- calvOS Project -

calvOS System Development

General System Requirements

Intended Audience:	Developers
Author(s):	Carlos Calvillo
File Name:	calvOS System Requirements.odt
Last save on:	05/01/09 - 02:30:33 PM
Total No. of Pages:	12
Document Maturity:	Draft

Revision History

Version	Date/ Maturity	Author(s)	Revision	Version Comment
0.1	07/05/09 Draft	Carlos Calvillo		Initial Draft for System Architecture.

Table of Contents

1 General Description

The aim of this document is to present the Higher-Level System Requirements for calvOS project. This requirements are as general as possible.

2 Terms Used

Terms used in this document.

Term	Description
SW	Software
HW	Hardware
OS	Operating System
MCU	Micro-controller Unit
I/O	Input/Output
GUI	Graphic User Interface
db	Database
Config	Configuration
DEC	Decimal

3 Development Tools (SYS-T)

3.1 Tools general requirements

Following basic requirements have been established for the tools used in the development of this project:

Req. Id.	Requirement
SYS-T_01	All the tools used for the development shall be Free and/or Open Source. This includes the tools used for documentation.
SYS-T_02	All the tools used for the development shall be able to be installed and run properly in at least Linux and Windows platforms.
SYS-T_03	Facts to be considered when choosing a tool for development are: - Provided support by development team or users by internet. - Size of the installed package. - Balance between complexity/required features.

In next sections we will provide a list of some tools that have been chosen so far. Some of them are mentioned to be mandatory this mean that these tools shall always be used for the specified

purpose. Some other tools are mentioned as not mandatory, in these cases the tools are just some suggestions but any other tool can be used for the specified purpose always that requirements in this section are met.

3.2 GUI Layer Tools.

Not defined yet.

3.3 Configuration Database Layer Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?
XML editor plugin for Eclipse	XML files handling.	Easy to install Eclipse plug-in.		No
XSD validation	Create and validate XML files based on XSD files.			No

3.4 Dynamic Data Layer Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?
XSL editor plugin for Eclipse	XSL files handling.	Easy to install Eclipse plug-in.		No
Xalan	XSL files processing.	XSL file processor used by command line.		Yes

3.5 Static Data Layer Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?
Eclipse IDE	Coding.	Very powerful and supported IDE.		No

3.6 Application Layer Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?
Eclipse IDE	Coding.	Very powerful and supported IDE.		No

3.7 Build Layer Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?

Perl	Perl code execution.			Yes

3.8 Other Tools

Tool Name	Used for..	Tool brief description	Tool's webpage	Mandatory?
OpenOffice	Documentation	Powerful Office application set.		Yes
Splint	Static code check (quality assurance)			No
Doxygen	Code automatic-documentation			No

4 Naming Conventions (SYS-N)

4.1 Folder's structure

TBD

4.2 Documentation

TBD

4.3 Input Files

Input files like source code, xml, xsd, data, etc..

4.4 Output Files

Output files like: executable files.

5 Coding Rules (SYS-C)

5.1 Coding Standards

TBD

5.1.1 C language

MISRA compliant (Items that doesn't affect code performance). Check if MISRA rules can be freely access or are proprietary.

5.1.1.1 Modules Naming Conventions

Each C module (typically compound by a source .c and header .h files) must follow the next name structure:

os_mainFunc_extraInfo.extension

The prefix "os_" is mandatory for any module of the calvOS system including that ones that doesn't belong directly to the operating system like the platform features.

The field "mainFunc" is mandatory and is a short string denoting the main functionality of the module. This must have maximum 8 characters. This can have mixed lowercase and uppercase characters, recommendation is to use as most as possible just lowercase characters.

The field "extraInfo" is optional and can be any string that provides some additional information that the one dennoted into the "mainFunc" field. This can have mixed lowercase and uppercase characters, recommendation is to use as most as possible just lowercase characters. Suggestion is to omit this filed for the "main" files that implement the functionality.

The field "extension" is mandatory and is the extension of the file.

Example 1: Let's imagine the following set of files that cover the scheduler feature:

File Name	Imaginary description
os_sched.c	Main Source code of the calvOS scheduler.
os_sched.h	Main Header file of the calvOS scheduler.
os_sched_cfg.h	File that contains some parameters that configures the scheduler.
os_sched_hal.c	Source file containing some implementation of the scheduler hardware abstraction layer.

Example 2: Let's imagine the following set of files that cover the IO feature:

File Name	Imaginary description
os_iomng.c	File implementing the main functionality of the IO manager.
os_iomng.h	Header File exporting the main functionality of the IO manager.
os_iomng_dig.c	File implementing some functions for handling of digital IOs.
os_iomng_pwm.c	File implementing some functions for handling of PWM IOs.

5.1.1.2 Variables Type definition

5.1.1.2.1 Simple Types

Used Name	ANSI C equivalent	Bit size	Values Range (DEC)
OSUINT8	unsigned char	8	0 … 255

OSUINT16	unsigned short int	16	0 … 65535
OSUINT32	unsigned long	32	0 … 4294967295
OSINT8	signed char	8	
OSINT16	signed short	16	
OSINT32	signed long	32	

5.1.1.2.2 Compound Types

Used Name	ANSI C equivalent	Bit size	Description
OSFLAG8		8	Union to allow bit by bit access or as a whole byte. Used to define 8 flags.
OSFLAG16		16	Union to allow bit by bit access or as a whole byte. Used to define 16 flags.
OSFLAG32		32	Union to allow bit by bit access or as a whole byte. Used to define 32 flags.
OSCOUNT8		8	Variable to be used as a 8-bit counter. Depends on compiler it can be declared as volatile.
OSCOUNT16		16	Variable to be used as a 16-bit counter. Depends on compiler it can be declared as volatile.
OSCOUNT32		32	Variable to be used as a 32-bit counter. Depends on compiler it can be declared as volatile.

