bitshift DYNAMICS

Qt Embedded Testrack

Abstract

This whitepaper provides an overview of the bitshift dynamics Qt Embedded Testrack services and technologies. It describes the daily challenges for companies that develop embedded systems and gives an overview how the Qt Embedded Testrack can support them. The paper is suitable for people who have a general understanding of embedded systems and application deployment.

Revision 1a, January 2016

bitshift dynamics GmbH · Neudorfer Str. 1 · 79541 Lörrach · GERMANY
Geschäftsführer: Alexander Nassian, Markus Pfaffinger
Registergericht Freiburg i. Breisgau, HRB 713747
E-Mail: info@bitshift-dynamics.com
Web: www.bitshift-dynamics.com
Phone: +49 7621 58673 - 0

Table of contents

Introduction

The development of an reliable embedded system is often a challenging task, even for experienced developers. Aside from hardware issues there are several components that have to be coordinated and configured properly in order to ensure reliability, stability and efficiency.

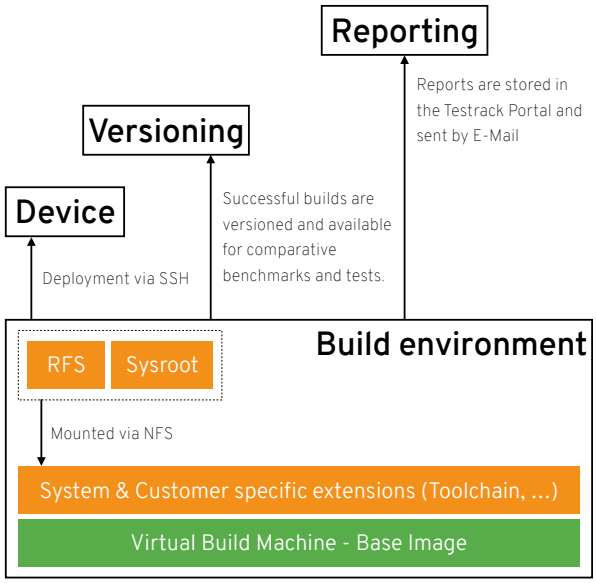
While the development is usually a well covered task, testing and benchmarking is very often underrepresented. A sophisticated testing and benchmarking concept of the application under real world conditions saves time and money and prevents frustration. As a matter of fact, testing and benchmarking the application often and as early as possible is crucial for a modern development lifecycle.

With the Qt Embedded Testrack we offer a service that guarantees a fully working and compatible Qt library on our customers embedded systems, as well as a comprehensive testing and benchmarking suite. Continuous tests and benchmarks of every single software change provides the earliest possible chance to fix problems before they affect the customer's product. It also ensures the conformity of the development to the defined requirements and gives active feedback on the performance and stability situation.

Qt Embedded Testrack overview

This chapter provides technical details about the hardware and software implementation of the bitshift dynamics Qt Embedded Testrack. It is also a first guide on how to integrate customer's software and hardware installation to our test and benchmark environment.

Build Server



Embedded systems need to provide SSH access in order to be enabled for automated build and test services. This can be done in advance, or when the hardware is installed, via the browser based Testrack Portal with the serial console.

In order to provide a consistent and reproducible build environment our build system is entirely based on snapshotted virtual machines. We provide a simple base image based on CentOS which can be further customized by our engineers or the customer itself. Any change to it can be reverted in case of changes that lead to build errors. Our build servers always use the virtual machine that is assigned to the embedded system that it builds software for. This method also ensures a clean separation between customer specific environments.

In cooperation with hardware vendors we also provide prebuilt virtual machines for specific profiles (hardware + toolchain + RFS) like for the Raspberry Pi 2 and others.

With our flexible CI system, any build and deployment or test configuration can be set up. The most common ones are provided as builtin configuration sets in the browser based Testrack Portal.

- **Qt Library**

  Before the embedded system is integrated in the automated Qt build system, if necessary we do hardware specific customizations to the Qt configuration and/or the RFS. These changes are versioned and will be applied to any updates to the Qt library, or the RFS. If any conflicting changes happen, we will fix and version them.
  Whenever a change is pushed to the public Qt repository, our build servers pull them and do a full rebuild on the library for all systems in the Testrack. Successful builds will be deployed to the systems, failed ones are pushed to our error reporting system that will inform the affected customers and our engineers.

