



# **The future of Embedded Software Development**

**A White paper prepared by  
Proven Software Solutions Ltd**

*Don't Replicate...Innovate*

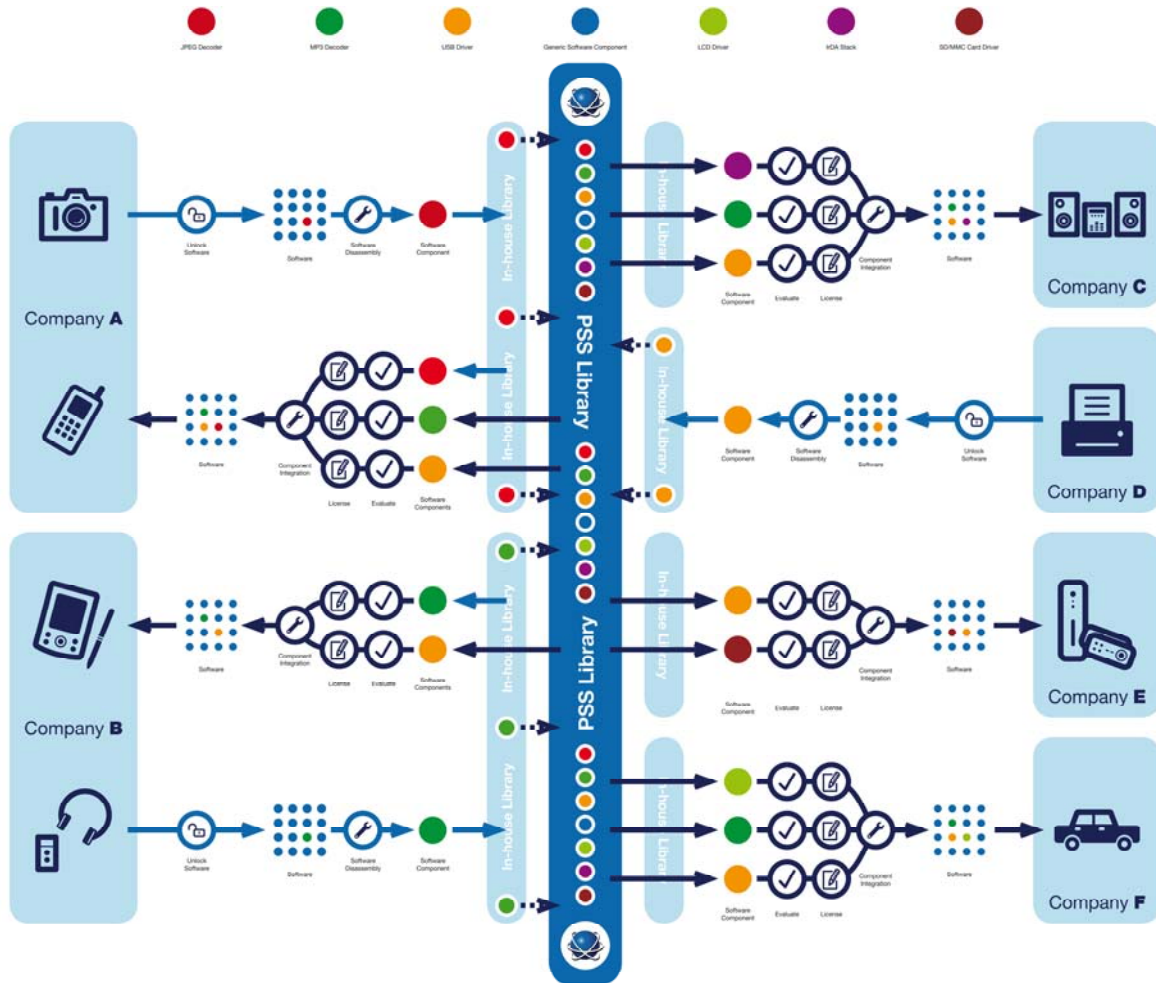
## **Introduction**

The trend in the embedded market sees software complexity rising, along with the cost and risk of developing it. This trend has resulted in over half of products missing their launch window and a quarter being cancelled altogether. Whenever this happens in any industry, market forces take over to create an alternative solution that addresses the issue. That alternative is software reuse.

