GDP Hands on Session

October 20, 2016  |  AMM Burlingame - All Members

Tom pollard: GDP maintainer. Developer at Codethink Ltd. tom.pollard@codethink.co.uk
Agustín Benito Bethencourt: Principal Consultant - FOSS at Codethink Ltd
● **Baselines**: outcome of the compliance program.
  ○ Yocto baseline (meta-ivi).
  ○ Baserock baseline.

● **Master**: rolling release: focused on auto system devs

● **GDP**: GENIVI Development Platform for apps devs.

● **New initiatives.**
  ○ GDP spins: community driven systems based on Master
  ○ GDP SDK: development tools
Master
Where collaboration takes place.

Latest automotive software available.
  - In OSS for automotive, GENIVI is upstream.

Targets FOSS auto system devs. & GDP contributors.

Build GDP from scratch for your favourite target or customise your build.
What is Master?

- **Rolling release** with the latest integrated software for automotive.
- Central integration point.
- Yocto (poky) based.
- Two main repos:
  - genivi-dev-platform
  - meta-genivi-dev
GDP: GENIVI Development Platform
Why GDP?

- It brings GENIVI components for automotive to the masses, including meta-ivi.
- Ideal for app developers and automotive newbies.
- Up to date stable software.
- Easier to consume and improved stability.
What is GDP?

- Acronym of **GENIVI Development Platform**
- FOSS and open delivery project.
- Published as binaries.
- GDP is based on Master (snapshot + stabilization).
- Available for several development boards & QEMU.
- Current stable version (**GDP-ivi9**)
  - Latest release: **GDP 11 RC2**.
GDP block diagram…
The diagram illustrates the components and layers of a technology stack. It starts with the Hardware: target boards at the bottom, followed by the BSP: firmware + kernel + drivers + ... layer. Above this is the meta-ivi repository, which includes Persistence, Common API, and lifecycle(*) layers. Below the lifecyle(*) layer, there are DLT and Audio Manager components. The Qt 5 layer contains QML, weston examples, Proof of Concepts Browser..., FSA, Automotive Message Broker, and RVI SOTA Client. The SDK(*) layer includes additional components, and at the top is the HMI Launcher.
GDP roadmap

GDP 11 Timeline
GDP 11 RC2

- **Released** on October 4th 2016. **Download** it!
- Demoed for the first time at ELCE.
- **GDP 11 RC2 highlights:**
  - Software: Yocto 2.1, Qt 5.6, AM 7.0, wayland-ivi-extension 1.10.9 (1.11 pre-release), meta-ivi 11...
  - Ports: QEMU, RPi2 & RPi3, Intel Minnowboard MAX/Turbot and Dragonboard 410c. Also build GDP for Renesas Porter & Silk from scratch.
GDP 11 RC3

- **Released** on October 18th 2016. [Download it!](#)
- Demoed for the first time at GENIVI 15th AMM.
- **GDP 11 RC3 highlights:**
  - New Application launcher and demo apps. Call for testing.
  - System based on RC2 + some new patches like wifi config in RPi3.
  - Available for RPi3 only.
    - Final release will be available for Intel Minnowboard MAX/Turbot, RPi2/3 and Dragonboard 410c. Also build GDP for Renesas Porter & Silk from scratch through Master.
GDP delivery project: other aspects
People behind Master & GDP

Delivery

- GDP maintainers
  - Changhyeok Bae, community.
  - Robert Marshall, Codethink Ltd.
  - Tom Pollard, Codethink Ltd.
  - Community testers.

- Other key people:
  - Meta-ivi & Renesas BSP maintainers, community management, devops/IT service, PMO, delivery team lead, GENIVI architect, LRT team …

Development

- GENIVI Expert Groups
- Community contributors
GDP tools

Tools GDP project uses today:

- **GitHub**: git repositories and code review.
- **JIRA**: bug tracker and task management tool.
- **Confluence**: wiki and blog.
- **go.cd**: integration/delivery mgnt.
- **Mailman**: genivi-projects@lists.genivi.org
- **IRC**: #automotive at irc.freenode.net
New outcomes

● New deliverables
  ○ First release of the Software Development Environment (SDE) for GDP.
  ○ First GDP spin: QtAS

● **Released** on October 18th 2016, at GENIVI 15th AMM.
Future of GENIVI delivery program (GDP)

● GDP 11 to be released before end of 2016
  ○ Consolidation of the new features.
  ○ Further system stabilization.

● New deliverables: consolidation.

● Infrastructure and services:
  ○ Improvements in the build capacity and deployment infrastructure.
  ○ More and better metrics.
  ○ Acceptance feature testing.
Call for action

- Documentation of the automotive software components
  - Need docu for newbies to extend the content critical path.
- More integration and use cases for the existing automotive components.
  - We have now a nice platform. What for? Make it meaningful for this industry!
- Testing
But above all…

More focus on automotive developers.

Check the latest GDP news.
Interesting links

● [www.genivi.org](http://www.genivi.org)
  ○ GENIVI [FAQ](http://www.genivi.org/faq)
  ○ GDP latest [GDP news](http://www.genivi.org/gdp_news)

● [GDP Master](http://www.genivi.org/gdp_master)
  ○ genivi-dev-platform
  ○ meta-genivi-dev

● Download:
  ○ GDP-11 RC2 and RC3
  ○ GDP-ivi9

● **Get involved:**
  ○ Get [the sources](http://www.genivi.org/get_sources)
  ○ Contribution [policies](http://www.genivi.org/contribution_policies)
  ○ Report [bugs](http://www.genivi.org/report_bugs)

● Follow up
  ○ Delivery status [reports](http://www.genivi.org/reports)
  ○ [GDP overview](http://www.genivi.org/gdp_overview) (weekly)
  ○ GDP [Out There](http://www.genivi.org/gdp_out_there)
Call for testing

GDP_11_RC3
Boards for the GDP hands on session
MinnowBoard Turbot

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Factor</td>
<td>MinnowBoard compatible 99x74mm</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel® Atom™ E3826 (2 x 1.46 GHz, 1MB cache, 7W, AES-NI)</td>
</tr>
<tr>
<td>DRAM</td>
<td>2GB DDR3L 1067MT/s, Memory down (non expandable)</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1x 1Gb Ethernet RJ45</td>
</tr>
<tr>
<td>Video</td>
<td>Intel HD Graphics</td>
</tr>
<tr>
<td></td>
<td>1x micro HDMI output</td>
</tr>
<tr>
<td>Storage</td>
<td>1x SATA2</td>
</tr>
<tr>
<td></td>
<td>1x MicroSD</td>
</tr>
<tr>
<td>I/O Connectors</td>
<td>1x USB 2.0 host</td>
</tr>
<tr>
<td></td>
<td>1x USB 3.0 host</td>
</tr>
<tr>
<td></td>
<td>8x buffered GPIO</td>
</tr>
<tr>
<td>Expansion Interface</td>
<td>MinnowBoard Max compatible Lure interface</td>
</tr>
<tr>
<td></td>
<td>High-speed expansion connector</td>
</tr>
<tr>
<td></td>
<td>Low-speed expansion connector</td>
</tr>
<tr>
<td>Console</td>
<td>Serial via FTDI cable</td>
</tr>
<tr>
<td>Boot Loader</td>
<td>TianoCore UEFI</td>
</tr>
<tr>
<td></td>
<td>CoreBoot / SeaBIOS</td>
</tr>
<tr>
<td>Power</td>
<td>5VDC via coaxial power jack</td>
</tr>
</tbody>
</table>

[3] [http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel](http://git.yoctoproject.org/cgit/cgit.cgi/meta-intel)
Renesas R-Car M2 Porter

R-Car M2 SoC
ARM®Cortex-A15 Dual Core 1.5-GHz
Multimedia Engine SH-4A 780 MHz
GPU
- PowerVR SGX544MP2 (3D)
- Renesas graphics processor (2D)
2 GB DDR3 memory (dual channel)
Three flash memory chips
- 4 MB SPI
- 64 MB SPI
- 128 MB NOR (one 128Mb bank or 2x 64MB banks)
Debug Ethernet (100 Mbps)
Storage connection
- one SATA rev. 3.1 port
- one SD card slot
- one microSD card slot
Analog Video In: ADV7180 Video Decoder
RCA jack
Audio codec: AK4643EN
Line In 3.5 mm jack
Line Out 3.5 mm jack
USB 2.0 port
microUSB port supports host, device and OTG modes
PCI Express x1 slot
CAN transceiver

Yocto / bitbake: introduction
Yocto Overview

- What layers to look into for recipes
  - bitbake = build tool
  - Open Embedded (OE) = build system
  - Yocto = umbrella project
  - poky = reference system

- Distribution: poky-ivi-systemd
- Machine: porter, intel-corei7-64, ...
- Package Extra Configuration
- Add more Packages in the build image
Recipe Overview

- Where to find to source code (git, svn, tar.gz, .c)
- What version
- Apply patches on it?
- Special commands.
- FILES_{$PN}
- RDEPENDS
- Build tree
- Native packages vs deployed packages

```
SUMMARY = "aaa"
DESCRIPTION = "bbb"
HOMEPAGE = "ccc"
LICENSE = "LGPL-2.1"
LIC_FILES_CHKSUM = "file://ddd/md5=597c8d49137513c98683e1d73158292f"

inherit cmake

PV = "hhh+git$(SRCPV)"

DEPENDS = "eee fff ggg"

SRC_URI = "iii, jjj.kkk"
SRC_URI += "file://fll.patch"
SRCREV = "955972390d16ca275159891cad29c2166217094d"
S = "$(WORKDIR)/mmm"
do_install_append() {
    mv $(D)/usr/include/hmm/* $(D)/usr/include
}
INSANE_SKIP_${PN} = "dev-deps"
```
Bitbake Build Process

The frameworks tags which targets to build

- fetch
- unpack
- patch
- configure
- compile
- install
- package

- populate_sysroot

Based on
1. Target to build
2. extra recipes in layer.conf
## Useful Bitbake Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bitbake &lt;image&gt;</code></td>
<td>Bake an <em>image</em> (add <code>-k</code> to continue building even if errors are found in the task’s execution)</td>
</tr>
</tbody>
</table>
| `bitbake <package> -c <task>` | Execute a particular *package*’s *task*. Default Tasks names: *fetch*, *unpack*, *patch*, *configure*, *compile*, *install*, *package*, *package_write*, and *build*.  
*Example:* To (force) compile a kernel and then build, type:

```
$ bitbake linux-$target -f -c compile
$ bitbake linux-$target
```
| `bitbake <package> -c listtasks` | List all tasks for package |
| `bitbake-layers show-layers` | Show layers |
| `bitbake -s | grep $package` | Check recipe version available / default to bitbake |
| `bitbake -e $package` | Shows all the steps bitbake will perform on a package |
Bitbake demo
1. This walkthrough will provide a generic example of how to modify an existing package included within the GDP system. It aims to provide a general overview of the workflow that can be used, involving tools recommended by the Yocto project such as quilt to create patches. This patch can then be added to the recipe, simulating a real workflow.

2. Removing the FSA icon from the HMI has been chosen as it provides visual feedback of the changes made to the source.

3. It is important to remember that rebuilding a package will cause any package that has a build dependency on that package to rebuild.

4. Content Model
This example will show the basic steps of adding an existing package to the generic GDP image, including finding & adding the recipe to the build environment.

This is a good starting point when attempting to add a package to a bitbake based system.

bitbake -s | grep $package to check if it’s already included in the bitbake environment. bitbake -g | grep $package will show if it’s already built for the image (note this does not mean deployed).

Ensure where possible to select a version of the recipe based against the same Yocto version as in use, currently this is 2.1 ‘Krogoth’ in GDP Master.
The simplest way to add a new recipe to the GDP, especially for testing is to use the "IMAGE_INSTALL_append = " package-name"" variable.

If a package recipe is correctly defined, it should build the packages and any needed runtime dependencies into your images sysroot.

If the package is from a new layer not already imported by GDP, follow the steps to add it including git submodules, init.sh & bblayers templating

If upstreaming the package addition into GDP, add the package name to an existing packagegroup if viable
This example will demonstrate how to add a new package to GDP, in the form of a simple Qt application.

A simple recipe will be created, and the package will be deployed to the system.

Diagram of [generic recipe creation process](#).

- **tspress**:
  - tspress tracks touchscreen presses and movement
  - Add recipe and update packagegroup-gdp-hmi and bitbake
  - To run, stop and restart weston: (systemctl stop weston; weston --tty=1 --no-config --log=weston.log & ) and then kill launcher, background and panel
  - Then run tspress

- **cepilosege**.
Note repository does not contain license file, bitbake will report this warning during packaging if incorrectly set against source, or will refuse to configure if value not given at all.

Depending on the dependencies of the package, a supporting package may need to be added to the GDP buildenv (meta-rust needed for SOTA client as an example).

Check licensing, particularly important if attempting upstream the recipes into GDP / meta-ivi.

To test the application works, kill weston and execute the binary over the ssh/serial connection.
Contribute
Contribute Patches (I)

Ensure you read over:

1. Contribute
2. How to contribute to GENIVI.

Generic steps to generate patchset to mailing list:

1. Ensure you’ve created your commits against the head of the master or supported release branch following genivi submission guidelines.
2. `git format-patch --cover-letter -M origin/$target -o outgoing/`  
   `edit outgoing/0000-*` (This is your cover letter)  
   `git send-email --to=maintainer@maintaineremployer.com --cc=genivi-projects@lists.genivi.org`  
   `--cc=username2@personalaccount.com outgoing/*.patch`
Contribute Patches (II)

This will generate the patch(set), allowing you to edit the default cover letter to add a blurb etc & finally send it. Patches to genivi-projects@lists.genivi.org are public and will be reviewed publically.

Please check Git Email Set Up for further help and examples.
So you’ve created & tested a change to GDP, the easiest way to contribute is using the Github Pull Request infrastructure. GDP repos are currently integrated with JIRA & go.cd support for CI testing, all of which are public.

Dealing with two tightly linked repositories has issues in terms of handling integration, certain scenarios requires changes in both repos to be tested simultaneously.

Examples
- Pulls
- GDP
- GDP policies
- Pull request example
- Go.cd example
Review CI Pipeline

- Workflow & Infra still in development - [http://go.genivi.org/go/pipelines](http://go.genivi.org/go/pipelines)
- We would expect all incoming PR’s to pass builds in these pipelines before merging into the ‘‘master’ branch.
- Still in development and ultimately CD tooling is needed for physical hardware tests.
- Dedicated release artifact pipelines, per target. [Tasks](#)
- [Maintain Open Source license during your product lifecycle.](#)
- [Quick demo if time permits](#) - This should be possible to show with whatever github PR demo we manage to make.
New in GDP 11 RC3
GDP-11 ‘New HMI’

- [Wiki page](#).
- [Pull request](#) and [commit](#).
- [HMI repository](#).
- New demo applications:
  - [FM Radio](#).
  - [Connected home](#).
  - [HVAC & commit](#).
- Currently released as ‘RC3’ for [RPi3](#), but master can be built for any target supported by GDP.
Interesting links

- SDE wiki page.
- Follow the improvements and bugs.
- SDE current repository

Tools included:
- Qt Creator
- Eclipse IDE

Other interesting links related with the GDP SDE:
- Yocto SDK
- DLT viewer manual.
Interesting links

- [Qt AS](#) spin wiki page.
- [Download](#) QtAS spin.
- QtAS [repository](#).
- [QtAS in Detail](#).
- QtAS [roadmap](#).
- [Contribute](#) to QtAS.
Thank you