "ALARM_DINPUT_CHANGE"
log_msg("start..",file_id)
```

```
--[[
stat = input_change_detect(device,pin_no,detect_type)

指定されたデバイスのDIN_xxxの立ち上がりエッジ、立下りエッジまたは、両方の検出を行う。
同一デバイスの同一ピンに対する検出を複数回行うことはできない。

**** 注意 ***************************************************************************
DeviceServer 起動後に最初にコールしたときは、それぞれのデバイス・ポートで前回のポート
値が無い為、必ず detect_type 2 で検査すると true になる。
エッジを確実に検出するためには、全てのデバイス・ポートに対してこの関数をコールして
前回のポート値を保存しておく必要がある。

この関数は、ALARM_DINPUT_CHANGE のイベントハンドラの中でしか使用できない
(スクリプトに渡されるパラメータに依存する)
**************************************************************************************

  stat:Boolean       指定したエッジを検出した場合は true, しなかった場合は false が返る
  device:String      デバイス名称
  pin_no:Number      検出対象の DINPUT ピン番号、 1,2,3,4 のいずれかひとつを指定
  detect_type:Number 検出するタイプを指定
    0:     値が 1 -> 0 に変わった時を検出
    1:     値が 0 -> 1 に変わった時を検出
    2:     値が 1 -> 0 もしくは 0 -> 1 に変わった時を検出
]]
function input_change_detect(device,pin_no,detect_type)
  local target_pin = "DIN_" .. tostring(pin_no)
  if (g_params["DeviceName"] == device) then
    local key_name = "DIN" .. ":" .. device .. ":" .. target_pin
    local stat,prev_state = get_shared_data(key_name)
    if not stat then error() end
    if not set_shared_data(key_name,g_params[target_pin]) then error() end

    if (detect_type == 0) then
      if ((prev_state == "0") and (g_params[target_pin] == "1" )) then return true else return false end
    elseif (detect_type == 1) then
      if ((prev_state == "1") and (g_params[target_pin] == "0" )) then return true else return false end
    elseif (detect_type == 2) then
      if (prev_state ~= g_params[target_pin]) then return true else return false end
    else
```

```
      error()
    end
  else
    return false
  end
end


-----------------------------------------------------
-- DINPUT の検出が抑止されているかを調べる
-- SUPPRESS_ALARM が共有データに定義されている場合は
-- スクリプトを実行しない
-----------------------------------------------------
stat,val = get_shared_data("SUPPRESS_ALARM")
if not stat then error() end
if val == "" then
  --------------------------------
  -- DINPUTポートの変化を検出する
  --------------------------------
  local device_name = g_params["DeviceName"]
  local change_port_list = {}
  if input_change_detect(device_name,1,2) then table.insert(change_port_list,"1") end
  if input_change_detect(device_name,2,2) then table.insert(change_port_list,"2") end
  if input_change_detect(device_name,3,2) then table.insert(change_port_list,"3") end
  if input_change_detect(device_name,4,2) then table.insert(change_port_list,"4") end
  if not script_exec("DINPUT_ALARM_ACTION","DeviceName,DevicePort",list_to_csv(device_name,
                      table.concat(change_port_list,","))) then error() end
  log_msg("DINPUT device = " .. device_name .. " port = " .. table.concat(change_port_list,","),file_id)
end
```

5.2.3 監視時間帯の有効フラグ操作スクリプトの作成

監視時間帯を共有データ（キー値 : SUPPRESS_ALARM）の有無で操作するためのスクリプトを２つ作成します。

監視時間帯に入った時に実行するスクリプト DIN_CHECK_ON を作成します。

ファイル名(DIN_CHECK_ON.lua) で DeviceServerのスクリプトフォルダに保管します。

```
file_id = "DIN_CHECK_ON"
log_msg("start..",file_id)
if not set_shared_data("SUPPRESS_ALARM","") then error() end
```

監視時間帯から抜けた時に実行するスクリプト DIN_CHECK_OFF を作成します。

ファイル名(DIN_CHECK_OFF.lua) で DeviceServerのスクリプトフォルダに保管します。

