Tool kit

Datasheet & Project evaluation

sales@straton-plc.com

CP_R2_019 F

straton



STRATON AUTOMATION, All Rights Reserved

The information contained in this document is the property of STRATON AUTOMATION. The distribution and/or reproduction of all or part of this document in any form whatsoever is authorized only with the written authorization of STRATON AUTOMATION. The technical data are used only for the description of the product and do not constitute a guarantee of quality in the legal sense of the term. We reserve the right to make technical changes.

Content

1. OVERVIEW 4

2. PRODUCTS 4

2.1. straton Runtime Tool Kit (Ref: VMTK) 4
2.2. straton Integration training (Ref: VMTRAIN) 5
2.3. Requirements and setup 5
2.4. straton IDE (Ref: IDExxx) 5
2.5. straton Runtime license for embedded system (Ref: T5RT) 6
2.6. straton annual subscription (Ref: ASUVMTK) 6

3. ARCHITECTURE OF THE STRATON T5 RUNTIME SOFTWARE 7

3.1. Definitions (1) 7
3.2. System dependent files and add-on features (2) 7
3.3. Standard T5 runtime files (3) 8
3.4. The application (4) 8
3.5. The straton Development Kit Wizard 9
3.6. Setting up your "C" project 13
3.7. Implementation strategy 14

4. ANNEX 1 15

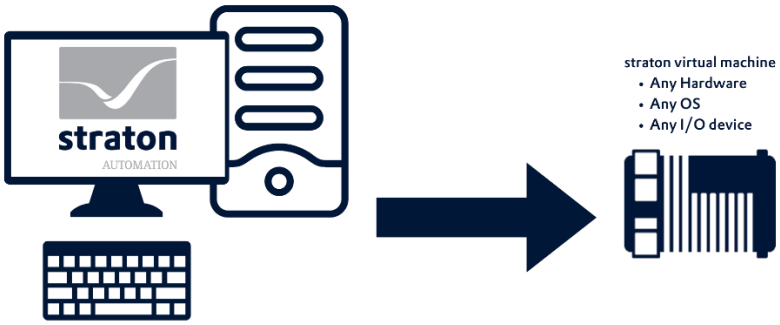
1. Overview

This document gives an overview of products proposed by STRATON AUTOMATION, allowing to integrate the Runtime on an electronic platform (using the Development Kit), customize the integration and program the applications.

2. Products

2.1. straton Runtime Tool Kit (Ref: VMTK)

The tool dedicated to the porting of straton is called the VMTK (Virtual Machine Tool Kit). It includes the tools to port the straton runtime on any platforms and add some function blocks and IOs.



- Contains all tools and information to port the straton runtime on any platform/operating system.
- Full Developer's Guide including guidelines and complete reference of all Runtime APIs.
- Includes complete Runtime source code and samples.
- The Runtime Wizard provides graphical tools for building your project, set up and automatically generates all ""C"" functions.
- Full support of MODBUS TCP and MODBUS RTU.
- Includes CAN protocol.
- User's guide for the creation of 'C' function block in the straton IDE.
- Complete documentation of the integration and interface description.
- Sample of 'C' function blocks for straton
- Including five time-limited licenses (for 12 months) for the Integrated Development Environment - IEC 61131-3, with Unlimited tags.

2.2. straton Integration training (Ref: VMTRAIN)

The integration package of straton is a training for the straton runtime porting on any platforms, explanations on how to add some function block and how to access to local IOs.

- Duration: 2 days.
- Location: STRATON AUTOMATION France or remote web sessions (For other locations: there is an extra cost for travel expenses)

2.3. Requirements and setup

The table to be fulfilled in Annex 1 is a non-exclusive list of the required information/materials necessary to create a straton Runtime.

2.4. straton IDE (Ref: IDExxx)

The straton Integrated Development Environment (also simply called the Editor) is the configuration and programming tool to develop application to be executed by the straton Runtime.

Integrated Development Environment features:

- Powerful text and graphic editors for IEC 61131-3 languages: SFC, FBD, LD, ST, and IL.
- Fast editing with keyboard.
- Powerful drag and drop features.
- Fast compiler with support of User Defined Function Blocks.
- Complete simulator and graphic debugger.
- HTML documents generator.
- Full online control and monitoring, including an ActiveX runtime.
- Fieldbus configuration tool (MODBUS, Profinet, CAN, IEC61850…).
- Licensing hardware key in option.