  Note Customers holding a commercial Qt license may have patches to the Qt library that are not mainlined. To ensure that these patches are not disclosed in any way, all Qt library builds for these customers are done on separate build servers that are physically located at the Testrack in our own datacenter.

  Note Builds of the opensource Qt library (including any opensource licensed patches) *may* be done on build servers that are not physically located in our own datacenter. We provide a high level of security, but these servers are not under our physical control. On request we also provide the build on our build server infrastructure for commercial license holders.

- **Customer specific applications**

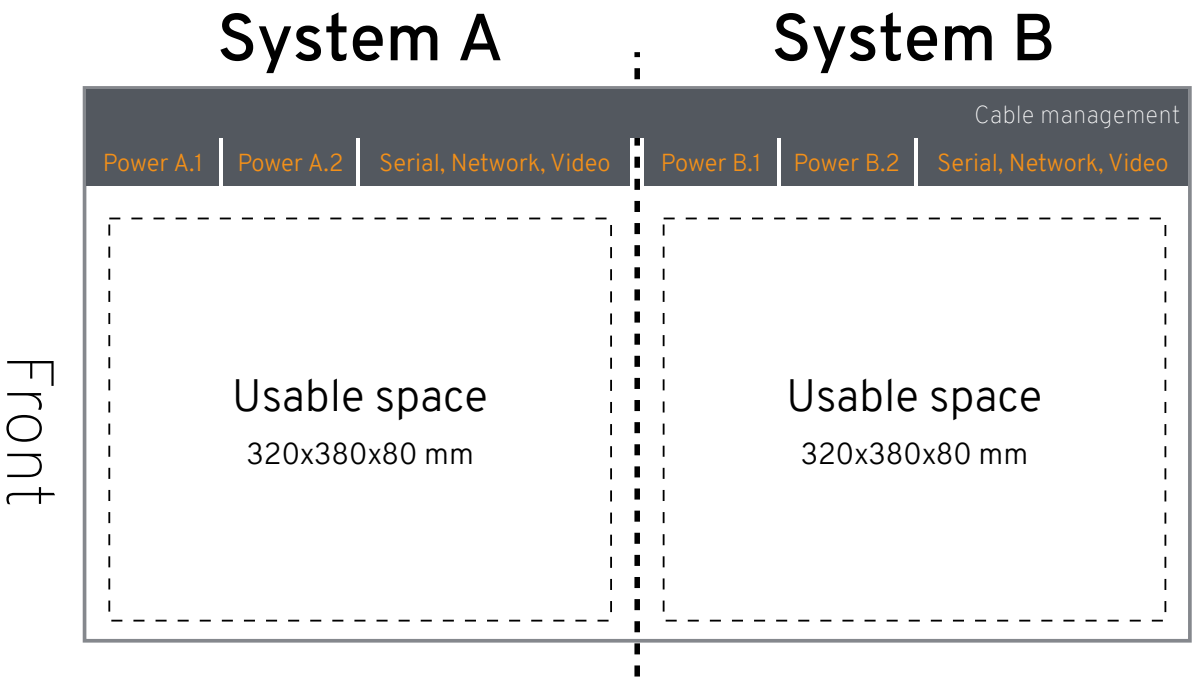
  The customer's application that runs on the embedded system for continuous testing may be delivered either in source or as binary filesystem overlay.
  Using the Testrack Portal, customers can setup a git repository that they can access to push their application's source code to it. Whenever a new build is triggered (eg. by a change in the Qt library or manually), the application will be build on one of our build servers. If the build was successful, it is automatically deployed to the specified embedded systems. Failed builds are pushed to our error reporting system that will inform the customer.

  Note Customer applications are *always* built on servers that are physically located in our own datacenter.

Hardware integration

The embedded system and any may necessary additional hardware is installed on a standardized Testrack Slot. These slots are based on easy slide rails based rack shelfs (standard depth of 800mm, 2HE) and provide all needed interfaces to the Testrack. Any slot is divided in half to provide space for two embedded systems. Embedded systems that need more than this standard space (320x380x80 mm) can be installed on individual Testrack Slots.

Some applications may need additional hardware that doesn't fit to our standardized or individual Testrack Slots such as oscilloscopes or laboratory clocks. This kind of hardware may also be integrated on an individual base in cooperation with our customers.



