Symbol naming conventions

Contents

1. Summary

1.1 Introduction

Symbol names are names which can be assigned to elements in the PCD (inputs, outputs, flags, registers, COBs etc.) but also to values.
If only the media type is specified but no address is given, PG5 will automatically allocate an address from the range of dynamic addresses (if possible).

By the use of symbols the readability of programs is increased and more flexible program structures may be achieved.

PG5 itself as well as FBox- and FB-libraries do also work with symbols. In order to avoid symbol collisions, a programmer is urged to follow some rules that guarantee that the user-defined symbols and the PG5 internal and FBox specific symbols do not conflict (e.g. do have the same name but not the same values).
These rules are described in this document. Additionally, this document does also point out the reserved words and symbol names that mustn't be used by users or FBox developers.

1.2 Symbols overview

There are several ways to declare several types of symbols. Below a short summary of the different symbol types and their declarators:

- Local symbols (declared by EQU)
- Global symbols (declared by PUBL and referenced by EXTN)
- Local macro symbols (declared by LEQU)
- Global macro symbols (declared by GEQU)

The local- and global symbols may be defined in the PG5 symbol editor (SSE) or directly in an IL file using declarators. If defined in the Symbol Editor, PG5 will handle the declarations itself for that the user doesn't have to care about them.

The scope of local symbols is the program file in which they are declared.

Local macro symbols can only be declared by the instructions LEQU and GEQU (or in the macro definition as formal parameters) and are visible in the macro in which they are declared.

Please refer to the chapter "The scope of symbols" for further information about the scope of the different symbol types.

2. Rules for symbol names

The following rules apply on symbol names, macro names (also FBox macro names!) group names and global symbols.

2.1 General rules

Any symbol (also group names, macro names and FBox names) used by PG5:

- Cannot begin with a digit (0-9)
- Can contain any alphanumeric character (ANSI character set)
- Can contain one or more underscore characters '_', providing the rules below are obeyed
- Can be up to 80 characters long (including the group names)
- Is not case-sensitive (Note: this may not be true for certain accented character, depending on the installed character set)

It is recommended not to use accented characters because they are depending on the installed character set of the PC (and therefore may cause problems when the project is compiled on another PC).

When working with the FBox builder:
Make sure you adapt the symbol- and macro names after importing a SAIA-internal FBox!

2.1.1 Reserved Words

The following words are reserved and therefore cannot be used as symbol names:

- Assembler declarators, e.g. PUBL, EXTN, EQU, DEF, LEQU, LDEF, MACRO, ENDM, EXITM etc.
- Medium control codes and data types (I, O, F, R, C, T, K, M, COB, FB, TEXT, X, SEMA, DB).
- MOV instruction special codes (N, Q, B, W, L, D).
- Condition codes (H, L, P, N, Z, E).
- Instruction mnemonics.
- Pre-defined symbols (see chapter "Predefined Symbols Created by build utility (S-Asm)").
- Symbols which begin with an underscore, these are reserved for use in SAIA libraries.

2.2 Rules for user defined symbols

These rules have to be applied by users when creating PG5 projects. The rules are also valid for FBox names, macro names and group names.

- The name mustn't begin with an underscore
- The name cannot contain a double underscore anywhere (to avoid collisions with third-party libraries)
- The symbol name must have at least two characters

2.3 Rules for third party libraries

Since the FBox macro names must be unique for every FBox and in order to avoid symbol collisions between user symbols and FBox symbols, the following rule is to be applied for all macro names, group names and (local) and global symbols used by an FBox.

- A user specific prefix (provided by the TCS from SAIA Burgess Controls, pcdsupport@saia-burgess.com), followed by a double underscore must be used, e.g. TCS__????.

Example for an FBox macro name:

```
TCS__FBox1  ;; FBox macro name for FBox1 from TCS
```

The formal macro parameters and the local macro symbols (defined by LEQU or GEQU) are not concerned because their scope is only the macro and therefore they do not conflict with the symbols defined outside the macro. (The symbols outside the macro are covered by the formal macro parameters if the names are the same.)

Please also refer to the chapter "The scope of symbols" and make sure that you only use EQU and PUBL declarators inside your FBox macro if appropriate.

2.4 Rules for SAIA libraries

(Macro names, group names and global symbols)

This chapter does only concern SAIA internal developers and is placed here for completeness of the rules.

- Must have one or two underscores at the beginning.
- Cannot contain a double underscore in the symbol or at the end (single underscore may be used).
- Each library should use a constant prefix (e.g. 3 characters) to simplify collision avoidance.
- Refer to document POPG4008 (Symbol Naming Conventions).doc that contains a table with the registered prefixes.

3. System symbols and pre-defined symbols

3.1 System symbols (A- and S-symbols)

System symbols are symbols declared by the PG5 Project Manager (SPM), editors/compilers or libraries distributed by SAIA Burgess Controls AG.
The user (or FBox developers) can reference these symbols but he is not allowed to modify them or to define new ones!
In SPM, the system symbols are listed after each build in the PG5's Symbol Editor.