Three versions are available:

Description	Ref
▸ Limited to 256 I/Os	IDE256
▸ Limited to 512 I/Os	IDE512
▸ Unlimited	IDEUN

Note: I/Os or tags are any straton variables connected to a profile, an I/O board or a Fieldbus.

2.5. straton Runtime license for embedded system (Ref: T5RT)

For any straton Runtime integrated in the produced platform, a Runtime fee needs to be acquired. This Runtime fee is materialized by a sticker provided by STRATON AUTOMATION.

Licenses are ordered by lot; one sticker corresponds to one license.

2.6. straton annual subscription (Ref: ASUVMTK)

An annual subscription is mandatory for the maintenance of the straton VMTK product. The annual subscription fee is payable in January for the incoming year.

An active maintenance plan allows to Upgrade/Update the straton Runtime, getting the source files of the latest version. It allows to use the latest version of the IDE too.

It includes:

- Support by phone and e-mail (5 hours/month)
- Up to 5 straton IDE licenses for the current year, with Unlimited tags

3. Architecture of the straton T5 runtime software

The T5 software is basically divided in four layers such as shown in the following diagram. This architecture is motivated by portability and system independence:

4 - Application
3 - Standard T5 runtime files
2 - System dependent modules and add-on features
1 - Definitions

When porting or integrating the T5 runtime, you have different styles of work according to the layer.

3.1. Definitions (1)

Most of portability issues for taking care of CPU, OS and compiler conventions are solved by a set of definitions. This is the first work you must do when implementing the T5 runtime on a new platform. All system dependent definitions are grouped in the t5vm.h file. Never enter changes directly in the file. Instead, use the T5DK wizard that provides a graphic interface for entering all definitions. This ensures that your T5 implementation can be easily maintained or upgraded on later releases of straton.

3.2. System dependent files and add-on features (2)

This layer contains all the files you may need to change or adapt according to your system and to your implementation strategy. The straton Development Kit proposes various templates for each of these files. You can directly use a template from the straton Kit and let STRATON AUTOMATION maintain it, or you may use a template to start and customize it, or you can start from a blank file and implement something new.

Below are the system dependent files:

- Socket interface
- Serial communication
- Handling of memory blocks
- Storing non-volatile variables
- File management
- Real Time clock
- "C" functions and FBs in a dynamic library
- I/Os in a dynamic library

In addition to standard features, you will generally have to include extra software to the T5 runtime to manage I/O devices and perform custom operations. The following files enable the integration of add-on features:

- Writing "C" functions and functions blocks
- Integrating built-in I/O drivers

3.3. Standard T5 runtime files (3)

This layer contains system independent files developed and maintained by STRATON AUTOMATION. The source code of all these files is delivered for compiling and debugging purpose. But you should never enter changes in these files. If you absolutely need to introduce a custom feature in a system independent file, contact STRATON AUTOMATION so that we can introduce a new hook or entry point that allows you to work in system dependent files.

Use the T5DK wizard to know the list of system independent files. This tool is also used for copying and maintaining system independent files in your "C" project.

