TECH FASS s.r.o. **www.techfass.cz**

APS 400 Config

Programmer's Guide, version 2.5

(Date of edition: 27. 3. 2008, valid for build 2.5.0.0)

1.1 Content

2

Installation, configuration and program environment

The APS 400 Config software is a complex development tool for customization of the APS 400 access control system. It contains tools for system programming, simple diagnostics, visualization and status control.

2.1 Hardware and OS requirements

Recommended operating system is Microsoft Windows NT 4.0 (SP4) or higher ones (Windows 2000, Windows XP). HW equipment related to the used OS is necessary for efficient work. Monitor with resolution 1024x768 pixels is suitable. Disc space requirements are minimal.

2.2 Software installation and configuration

The software is available free of charge, link to the latest installation files is located on the web site http://www.techfass.cz/aps_400_dn_cz.html. The APS 400 Config software can also be distributed on CD ROM or other media for a particular order.

Follow the recommendations after running the installation file (`setup.exe`). No other configuration is necessary after finishing the set up procedure. The configuration file (`APSConfig.ini`) is being saved when exiting the program. When you change the parameters setting inappropriately and the default setting is demanded, simply close the program, erase the configuration file and run the program again.

2.3 Main program window, desktop, tool bar, status bar

APS Config main program window (*fig. 2.1*) contains standard parts (menu, tool bar, status bar ...).

The desktop is divided into three basic parts:

- *Application tree* … defines the hardware structure of the application.
- *Module inspector* … displays and allows editing the object properties selected in the application tree.
- *Script editor … text editor for creation of a program defining the application behavior.*

All three parts are interrelated. Thus the changes in one of these parts cause changes in the other two ones automatically.

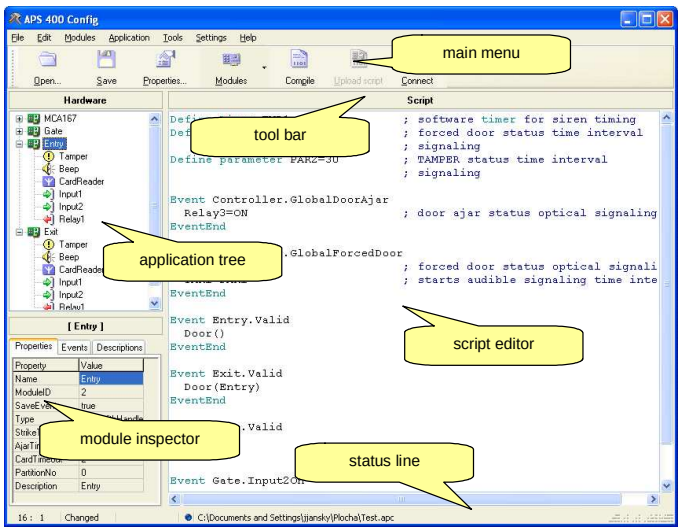


Fig. 2.1: Main program window

2.4 Data files used

The application source file is saved as one file with the extension `.apc`. Any time when the file is opened a backup file is created in the same folder as the source file. It has the same name but behind its suffix the extension `.bat` is added. After the source program is compiled successfully the configuration file for administration software is saved in the same folder as the source file. It has the same name and `.cfg` extension.

2.5 Main menu commands summary

All commands located in the program main menu are described in the table in the *fig. 2.3*. In order to show the list complete, the table in *fig. 2.2* contains also commands not included in the main menu (they can be run only by a keyboard shortcut). The function of the most of commands is the same as in the majority of common programs (file management, text editing, help…), therefore the commands specific for this program will be described in this text only.

Name	Shortcut	Description
Smart tag	Ctrl+ space	To be used when writing the script – the command displays a list of modules and devices that can be at given place of the script used
Fast constructions	Ctrl+J	To be used when writing the script – the command displays a menu with pre-defined parts of the script that can be inserted into the editor
Find a module	Ctrl+M	To be used when writing the script – after the command is run the module corresponding to the currently edited macro is found (highlighted)
Indent a block to the right	Shift+ Ctrl+I	Indents all selected lines 2 spaces to the right
Indent a block to the left	Shift+ Ctrl+U	Indents all selected lines 2 spaces to the left (if it is possible)

Fig.2.2: Commands not included in the main menu

Menu / command caption		Shortcut	Description
File		**Alt+S**	**Working with files**
	New	-	Creates a new (empty) file.
	Open	-	Opens an existing application file.
	Save	-	Saves the opened application file.
	Save as	-	Saves the opened application file under a new filename.
	Print	-	Prints out the applicaton report.
	Close	Alt+F4	Closes the APS Config application.
	Last opened files	-	Four commands for opening the last opened files.
Edit		**Alt+A**	**Script editor tools**
	Back	Alt+←	Takes back the last change in the script editor (if possible).
	Cut	Ctrl+X	Cuts the selected text into the windows clipboard.
	Copy	Ctrl+C	Copies the selected text into the windows clipboard.
	Paste	Ctrl+V	Pastes the text from the Windows clipboard into the script editor.
	Delete	Del	Deletes the text selected in the script editor.
	Select all	Ctrl+A	Selects all text in the script editor.
	Find	-	Opens the Find text dialog.
	Replace	-	Opens the Replace text dialog.
	Make comments	Ctrl+K	Converts lines selected in the script editor to the comments.
	Delete comments	Ctrl+L	Removes the comment marks form the lines selected in the script to editor (if the ";" mark is the first not-space character in the line).
	Collapse application tree	Ctrl+B	Collapses all nodes in the application tree.
	Sort macros	-	Sorts all macros in the script editor in the same order as defined in the application tree.

Fig. 2.3: Main menu commands summary – the 1st part

Modules		**Alt+M**	**Working with modules**
	Delete module	-	Deletes the module selected in the application tree.
	Controller MCA 168	-	Inserts the controller MCA 168 into the application tree.
	Network reader	-	Inserts the network reader module into the application tree.
Application		**Alt+E**	**Working with application**
	Compile	F9	Compiles the application to the binnary code.
	Upload script	Alt+C	Uploads the compiled binnary code to the controller, if the source file is changed the compilator is called before.
	Properties	Alt+V	Opens the Application properties dialog.
	Connect/Disconnect	-	Opens/closes the connection between the PC and the controller.
	Change application password	-	Opens the Change application password dialog.
Tools		**Alt+R**	**Visualisation and tuning**
	Download cardholders	-	Downloads the card IDs table from the controller memory.
	Download access groups	-	Downloads the access goups setting from the controller memory.
	Download time zones	-	Downloads the time zones setting from the controller memory.
	Download holidays table	-	Downloads the holidays setting from the controller memory.
	Application parametres	-	Downloads all application parameters from the controller memory.
	Controller properties	-	Downloads the controller properties.
	Network status	-	Starts the network status visulaization.
	Master module status	-	Starts visualization of the controller status.
	Network reader status	-	Starts visualization of the selected network reader module status.
	Download events archive	-	Starts downloading of the access system events archive.
	Show registers	-	Starts visualization of the registers values.
	Show timers	-	Starts visualization of the timers values.
	Upload service setting	-	Uploads the service settings into the controller.
Settings		**Alt+N**	**Program parameters setting**
	Script editor	-	Setting the script editor font and colors.
	Fast constructions	-	Opens the Fast constructions definition dialog.
	Communication servers	-	Opens the Customer communiaction servers paths dialog.
	Tools	-	Opens the Service settings definition dialog.
	Open scenario	-	Opens the saved scenario file.
	Save scenario	-	Saves current windows scenario into the selected file.
Help		**-**	**Working with help**
	Contents	F1	Opens the Help content.
	About	-	Opens the About application window.

Fig. 2.3 Main menu commands summary – the 2nd part

2.6 Program set up

Both the appearance and the behavior of the program at some operations can be influenced by APS Config software. Dialogs for all settings are activated with the commands from the *Settings* menu.

- *Fast constructions* are pre-defined parts of the text that can be inserted in the editor by hitting the keyboard shortcut CTRL+J. The dialog contains a list of defined constructions and buttons for their creation, editing and deletion. Two text parameters are defined for each construction: Menu title and Inserted text (fig. 2.4) The Menu title can be any text displayed in the menu activated by the shortcut. The Inserted text is the text inserted in the script after the item in the menu is selected. The character "%" defines the position of cursor after text insertion.

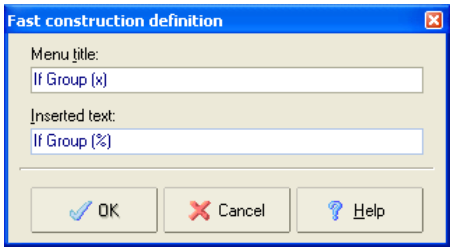


Fig. 2.4: Fast construction definitions

- Standard *comunication servers* form a part of program installation. The description of setting up and configuration of non-standard communication servers can be found in the documentation enclosed.
- *Tools and environment* dialog contains three folds to set:
  - ID codes of access cards and time plans.
  - ID code of designated card for validation of the reader module HW address setting (the modules supporting this function only) and a number of collected ID cards from the memory in case of controller diagnostics.
  - Colors used in the source text editor.
- *Reports setting* dialog enables to insert a company logo and enter 4 lines of text information into the programming protocol. Insert `bmp` pictures only with the side ratio 2:1 to avoid their distortion during printing.
- Commands *Save scenario* and *Open scenario* from *Settings* menu are used to enable recovering the settings of the position of visualization windows used before. Using the scenarios is conditional on online communication with the controller only.

2.7 Tools

The APS 400 Config software contains a number of tools that make the application design easier. All of these tools require online connection with the controller (more in chapter 3.1: Application structure and properties). The tools can be divided into three categories:

- Tools for *memory data check* of the controller.
- Tools for the *system status visualization*.
- Tools for *setting of the system parameters* (service cards, HW address setting,..).

Controller memory data check

Tables of cardholders' ID, access groups, time zones, holidays, application parameters and controller properties can be downloaded to a PC. Correct data collection is conditional on agreement of the application opened in APS Config software and the application uploaded in the controller.