5.1.1.2.3 Symbolic Constants

Constants defined with the #define pre-compiler directive must be written in UPPERCASE. Constants defined with the const attribute must follow the corresponding naming rules showed in chapter 5.1.1.3 - Variables Naming Convention.

Example:

```
#define SOME_SYMBOLIC_CONSTANT
```

5.1.1.3 Variables Naming Convention

Just variable's prefixes are required to follow certain naming rules. In general a variable would be as follows:

prefix_variableName

Prefix must reflects the scope and the type of the variable as showed in table ASDASD. The variableName can be any string C-compliant that denotes the usage of the variable. Recommendation is to use words separated by the 1st letter in Uppercase as in variableName.

Variable Main Type	Compound Type	Variable Scope	Prefix	Example
OSUINT8 / OSFLAG8		Global	ru8	ru8_variableName
		Local	lu8	lu8_variableName
		Constant	cu8	lu8_constantName
	array	Global	rau8	rau8_variableName
		Local	lau8	lau8_variableName
	pointer	Global	rpu8	rau8_variableName
		Local	lpu8	lau8_variableName
OSUINT16		Global	ru16	ru8_variableName
		Local	lu16	lu8_variableName
	array	Global	rau16	rau8_variableName
		Local	lau16	lau8_variableName
	pointer	Global	rpu16	rau8_variableName
		Local	lpu16	lau8_variableName
OSUINT32		Global	ru32	ru8_variableName
		Local	lu32	lu8_variableName
	array	Global	rau32	rau8_variableName
		Local	lau32	lau8_variableName
	pointer	Global	rpu32	rau8_variableName
		Local	lpu16	lau8_variableName
OSINT8		Global	rs8	ru8_variableName
		Local	ls8	lu8_variableName
	array	Global	ras8	rau8_variableName
		Local	las8	lau8_variableName
	pointer	Global	rps8	rau8_variableName
		Local	lps8	lau8_variableName
OSINT16		Global	rs16	ru8_variableName
		Local	ls16	lu8_variableName
	array	Global	ras16	rau8_variableName
		Local	las16	lau8_variableName

	pointer	Global	rps16	rau8_variableName
		Local	lps16	lau8_variableName
OSINT32		Global	rs32	ru8_variableName
		Local	ls32	lu8_variableName
	array	Global	ras32	rau8_variableName
		Local	las32	lau8_variableName
	pointer	Global	rps32	rau8_variableName
		Local	lps16	lau8_variableName
1-bit length		Global	rbi	rbi_variableName
		Local	lbi	lu8_variableName
X-bit length		Global	rbiX X = [2,3,...]	rbi2_twoBitsVariable
		Local	lbiX X = [2,3,...]	lbi3_ThreeBitsVariable
Unions		Global	ru	ru_someUnionVariable;
		Local	lu	lu_someLocalUnionVariable;
Structures		Global	rs	rs_globalStructureVariable;
		Local	ls	lu_someLocalUnionVariable;

5.1.1.4 Functions Naming Convention

The structure of a function must be:

filename_functionName

The "filename" prefix is mandatory and must match with the "mainFunc" field of the name of the file where the function is implemented (see Chapter 5.1.1.1 - Modules Naming Conventions for details about the "mainFunc" field). The "functionName" part can be any string C-compliant that denotes the main functionality of the function, suggestions for this is to use words separated by their 1st letter in UPPERCASE as in functionName.

5.1.1.5 Coding Style

In this section we will provide some set of coding style rules, our purpose is to present at the end standardized code style independent on who is making the code.

The start-of-block "{" and end-of-block "}" characters must be placed alone in a line. See the following examples:

Example	Code	Example	Code

"if-else" structure	`if(some-logic)`
	`{`
	`  statement1;`
	`}`
	`else`
	`{`
	`  statement2;`
	`}`
		"for" structure.	`for(;;)`
			`{`
			`  statement1;`
			`}`

Function	`void module_funcName(void)`
	`{`
	`  statement1;`
	`}`
		"Switch" structure	`switch(varName)`
			`{`
			`  case 0:`
			`  statement1;`
			`  break;`
			`  default:`
			`  statement2;`
			`}`

5.1.1.6 Code Comments

Code must be as most commented as possible, this in order to make the code understandable easily by either end-users and developers.

Code must follow Doxygen comments. **(analyze if it is a good idea)**

5.1.1.7 Code Quality

Asd

5.1.2 Language used in GUI

5.1.2.1 Variables Naming Convention

TBD

5.1.2.2 Functions Naming Convention

TBD

5.1.3 XML files

TBD

5.1.4 XSL files

TBD

5.1.5 Build Layer language

TBD

6 Files Format (SYS-F)

TBD

7 General Development Requirements (SYS-D)

TBD