What's the FOSS about?

What are the implications of using Free and Open Source Software in embedded systems and what should you bear in mind? By **Caroline Day**.

Free and Open Source Software – FOSS – is becoming more widely used by developers and engineers. Broadly speaking, in order to be FOSS, software will be distributed freely for anyone to view, use, copy or change, with at least a right of access to the source code. 'Free', in this instance, means 'freedom', rather than 'without financial cost', although FOSS is often financially free.

It is tempting to assume that because the software is 'free' it will also be low quality. However, this is not necessarily the case. The open source community is skilled, vocal and relishes bug hunting. For instance, an organisation or individual may submit code to a software project on a code repository such as Github (**github.com**); other users will often hunt out poor coding standards and errors, resulting in the code being improved or corrected.

A professional embedded systems code developer will likely have benefitted from the high quality of code resulting from FOSS software projects through use of software systems such as the Apache web server, Samba file sharing software or any of the many flavours of the Linux distribution. Indeed, Linux currently splits the bulk of the embedded operating system market with Microsoft's Windows Embedded Compact (previously Windows CE).

"It is worth considering if any original code you have developed should be made open source."
Caroline Day

By way of an example of the benefits of FOSS, let's say you are interested in designing a thermostat that can be controlled over the internet – perhaps having been inspired by the Nest Learning Thermostat or the stir created by Google's purchase of Nest Labs.

As you are not the first person to have this idea, it may save you time to consider whether people who have worked on similar projects have made their code available as FOSS – either as a whole project or as code enabling some specific functionality. A good starting point may be Nest Labs itself, which used FOSS in developing its product. In order to comply with FOSS licences, portions of the code underlying the thermometer are available, under FOSS terms, at **nest.com/legal/compliance**

Looking further afield, Github, for example, lists several projects detailing the build of an internet connected thermostat under various FOSS licences, including the MIT licence and GNU Public Licence (GPL).

There may be many sources of code, potentially overlapping in functionality, and it could be difficult to decide which code, if any, you should use. When making this decision, your first focus might be on finding code which provides certain desirable features and functionality. You may also consider the reactions of the developer community to the published code and start asking questions like 'have people tried it?' or 'did it work?'. After all, not all FOSS software projects are created equal: large popular projects, such as the Linux kernel or the Apache web server, have many active users and submitted code is likely to undergo vigorous peer review. Smaller or less

Practical tips on using FOSS:

- Do you want your modified code to be confidential? If so, avoid 'copyleft' licensed FOSS, such as GPL
- Consider the reputation of the source
- Has the code has been discussed and implemented by others? Is it taken from a popular and currently active software project? If so, the open source community is helping you with quality control and debugging (albeit without any warranty).
- Once you have chosen FOSS code, note the source.
- Bear in mind there may be other rights, such as patents, which could prove a barrier to commercialisation.

popular projects may tend to be more error prone.

However, it is also worth considering the licences applied to the code. For example, some licences are relatively permissive, making few demands on a developer wishing to use the code. Such licences may include clauses requiring a restatement of any licence terms in the original software, or requiring attribution of the source. The MIT licence, Berkeley Software Distribution (BSD) and Apache licences are examples of such relatively permissive licences.

Other licences – sometimes termed 'copyleft' or 'viral' licences – require that all software, even if you have added to or modified it, must be made available publicly, usually under the same licence terms. One example of such a license is GPL (currently, GPL Version 3).

Even with this more onerous requirement, GPLv3 is popular and GPL software has the backing of a community dedicated to open source, meaning it can often be high quality. It is important to be aware of the obligations imposed and, if appropriate, your employer's view in relation to them.

If you are considering acquiring FOSS from a variety of sources, it is also worth considering whether the licences are compatible in nature – not all are. For example, while it is generally accepted that Apache code can be included in a GPLv3 project, the reverse is not true.

Once you have selected your code, it is advisable to make a list of your sources so you can ensure you fulfil all obligations.

Consider other IP aspects

Even if you have gathered your software from an appropriate source and are prepared to comply fully with all FOSS licence requirements, it is still wise to consider other aspects of intellectual property.

For a start, FOSS generally comes with no warranties: not only will you have no guarantee that it is functional or useful, you also have no guarantee the publically available code is itself free of copyright infringing material, or that you could use the code without the risk of patent infringement.

After all, a patent could belong to a party entirely unconnected to the code contributors. A potential competitor would have to be wary of such third party rights, even if the software is easy to find and free to copy.

You should also consider your own intellectual property. The protectionist nature of patents, for example, and the requirement to maintain confidentiality of an invention before filing a patent are philosophically hard to square with some of the principles behind the liberal, collaborative FOSS community. Further, there is a (debatable) suggestion that a contributor to a FOSS project licences – at least implicitly – related patented technology. Some FOSS licences make specific mention of patent rights in the licence terms. Enforcing of a related patent by a party who has contributed to a FOSS project may therefore bear the risk of additional uncertainty and the prospect of reputational damage, and the use of FOSS may be inappropriate for these reasons.

Finally, while the above examples discuss modifying existing code, it is worth considering if any original code you have developed should be made open source. There can be considerable benefits: the FOSS community may provide improvements and manpower beyond anything you could provide alone, and opening up your code may allow complementary products to be developed, potentially both boosting the appeal of (and advertising) your product.

Author profile:
Caroline Day is a UK and European patent attorney with Haseltine Lake (**www.haseltinelake.com**)