After the corresponding command is selected the inquiry is sent to the communication server and subsequently filled dialog with data downloaded is displayed. Each of these dialogs contains buttons *OK* and *Cancel*. Pushing any of these buttons closes the dialog. Pressing *OK* button saves the content of the table is into a text file, in which each line corresponds to one line of the table, and the columns are separated by semicolons.

Particular text files are saved in the same folder as the `APSConfig.exe` file and their names are as follows:

- `ReportCards.txt` … for cardholders' ID.
- `ReportPlans.txt` … for time zones.
- `ReportHolidays.txt` … for holidays.
- `ReportParams.txt` … for application parameters.
- `ReportGroups.txt` and `ReportIDS.txt` … for access groups and related authorization flags.
- `ReportProperties.txt` … for parameters of the controller.

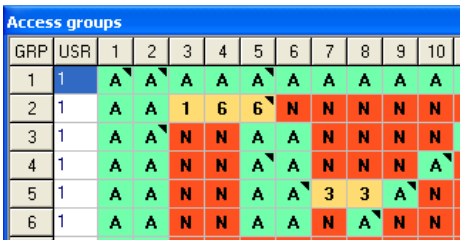


Fig. 2.5: Access groups

Some details related to the access groups download should be mentioned:

The access level of particular access point is defined by a value saved in the intersection of the access group number assigned to the card ID and the particular access point reader HW address. At each intersection the general access ("A" – displayed in the green field), no access ("N" – in the red field) or access according to a time zone (the frame number is displayed – in the yellow field) can be defined, see *fig. 2.5.*

If the access is not forbidden there is a possibility to define so called authorization flag (e.g. it can signify the authorization for IDS control). In case this flag is defined a black triangle is displayed in the right upper corner of the field. In the text file `ReportIDS.txt` these values are saved as "A" – flag is assigned and "N" – flag is not assigned.

System visualization tools

The tools for *system status visualization* allow on-line overview of all inputs, outputs and logical statuses of particular system modules, their remote control, user events activation, and, partially, APS Bus communication control. All visualization windows are "stays on top" and can be opened simultaneously. At the same time the APS Config program can be operated without any restrictions.

The basic visualization tool is the *network status* monitoring (*fig. 2.6*). Every network reader module is represented by one icon showing its communication status. The modules that are not connected to the network are gray, standard communicating modules are green, the ones where the communication was lost and the controller has been trying to restore the communication again are red.

Clicking any module icon by the mouse right button displays a local menu with following commands:

Show details ... displays a window with detailed visualization of the module status.

Connect module ... commands the controller to restore the communication with the module.

Disconnect module ... commands the controller to disconnect the communication with the module.

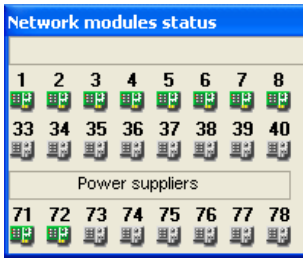


Fig. 2.6: Network status

The detailed *network reader* status monitoring can be opened in two ways. Either from the *Network status* window as described above or using the *Network reader status* command from *Tools* menu. The status window (*fig. 2.7*) contains (from left):

- An icon showing the communication status (it has the same meaning as in the *Network status* window).
- Tamper input icon (yellow = activated).
- Reading device icon (gray = not active, green = valid ID, blue = invalid ID, red = unknown ID).
- Alarm status icons "Forced door" and "Door ajar".
- Both inputs and outputs icons (green = open, red = close).

Fig. 2.7: Network module

Power supplier status monitoring can be opened in the same way as the Network reader status. The status window (fig. 2.8) contains (from left):

- An icon showing the communication status (it has the same meaning as in the *Network status* window).
- Tamper input icon (yellow = activated).
- AC current status icon (green = connected, red = disconnected).
- Battery status icon (green = O.K., red = discharged, grey = disconnected).
- DC output status icon (green = O.K., red = overloaded/short cut, black = disconnected).

Fig. 2.8: PSU status

For a detailed visualization of the controller status there is a *Controller status* window, see *fig. 2.9*. A part of the window is composed from the icons representing inputs and outputs and global logical system statuses. The rest of icons contain buttons whose meanig is obvious from their description.

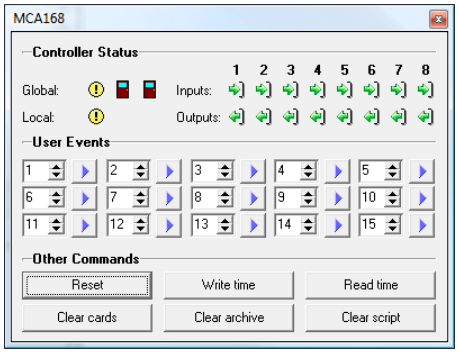


Fig. 2.9: Controller status

The outputs can be controlled from the local menu in all cases, accessible by clicking an appropriate icon which contains the dual command *Connect* and *Disconnect*.

Exceptional standing has status visualization windows of registers and software timers *Show registers* and *Show timers*. It is necessary to compile the relevant application which is downloaded in the controller before to have correct overviews. Remote setting of both parameters is possible.

The last tool of this category is the *Download events archive* command. The assumption of correct overview is to compile the relevant application downloaded in the controller before.

Tools for setting up the system parameters

Upload service setting to the controller makes the system debugging easy. Access levels defined in the Settings menu are uploaded to controller memory – there is no need to install server and administration software.

Set reader HW address (*fig. 2.10*) is used for the reader modules which address can't be set using a keypad, jumpers or DIP switch (e.g. NREM 61).

Enter the desired HW address into particular field and hit the "Set" button. Then take the card set as the "setting one" in the *Settings* menu and enclose it to the appropriate reader module. Address setting is acknowledged by a long beep.

Do not set the same address as already present on the APS Bus!

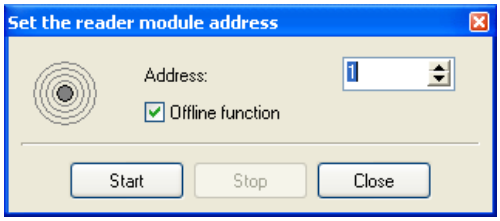


Fig. 2.10: Setting up the network module address

3

Applications in APS Config environment

3.1 Application structure and properties

An application in APS Config environment defines the HW structure of the APS 400 system installation, its properties and script defining its behavior.

The application hardware structure is displayed in the application tree. Every application contains following elements: one master module (MCA 168), up to 64 net reader modules and up to 8 system power suppliers (PSU 71). The particular modules can be inserted into the application tree by choosing the required module from the *Modules* menu or using a local menu accessible by clicking the right mouse button on the application tree. Modules can be removed from the application using the *Delete module* command from the same menu.

Several properties of the application can be defined in the *Application properties* dialog (*fig. 3.1*), which can be displayed using the *Properties* command from *Application* menu. This dialog contains two folds: *Application* for general properties definitions and *Connection* for setting up the way of controller connection to a PC.

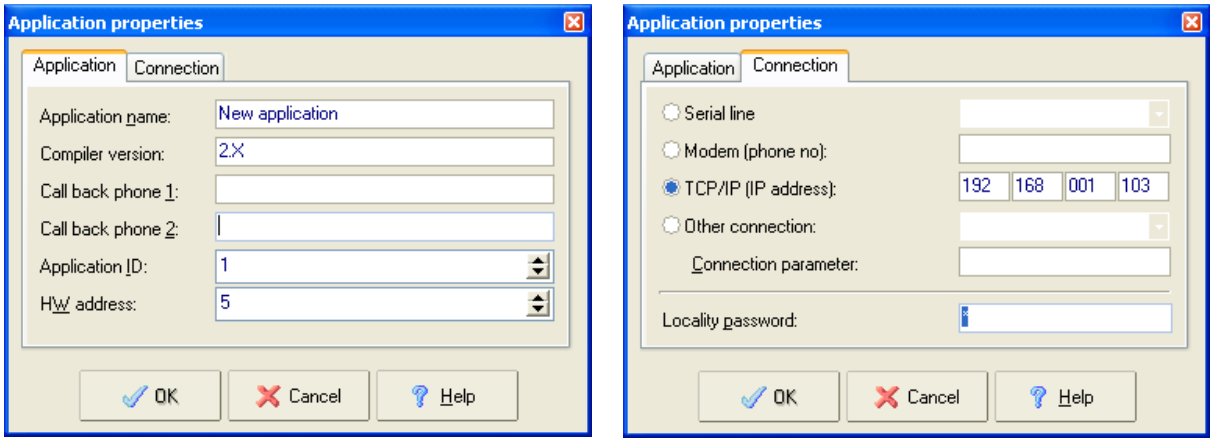


Fig. 3.1: Application properties

General application properties are:

- *Application name* ... any text string, max 30 characters.
- *Compiler version* ... number of the compiler version for which the application is created, this number can't be changed.
- *Call back phone 1* and *Call back phone 2* ... for applications connected via modem, the command for calling the upper system (PC) using one of these two phone numbers can be used in the script.
- *Application ID* ... numerical identifier. It has a special meaning in case of modem connections (call back) only.
- *HW address* … setting up HW address of the controller.

The way of system controller communication has to be defined in the fold of connection properties; connecting a MCA 168 controller is possible only by setting a TCP/IP connection to a communication server APS 400 nServer.NET.

- *Serial line* ... communication via RS 232 interface; connection parameter is the number of the serial port used.
- *Modem* ... communication via modem; connection parameter is the controller phone number.
- *TCP/IP* ... communication via computer network, connection parameter is the IP address of the server.
- *Other connection* ... the communication is established in a different way of the modes of communication mentioned above. The way of setting up the non-standard communication modes is described in the manuals of appropriate communication software.
- *Locality password* ... numerical code from interval <0; 99999999>, its value has to correspond to the value stored in the controller memory. The default value is "0". This value is set in the controller after its restart while the switch DIP no.10 is switched on (se the manual of MCA 168 controller). The password can be changed after the communication is established using the *Change application password* command from the *Application* menu.