System symbols are used for special function library symbol names.

System symbols are declared as follows:

- the first group level has a single character
  - 'S' for system symbols
  - 'A' for optional symbols in application libraries
- the second group level is a registered name for each package or the registered macro prefix of third-party developers
- subsequent group levels and symbol names can be anything

3.1.1 S-symbols

S-symbols are always defined without any special decision of the user.

The developer of the library is responsible to avoid collision of the S-symbols. Any symbol generated without a user defined group, must be declared as S-Symbols (E.g. in OBL file).

Examples:

```
S.HMI.Contrast    ;;Register to set the display contrast in HMI application
S.IPD.Library     ;;Library for the IP-Data mode.
```

3.1.2 A-symbols

A-symbols are optionally generated on decision of the user (e.g. if the user specifies an FBox name). A part of the symbol (after the registered group) is defined by the user. E.g. in Fupla the user defines an FBox name that will be used to build the A-symbols specific to the FBox.

The user is responsible to avoid collision by duplicated FBox names.

Example:

If the user specifies an FBox name (in this example: PID_1) for an Heavac PID controller, the following A-Symbol will be generated automatically (together with a bunch of symbols for this FBox). This mechanism is very helpful e.g. for the HMI objects of Heavac FBoxes.

```
A.HVC.PID_1.YOut  ;;Register that contains the output value of a PID
                  ;;controller of the Heavac library
```

In case a third-party FBox developer wants A-Symbol for one of his FBoxes, he has to use his macro prefix at the second level of the group name. Below an example of the code in the FBox macro:

```
$IFNB <name>                        ; The A-symbol is only generated if
                                    ; An FBox name is provided
$GROUP A.TCS__.name                 ; second level is the macro prefix
FlagBase          EQU   stc_ADJ     ; Comment shown in the Symbol Editor
                  PUBL  FlagBase    ; Make the symbol public
$ENDGROUP
$ENDIF
```

The A-symbol that will be generated if the FBox has given a name will be A.TCS__.FBoxName.FlagBase. TCS__ in this example is the reserved prefix of the FBox developer.
Note that the parameter “Accept name” (General tab in the workspace in the FBox Builder) must be set to “User defined” and not to “Default”.

For third-party libraries it is not allowed to use anything else except of the macro prefix provided by Saia as second level of the A-symbol.

3.2 Predefined Symbols Created by build utility (S-Asm)

These symbols are generated internally by the assembler or linker when a build is done. They can be referenced but mustn’t be modified by the user program.

`__CSTART__`	Assigned by the linker to the line number of the first line of code in the module.
`BLOCKNUM`	The number of the current block.
`BLOCKTYP`	The type of the current block as an ASCII character.
`__SAIASystemBuildTime__`	32-bit value of PC’s internal clock when the build was started.
`__SASMVERS__`	Version number of the assembler which created the OBL or OBJ file (may be different for every module).
`__PGVERS__`	The PG programming package version number, 5=PG5.
`__PGBUILD__`	The PG programming package build number (software version), e.g. 12003 = version 1.2.003.

4. The scope of symbols

This chapter explains where symbols are “visible”. In the short descriptions below the expression “module” is often mentioned. A “module” is the program file that contains the declarator.

- If the declarator is part of an IL source file, the “module” stands for this IL file.
- If the declarator is part of a macro, the “module” is the file in which the macro is expanded.

This explanation is important in relation to the symbols used in FBox macros. While there is no problem to declare all symbols as local symbols in IL files, it will cause problems (in fact multi defined symbols) if the macro contains EQU declarators and the FBox is places several times in the same Fupla file.

4.1.1 Local Symbols

Local symbols are defined by the declarator EQU (EQUate) and their scope is the module in which the equate statement is written in. Once a symbol is declared by the EQU declarator, it cannot be modified any more.
The second possibility to declare a symbol whose scope is the module is the declarator DEF (DEFine). A symbol defined by DEF may be re-defined later in the same module but cannot be made public.

4.1.2 Global Symbols

When a symbol is declared public (by the PUBL statement), its value is global. The symbol can be referenced from any module as an external using the EXTN declaration.
Before a symbol can be made public, it must be declared (with EQU); defined symbols (using DEF) cannot be made public.

4.1.3 Local macro symbols

As for the formal macro parameters, the scope of symbols declared by LEQU or LDEF is only inside the macro itself. LEQU may only be used inside in a macro.

This allows symbols to be defined within macros which do not produce "multi-defined symbol" errors if the macro is called more than once in the same module (e.g. the same FBox is placed several times in a Fupla file).

4.1.4 Global macro symbols

Symbols declared with LEQU or LDEF cannot be accessed directly by any nested macros, for this you must use GEQU or GDEF.

A symbol declared with GEQU is visible in the macro in which it is declared and in all macros that are called from this macro.

5. Sources

This document is an extract from the following documentations:

- Symbol Naming Conventions for the SAIA PG5, Doc # PO-EPG4-008 Rev.25
- SAIA Instruction list online help (installed together with PG5)