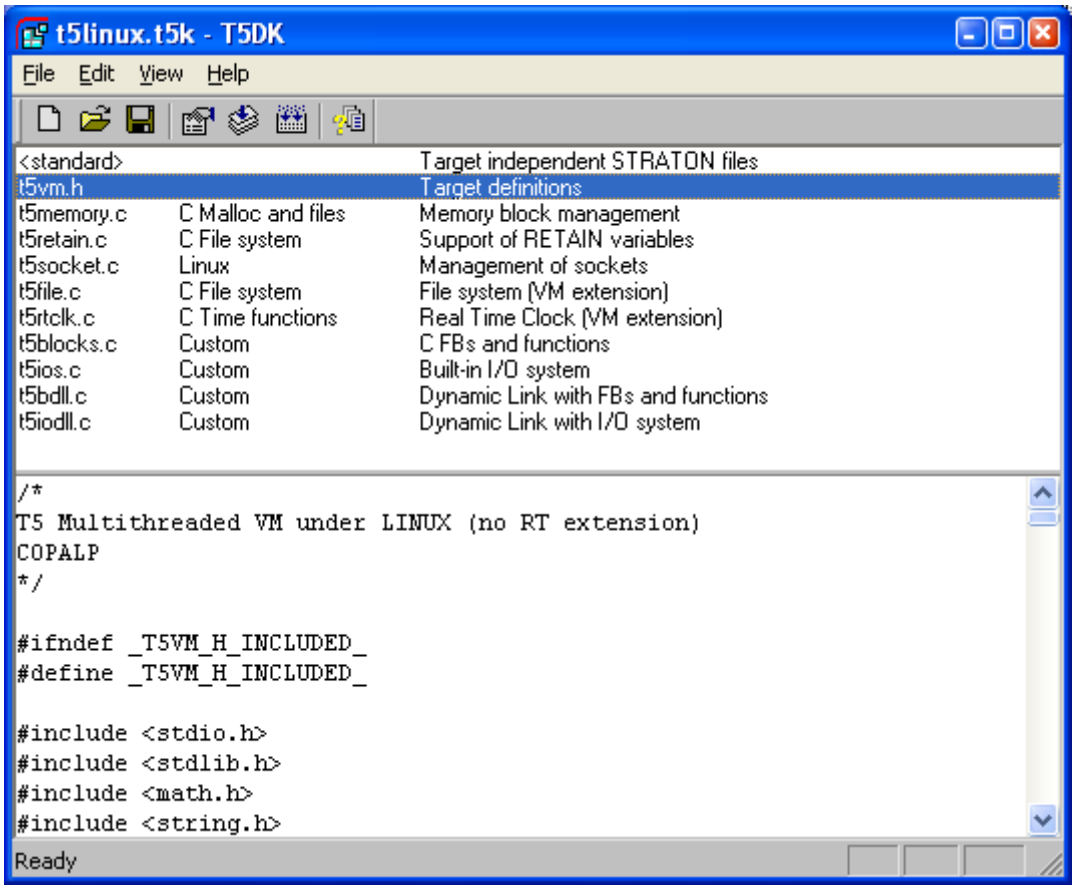
3.4. The application (4)

All T5 runtime components implemented in the standard T5 runtime files are available as APIs documented in this guide. The top layer is the main target application that links and chains all high-level services proposed in T5 runtime components. The top layer may have very different forms according to the selected implementation strategy (monotasking, multi-threading, multi-processing...) and may also include links to specific components running together with the T5 runtime in the target platform.

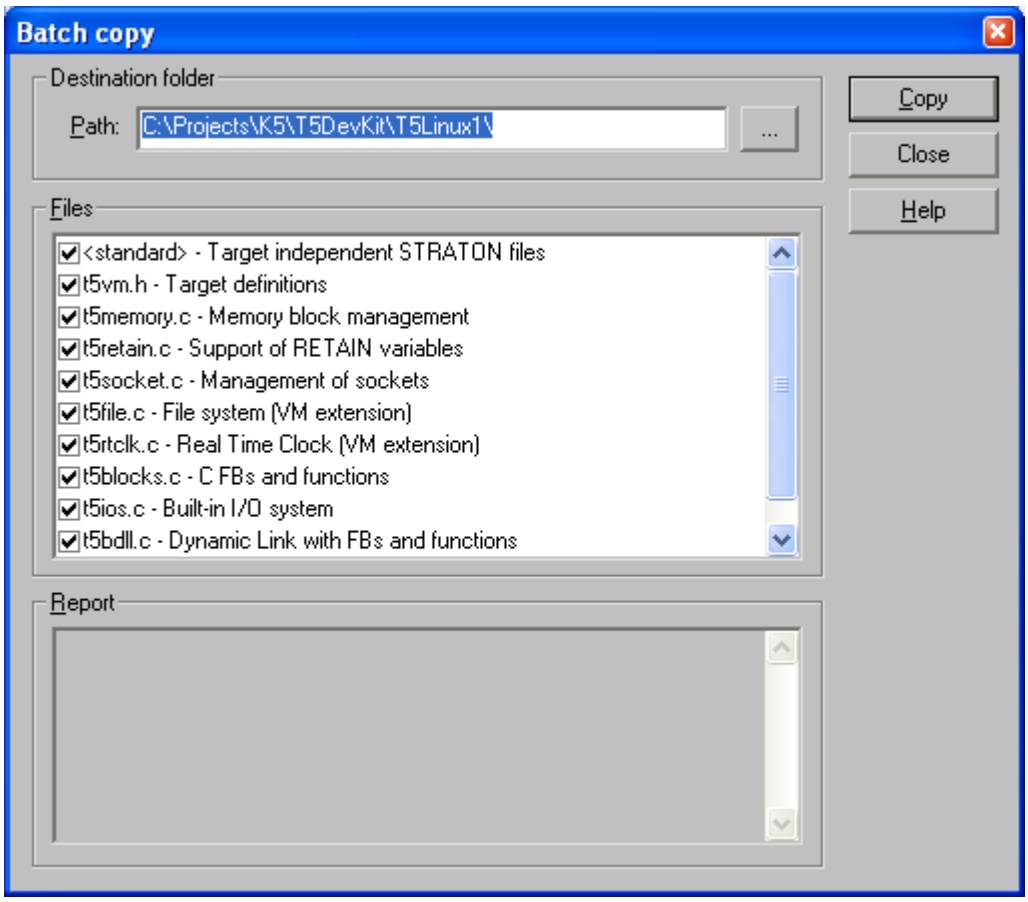
The straton Development Kit proposes a set of samples for top layer. You can use samples for starting a new implementation. Even in the case of a completely specific implementation, you can look at available samples and find some pieces of source code you can re-use in your application.