Remote access

After the physical integration of a customer's embedded system, access to the hardware is provided in several different manners. Additional access methods may be implemented if a customer requests them.

1. **Power supply**
   Every testrack slot is equipped with two remotely switchable, metered, power supplies that are UPS backed. This is essential when accessing the system at a state before any operating system has started to reboot, or shutdown for maintenance. Access to the live and historical data for power consumption and switch states is granted via the Testrack Portal interface.
2. **Serial console**
   Embedded systems that are equipped with a standard RS232 or USB serial port can be accessed via the browser based Testrack Portal, or via our SSH-Serial gateway. This is necessary when e.g. accessing the bootloader menu.
3. **Video**
   If available, VGA and HDMI are connected to converters that allow streaming the output to the browser based Testrack Portal. The generated video stream may be used for screenshots for error reports, unit testing and many other use cases.
4. **VPN (SSH and others)**
   Customers are separated completely by VPNs (one VPN per customer) in which all embedded systems of the customer are placed in. The devices in the VPN may have, configurable, direct outbound internet access. Customers are able to configure any additional access method (SSH, Telnet, FTP, ...) to the embedded systems via the browser based Testrack Portal.

These interfaces also can be programmatically controlled by a script and C++ library we provide to our customers. This can be used for highly customized test applications like a screenshot tool that provides a view on any implemented GUI state. Another routine that was already implemented is a test for display flicker after power on.

Test suites

Testing and benchmarking is an essential feature of the Qt Embedded Testrack. After every successful Qt or application build and deployment we run three stages of tests and benchmarks.

1. **Standard Tests and Benchmarks**

   The first run contains standard, publicly available unit tests and benchmarks. These group contain unit tests that are shipped with Qt itself, but also standard tools like GFXbench, Quake 3, sysbench, nbench, iperf, mbw, hdparm and others.

   In this area we use synthetic benchmarks for the purpose of comparing systems as well as organic benchmarks to get a better view on real world applications.

2. **bitshift dynamics Test Suite**

   Over the years we developed our own set of tools, tests and benchmarks that come from real application's requirements and problems. In addition to the standard toolset we run our test suite also on every successful Qt build and deployment.

   These test for example cover the correct handling of touch screens, applications reaction to fuzzy inputs on various interfaces, the system's and application's behavior on different load situations, and many more.

3. **Customer's Test Suite**

   Via the Testrack Portal we offer a simple interface to our customers where custom test and benchmark applications may be deployed and run. The results of these tests will also be included in the generated reports. Through our programmable interface to the Testrack it is possible to define highly customized test routines that also may contain interaction with the hardware like touch screen or display tests.

   In order to mock external hardware that customers don't want to install in the test environment it is also possible to develop test routines that simulate I/O on a variety of standard interfaces like GPIO, I2C, SPI, RS232, RS485, etc..

Reporting

In the end of the day, reporting is the most important feature of the Qt Embedded Testrack. Any build and any test run has no value if there is no report that contains all the needed information. Because of this fact the Report Center in the Testrack Portal is the most configurable yet simple module in the whole ecosystem. Users with a very high demand of customization can include any information that is created during configuration, build, test or benchmark to the report and use in further evaluations. There are also several predefined profiles for different aspects of an embedded system application that can be used right out of the box. These profiles can be assigned to any registered person or department of the customer so that everyone can focus on the information that is most important to them.

A typical report consists of two main parts, the overview and a package containing the raw data for further processing. The overview document is delivered as a human readable PDF document and is primarily focused on deviations between different builds of the system, Qt and the customer's application. These deviations are the most common source of performance and stability issues. Secondly the report shows alle the important test and benchmark results compared to the previous, the 10th, the n/2th and the very first generation of the system, Qt library and application to give a first rough overview about the development of performance, stability and overall system quality.

The data package contains all raw information that were created during the whole run:

1. **Scripts used to create the system, Qt library and application**
2. **Any configure, build, test and benchmark output**
3. **Detailed power consumption data and diagrams**
4. **The full protocol of the affected Testrack slot which contains any changes and actions done to is whether it was done in hard- or software**

The Testrack user may also define rules for automatic report delivery via mail or upload to FTP or different cloud services. So it is possible to have all reports stored for later analysis, but get immediately informed if any performance or other application requirement was violated.