3.2 Modules

Every hardware module installed in the system is represented by an object having inputs, outputs and other periphery in the application tree. The general structure of the controller and a network module is evident from the following diagram (*fig. 3.2).*

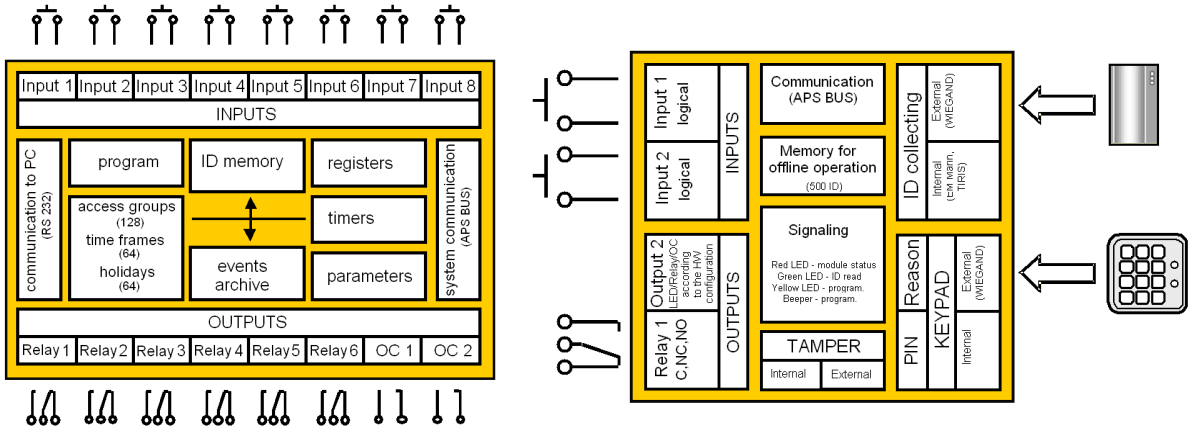


Fig. 3.2: Block diagrams of the controller and a network module

Each module or device has a number of parameters defining its behavior and processing in administration software. They can be divided into three categories:

- *Properties* … general definitions of properties of the modules or devices.
- *Events* … system status change definitions due to the particular status change (e.g. switching on the input).
- *Descriptions* … define a text assigned to every archive mark generated by the module or device if displayed.

All of these parameters can be edited in the *Module inspector* (*fig. 3.3*) located in the left lower part of the main program window.

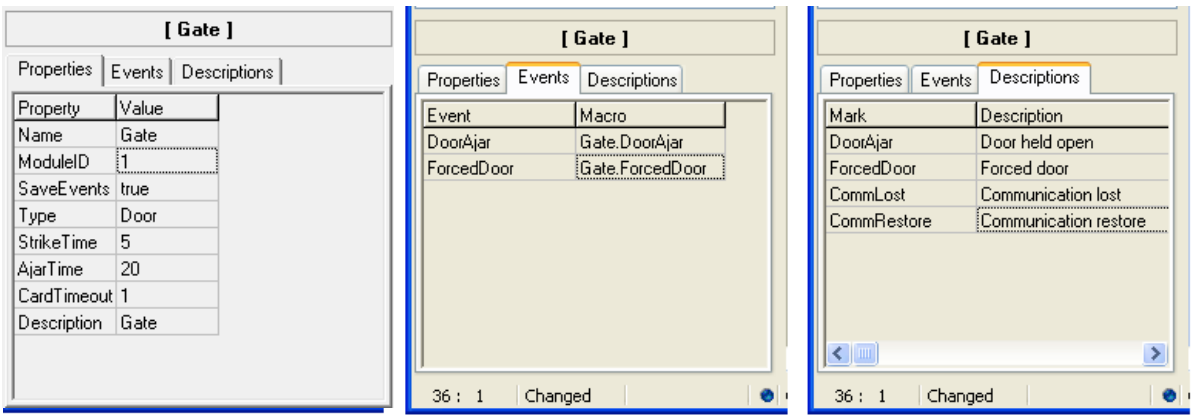


Fig. 3.3: Module inspector

Three properties are obligatory for all modules:

- `Name` ... a unique text identifier of a module in the application. It can contain characters “A” .. “Z”, “a” .. “z”, “0” .. “9”, “_” and “-“. Up to 25 characters can be used for the name. The name can not be identical with any keyword of the macrolanguage or with any other name defined in the application.
- `ModuleID` ... a unique number assigned to a module. It is assigned according to following rules:

  1) Controller ... `Module ID = 0` (always)
  2) Network reader module ... `Module ID = module HW address (1 – 64)`
  3) Power supplier ... `Module ID = module HW address (71 – 78)`

  Similarly as the Name, also the ModuleID must be unique within an application.
- *`Description`* `... it is any textual description of a module. It can be transfered to the administration software.`

Every device has an obligatory name (it is assigned implicitly and can not be changed). Set of data types properties and possibility of their editing can be seen in the table bellow (*fig. 3.4*).

Data type	Value range	Example	Editation
Number	0..255	ModuleID	Enter into the edit line
Logical	True, false	SaveEvents	Enter into the edit line, left mouse button double click
Enumerated – reader module type	General, Door, DoorWithHandle	Type (by the network reader module)	Enter into the edit line, left mouse button double click
Text	Any text string – up to 50 characters	Description	Enter into the edit line

Fig. 3.4: Data types

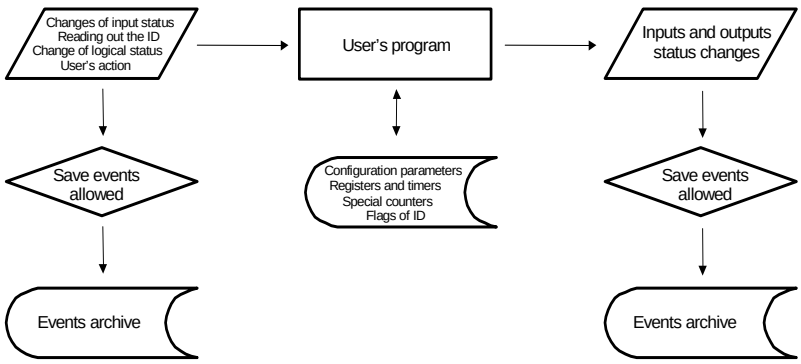
The list of module/device *events* specifies all events that can trigger a macro. When an event occurs an archive mark is recorded into the system events archive (if its recording is allowed in the module properties)

If a *macro* is created for any system event its name is displayed in the corresponding edit line of the *Module inspector*. If you wish to create a macro, double click on the edit line – the program will record the macro name into the *Module inspector* and will declare it simultaneously in the script.

The list of archive marks *descriptions* enables editing of the text descriptions displayed when the system event archive is analyzed. The complete list of system archive marks can be seen in appendix 5.4. “System archive marks overview”.

3.3 Events and macros

The APS 400 program is performed via the controller, which detects the changes of status of its own and of all slave network modules. Based on this status changes the controller generates so called *events*, which can trigger related user programs – macros defining the reaction of the system. The way of generation and processing of events is shown in following diagram, *fig. 3.5.*



Figr. 3.5: The way of events processing

Within a macro processing, especially the `Delay` command, repeated activation of the same macro can appear. The controller has to decide if to interrupt the running macro (and execute it from the beginning) or leave it until finished. The behavior of the controller is determined by the setting up of the 5th configuration DIP switch, see the MCA 168 manual.

Note: Interrupting of macros is recommended in most of applications; in case of doubt ask your technical support.

Most of the events are triggered by a status change of any peripheral device. Events activated by a user's identification number (`Valid`, `Invalid`, `Unknown`) are processed a little more complicated. Thereafter the ID is read by a network reader, the controller decides which event to generate. The progress of this processing can be seen in *fig. 3.6*.

The whole model of access rights is rather complex. It contains counters of present users, programmable flags fixed on particular ID and other mechanisms suitable for various types of applications. The description of this mechanism contains following documents:

- http://www.techfass.cz/files/t_aps_400_opravneni_en.pdf.

Detailed technical description of frequent applications:

- http://www.techfass.cz/files/t_aps_400_ijidelna_en.pdf,
- http://www.techfass.cz/files/t_aps_400_iadministrator_vis_en.pdf.

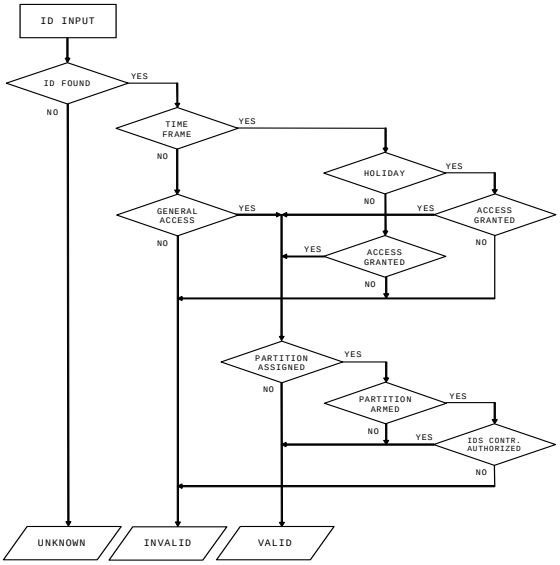


Fig. 3.6: Access rights

3.4 Script

The script contains all macros defining the system behavior, comments, and declarations of registers, timers, user archive marks and parameters.

No upper case and lower case letters are distinguished; e.g. for the module name is valid:

```
Reader1=reader1=READER1
```

Macros

`Macro` has following format:

```
Event ModuleName.EventIdentifier
… Body of macro …
EventEnd
```

The body of macro is composed of a sequence of commands defining the system behavior. The key words `Event` and `EventEnd` are obligatory and define the beginning and the end of the macro. `ModuleName` is the name assigned by a property `Name` in the module inspector. `EventIdentifier` is an implicitly defined name of a touched event and cannot be changed.

Comments

`Comment` is a part of the script beginning with the character ";". All characters behind, up to the end of the line, are then ignored when the script is compiled. Comments serve for explaining more complicated parts of the program code, variables etc. It is recommended to use rich comments; they will help to facilitate understanding the program code when it is customized in the future.

```
Event ModuleName.Event.Identifier
   ; This is a comment
EventEnd
```

Registers

`Register` is numerical variable valid globally in the whole script (in all macros). Any number from the interval <0; 255> can be written in the register and the value can be tested at other place of the script. Up to 250 registers can be used in an application. Register declaration can be located at any place of the script but it is recommended to place all declarations at the beginning of the script.

`Register` declaration has following format:

```
Define register RegisterName
```

Software timers

`Software timer` is numerical variable as well as the `register` but with different meaning. If the timer value is set to 255 the timer switched off. If the value is less then 255 it is decremented every second by 1 until zero. The program can assign a numerical value to the timer (meaningful are values 1-254), which triggers counting-down the time interval corresponding to the assigned value in seconds. It can be tested anywhere in the script while the timing is still running (value > 0) or if it has already stopped (value = 0). If the condition for timer zero value is reached the value is set to 255 (the timer is switched off). Up to 254 software timers can be used in an application.

`Software timer` declaration has a following format:

```
Define timer TimerName
```

User parameters

`User parameter` is valid in the whole script too. It can be used as a parameter of the `Delay` command and it can assign a value to the `register` or the `timer`. Up to 254 users parameters can be used in an application.

`User parameter` declaration has following format:

```
Define parameter ParameterName=Value
```

Using of the parameter is suitable when the same value has to be used in more program places or where the changes of its value from the administration software are required.

User archive marks

`User archive marks` are used for saving the particular program code passing into the events archive. Archive marks are valid in the whole script as well. Up to 254 users marks can be used in an application.

`User archive marks` declaration has following format:

```
Define mark MarkName=Description
```

User archive marks are often used during the program debugging. In “sharp“ applications they can be used for storing the system statuses, which cannot be distinguished from the standard system archive marks.

3.5 Program compilation, upload to the controller memory

After the script is ready the application needs to be compiled to the code understandable for the controller. The compilation can be run using the *Compile* command from the *Application* menu. If there are no syntax errors found in the script the code can be uploaded to the controller (using the *Upload script* command) and the system behavior can be tested. In the opposite case, a list of found errors is displayed in the lower part of the editor. The compiler can check the syntax errors only, i.e. an error message appears if it does not “understand“ some part of the script. The compiler cannot find out any logical errors!

Note 1: The macro is canceled if its body is empty during compilation.

Note 2: When the module is renamed in the Module inspector the change comes out in the script editor automatically.

4

System programming

4.1 Standard system functions

There are two ways of the standard functions programming in APS 400 Config environment:

- "Manually" as described in the following chapters.
- Using the configuration dialogs related to the standard programming of a particular module.

Configuration dialog related to the selected module of application tree can be opened using the *Standard function* command from the *Modules* menu or from a local popup menu of the application tree.

Function of configuration dialogs

All functions performed in APS 400 access control system must be defined by the script or by setting up the values of some properties. The configuration dialogs generate the source texts of standard functions automatically based on entered values.

Common items to configuration dialogs

All configuration dialogs have following common items:

- Text box *Module name (in the script)* for setting up the `Name` property.
- Text box *Module description (in the events archive)* for setting up the `Description` property.
- *Defaults* button - when pressed the default values of the module are set.

Standard controller functions

Global alarm warnings can be assigned to the controller in configuration dialog (*fig. 4.1*). Particular warning is activated through an optional controller output setting (`Relay 1` - `Relay 8`).

The alarm status of the output is configurable. Standard is NC contact. NO contact can be chosen by checking `NO` status.

Meanings of particular alarm statuses and their default setting are listed in following table, see *fig. 4.2*.

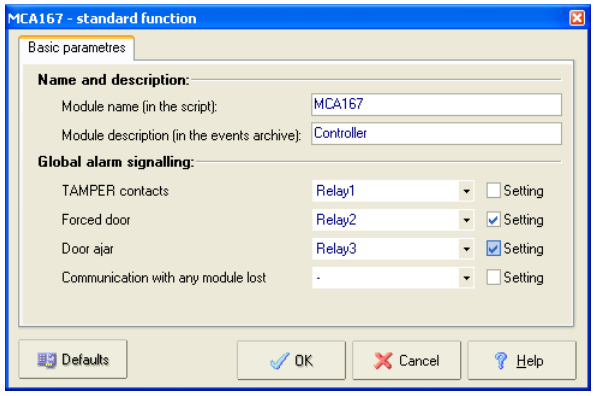


Fig. 4.1: Standard controller functions

Warning	Meaning	Output	Status
Tamper contacts	At least one tamper contact was activated in the system (controller, netwok modules)	Relay 1	Open
Forced door	At least one network module has Forced door status	Relay 2	Open
Door ajar	At least one network module has Door ajar status	Relay 3	Open
Communication lost…	The communication with at least one network module is lost	Relay 4	Open

Fig. 4.2: Controller alarm warnings

Network module standard functions

The way of door control can be defined in the `Basic parameters` fold; single or double sided control with any other reader module (choose the one in the list, see *fig.4.3a*). The text box for Module type definitions is the last item of this fold. The meanings of particular types are described bellow.

It is strongly recommended to control the door from the more secure reader module (inside the building) in case of double sided control. The lower secure reader module then ensures reading out the media ID or a PIN code only and its output and inputs are free for other utilization.

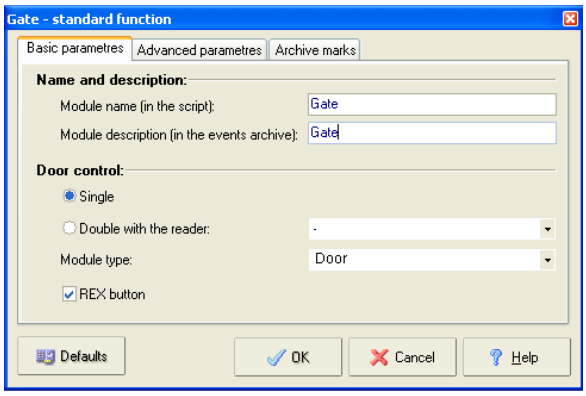


Fig. 4.3a: Network module standard functions

Advanced parameters

The `Advanced parameters` fold (*fig.4.3b*) contains a text box for specifications of Strike time, Door ajar time (after expiration of this parameter a Door ajar warning is triggered – if the Module type is set to `Door` or `DoorWithHandle`) and the Last card timeout parameter.

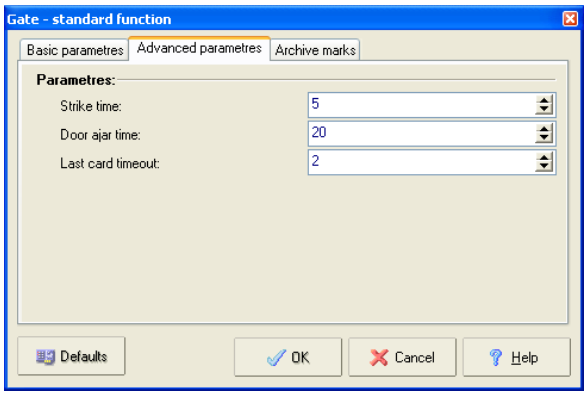


Fig. 4.3b: Network module standard functions

Archive marks

The “Archive marks“ fold contains a text box for text definitions of selected events.

Defaults

When this button is pushed the default settings of the module are recovered, i.e. single side door control, module type `General`, Strike time 5s, Door ajar 20, Last card timeout 1s, REX button checked. The descriptions of chosen events are set according to the *fig. 4.3c*.

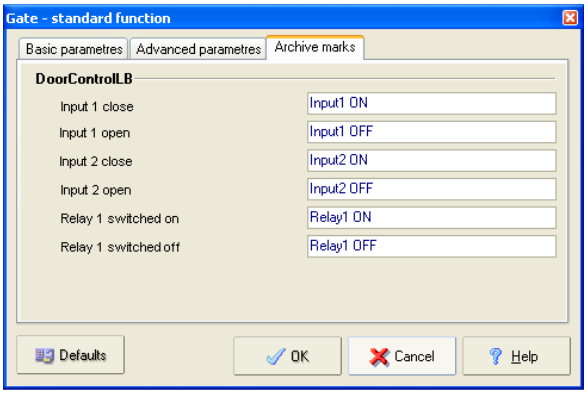


Fig. 4.3c: Network module standard functions

4.2 Macros

The body of macro is composed of a sequence of commands that should be carried out by the system after the macro is run. Particular commands are carried out sequentially in the order they were written in the body of macro. Only one command with an eventual comment can be placed on each line. The programmer can work with all modules and devices defined in the whole application in the body of the macro. Every device in the application has a unique identifier in following format:

`ModuleName.DeviceName`

Where `ModuleName` is the name assigned to the module in the property `Name` of module inspector, and `DeviceName` is the name of device generated by the environment. The names are unique within the module. Devices forming a part of the module generating a macro can be identified only by their names in the script (the compiler assumes automatically that the device is a part of the module that generates the event).

The macro-language statements can be divided into three groups: assignments, conditions and system commands.

Assignments

Assignments enable the program to set the values of system outputs, registers and timers. The operator has following format:

`DeviceName=Value`

The type and range of used values depend on the type of status to be set. The `On` and `Off` values are to be used for the statuses of logical devices (inputs, outputs...), values within the interval `<0; 255>` for numerical values of registers and timers. Using of the operator is explained in the example 4.1. Two macros are defined in the example – for network reader module input (`Input1`) turned on and off. The macros copy the status of this input to the controller's (`MasterModule`) output 1 (`Relay1`).

`Example 4.1: Macros for copying input status`

```
Event Reader1.Input1On       ; macro for closing the Input1
  MasterModule.Relay1=On     ; turns the Relay1 on
EventEnd

Event Reader1.Input1Off      ; macro for opening the Input1
  MasterModule.Relay1=Off    ; turns the Relay1 off
EventEnd
```

Conditions

Simple logical expressions can be interpreted in the macro-language and the program can be branched on its base. Following construction serves for interpreting the logical expressions:

```
If LogicalExpression then
... body of the condition executed if the expression is true ...
Else
... body of the condition executed if the expression is false ...
EndIf
```

A shorter notation can be used if both conditions are not necessary:

```
If LogicalExpression then
... body of the condition ...
EndIf
```

The logical expression can test system input and output statuses, numerical values of registers, statuses of software timers and logical system variables. An overview of supported conditions is listed in appendix 5.3: "The most important constructions of APS 400 macro-language".

Only simple logical expressions can be interpreted in the conditions (only one logical status, one numerical value or one system status). The only logical operator used in logical expressions is NOT (negation). However, the conditions can be submerged (at maximum 20 conditions) and the absence of `AND` and `OR` operators can be substituted for an appropriate combination of logical conditions. *Example 4.2* demonstrates substitution for the `AND` operator, *example 4.3* for the `OR` operator, and the testing of system variables demonstrates *example 4.4*.

Example 4.2: Substitution for the AND operator (submerged conditions)

```
Event Reader1.Input1On
  If Reader2.Input1=On then
    If Reader3.Input1=On then
       Relay1=On
    EndIf
  EndIf
EventEnd
```

Example 4.3: Substitution for the OR operator (conditions with the same body)

```
Event Reader1.Input1On
  If Reader2.Input1=On then
    Relay1=On
  EndIf
  If Reader3.Input1=On then
    Relay1=On
  EndIf
EventEnd
```

Example 4.4: Testing of system logical variables.

```
Event MasterModule.Input1On
  If AllCountersEmpty then   ; nobody present
    Signal(Reader1,2)        ; two short beeps of the Reader1 module
  Else                       ; somebody present
    Signal(Reader1,4)        ; one long beep of the Reader1 module
  EndIf
EventEnd
```

Triggering of system functions

The last group of commands ensures triggering of various functions implemented in the system itself. The overview of these functions is listed in appendix 5.3: “The most important constructions of APS 400 macro-language”. Using of the `Signal` command is shown in *example 4.4*.

Registers

Any value from <0; 255> interval can be assigned to the register, a constant can be added or subtracted, and the value can be tested in the condition.

Basic commands for managing the registers are:

```
RegisterName = Value                          ; assign value to the register
RegisterName = RegisterName + constant        ; add constant to the register
RegisterName = RegisterName – constant        ; subtract constant of the register
If RegisterName = Value then                  ; test the register value
If RegisterName > Value then                  ; test the register value
```

Logical operator `NOT` can be used in both conditions.

The register used as a branch condition of system status (standard operation / building evacuation) is shown in *example 4.5*.

Connections and function of the system are as follows:

A switch (e.g. an output from fire alarm system) determining the "Evacuation" status (switched ON = Evacuation, switched OFF = Standard operation) is connected to the input 1 of the controller. There are two net reader modules installed in the system. Their outputs (e.g. controlling the door locks) should be permanently turned on in case of the evacuation. The evacuation status signalization has to be indicated every 10 s by the reader modules.

Example 5.6: Use of registers

```
Define register Evacuation   ; register definition

Event MasterModule.Init      ; event from the script initialization
  If Input1=Off then         ; testing input 1
    Evacuation = 0           ; setting status to Standard operation
  EndIf
  If Input1=On then          ; testing input 1
    Evacuation = 1           ; setting status to Evacuation
  EndIf
EventEnd

Event MasterModule.Timer10sec      ; event from hardware timer 10s
  If Evacuation = 1 then     ; testing the Evacuation status
    Reader1.Beep = On        ; switching on the beep of reader 1
    Reader2.Beep = On        ; switching on the beep of reader 2
    Delay(2)                 ; wait for 2s
    Reader1.Beep = Off       ; switching off the beep of reader 1
    Reader2.Beep = Off       ; switching off the beep of reader 2
  EndIf
EventEnd

Event MasterModule.Input1On  ; event from setting the controller input 1
  Evacuation     = 1         ; setting the Evacuation status
  Reader1.Relay1 = On        ; switching on the reader 1 relay
  Reader2.Relay1 = On        ; switching on the reader 2 relay
EventEnd

Event MasterModule.Input1Off ; event from resetting the controller input 1
  Evacuation     = 0         ; setting the Normal activity status
  Reader1.Relay1 = Off       ; switching off the reader 1 relay
  Reader2.Relay1 = Off       ; switching off the reader 2 relay
EventEnd

Event Reader1.Valid          ; valid card on reader 1
  If Evacuation = 0 then     ; testing Standard operation status
    Door()                   ; calling standard door function
  EndIf
EventEnd

Event Reader2.Valid          ; valid card on reader 2
  If Evacuation = 0 then     ; testing Standard operation status
    Door()                   ; calling standard door function
  EndIf
EventEnd
```

Software timers

Setting and testing the software timers is analogous to the registers except adding or subtracting constants (not supported). *Example 4.6* is extended *example 4.5*. The timer is used for acoustic signalization control during the evacuation.

Expansions of connection are as follows:

The siren (has to be activated for the first 2 minutes of evacuation) is controlled by an output 1 of the controller. The optical signalization (indicates the whole duration of evacuation period) is switched on and off by output 2 of the controller.

Example 4.6: Use of software timers

```
Define register Evacuation   ; register definition
Define timer Siren           ; timer definition

Event MasterModule.Init      ; event from the script initialization
  If Input1=Off then         ; testing input 1
    Evacuation = 0           ; setting the status to the Standard operation
    Siren      = 255         ; resetting the Siren timer
    Relay1     = Off         ; switching off the optical indicator
    Relay2     = Off         ; switching off the siren
  EndIf

  If Input1=On then          ; testing the controller input 1
    Evacuation = 1           ; setting the Evacuation status
    Siren     = 120          ; running the siren timer (120s)
    Relay1     = On          ; switching on the optical indicator
    Relay2     = On          ; switching on the siren
  EndIf
EventEnd

Event MasterModule.Timer1sec ; event from the hardware timer 1s
  If Siren = 0 then          ; siren time expired?
    Relay2 = Off             ; switching off the siren
  EndIf
EventEnd

Event MasterModule.Timer10sec      ; event from the hardware timer 10s
  If Evacuation = 1 then     ; testing the evacuation status
    Reader1.Beep = On        ; switching on the beep on reader 1
    Reader2.Beep = On        ; switching on the beep on reader 2
    Delay(2)                 ; waiting for 2s
    Reader1.Beep = Off       ; switching off the beep on reader 1
    Reader2.Beep = Off       ; switching off the beep on reader 2
  EndIf
EventEnd

Event MasterModule.Input1On  ; event from setting the controller input 1
  Evacuation     = 1         ; setting the Evacuation status
  Reader1.Relay1 = On        ; switching on the reader 1 relay
  Reader2.Relay1 = On        ; switching on the reader 2 relay
  Siren          = 120       ; starting the Siren timer (120s)
  Relay1         = On        ; switching on the optical indicator
  Relay2         = On        ; switching on the siren
EventEnd

Event MasterModule.Input1Off ; event from resetting the controller input 1
  Evacuation     = 0         ; setting the Normal activity status
  Reader1.Relay1 = Off       ; switching off the reader 1 relay
  Reader2.Relay1 = Off       ; switching off the reader 2 relay
  Siren          = 255       ; resetting the Siren timer
  Relay1         = Off       ; switching off the optical indicator
  Relay2         = Off       ; switching off the siren
EventEnd
```

```
Event Reader1.Valid          ; valid card on reader 1
  If Evacuation = 0 then     ; testing the Standard operation status
    Door()                   ; calling standard door function
  EndIf
EventEnd

Event Reader2.Valid          ; valid card on reader 2
  If Evacuation = 0 then     ; testing the Standard operation status
    Door()                   ; calling standard door function
  EndIf
EventEnd
```

User parameters

User parameters can be used in following constructions:

`Delay(ParameterName)`

`RegisterName= ParameterName`

`RegisterName = ParameterName`

`Delay(ParameterName)`

Typical use of user parameters is entering the time strike (after pushing a button or reading a valid card), see *example 4.7*.

Connections and function are as follows:

The door lock is connected to output 1 (`Relay1`) of the network reader module Reader1. The input 1 of Reader1 module scans the door contact (turned on if the door is closed). The door lock is released after the input 2 of Reader1 module is switched on, and locked after the door is open or a time interval (defined by `StrikeTime` user parameter) is expired.

`Example 4.7: Use of user parameters`

```
Define parameter StrikeTime = 10    ; user parameter definition

Event Reader1.Input2On       ; event from pushing the button (Reader1,input2)
  If Input1=On then                ; testing the door status
    Relay1   = On                  ; releasing the strike
    Delay(StrikeTime)              ; waiting for the StrikeTime seconds
    Relay1   = Off                 ; locking the strike
  EndIf
EventEnd

Event Reader1.Input1Off            ; event from disconnecting the door contact
  Relay1 = Off                     ; locking the strike
EventEnd
```

Archive marks

Archive marks can be stored into the archive using following command:

`Mark(MarkName)`

Example 4.8 is extended *example 4.7*. The system functionality is extended by the archive mark storing in case of the door stayed not open within the strike time interval.

Example 4.8: Use of archive marks

```
Define parameter StrikeTime=10      ; definition of the user parameter
Define register DoorOpened          ; definition of the register
Define mark DoorNotOpened=Door was not opened    ; definition of the
                                                 ; archive mark

Event MasterModule.Init             ; event from the script initialization
  DoorOpened=0                      ; setting the register value
EventEnd

Event Reader1.Input2On              ; event from setting the reader 1 input 2
  If Input1=On then                 ; testing the Door status
    DoorOpened = 0                  ; setting the register
    Relay1   = On                   ; releasing the strike
    Delay(StrikeTime)               ; waiting for the defined time [s]
    Relay1   = Off                  ; locking the strike
    If DoorOpened=0 then            ; testing the register value
      Mark(DoorNotOpened)           ; putting the archive mark into the archive
    EndIf
  EndIf
EventEnd

Event Reader1.Input1Off             ; event from disconnecting the door contact
  Relay1 = Off                      ; locking the strike
  DoorOpened = 1                    ; setting the register value
EventEnd
```

5

Appendix

5.1 APS 400 system modules

MCA 168 controller

MCA 168 controller– properties		
Name	**Type**	**Description**
Name	text	Defines a unique text identifier of the module. The module is represented by this identifier in the script. Characters ('0..9', 'a'..'z', 'A'..'Z', '-', '_' can be used for defining the Name.
ModuleID	numeric	Defines the module numerical identifier for internal system use. It must be always equal to "0" for the controller.
SaveEvents	logical	Defines whether the archive marks generated by the module should be saved in the events history archive.
Description	text	Any text representing module in the administration software APS 400.
MCA 168 controller– events		
Name	**Conditions for event triggering**	
GlobalDoorAjar	DoorAjar status transition (any door stayed open for a longer period then allowed).	
GlobalForcedDoor	ForcedDoor status transition (any door forced in the system).	
GlobalDoorAjarOK	GlobalDoorAjarOK status transition (the last DoorAjar status shut down).	
GlobalForcedDoorOK	GlobalForcedDoorOK status transition (the last ForcedDoor status shut down).	
GlobalTamper	Any module gets tampered.	
GlobalTamperOK	The last tamper canceled in the system.	
GlobalCommLost	Loss of any system module communication.	
GlobalCommRestore	Communication with all system net modules gets restored.	
Init	User program (script) is initialized.	
Reset	After the controller reset.	
MCA 168 controller– descriptions		
Name	**Saving marks conditions**	
GlobalDoorAjar	GlobalDoorAjar event generation.	
GlobalForcedDoor	GlobalForcedDoor event generation.	
GlobalDoorDoorAjarOK	GlobalDoorAjarOK event generation.	
GlobalForcedDoorOK	GlobalForcedDoorOK event generation.	
GlobalTamper	GlobalTamper event generation.	
GlobalTamperOK	GlobalTamperOK event generation.	
GlobalCommLost	GlobalCommLost event generation.	
GlobalCommRestore	GlobalCommRestore event generation.	
Init	Init event generation.	
Reset	Reset event generation.	
ModemConnection	At the remote connection to the system by a modem.	
LineBusy	At not successful upload via a modem thanks to line busy.	
MCA 168 controller– list of devices		
UserEvents, Timer1Sec, Timer10Sec, PSU, Beep, Tamper, Input1..Input8, Relay1.. Relay8.		

Fig. 5.1: MCA 168 controller

Network reader module

Network reader module – properties		
Name	**Type**	**Description**
Name	Text	Defines a unique text identifier of the module. The module is represented by this identifier in the script. Characters ('0..9', 'a'..'z', 'A'..'Z', '-', '_' can be used for defining the Name.
ModuleID	Number	Defines the module numerical identifier for internal system use. It is equal to the module hardware address setting <1;64>.
SaveEvents	Logical	Defines whether the archive marks generated by the module should be saved into the events history archive.
Type	Enumerated – reader module type	Defines the reader behaviour. Detailed description of the module types contains the table on *fig.5.3*.
StrikeTime	Number	Maximum door strike release time [s] (used by the Door function).
AjarTime	Number	Restriction of door open period, when elapsed, the DoorAjar status is set.
CardTimeout	Number	Period during which the last read access card is ignored by this module.
Description	Text	Any text representing module in the administration software APS 400.
Network reader module – events		
Name	**Event triggering conditions**	
DoorAjar	DoorAjar status transition.	
ForcedDoor	ForcedDoor status transition.	
Network reader module – archive marks		
Name	**Saving marks conditions**	
DoorAjar	DoorAjar event generation.	
ForcedDoor	ForcedDoor event generation.	
CommLost	Loss of the module communication.	
CommRestore	Restoring of the module communication.	
Network reader module – list of devices		
Tamper, Beep, CardReader, Input1, Input2, Relay1, LED3 (Relay2)		

Fig. 5.2: Network reader module

`Type` property

The `Type` property of the network reader needs a detailed explanation of its operation in `Standard connection` using the `Door` command.

The standard connection of the reader module supposes the door contact connected to the first module input (door closed = door contact closed), the door lock connected to the first relay (NO, NC), and optional handle contact (handle down = handle contact closed) connected to the second module input. Wiring diagrams for various types of door locks and openers are on http://www.techfass.cz/aplikace_en.html.

The basic solutions of settings the module type at calling the `Door` function are shown in *examples 5.1*, *5.2* and *5.3*. It is recommended to test the function on an operating system during studying the examples and also test the system behavior at changes of the `StrikeTime`, `AjarTime` and `CardTimeout` properties.

Type	Meaning and usage
General	General module not programmed in a standard way.
Door	Module programmed as standard with connected door control. Reports forced door and door ajar status automatically.
DoorWithHandle	Module programmed as standard controllig door with handle from inside. Handle contact is connected to the 2nd module input. Reports forced door and door ajar status automatically with respect to the handle status.

Fig. 5.3: Meaning and usage of Type property

`Door` function

Triggering of the `Door` function actuates simultaneously the output relay and beeper untill the door is open (door contact is disconnected) or the preset interval `StrikeTime` is elapsed. In case of non-authorized entry (i.e.

disconnecting the door contact without preceding controller command or handle pulling) the `ForcedDoor` event is triggered. If the door stays open until the pre-defined parameter `AjarTime` expires, the `DoorAjar` event is activated.

`ForcedDoor` event

The `ForcedDoor` event can be activating by the modules using `type` property set to `Door` or `DoorWithHandle`. In both cases triggering of the event is conditional on disconnecting the 1st module input at non-actuated output relay status. Moreover the 2nd module input cannot be connected (handle down) in the `DoorWithHandle` module type, it is considered as authorized door open.

`DoorAjar` event

The `DoorAjar` event is activating by the modules using `type` property set to `Door` or `DoorWithHandle` when the door stays open (the 1st module input is open) after the pre-defined parameter `AjarTime` expires. If the module occurs in `ForcedDoor` status the `DoorAjar` is not reported.

Examples:

Example 5.1: Single side door control by Reader1 module, type DoorWithHandle, handle connected to the 2nd module input.

```
Event Reader1.Valid
  Door()       ; Door open based on valid user's identification.
EventEnd
```

Example 5.2: Single side door control by Reader1 module, type Door, REX button connected to the 2nd module input.

```
Event Reader1.Valid
  Door()       ; Door open based on valid user's identification.
EventEnd

Event Reader1.Input2On
  Door()       ; Door open based on pushing the REX button.
EventEnd
```

Example 5.3: Double side door control by the twin of reader modules, type of Reader1 – Door, type of Reader2 – General, door connected to the Reader1 module.

```
Event Reader1.Valid
  Door()
EventEnd

Event Reader2.Valid
  Door(Reader1)
EventEnd
```

Power supply unit

System PSU – properties		
Name	**Type**	**Description**
Name	Text	Defines a unique text identifier of the module. The module is represented by this identifier in the script. Characters ('0..9', 'a'..'z', 'A'..'Z', '-', '_' can be used for defining the Name.
ModuleID	Number	Defines the module numerical identifier for internal system use. It is equal to the module hardware address setting <71;78>.
Description	Text	Any text representing module in the administration software APS 400.
System PSU – events		
Name	**Event triggering conditions**	
StatusChanged	At any flags or tamper change.	
System PSU – archive marks		
Name	**Saving marks conditions**	
CommLost	Loss of the module communication.	
CommRestored	Restoring of the module communication.	
DCOverload	Output overloaded.	
DCOK	Output O.K.	
BatteryLow	Battery discharged	
BatteryOK	Battery O.K.	
ACLost	Loss of mains.	
ACRestorted	Mains restored.	
BatteryConnected	Battery connected.	
BatteryDisconected	Battery disconnected.	
DCOn	Output voltage turned on.	
DCOff	Output voltage turned off.	
System PSU – list of devices		
Flags, Tamper		

Fig.5.4: System power supplier PSU 71

5.2 APS 400 system devices

Every APS 400 device type properties, events and archive marks are detailed described in this chapter. If more than one device occur in a module the devices are differed by a serial number (e.g. `Input1`, `Input2`, …) generally marked (`Input`*X*, `Output`*X*, …) where *X* is the device serial number.

Beep device

`Beep` device represents the module beeper, which status can be controlled by assignment command (values `On`, `Off`) and it can be used as `Invert` function argument.

Beep device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.

Fig. 5.10: Beep device

CardReader device

`CardReader` device represents reading part of network modules; its status cannot be software controlled. After the card ID is read its validity (*fig. 3.6*) is analyzed and related event `Valid`, `Invalid` or `Unknown` is triggered. Saving the archive marks of this device cannot be forbidden.

CardReader device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
CardReader device – events		
Name	**Event triggering conditions**	
Valid	Reading valid card ID.	
Invalid	Reading invalid card ID.	
Unknown	Reading unknown card ID.	
Invalid PIN	Invalid PIN code 5 times over entered.	
CardReader device – archive marks		
Name	**Saving marks conditions**	
Valid	At Valid event.	
Invalid	At Invalid event.	
Unknown	At Unknown event.	

Fig. 5.11: CardReader device

Flags device (PSU 71)

Power supply unit `Flags` device represents particular status information of PSU 71 power supplier. The statuses can be tested in condition commands, see *example 5.6*.

Flags device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.

Fig.5.14: PSU71 Flags device

Example 5.6: Testing the PSU 71 power supplier flags

```
Event PSU71.StatusChanged       ; status change of PSU (HW address=71)
  If not Flags(fACPresent) then ; test of AC present flag
    MasterModule.Relay1=On      ; activation of the controller relay1
                                ; e.g. for signaling (optic or acoustic)

  EndIf
EventEnd
```

List of all sense flags; see the table in *fig. 5.15*.

Flag	Meaning
fOverload	Output overloaded.
fBatteryPresent	Battery connected.
fLowBattery	Discharged battery.
fBoxTamper	PSU box tampered.
fDCOn	Output voltage turned on.
fACPresent	Mains on.

Fig.5.15: List of PSU 71 flags

InputX device

`InputX` represent the X^{th} – module input. Its status cannot be software controlled but can be tested.

Input*X* – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
SaveEvents	Logical	Defines whether the archive marks generated by the device should be saved into the events history archive.
Input*X* – events		
Name	**Event triggering conditions**	
Input*X*On	Input*X* close.	
Input*X*Off	Input*X* open.	
Input*X* – archive marks		
Name	**Saving marks conditions**	
Input*X*On	At the Input*X*On event.	
Input*X*Off	At the Input*X*Off event.	

Fig. 5.17: Inputs

PSU (Power Supply Unit) device

`PSU` device represents an extra controller input determined for standard (not APS bus connectible) power supplier output connection. Its status can be tested by condition command.

PSU device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
SaveEvents	Logical	Defines whether the archive marks generated by the device should be saved into the events history archive.
PSU device – events		
Name	**Event triggering conditions**	
PSUOn	Output voltage O.K.	
PSUOff	Output voltage K.O.	
PSU device – archive marks		
Name	**Saving marks conditions**	
PSUOn	At PSUOn event.	
PSUOff	At PSUOff event.	

Fig. 5.18: PSU device

RelayX, AuxOutput

`RelayX` or `AuxOutput` device represent the hardware module outputs. The output status can be software controlled using assignment command (values `On`, `Off` can be assigned). The device can be used also as `Invert` function argument and can be tested.

Relay*X*, AuxOutput device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
SaveEvents	Logical	Defines whether the archive marks generated by the device should be saved into the events history archive.
Relay*X*, AuxOutput device – archive marks		
Name	**Saving marks conditions**	
Relay*X*On, AuxOutputOn	Relay*X*, AuxOutput close.	
Relay*X*Off, AuxOutputOff	Relay*X*, AuxOutput open.	

Fig. 5.20: Relays

Tamper device

`Tamper` device represents a module protective contact. It triggers particular event and can be tested in conditional commands. Any module tamper status change influences the system `GlobalTamper` status.

Tamper device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
Tamper device – events		
Name	**Event triggering conditions**	
Tamper	Switching tamper contact off.	
Tamper device – archive marks		
Name	**Saving marks conditions**	
Tamper	Tamper event triggering.	
Tamper OK	Switching tamper contact on.	

Fig. 5.23: Tamper device

Timer1sec, Timer10sec device

`Timer1sec` or `Timer10sec` represents controller hardware timers. These timers generate event each 1s, or each 10s and macros can respond to these events. It is necessary to keep in view the macros are called periodically and their code has to be executed before next triggering the same macro. Therefore it is not available to use commands `Delay` and `Door` or to control the IDS panel.

Timer1sec, Timer10sec device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
Timer1sec, Timer10sec device – events		
Name	**Event triggering conditions**	
Timer1sec Timer10sec	Events Timer1sec, or Timer10sec are triggered by the system each 1s, or each 10s.	

Fig. 5.24: Timer1sec, Timer10sec device

UserEvents device

`UserEvents` device represents 8 events, which can be generated from a host PC. User events can also be triggered from other macros by `Execute(X)` command, where `X` is the user event number. `Execute` command cannot be used in user events macros.

UserEvents device – properties		
Name	**Type**	**Description**
Name	Text	Defines the unique text identifier within the parent module. It is assigned automatically by the environment and cannot be changed.
SaveEvents	Logical	Defines whether the archive marks generated by the device should be saved into the events history archive.
UserEvents device – events		
Name	**Event triggering conditions**	
UserEvent1 … UserEvent8	Sending a command to trigger the UserEvent*X* from host PC or calling the Execute(*X*) function from any macro in the script.	
UserEvents device – archive marks		
Name	**Saving marks conditions**	
UserEvent1 … UserEvent8	At UserEvent*X* event.	

Fig. 5.26: UserEvents device

5.3 The most important macro-language constructions overview.

With respect to continuous increasing of APS 400 system functionality we recommend to update this list occasionally on http://www.techfass.cz/aps_400_macrolanguage_en.html. Also you can find there the links, examples and other related information sources. The list bellow corresponds to the controller firmware version 202 and the compiler version 2.15 from the 6th of July 2005.

Break

Terminates the macro executing; can be used in all events.

ClearAllCounters

Deletes all *PRESENT* flags of access cards and clears access groups and users' codes counters of present cardholders at the same time; can be used in all events.

ClearGroupCounter(x)

Clears the access group counter of present cardholders defined by parameter *x*, deletes *PRESENT* flags of all cards assigned to defined access group and decrements adequate the user's code counter; can be used in all events.

ClearPresent

Deletes *PRESENT* flag of just read access card and clears the relevant access group and user's code counters of present cardholders at the same time; can be used in *Valid* event.

ClearUserCounter(x)

Clears the user's code counter of present cardholders defined by parameter *x*, deletes *PRESENT* flags of all cards assigned to access groups with *x* user's code set and decrements adequate the access groups counters; can be used in *Valid* event.

Define mark x=Description

Archive mark definition; it is entered beside the macros to the script.

Define parameter x=Name

Parameter definition; it is entered beside the macros to the script.

Define register x=Name

Numerical register definition; it is entered beside the macros to the script.

Define timer x=Name

Software timer definition; it is entered beside the macros to the script.

Delay(x)

Stops the macro executing for a pre-defined time in seconds defined by *x* parameter; can be used in all events except *Timer1sec* and/or *Timer10sec*.

Dial(x,y)

Establishes modem connection to a host PC. *X* parameter defines index of dialed phone number (1 or 2); *y* parameter defines the user archive mark describing the reason of the connection; can be used in all events.

Display(x) / Display(ReaderN,X)

Displays a symbol on current module, or *ReaderN* module (if the hardware contains the display). Meaning of *x* parameter is as follows:

0 - 9 ... digit 0 - 9
10 ... hyphen
11 ... three level lines
12 ... E character

13 ... decimal point
14 ... all segments on (inc. decimal point)
15 ... all segments off

Door() / Door (ReaderN)

Executes standard *Door* function; can be used in all events.

EraseCard

Erases the current card from controller memory table; can be used in *Valid* event.

Execute(x)

Activates the user event defined by *x* parameter; can be used in all events except the *UserEventN* events (recursion not allowed).

If AllCountersEmpty then

Evaluates the zero status of all access groups and user's code counters of present cardholders; can be used in all events.

If Authorized then

Evaluates extra authorization flag of cardholder's access levels; can be used in *Valid* event.

If Buffer(x) then

Checks filling in the controller events archive buffer if filled up more then defined by *x* parameter (in percentage) the result is true; can be used in all events.

If Byte1=x then

Compares entered value *x* to the value stored in *Byte1* column of just read card ID. If the values match the result is true; can be used in *Valid* and/or *Invalid* events.

If Byte2=x then

Compares entered value *x* to the value stored in *Byte2* column of just read card ID. If the values match the result is true; can be used in *Valid* and/or *Invalid* event.

If Group(x) then

Evaluates assignment of current card ID to the access group no. *x*; can be used in *Valid* event.

If GroupEmpty(x) then

Evaluates the zero status of access group no. *x* counter of present cardholders; can be used in all events.

If KeyCode(x) then

Compares the pressed key code on the reader module with reason keypad to entered code *x*; can be used in *Valid* event.

If ModuleLost(x) then

Checks the communication status of the network module with hardware address *x* the result is true if the communication is not established; can be used in all events.

If Present then

Evaluates the *PRESENT* flag of current card ID; can be used in *Valid* event.

If TimePlan(x) then

Evaluates the current validity of the time zone no. *x*; can be used in all events.

If User(x) then

Evaluates if the *x* number is equal to access group user's code assigned to current card; can be used in *Valid* event.

If UserEmpty(x) then

Evaluates the zero status of present cardholders' counter of the access groups with user's code no. *x* assigned; can be used in all events.

If x then ... endif / if x then ... else ... endif / if not x then ... endif / if not x then ... else ... endif

Evaluates the *x* logical condition. The result can be negated using the *not* operator. Nesting depth is 20 levels; can be used in all events.

Invert (x)

Inverts the status of logical output defined by its name *x*; can be used in all events.

Mark(x)

Stores the archive mark defined by *x* name to controller events archive buffer; can be used in all events.

SetByte1(x)

Sets *x* value to *Byte1* column of current card to the card ID table of the controller memory; can be used in *Valid* and/or *Invalid* event.

SetByte2(x)

Sets *x* value to *Byte2* column of current card to the card ID table of the controller memory; can be used in *Valid* and/or *Invalid* event.

SetModePIN(x)

Switches over the defined reader module *x* to identification mode "card+PIN". Reader modules with PIN pad only can be switched the other ones ignore this setting; can be used in all events.

SetModePINFree(x)

Switches over the defined reader module *x* to identification mode "card only". Reader modules with PIN pad only can be switched the other ones ignore this setting; can be used in all events.

SetPresent

Sets the *PRESENT* flag to the current card and increments relevant access groups and user's code counters of present cardholders; can be used in *Valid* event.

Signal(x) / Signal(ReaderN,X)

Triggers the acoustic signal on current reader module, or *ReaderN* module. Meaning of *x* parameter is as follows:

1 ... one short beep
2 ... two short beeps
3 ... three short beeps
4 ... one long beep
5 ... five short beeps
6 ... erase card memory in the reader module

5.4 System archive marks summary

Event ID	Description	Event ID	Description	Event ID	Description
Controller archive marks (ModuleID=0)					
4	Input1 On	40	Reserved	132	Relay 3 On
5	Input1 Off	41	Reserved	133	Relay 3 Off
6	Input2 On	42	Reserved	134	Relay 4 On
7	Input2 Off	43	Reserved	135	Relay 4 Off
8	Tamper	45	Global door ajar	136	Relay 5 On
16	Input3 On	46	Global forced door	137	Relay 5 Off
17	Input3 Off	47	Global door ajar OK	138	Relay 6 On
18	Input4 On	48	Timer 1 sec	139	Relay 6 Off
19	Input4 Off	49	Timer 10 sec	140	Relay 7 On
20	Input5 On	50	Init	141	Relay 7 Off
21	Input5 Off	51	Global tamper	142	Relay 8 On
22	Input6 On	52	Global tamper ok	143	Relay 8 Off
23	Input6 Off	53	Global comm lost	144	Reserved
24	Input7 On	54	Global comm restore	145	Reserved
25	Input7 Off	55	PSU On	146	Reserved
26	Input8 On	56	PSU Off	147	Reserved
27	Input8 Off	60	User event 1	148	Reserved
28	Reserved	61	User event 2	149	Reserved
29	Reserved	62	User event 3	150	Reserved
30	Reserved	63	User event 4	151	Reserved
31	Reserved	64	User event 5	152	Reserved
32	Reserved	65	User event 6	153	Reserved
33	Reserved	66	User event 7	154	Reserved
34	Reserved	67	User event 8	155	Reserved
35	Reserved	68	Global forced door ok	156	Reserved
36	Reserved	128	Relay 1 On	157	Reserved
37	Reserved	129	Relay 1 Off	158	Reserved
38	Reserved	130	Relay 2 On	159	Reserved
39	Reserved	131	Relay 2 Off	160	LineBusy
PC programming archive marks (ModuleID=253)					
0	Connected	2	Parametres upload	-	-
1	Script upload	3	Access data upload	-	-
Network module archive marks (ModuleID=HW address)					
1	Valid	7	Input2 Off	66	Comm. lost
2	Invalid	8	Tamper	67	Comm. restored
3	Unknown	9	DoorAjar	161	AuxOutput On
4	Input1 On	10	Forced Door	162	AuxOutput Off
5	Input1 Off	64	Relay1 On	249	Tamper OK
6	Input2 On	65	Relay1 Off	250	Invalid PIN

Fig. 5.28: System archive marks – the 1st part

PSU archive marks (ModuleID=HW address)					
8	Tamper	72	Battery low	77	Battery disconnected
66	Comm. lost	73	Battery OK	78	DC On
67	Comm. restored	74	AC Lost	79	DC Off
70	DC overload	75	AC Restored	249	Tamper OK
71	DC OK	76	Battery connected		

Fig. 5.28: System archive marks – the 2nd part

5.5 Complete APS 400 application example

Required system functions

Simplified diagram of standard access control application is shown in *figure 5.29.* Application requirements are as follows:

- The main entrance control should be both, from the readers or from handsets.
- Current cardholders' presence registration is required.
- Door alarm and tamper alarm statuses should give a signal by audible and visual indicator. Audible signalization has to be active for a pre-set time after the alarm statuses are triggered or stopped using a button or host PC before the pre-set time is elapsed.
- Door ajar status should give an optic signal only.

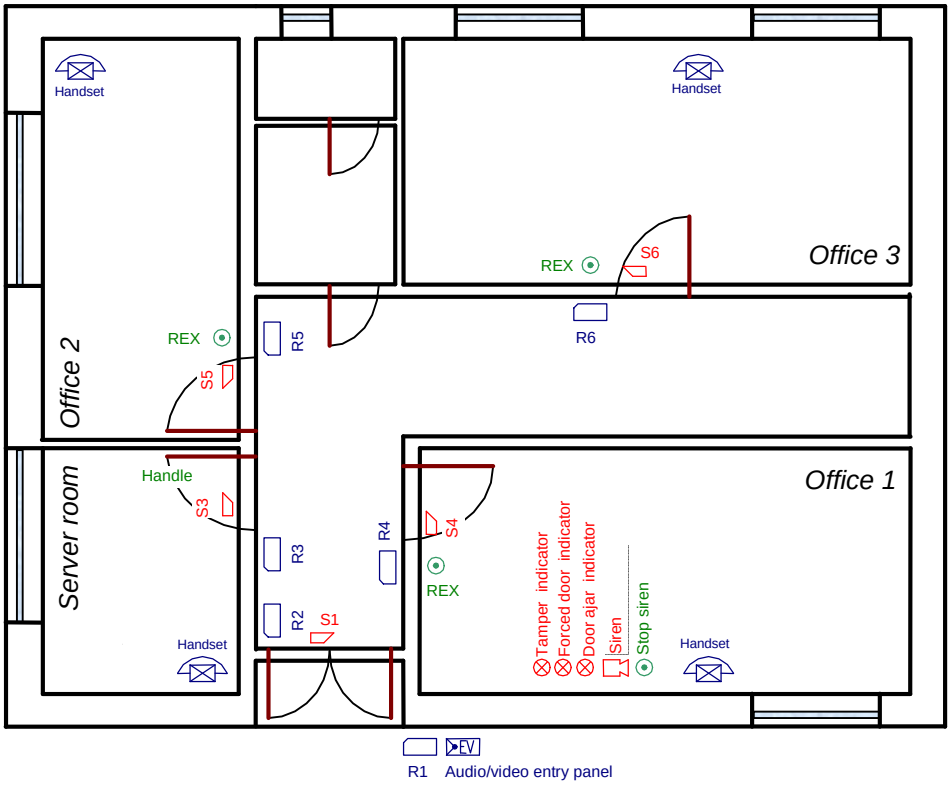


Fig. 5.29: Simplified application diagram

Hardware solution

The main entrance double side reader control (R1, R2) is necessary because of the current cardholders' presence registration is required. E.g. motor lock can be used for the main entrance. Door opening by the handset is used for visitors only.

Server room door can be equipped e.g. by electric lock with handle contact and one reader module R3.

The offices doors are fitted with door openers controlled by readers from outside and by request to exit (REX) buttons from inside.

It is recommended to use locks and openers equipped by door contacts.

Controller wiring – inputs:

- Output relay of audio/video entry system … input 1
- Reset button for acoustic signalization … input 2

Controller wiring – outputs:

- Siren … relay 1
- "DoorAjar" status indicator … relay 3
- "ForcedDoor" status indicator … relay 4
- "TAMPER" status indicator … relay 5

Inputs and outputs of reader modules are connected in standard way, see:

http://www.techfass.cz/aplikace_en.html

Note 1: It is necessary to ensure the emergency exit from inside areas according to the regulations of particular country independent of access control system operation!

Software solution

Insert MCA 168 controller (name `MasterModule`) and six network reader modules (name `R1…R6`) into APS Config application. Define modules descriptions (property `Description`) as follows:

MCA168 controller… Master module
R1 … Main entrance
R2 … Main exit
R3 … Office 1
R4 … Server
R5 … Office 2
R6 … Office 3

Both (`R1, R2`) main entrance reader modules have different type set. `R2` module (type `Door` is set) controls the door directly and generates the alarm signals, `R1` module (type `General` is set) controls the door through the `R2` module. Exit and entry function of `R1, R2` reader modules can be set in communication server software, see the manual: http://www.techfass.cz/files/m_aps_400_iserver_en.pdf

`R4` reader module is set (type `DoorWithHandle`). Other reader modules are set (type `Door`).

List of the script:

```
Define timer TMR1                ; software timer for siren timing
Define parameter PAR1=30         ; forced door status time interval
                                 ; signaling
Define parameter PAR2=30         ; TAMPER status time interval
                                 ; signaling

Event Controller.GlobalDoorAjar
  Relay3=ON                      ; door ajar status optical signaling
EventEnd

Event Controller.GlobalForcedDoor
  Relay4=ON                      ; forced door status optical signaling
  TMR1=PAR1                      ; starts audible signaling time interval
EventEnd

Event Controller.GlobalTamper
  Relay5=ON                      ; TAMPER status visual signaling
  TMR1=PAR2                      ; starts audible signaling time interval
EventEnd

Event Controller.GlobalTamperOK
  TMR1=0                         ; stops acoustic signaling
  Relay5=OFF                     ; stops optical signaling
EventEnd
```

```
Event Controller.GlobalDoorOK
  Relay3=OFF                    ; stops optical signaling
  Relay4=OFF                    ; stops optical signaling
  TMR1=0                        ; stops audible signaling
EventEnd

Event Controller.UserEvent1     ; stops acoustic signaling by a PC
  TMR1=0
EventEnd

Event Controller.Timer1sec
  If TMR1>0 then
    Invert(Relay1)              ; intermittent acoustic signal
  Endif
  If TMR1=0 then                ; stops acoustic signaling by the timer
    Relay1=OFF
  Endif
EventEnd

Event Controller.Input2On      ; stops acoustic signaling by a button
  TMR1=0
EventEnd

Event Controller.Input1On      ; door function from entry system handset
  Door(R2)
EventEnd

Event R1.Valid
  Door(R2)
EventEnd

Event R2.Valid
  Door()
EventEnd

Event R3.Valid
  Door()
EventEnd

Event R3.Input2On
  Door()
EventEnd

Event R4.Valid
  Door()
EventEnd

Event R5.Valid
  Door()
EventEnd

Event R5.Input2On
  Door()
EventEnd

Event R6.Valid
  Door()
EventEnd

Event R6.Input2On
  Door()
EventEnd
```

End of list.

5.6 Programming protocol

Every report page contains identical header and the bottom with its serial number. It is possible to modify the left part of a report header by setting some parameters, see chapter 2.6. “Program set up”.

The report is divided into three parts:

- *General parameters* … contains general information about the application, e.g. filename, date and time, connection type etc.
- *Modules list* … contains the list of all modules (including subordinate devices) used in the application. List of all properties (normal font) and of archive marks description (italic font) is included.
- *Script* … contains complete list of the script.