



User's guide



A3600AX ActiveX component

Application:

📁 Adash 3600 on-line monitoring system communication component

Characteristics:

📁 Implements public programming interface to access A3600 online system data from various applications

*Ref: 11092002 OS
31012003 JC*

Contents

Purpose of the component.....	2
Description of the component	2
Timing.....	3
Dynamic data.....	3
Guids and ProgIDs	3
Distribution and installation	3
Methods and propertis	4
Methods	4
Properties	5
Events	6
Visual side of the component.....	6
Example.....	7
A3600AX – new interface A3600FLASH	9
Guids and ProgIDs	9
Methods and propertis	9
Example.....	12

Purpose of the component

The A3600AX component simplifies user data communication with the Adash A3600 online monitoring systems.

The component is dedicated to use in such tools as DHTML, MS Excel (generally VBA or Visual Basic), MS C++, Wonderware InTouch, Citect, Labview etc. Generally, the main application **must** support **ActiveX** components. And, of course, It is assumed the user has basic knowledge of programming in the tool selected.

Description of the component

The component implements A3600 systems serial communication protocol as basic buffering of the downloaded data. Ease of use is achieved by several methods, properties and events, where the user is not bound to implement own data polling mechanism, which is inherently present in the component implementation.

In closer look the component use another thread to download data by serial communication from the A3600 system, so the main application is not load by the communication. If some of the methods are synchronous, it's noticed in detailed description (see chapter **Methods and Properties**).

The component also brings properties for determining data **timestamp** and data **validity**.

Internal mechanism of the data download also implements error communication free function, so any error is signaled and line is repetitively tested, if communication can proceed.

Newly from version 1.5. of the component the dynamic data download is implemented.

Timing

Due to the principal mechanism of how data is stored in the NET module of the A3600 system, download of the data is performed in scans, each scan is followed by next scan at time incremented by **RefreshRate** property.

During one scan multiple measured values can be read from NET module buffer. How many values are read depends on RefreshRate.

Approximately it can be said that one measurement is done in 1 second by the A3600 system (one **measurement**, not one complete channel!), so for example, for **RefreshRate=10** 10 measured values will be read during one scan.

Dynamic data

Equivalent mechanism to the static data download is used in the case of the dynamic data. The component tries to determine whether some dynamic data are present on the MAIN module or not. When presence of a dynamic data measurement resultset is determined, the component downloads and internally caches the fresh resultset, which is held until the method [GetDynamicArray](#) is called and resultset is fetched.

RefreshRate property **does not** affect dynamic data checking and downloading.

To enable dynamic data processing [DynDataEnabled](#) property of the component must be set to True.

Guids and ProgIDs

The component has these GUIDs and PROGIDs:

PROGID = A3600COM.A3600COM.1

A3600COM.A3600COM.1 = 'A3600COM Class'