```
file_id = "DIN_CHECK_OFF"
log_msg("start..",file_id)
if not inc_shared_data("SUPPRESS_ALARM") then error() end
```

⚠ 注意

スクリプト中に日本語を記述するときは、スクリプトファイルを UTF-8N 形式で保存してください。Shift_JISや UTF-8 BOM付き形式などで保存すると、DeviceServer でエラーが発生します。Windows付属のワードパッドやメモ帳ではこの形式で保存できませんので、別途 UTF-8N 形式で保存可能なエディタソフト（*1）を使用してください。

（*1）TeraPad 等のソフトウエアがよく使用されています。

6 Windows タスクスケジューラの設定

定期的に決められた時刻でスクリプトを実行するために、Windows のタスクスケジューラを利用します。また、コマンドプロンプトからスクリプトを実行するためのプログラムは、ABS-9000 DeviceServer インストール時に保管されている ScriptExecCmd.exe を使用します。

この例では、タスクスケジューラのジョブ登録は DeviceServer の動作している PC で行います。

6.1 ScriptExecCmd.exe プログラムの設定

ScriptExecCmd.exe プログラムを "C:¥Program Files¥AllBlueSystem" に配置してください。また、同一フォルダに ScriptExecCmd.ini ファイルを作成して、ユーザー名とパスワードを予め指定しておいてください。

ScriptExecCmd.iniファイルを下記の内容で作成します。ユーザー名とパスワードは環境に合わせて変更してください。ScriptExecCmd.exe を実行したときに、iniファイルが見つからない場合は、自動的にiniファイルがデフォルト値で作成されます。

```
[ScriptExecCmd]
HostName=localhost
UserName=user_name
Password=user_password
KeyList=
ValList=
```

ScriptExecCmd.exe プログラムと ScriptExecCmd.ini ファイルの設定方法については “DeviceServer ユーザーマ

ニュアル” 中の ”その他のプログラム” の章を参照してください。

6.2 タスクスケジューラへの JOB 登録

ここでは、DINPUT 監視時間帯の開始時刻を 22:00、終了時刻を 05:00 とする例で説明します。

DeviceServer の動作する PC で、コマンドプロンプトを起動して、下記のコマンドを実行してください。必ず、システム管理者権限を持った Windows アカウントでログインしてから実行してください。

```
at 22:00 /every:M,T,W,Th,F,S,Su "C:¥Program Files¥AllBlueSystem¥ScriptExecCmd.exe" DIN_CHECK_ON
at 5:00 /every:M,T,W,Th,F,S,Su "C:¥Program Files¥AllBlueSystem¥ScriptExecCmd.exe" DIN_CHECK_OFF
```

Windows タスクスケジューラと “ATコマンド” についての詳しい説明はマイクロソフト社のドキュメントを参照してください。

7 備考

DeviceServer の起動直後や再起動後は、監視状態に設定しています。タスクスケジューラで設定した時刻になると本来の監視状態に移行するようになります。

8 このドキュメントについて

8.1 著作権および登録商標

8.2 連絡先

オールブルーシステム (All Blue System)
ウェブページ http://www.allbluesystem.com
メール contact@allbluesystem.com

8.3 このドキュメントの使用について

このドキュメントは、ABS-9000 DeviceServer の一般的な使用方法と応用例について解説してあります。お客様の個別の問題について、このドキュメントに記載された内容を実際のシステムに利用するときには、ここに記載されている以外にも考慮する事柄がありますので、ご注意ください。特に安全性やセキュリティ、長期間にわたる運用を想定してシステムを構築する必要があります。

オールブルーシステムでは ABS-9000 DeviceServer の使用や、このドキュメントに記載された内容を使用することによっ

て、お客様及び第三者に損害を与えないことを保証しません。 ABS-9000 DeviceServer を使用したシステムを構築するときは、お客様の責任の下で、システムの構築と運用が行われるものとします。