March 1st, 2019

Ac6 System Workbench for Linux

Abstract

Ac6 System Workbench for Linux is an integrated development environment for Linux embedded systems development. It provides a unified environment for the whole Linux-based embedded system development process. Of course there are a few other solutions, some of them open-source, but they need to combine independent tools, some command line based, some with a graphical user interface. System Workbench for Linux is the only one to integrate the whole process in a single tool.

Furthermore it can be integrated with most microcontroller IDEs based on Eclipse, like System Workbench for STM32, to support development of asymmetric multicore systems combining the real-time capacities of a microcontroller running a bare-metal or RTOS-based applications and the flexibility of a Linux-based application.

Presentation

Ac6 System Workbench for Linux assists the developer in every steps of system realization:

- Porting the Linux kernel on the board (if required)
  - When a standard board is used, the manufacturer proposes a tailored kernel which will just require integration in the final system.
  - On the other hand, when a new board was developed, often based on a reference design, this requires porting from either a « standard » kernel, or a kernel tailored to the board used as reference.
  - In most cases that can be done just by creating and compiling a target-specific device-tree, that System Workbench for Linux will automatically compile and deploy.

SARL au capital de 138600€ - SIRET 449 597 103 00026 - RCS Nanterre - NAF 6202A

21, rue Pierre Curie - 92400 Courbevoie - Tél : 01.41.16.80.10 - Fax : 01.41.16.07.78

http://www.ac6-tools.com

- Kernel configuration and specific drivers development
  - Even when using a standard board, it is often needed to adapt the kernel configuration.
  - System Workbench for Linux allows you to edit the default kernel configuration to fit your needs
- Driver development
  - Specific drivers may need to be added for additional hardware needed by the developed system.
  - System Workbench for Linux allow you to create your own drivers and debug them on your target using all the power of an Eclipse-based debugging environment
- Choice of applicative packages to integrate in the system image.
  - These packages are mostly open-source packages
  - Some packages will depend on other packages, possibly optional
  - In certain cases, several possibilities can be chosen for a given function; the appropriate alternative will be chosen depending on additional services provided and system constraints.
- Building standard packages
  - These packages can use various build mechanisms (eg auto-tools, make, cmake…), which must be managed and automated.
  - It is indeed not possible to manually build the different packages, as a platform may easily require several tens of packages…
- Development, test and validation of your application
  - To build a full system, it is also necessary to assist the development team in developing and debugging the application.
  - This application will often be developed under Eclipse, to take advantage of its features.
  - This requires adapting Eclipse to cross-compilation and development.
- Support of asymmetric multicore environments
  - Modern System on Chips include several processors, often Cortex-A cores running Linux and a Cortex-M core for critical real-time applications
  - Applications will have to be split in two parts that must communicate together
  - This requires adapted communication frameworks and the ability to develop both parts in the same IDE
- Building the final image
  - Depending on the hardware platform, various options must be provided for deployment on systems completely flash-based (diskless).
  - Update mechanisms are also needed.
- Configuration management of your embedded system
  - Embedded systems are often tailored for each specific customer
  - That means that sophisticated configuration tools are needed to be able to manage several variants of a base configuration

Open Source Solutions

Various free solutions exist on the market to support some of these different steps. Independent tools exist for application development, configuration management, platform building, etc. However, no global solution takes care of the whole embedded system development; moreover existing partial solutions for platform development have various pros and cons, as developed below.

- Buildroot
  - OpenSource tool.
  - Relatively friendly tool which can build a cross-toolchain and a Linux platform.
  - The platform configuration is menu-driven into a terminal-like interface (menuconfig).
  - It doesn’t manage the Linux kernel modifications, without manually changing the kernel configuration.

    - It cannot simply store a complete configuration, without manual manipulations to save and restore it.
    - Adding new packages to buildroot require modifying the buildroot tool itself, therefore editing and modifying script files.
    - It doesn't manage applications development.
    - It generates an "empty" platform without the applications which, later, requires manual integration.