3.5. The straton Development Kit Wizard

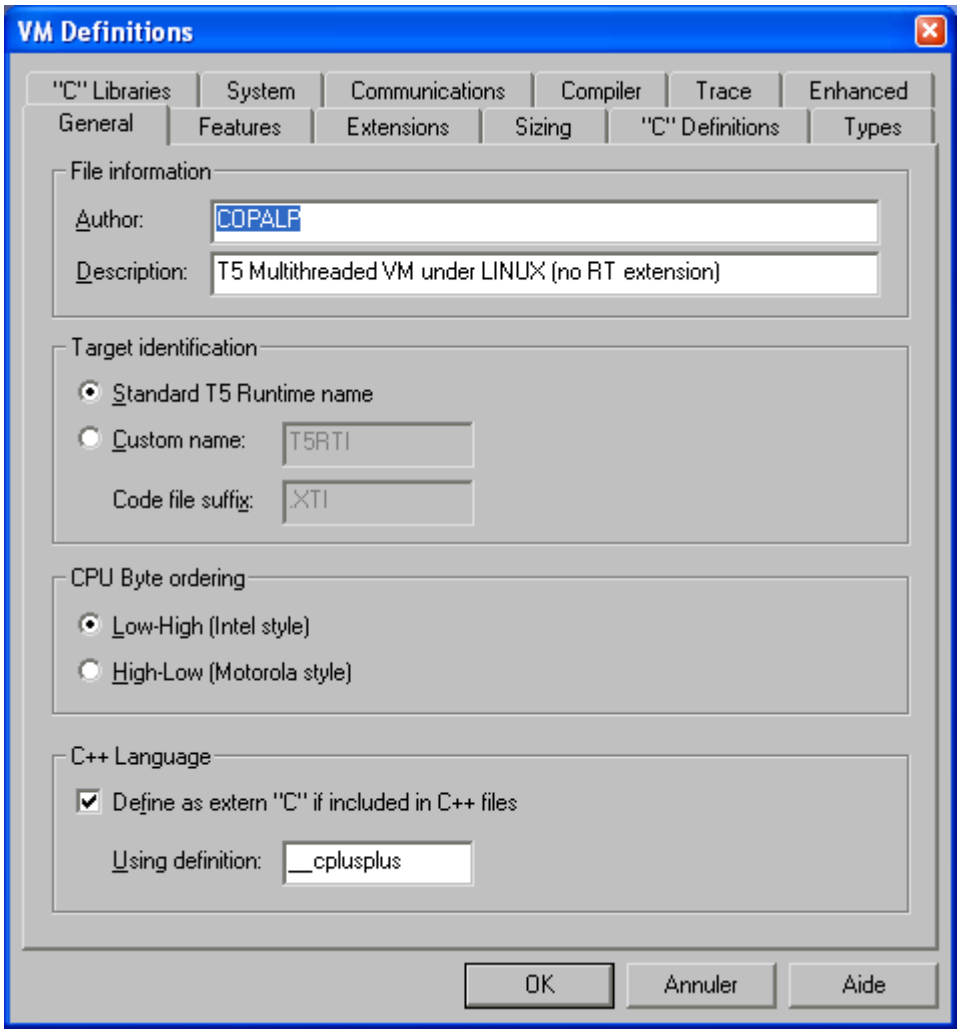
The Wizard provided with the straton Development Kit enables you to enter all required system or CPU definitions and generate / maintain T5 runtime source files in your "C" project:



It guides you for the implementation of all system specific modules and offers for each of them a set of ready-to-use templates. Additionally, the Wizard provides tools for copying / maintaining standard files (the ones you don't need to change) in your project:

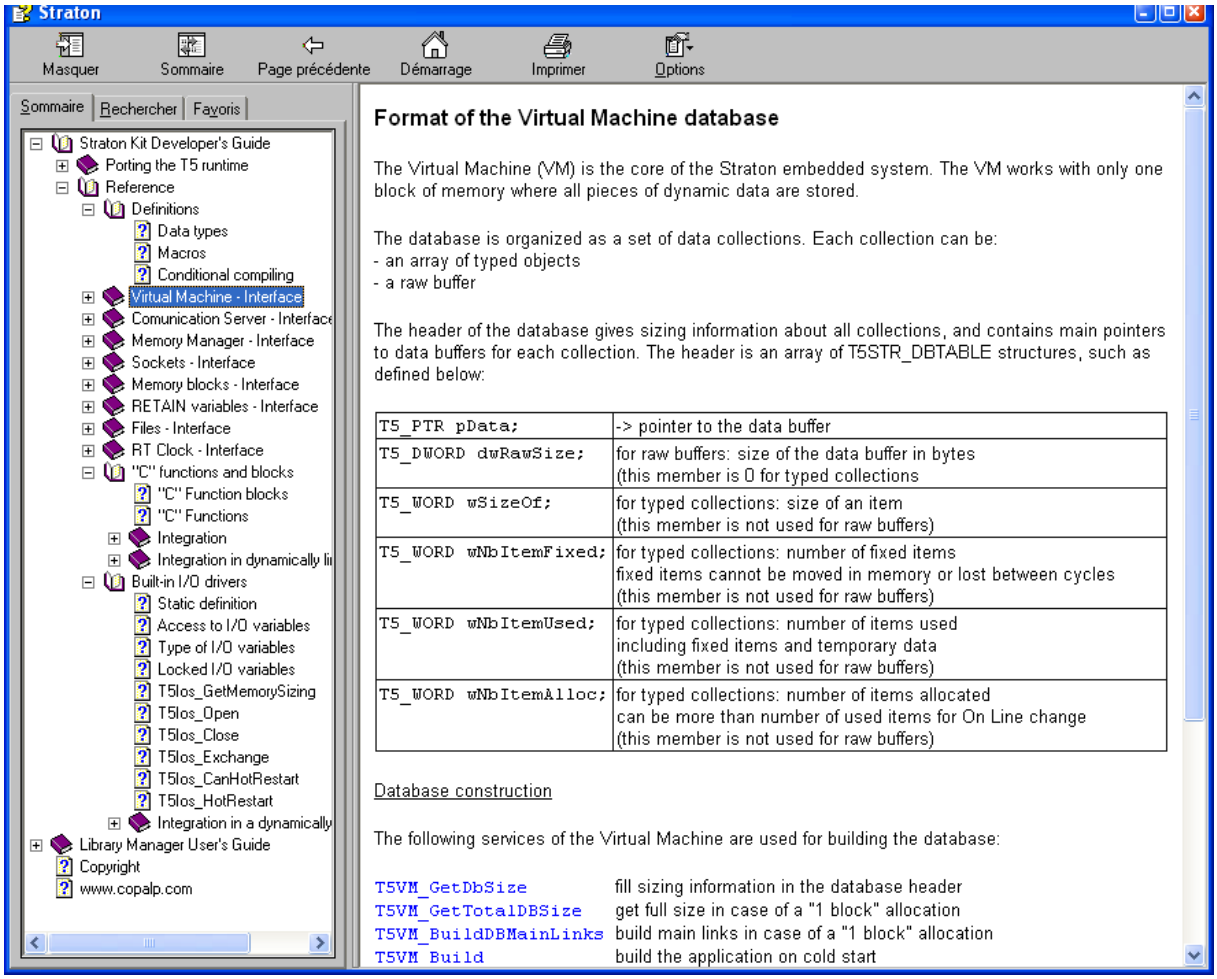


The Wizard also provides an easy to use graphic interface and guides you through the definitions required for "C" portability and hardware independence:



The straton Development Kit is delivered with complete source code of the T5 runtime software so that you can freely recompile and debug it.

The very complete online documentation provides guidelines for managing low-level implementation of hardware dependent features, completed with a full reference of all straton interfaces (APIs). Ready-to-use examples of T5 Runtime implementation will help you building quickly first prototypes of your embedded system.



3.6. Setting up your “C” project

When you start developing an embedded system that integrates the straton T5 runtime, the first thing you need is to setup the "C" project. You must perform the following operations:

1. Initiate the project skeleton
2. Include all straton files to the project
3. Setup compiling options

Initiating the project skeleton is strongly dependent on the "C" compiling tool you are using. Some tools may provide you wizards to initiate the project, or you can simply start with a simple "hello" application. In all cases, you must create the main entry point of your application, as straton does not provide it as a standard module. Alternatively, you also can start with a sample application delivered with the straton Development Kit and adjust it to your needs.