CLSID = 9B77DECB-31EA-49C3-BE91-9DCA2BE1746A

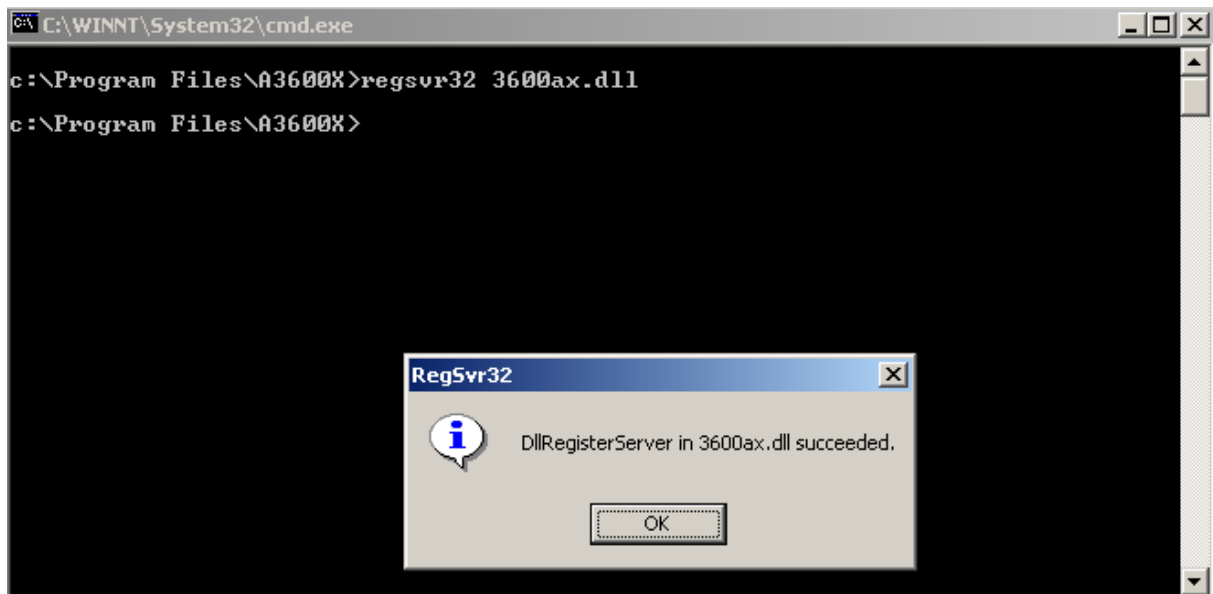
Distribution and installation

Distribution contains self extracting installation file **SETUP36AX.EXE** and contains also registration of the component.

The component is one file **3600AX.DLL**. Placement of the file is irrelevant.

Installation consists of component registration, which is done by included batch file **3600AXREG.BAT**, or manually from command line:

```
c:\Program Files\A3600X>regsvr32.exe 3600AX.dll
```



Methods and properties

In the description is used Visual Basic syntax and C++ IDL syntax

Methods

```
Sub OpenComm(nCom As Long, nBaudRate As Long, nTimeout As Long)
```

nCom	COM port number counted from 1
nBaudRate	baud rate set for A3600 system module NET, def.57600
nTimeout	timeout of serial communication in milliseconds

synchronous

Opens communication with the A3600 system on given port with given baud rate and sets communication timeout.

```
Sub CloseComm()
```

asynchronous

Closes communication with A3600 system.

```
Function GetDynamicArray(nChannel As long, sType As String, sSubType As String) As Variant
```

synchronous

Fetches dynamic data resultset of the given type and subtype.
 Resulting variant is empty if no resultset is currently present.
 If some resultset is present, array is returned. The array contains parameters and result.
DynDataEnabled must be set to True, else no records will be returned.

nChannel	Input channel of the A3600 system, 1-64 values possible (the number is counted as Module*8+ChannelOnModule)
sType	string representing desired dynamic data type "TIME" Time signal resultset is requested "SPEC" Spectrum of the signal resultset is requested
sSubType	string representing measuring type: "LF" How frequency measurement "HF" High frequency measurement "LIN" Linear measurement (10-10000 Hz band) "ENV" Envelope detection measurement "PROC" process parameter (f.e. temperature)

Example:

```
'A3600COM1 is instance of the A3600AX ActiveX component placed on the workspace
Dim V, nNumParams &, nLines&, nStep double
```

```
' try to fetch dynamic data
V = A3600COM1.GetDynamicArray(n, "TIME", "LIN")

' test if present
If NOT IsEmpty(V) AND IsArray(V) Then
  'process parameters
  nNumParams = V(0)      'number of parameters
  nLines      = V(1)      'number of the measured or computed records
  nStep       = V(2)      'time or frequency step of the signal

  ' process results - do your work here
  For n = nNumParams To UBound(V)
    V(n) = 1.2345 * V(n) 'e.g. ...
  Next
End If
```

Properties

Property RefreshRate As Long

Sets refresh rate in milliseconds. Refresh rate indicates time interval between two data loads. **Does not** affect dynamic data checking and downloading.

Property Status As Long

Returns current component status:

0	ready, communicating
1	not ready, not communicating
2	ready, communication stopped (after CloseComm call)
3	ready, communicating, but data error occured (broken serial line f.e.)

Property DynDatatatus (nChannel As Long, sType As String) As Long

Returns code of the current present cached dynamic records in the internal component cache. The resulting code is sum of binary values 1(HF), 2(LF), 4(LIN), 8(ENV), indicating which of the signals are currently in the cache.

Property Value	(nChannel As Long , sType As String) As Double
Property Alert	(nChannel As Long , sType As String) As Double
Property Danger	(nChannel As Long , sType As String) As Double
Property Timestamp	(nChannel As Long , sType As String) As Date
Property Valid	(nChannel As Long , sType As String) As Long

nChannel	Input channel of the A3600 system, 1-64 values possible (the number is counted as Module*8+ChannelOnModule)
sType	string representing measuring type: "LF" How frequency measurement "HF" High frequency measurement "LIN" Linear measurement (10-10000 Hz band) "ENV" Envelope detection measurement "PROC" process parameter (f.e. temperature)

Property **Value** returns last read value of the given channel and measurement type.

Property **Alert** returns the alert value of the given channel and measurement type.

Property **Danger** returns the danger value of the given channel and measurement type.

Property **Timestamp** returns last read value timestamp of the given channel and measurement type.

Property **Valid** returns 1 if value of the given channel and measurement type was read valid, 0 if not.

Property DynDataEnabled As Boolean
--

Enables/disables dynamic data checking and downloading on the background.

Events

Event DataLoaded (Timestamp As Date)

This event is fired by component whenever scan is performed and some data are loaded. If it's needed it is up to the user to determine by the timestamp if data are new.

Implementing the event simplifies data read, user must not implement any timers. See example code for MSEXcel.

Visual side of the component

More than graphic the component exposes text info about current communication status.

```

----- A3600COM ACX ver. 1.4 -----
Overall status:Ready, communicating
Comm status:Comm OK
Refresh rate:2500 [ms]
#scans:128 #successes:128
Mod.   1   | 2   | 3   | 4   | 5   | 6   | 7
Chan. 12345678123456781234567812345678123456781234567
LF    xxxx-----
HF    xxxx-----
LIN   xxxx-----
ENV   xxxx-----
PROC  -----
TIME  1-----
SPEC  82-----

```

Meaning of most text rows is clear, here is description of some of them:

#scans - number of performed scans (both successful and unsuccessful)

#successes - number of performed successful scans

Mod. - index of multiplexer module MUX for given channel on row **Chan.**

Chan. - index of the channel of MUX for given module **Mod.**

LF, HF, LIN, ENV, PROC - if there is character '-' in the column of the MUX/channel, the given signal **was not read** from the NET module; if the character is 'x' the signal **was read**.

TIME, SPEC - time and spectrum dynamic data status. The value is result of the sum of the signal path codes - every signal path has it's own code, LF=1, HF=2, LIN=4, ENV=8. So, in the snapshot above the TIME dynamic resultset is present for signal path LF on channel no.1, SPECTRA resultset is present for ENV on ch.#1 and for HF on ch.#2. If combination of resultsets is present for given channel, sum of the codes is showed in hexadecimal form (e.g. C hex (12 decimal) is 8+4, so ENV and HF is present)

Example

The script below implements start of the communication with A3600 system on COM2 and baud rate 57600 Bd, stop of the communication and event **DataLoaded** handling. Script is intended for use in MS Excel spreadsheet and the example is distributed along with component dll.

' there must exists A3600COM1 instance

Option Explicit

' we implement event DATALOADED, which is called whenever some data are
' loaded from A3600 system ActiveX A3600COM do it self

Private Sub A3600COM1_DataLoaded(ByVal Time As Date)

' values are defined as variants, we will override them by strings
' if values are not present for the channel

Dim i As Integer, LF, HF, LIN, ENV, Timestamp As Date

' iterate through 1st. 16 channels

For i = 1 To 16

' load each value separately

LF = A3600COM1.Value(i, "LF")

HF = A3600COM1.Value(i, "HF")

LIN = A3600COM1.Value(i, "LIN")

ENV = A3600COM1.Value(i, "ENV")

' filter invalid data

```

If (A3600COM1.Valid(i, "LF") = 0) Then LF = "----"
If (A3600COM1.Valid(i, "HF") = 0) Then HF = "----"
If (A3600COM1.Valid(i, "LIN") = 0) Then LIN = "----"
If (A3600COM1.Valid(i, "ENV") = 0) Then ENV = "----"

' simplified - we display timestamp of the LF values only !
Timestamp = A3600COM1.Timestamp(i, "LF")

' now display values from column 'H'
Worksheets("A3600").Cells(i + 1, 8) = LF
Worksheets("A3600").Cells(i + 1, 9) = HF
Worksheets("A3600").Cells(i + 1, 10) = LIN
Worksheets("A3600").Cells(i + 1, 11) = ENV
Worksheets("A3600").Cells(i + 1, 12) = Timestamp

Next
End Sub

' open communication
Private Sub OpenA3600()

' set refresh rate
A3600COM1.RefreshRate = 5000
' open communication COM2,57600Bd, 2000msec timeout
A3600COM1.OpenComm 2, 57600, 2000

End Sub

Private Sub CloseComm()
A3600COM1.CloseComm
End Sub

```

A3600AX – new interface A3600FLASH

This interface allows administration of FLASH card and read data from FLASH card.

Guids and ProgIDs

The component has these GUIDs and PROGIDs:

PROGID = A3600FLASH.A3600FLASH.1

A3600FLASH.A3600FLASH.1 = 'A3600FLASH Class'

CLSID = 8F220727-B90C-46D2-85E9-0DE980A1E184

Methods and properties

FLASH card administration

```
Function FindFlash() As Boolean
```

automatically find FLASH card

Result - true – FLASH card founded

- false – some error, use function [GetLastError\(\)](#)

This function is allways successful, if you use USB Compact flash reader. If you use e.g. PCMCIA adapter this function don't work correctly, because the PCMCIA card driver map the FLASH card as "hard disk" not as removable device. If you have PCMCIA adapter you must use function [SetFlashLocation\(\)](#).

```
Function FormatFlash() As Boolean
```

format FLASH

Result - true – FLASH card formatted

- false – some error, use function [GetLastError\(\)](#)

This function initializes FLASH card for using with A3600 system. New FLASH card must be formatted with this function. This function can take a few of minutes. If you use for FLASH card (or data file) localization function [SetFlashLocation\(\)](#), you can't use this function.

```
Function ClearFlash() As Boolean
```

clear FLASH

Result - true – FLASH card cleared

- false – some error, use function [GetLastError\(\)](#)

This function clear stored data on FLASH card. This function can take a few of minutes. If you use for FLASH card (or data file) localization function [SetFlashLocation\(\)](#), you can't use this function.

```
Function SetFlashLocation(sPath As String) As Boolean
```

set location of data file

sPath - location (directory) of data file or logical drive of FLASH card
e.g. [SetFlashLocation](#)("C:\temp\Data3600.mem")

Result - true – sPath setting is OK
- false – some error, use function [GetLastError\(\)](#)

You can copy data file from FLASH card to hard disk and than use this function for reading data.

Data file operations

```
Function OpenFlashFile() As Boolean
```

open file for reading data

Result - true – opening is OK
- false – some error, use function [GetLastError\(\)](#)

You must use this function before reading data from data file.

```
Function CloseFlashFile() As Boolean
```

close file after reading data

Result - true – closing is OK
- false – some error, use function [GetLastError\(\)](#)

You must use this function after reading data from data file.

```
Function MoveFirst() As Boolean
```

set position to first data record

Result - true – setting is OK
- false – some error, use function [GetLastError\(\)](#)

Use this function for setting initial position in data file.

```
Function MoveLast() As Boolean
```

set position to last data record

Result - true – setting is OK
- false – some error, use function [GetLastError\(\)](#)

Use this function for setting initial position in data file.

```
Function MoveNext() As Boolean
```

set position to next data record

Result - true – setting is OK
- false – some error, use function [GetLastError\(\)](#)

Use this function for moving from actual position to next position (+1) in data file.

Function `MovePrev()` As `Boolean`

set position to previous data record

Result - true – setting is OK
- false – some error, use function `GetLastError()`

Use this function for moving from actual position to next position (-1) in data file.

Data record manipulation

Property `GetMeasurementType()` As `String`

get type of measurement (static,dynamic)

Result - “static” – actual position is on static data measurement
- “dynamic” – actual position is on dynamic data measurement
- “none” – some error, use function `GetLastError()`

This function detects the type of measurement on actual position. With result of this function you can decide, if you use for data reading function `GetStaticData()` or `GetDynamicInfo()`, `GetDynamicData()`.

Property `GetStaticData()` As `Variant`

get data and info about static measurement

Result of this function is a `Variant` array with this specification:

- [0] – number of items in this array, if all is OK the value is 9, if the value is 0, use function `GetLastError()`
- [1] – serial number of module MAIN
- [2] – number of channel
- [3] – type of signal – LF, HF, LIN, ENV
- [4] – static measurement value
- [5] – alert value from A3600 system
- [6] – danger value from A3600 system
- [7] – measurement unit e.g. “mm/s” or “g”
- [8] – status, 80h – static data OK
 0 – error overflow
 1 – error ICP
 2 – error timeout
- [9] – date and time of measurement

Property `GetDynamicInfo()` As `Variant`

get info about dynamic measurement

Result of this function is a `Variant` array with this specification:

- [0] – number of items in this array, if all is OK the value is 12, if the value is 0, use function `GetLastError()`
- [1] – serial number of module MAIN
- [2] – number of channel
- [3] – type of signal – LF, HF, LIN, ENV
- [4] – type of dynamic measurement - “time” or “spec”
- [5] – number of items in dynamic data array
- [6] – measurement unit e.g. “mm/s” or “g”
- [7] – step in x-axe
- [8] – x-axe unit e.g. “Hz” or “ms”,
- [9] – date and time of measurement

- [10] – number of averaging
- [11] – rotation – now not used
- [12] – rotation unit – now not used

Property `GetDynamicData()` As `Variant`

get data from dynamic measurement

Result of this function is a `Variant` array with this specification:

- [0] – number of items in this array, if the value is 0, use function `GetLastError()`
- [1] – dynamic value
- .
- .
- .
- [n] – dynamic value

Special operation

Property `GetLastError()` As `Variant`

get last error

Result of this function is an `Integer` value with this specification:

- 1 ERR_FLASH_STRUCT_BAD
- 2 ERR_FLASH_FILE_NOT_OPEN
- 3 ERR_FLASH_NOT_DATA_READ
- 4 ERR_FLASH_EMPTY
- 5 ERR_FLASH_HEAD_BAD
- 6 ERR_CYCLIC_READ_BAD
- 7 ERR_NO_FIND_BEGIN_END
- 8 ERR_NO_DATA
- 9 ERR_NO_STATIC_DATA
- 10 ERR_NO_DYNAMIC_DATA
- 11 ERR_FORMAT_HARD_DISK
- 12 ERR_FORMAT_NOT_FOUND
- 13 ERR_MAKE_FORM_FILE_BAD
- 14 ERR_FLASH_BAD_LOG_DISC
- 15 ERR_FLASH_BAT_FILE_BAD
- 16 ERR_FLASH_FORM_PROC_BAD
- 17 ERR_SERNUM_LICENSE
- 30 ERR_UNDOCUMENTED_ERROR

Example

```
Dim oFlash As A3600AXLib.A3600FLASH

Set oFlash = New A3600AXLib.A3600FLASH
'localization of data file of FLASH card with oFlash.FindFlash
oFlash.SetFlashLocation ("E:\sdileni\A3600_Flash_File\2003.01.27\Data3600.mem")
'open data file
oFlash.OpenFlashFile
```

```

Dim val As Double
Dim i As Integer
Dim b As Boolean
Dim s As String

i = 1
'set init position
b = oFlash.MoveFirst()
Do While b
    'get type of actual measurement
    s = oFlash.GetMeasurementType

    If s = "static" Then
        aV = oFlash.GetStaticData
        If aV(0) = 0 Then
            Err = oFlash.GetLastError
            MsgBox (Err)
        Else
            Worksheets("Flash").Cells(10 + i, 1) = i
            For j = 1 To aV(0) Step 1
                Worksheets("Flash").Cells(10 + i, 1 + j) = aV(j)
            Next
        End If
    End If

    If s = "dynamic" Then
        'get info about dynamic measurement, if you want data use aV =
        'oFlash.GetDynamicData
        aV = oFlash.GetDynamicInfo
        If aV(0) = 0 Then
            Err = oFlash.GetLastError
            MsgBox (Err)
        Else
            Worksheets("Flash").Cells(10 + i, 1) = i
            For j = 1 To aV(0) Step 1
                Worksheets("Flash").Cells(10 + i, 1 + j) = aV(j)
            Next
        End If
    End If

    b = oFlash.MoveNext
    i = i + 1
Loop

oFlash.CloseFlashFile
Set oFlash = Nothing

```