Hardware design reuse is a tried and tested solution to tackling complexity and, frankly, the sheer effort involved in developing new products. Reusing hardware is commonplace today and consequently the cost of hardware design is falling in relative terms. The electronics industry recognises that the same concept for embedded software reuse needs to be established.

Like hard Intellectual Property (IP), software is a commodity that can be traded. It already happens with new software, in the form of Independent Software Vendors (ISVs), Value Added Resellers (VARs), Systems Integrators and Consultants – who all trade on their skills in creating embedded software and software development tools. In the case of System Integrators and Consultants, any IP they create will often belong to the OEM or DM who commissioned the job. Additional to that, is the wealth of embedded software that has been developed in-house, by OEMs and DMs, for specific projects. All of this software may already be reused in-house by the OEM/DM, but it could also be reused externally, by other manufacturers.

The figure below shows examples of how embedded enabling software can be isolated and turned into a component, by Proven Software Solutions (PSS), to then sit within the OEM's/DM's library for internal reuse or which could be turned into a commercial product by PSS for licensing to those that need that functionality.



The fact that most enabling software has to be developed in the first place indicates it isn't always available anywhere else, such as from a silicon supplier or OS vendor. This means if another OEM or DM using a similar platform needs a similar function, they will have to develop the software too. Clearly this isn't an efficient use of resources, but Proven Software Solutions is changing that, by providing a cost effective and resource-friendly framework for software reuse.

### Identifying software for reuse - Licensor

Elements of embedded software are either *differentiating* or *enabling*. *Differentiating* software is your 'secret sauce', the bit that sets you apart from the rest of your competition. Releasing this to the embedded community, some of whom may be in competition with you, is obviously not a good idea! *Enabling* software is the stuff that needs to be there to support your product – such as a communications protocol, device driver or peripheral interface – but on its own doesn't really add much value to, or differentiate, your end product. It is however, an asset

that, if released for licensing for a fee, could return enough to cover your original investment or even make a profit on.

Enabling software can represent a large majority of your embedded code, costing just as much to develop as your differentiating software but adding much less value to the end product. It is this software that PSS targets for reuse, where the benefits far outweigh any downside. In fact, reusing enabling software is a win-win situation for all involved.

The proliferation of freely available source code today is dwarfed by the amount owned by OEM/DMs. The ‘value’ of free code is only as good as its traceability. Our philosophy is to supply *proven* software, the definition of which is covered by four scenarios. The software must be fully developed, debugged and deployed, and the first two scenarios relate to this; the number of units or family variants produced and deployed without reported failure goes a long way towards classifying software as proven. The third scenario relates to production runs of lower volumes, but where the longevity of service without reported failure demonstrates reliability, in this case we document the number of active hours. The last scenario is relevant to code that has been developed in accordance to an industry recognised quality standard; here the actual volumes involved are less important. We have a supplementary definition for software that has been fully developed and debugged but for whatever reason has failed to reach deployment – this code is referred to as proof of concept and can also be supplied for reuse.