You should use the T5DK Wizard for copying to your project folder all straton files. When this is done, you must add the straton files to your "C" project or makefile. This includes:

- The straton definition file (created using the T5DK Wizard)
- The system dependent files:
  - Socket interface
  - Serial communication
  - Handling of memory blocks
  - Storing nonvolatile variables
  - File management
  - Real Time clock
  - "C" functions and FBs in a dynamic library
  - I/Os in a dynamic library
  - Writing "C" functions and function blocks
  - Integrating built-in I/O drivers
- All the standard straton system independent files
- The list of standard files is given in the preview window of the T5DK wizard.

Now you are ready to develop your embedded application, i.e. to customize some system dependent files and to write the top-level application using APIs of straton components.

3.7. Implementation strategy

The straton software enables different styles of implementation. You need to select an implementation strategy depending on the real time requirements of your embedded system and the capabilities of the Operating System you use.

Basically, the straton T5 Runtime performs two kinds of operations:

- Survey and serve requests from the communication, enables downloading and remote control of applications (start / stop / change) from the remote debugger.
- Execute cycles (actually run the application)

Executing cycles obviously requires more real time than serving the communication. If you use a multitasking Operating System or scheduler, it is worth to separate the straton software into two tasks, and to give a higher priority to the execution of cycles.

We can distinguish the following typical implementation strategies:

Monotasking	This is the easiest way to implement the T5 Runtime, just by chaining in a simple loop the VM cycles and the serving of communication requests. This implementation does not ensure real time of VM cycles. Obviously, the scheduling of cycles is disturbed by networking in that case.
Multithreaded	Separating the execution of VM cycles in another thread and giving it a higher priority provides better real time behavior. A multithreaded implementation is easy to establish as the database is simply shared by the threads on the same process and does not require the implementation of named shared memory.
Multi process	This is the most sophisticated implementation, with the execution of cycles handled by a separate process. The database must be implemented in named shared memory. This implementation provides an open architecture for linking other processes to the database.
Custom	straton also enables more particular implementations, where a custom application, working with its own execution rules, takes care of scheduling T5 Runtime services.

The T5 Runtime Development Kit includes sample applications that demonstrate various possible implementations. Look at them in order to estimate your work and select the best strategy for implementing the T5 Runtime in your embedded system.

4. Annex 1

Please fulfil all the missing information's and send it by email to: sales@straton-plc.com

CUSTOMER INFORMATION:

Company name:_____

Contact name: _____

Position:_____

E-mail: _____

Phone N°:_____

The following table is a non-exclusive list of the required information/materials in order to create a straton Runtime. Thank you to fulfil the "Customer" column.

Requirements	Description	Customer
QUALIFICATION STEP		
▸ Application	• What is the application for this project • Does your company plans to develop the application (with IDE/IEC61131), or will you sell the device ready for development	
▸ Hardware	Hardware Manufacturer and board (reference) CPU: • 32 or 64B, • Frequency, • RAM memory • Flash memory	
▸ Environment	Operating System, version, and its distribution	
▸ I/Os	Types of I/OS and approximately their number	
▸ Protocols to use	Which protocols have to be used, and are they from STRATON AUTOMATION or third parties	

WORKSHOP STAGE		
Login / Password	All necessary information for the login in the hardware device.	
Remote connection	Type of link for console SSH, telnet, serial including IP address, baud rate	
TCP/IP	How to change IP address and make it permanent in the system	
File transfer	How to transfer binary file to platform: scp, ftp.	
Power supply	Required power supply (Volt/Ampere) and connector cable.	
Development environment	Required development environment: Debian 2.6, Ubuntu 3,1, QNX 6,5 Momentics, VxWorks 6,8, Windows 32/64 bits, homemade OS…	
Compilation tool chain 'C'	Preferred VMWare/VirtualBox with tool that contains libraries, cross compiler and sample of makefile or full installation tool chain	
Compilation tool chain 'C++' with STL Library (Required for some protocol stacks)	Preferred VMWare/VirtualBox with tools that contains libraries, cross compiler and sample of makefile or full installation tool chain	
Local I/Os	'C' sample of access to the IOs drivers and schema of how IOs can be wired, special cable in case of special connectors, bridge to connect inputs with outputs	

Signature & stamps of the company: Date: