Code sharing

Trusted Firmware M

Tamas Ban
Arm
Motivation

• Cortex-M devices are usually constrained in terms of RAM and flash.
• Secure boot and runtime crypto service has overlapping functionalities.
• Same peripheral drivers are used in bootloader and runtime TF-M.
• Flat memory space, relocation usually not supported.

• **Reuse the common code from bootloader and reduce the memory footprint.**
How does it work?

• No standard solution, toolchain dependent
• Manual investigation for shareable code
• Adjust symbol template file, which contains the name of shareable functions
• Sharable function name and address is extracted from bootloader at link time
• The artefacts of shareable code is added to secure firmware at link time
• To avoid symbol collision, shared symbols in secure firmware libraries must be mark as WEAK
• Linker picks up symbols from bootloader code instead of libraries (mbed-crypto, platform_s, etc.)
What type of code can be shared?

- Public functions and global variables
- Easy to share functions with local variables only
- Functions relying on global variables is a bit tricky
- Global variables must be placed to shared symbol section
Gain in flash utilization

- MinSizeRel build type
- Version: 465f73cdefba5fac6b7430baa4a424d789fed8f + code sharing patches
- MCUboot image encryption is disabled
- Size of secure image:

<table>
<thead>
<tr>
<th></th>
<th>ConfigDefault</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARMCLANG</td>
<td>GNUARM</td>
<td>ARMCLANG</td>
<td>GNUARM</td>
<td>ARMCLANG</td>
</tr>
<tr>
<td>CODE_SHARING=OFF</td>
<td>130012</td>
<td>123440</td>
<td>83612</td>
<td>80244</td>
<td>55008</td>
</tr>
<tr>
<td>CODE_SHARING=ON</td>
<td>120560</td>
<td>114368</td>
<td>77524</td>
<td>74340</td>
<td>53484</td>
</tr>
<tr>
<td>Difference</td>
<td>9452</td>
<td>9072</td>
<td>6088</td>
<td>5904</td>
<td>2524</td>
</tr>
</tbody>
</table>

- MCUboot image encryption is enabled: saving is up to ~13-15KB
Useability

- If bootloader is immutable then bug in shared code cannot be fixed with firmware upgrade.
- Global variables must be to shard symbol section.
- Shared global variables might need to be reinitialized in SPE explicitly, low level start-up does not do it.
- Compiler flag alignment(?)
- Shared code artefacts must be archived because they are needed when new secure image is built.
- [https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/4587](https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/4587)