- Open Embedded, Poky, Yocto
    - Those are three successive evolutions of an OpenSource tool.
    - As buildroot, they can build the toolchain and the Linux platform to be embedded.
    - Creating a new platform or adding new packages requires scripts edition in a specific language (bitbake).
    - They cannot simply store a complete configuration, without manual manipulations to save and restore the configuration.
    - They don't manage applications development.
    - They offer an Eclipse plugin to edit the configuration scripts, with text-based, syntax coloring, editors, which can be also used to launch the platform configure and build tasks.
    - They generate an "empty" platform without the applications which, later, requires manual integration.

Ac6 System Workbench for Linux's innovates:

Ac6 System Workbench for Linux does the merge, in a single tool, of all features of existing tools.

- All interactions with the tool are performed in graphic mode, using a unified Eclipse-based interface.
    - Only LinuxLink proposes such a unified interface, but does not allow convenient addition of new packages to the library.
- It allows in a simple way, using a fully graphical interface, to add additional packages to the packages library.
    - Other tools (buildroot, Yocto) requires script files edition.
- It provides predefined packages, tested and validated, for features required by an embedded system.
    - Packages are provided in various libraries, sorted by target board, SoC, functionality, etc.
    - Furthermore, as the System Workbench for Linux process is similar to the Yocto process, it is possible to use Yocto recipes as a base for integrating new packages.
- It supports the development and debug of applications
    - Buildroot and Yocto don't manage this function at all.
    - In an asymmetric multicore environment, the MCU and Linux applications can be developed and debugged simultaneously
- It supports adaptation of the kernel configuration and device-tree to your target board
    - Buildroot and Yocto don''t help at developing these and their integration in the Yocto environment needs the creation a a full BSP Yocto layer
- It supports the development and debug of user-specific drivers
    - OpenSource tools don't offer development support at all.
- It allows integration in the library of all user-defined components (Linux and MCU applications, drivers, kernel configuration and device tree) so they can be used in other projects.
    - OpenSource tools you need to manually modify the build environment (adding makefile fragments for Buildroot or layers with bitbake code for Yocto).
- It manages the final platform as a whole Eclipse project.
    - Only System Workbench for Linux is integrated to Eclipse at the platform level; with other tools Eclipse is used only for application development.

- It manages the final image, ready for deployment.
    - Other tools, create generic Linux platforms, in which you must integrate your applications manually.
    - Other tools also don't generate complete deployable images, but require the developer to manually create his deployable images.
    - On the opposite, System Workbench for Linux will create directly filesystem images, that may be splitted on different storage devices (eMMC, SD, Nor or Nand flashes based on the architecture defined by the user, generating also the runtime file needed to moun
- It integrates nicely with source code management systems for system configuration management
    - All the information needed to recreate a given platform is included in only two System Workbench for Linux components, the Platform and Root Filesystem projects.
    - These two Eclipse projects can simply be managed using any configuration management system integrated in Eclipse, based on GIT or Subversion for example.
    - They contain the whole definition of the delivered platform that can be rebuilt identically when needed, for maintenance or evolution.

Asymmetric multicore support

On asymmetric System on Chips (Cortex-A + Cortex-M), System Workbench for STM32 can be integrated with microcontroller development tools based on Eclipse like System Workbench for STM32 to develop and debug simultaneously:

- The critical real-time parts running on the Cortex-M core
    - This could be programmed either as a bare-metal application or using a RTOS like FreeRTOS
- The non-critical parts, like GUI, network communications, etc. on the Cortex-A running Linux using standard toolkits available on Linux, including:
    - Networking frameworks: TCT/IP, SSH, SSL…
    - Graphic toolkits: Qt, X11, Weston, Wayland, OpenCL, OpenCV
    - Multimedia: Alsa, video for Linux, gstreamer
- The communication between both parts using the OpenAMP standard protocol
    - OpenAMP allows for communication and synchronization between the MCU and the MPU
    - The real-time behavior of the MCU is nor disturbed by OpenAMP nor the MPU under Linux