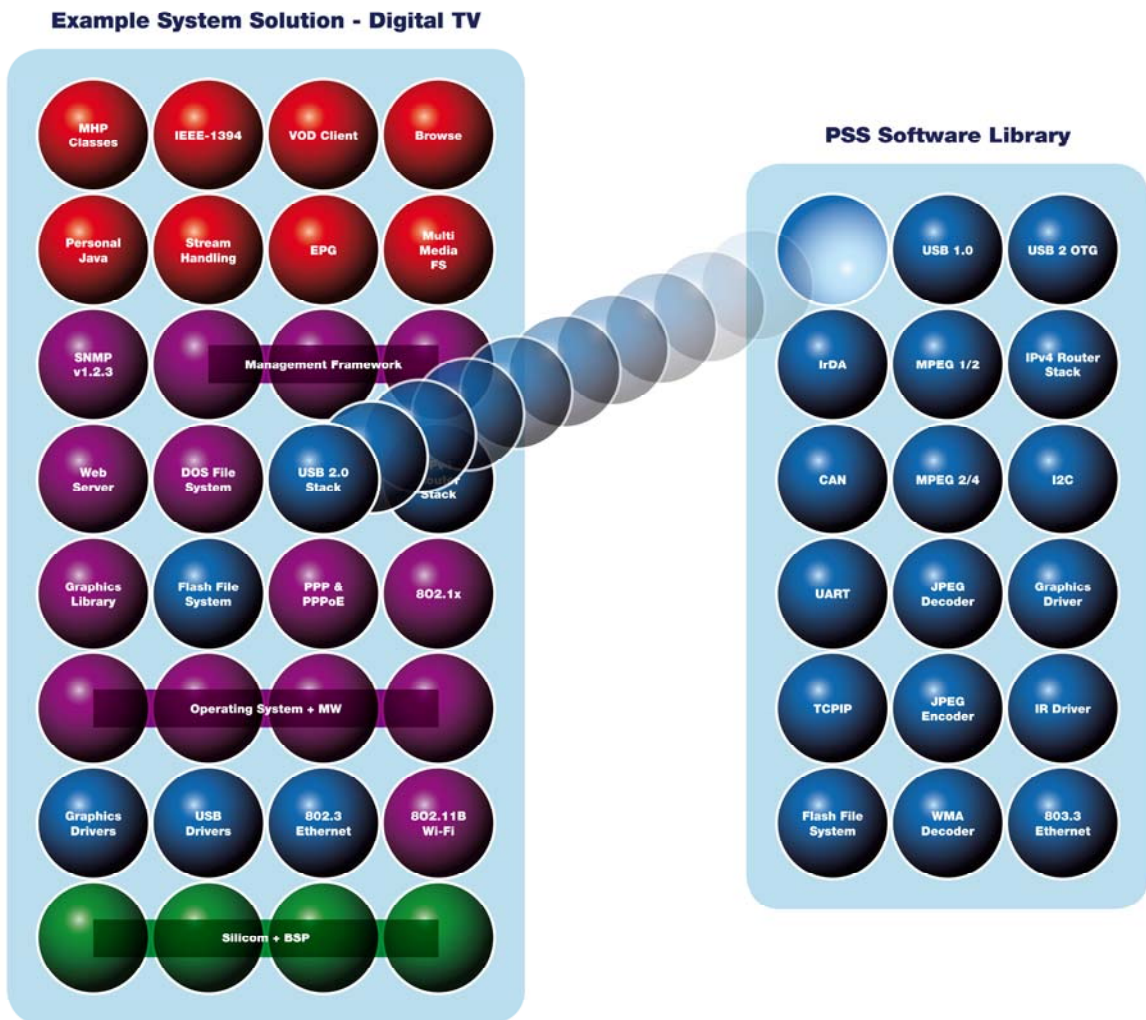
The value of proven software is clear, but as it isn’t open source, it isn’t protected by the likes of a General Public Licence (in fact, the implication of a GPL would preclude its distribution and protection under our licence agreements). To enable the distribution of proprietary software and protect the rights of the owner, the PSS Licence Agreements were created. They comprise a licence for software suppliers and a licence for software users. This way, both the owner and the licensee are protected.

### **Retargeting Embedded Software - Licensee**

All embedded software is effectively tailored to an end application, because it’s written to run on a specific processor within a family, or interface to specific off-chip devices, but this isn’t a stumbling block when it comes to software reuse. PSS believes that restricting access to the source code isn’t an option when it comes to software reuse; the licensee not only needs to see the code, but enjoy the freedom to modify it. That’s why PSS supplies embedded software as source code wherever possible.

Even if you are targeting a different platform, or your target processor is yet to be specified, you can still make significant savings by licensing proven, enabling embedded software from PSS and immediately begin integration.

The figure below shows an example of the software elements needed for a digital TV. The green represents the hardware and BSP from the silicon supplier and the purple the software potentially available from an OS vendor. The red represents the application layer where in-house development should be focussed. The blue elements, which add no value just cost, time and risk, would historically also need to be developed either in-house or by the services of a systems integrator. Instead the blue elements can be licensed from PSS and integrated into the system.



## **Adding Value for Software Engineers**

Software development is a product of the engineer's talent, experience and knowledge. There is no substitute for experience, enabled by natural talent, when it comes to developing high-quality code. But if an engineer is tasked with developing software for which he has no experience, even the best engineer faces a learning curve. If the software adds value to an end product, by introducing innovative or advanced functions, and gaining this knowledge strengthens the software team's overall capabilities, then this can be considered worthwhile. But, what if the software function required, though complex, adds little or no value to the end product? The specification would still need to be learned, the code developed, tested and debugged which could all take as long as one of those differentiating functions that does need to be developed in-house. Worst-still, what if this piece of development meant that, in order hit a product's launch window; one of those differentiating functions couldn't be included? Feature deficiencies are suffered by a third of products launched.

### **So much software...**

The amount of software available for reuse is only limited by the number of OEM/DMs developing products, while the 'shelf life' of enabling software is only limited by the availability of the architecture it runs on. For instance, the 68HC08 family of 8bit microcontrollers from Freescale has been around for tens of years and is still available. However, the pace at which end product features move means the useful shelf life of embedded software is more realistically around five years.

Europe's 4000 or so OEMs/DMs collectively develop around 1Billion lines of enabling software every year. With a shelf-life of 5 years, that means there's around 5Billion lines of code that could be reused instead of (re)developed at any given time! Releasing even a fraction of this software into the Embedded Systems market will rapidly lead to lower development costs, shorter development cycles and a higher return on investment.

Processing that much code represents PSS' investment in the concept, but we can't do it without your help. If you have enabling software – and if you're an OEM or DM then you undoubtedly do – PSS can help you turn it into a reusable commodity, which other developers like you are ready to license.

You could submit your code for reuse today, meaning it could be working harder for you tomorrow. And the more software we have to offer, the more chance of us having the software you're looking for, for your next project.

## Support

It is PSS' policy to only accept software that meets our definition of proven, but even proven software may need to be supported when being integrated in to a larger system. It is also our policy not to require software suppliers – namely, you – to provide this support, because we recognise that your primary business is developing end products, not supporting software. To overcome this problem we have implemented localised Support Networks, typically comprising Systems Integrators, whose business *is* supporting Third Party software. Under a Non Disclosure Agreement, and only with your permission, this support network will familiarise itself with the software and offer to support it on your behalf. This lifts the burden of support from your shoulders, making it even easier to submit software to PSS for reuse.

## Copyright

If you choose not to support your code and PSS handles this on your behalf through the Support Network, you may also choose to maintain a level of anonymity as a supplier. To this end, we will need to remove any comments in the software that reference you, the supplier, directly. Typically this will include removing any copyright inserted by you. Copyright in the United Kingdom is automatic, however there is no register of copyrights and it can be hard to enforce. Because PSS holds Golden copies of your software – which cannot be changed once filed – by adding our own Copyright by Proxy (example below), we ensure both your copyright and anonymity are protected.

Example Copyright by Proxy:

```
/******
```

```
Copyright 2005 Proven Software Solutions Ltd
```

```
The copyright for this software is held by Proven Software Solutions Ltd on behalf of its customer,  
customer identification number PSS_XXXX
```

```
*****/
```

## Summary

Software reuse represents an evolutionary step in the embedded electronics industry. Software complexity is doubling every two years and software development is rapidly becoming the primary cost in R&D, as well as the biggest threat to missed product launches and feature deficiencies. Reusing enabling software is the only viable solution to this challenge. Proven Software Solutions is dedicated to promoting embedded software reuse to the benefit of the electronics industry worldwide.

## Contact Details:

Proven Software Solutions is a UK Registered Company based in Aylesbury, Bucks, UK.

Mailing address:

Proven Software Solutions Ltd  
5 Weill Road  
Aylesbury  
Buckinghamshire  
HP21 9RH  
UK

Telephone: +44(0)870 950 3048

Web: [www.proven-software.com](http://www.proven-software.com)

Email: [info@proven-software.com](mailto:info@proven-software.com)