Deal with Component Shortages Using Abstraction Based Connectivity Solutions

Trish Messiter; Gokhan Tanyeri
Clarinox Technologies Pty Ltd
Melbourne, Australia
firstname@clarinox.com

***Abstract*—Supply chain disruptions caused by the pandemic have had major repercussions on the embedded systems industry. Firms have been forced to switch from one supplier to another in an attempt to fulfill their customers' orders, and engineering teams have been faced with an enormous amount of rework. This has not only required hardware redesign but has also necessitated software changes. Abstracting the connectivity components of a design from the hardware and Real Time Operating System (RTOS) layer is one method to relieve part of the rework burden. This presentation will discuss abstraction based connectivity solutions to enable the same software libraries and application layers to run across dozens of different target platforms. This approach both assists in coping with current component shortages, and provides increased flexibility for the future. Abstraction based connectivity addresses many issues related to component shortages. Can't get one microcontroller unit (MCU)? Switch to another without needing to rewrite all your software. Want another member of your product family at a different price-point? Design the new hardware with the confidence that portions of your current software can still be reused. Abstraction based solutions can enable companies to move forward with their designs in a timely and cost-effective way, despite the uncertainty of the current environment.**

***Keywords*—connectivity; Bluetooth; Wi-Fi; software architecture; abstraction**

I. Introduction

There is no doubt that over the last two and a half years, the pandemic has had a major impact upon every country, every industry and every profession. Labor shortages, freight disruption, work-from-home orders and chip shortages have challenged engineering projects everywhere. The challenge presented by COVID-19 has been compounded by political tensions and increased extreme weather events. One of the worst areas impacted has been the production of semiconductors. This shortage has often been framed within the context of the automotive industry, although its ramifications extend across the whole embedded systems industry, which touches all industry sectors.

Experts estimate that the global semiconductor shortage will extend until at least the end of this year. Fig. 1. Shows predictions for the Gartner Index of Inventory Semiconductor Supply Chain Tracking, with values <1.0 indicating a shortage. Based on these trends, Gartner predicts a return to normal inventory levels will not happen until Q3 or Q4 of 2022 [1]. Other industry leaders expect shortages to continue impacting supply chains well into 2023 [2].

Given the ongoing demand for chip-based solutions across all sectors, waiting is not an option for most companies. Engineering is about finding viable solutions when challenges present themselves. In this paper we discuss the use of software abstraction techniques to avoid platform-specific limitations and enable the reuse of existing software across platforms. We'll explore the value of this approach as a way to avoid the issues related to hardware shortages. We ground our explanation using examples from real projects.

II. Challenges Arising from Component Shortages

Geopolitical tensions had been causing concern about continuity of supply chains even prior to the pandemic.

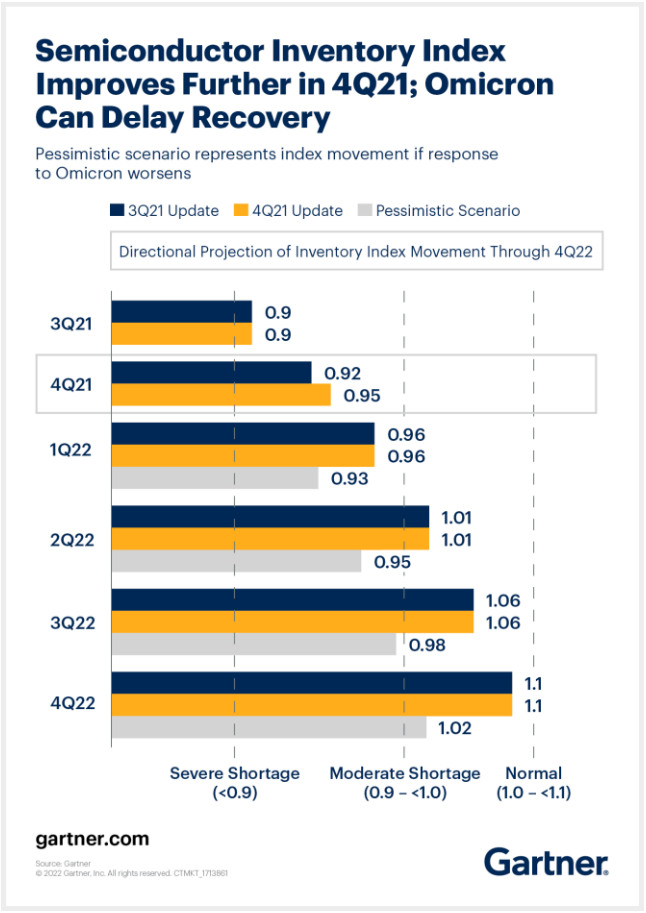

Fig. 1. Predicted duration of semiconductor shortage

However, it was during the periods of lockdowns and stay-at-home orders put in place to deal with the threat from COVID-19 that chip manufacturing was halted, leading to a reduction in supply. Simultaneously, the lockdowns fueled rising demand. This supply chain disruption has forced companies that rely on a supply of semiconductors to switch from one part to another, or one supplier to another, in an attempt to fulfill their customers' orders.

In the embedded systems industry, the flow-on effect of these shortages is an enormous amount of rework for engineering teams. This has not only required hardware redesign but has also necessitated embedded software changes. The effects are many and include:

- Extra cost for redesign, redevelopment, and recertification to comply with industry standards
- Production delays and stoppages across many industry segments due to inability to obtain required component/s
- Possibility of plant shutdowns in cases of prolonged production stoppages
- Reduced development efficiency due to not having access to required tools from home
- Slower development times due to difficulties sharing the development experience and debugging information while working remotely

A mixture of responses occurred to the shortages. Semiconductor vendors made moves to increase production to help mitigate the situation. Stockpiling occurred, either at state or company level, and limits were imposed upon exports to better respond to market volatility and political pressures [3]. These long-term measures, however, were not sufficient for a company that needs to fill customer orders in the short term. Companies have needed to come to terms with the need to change out the hardware and software components upon which their products were based. The considerable amount of work required to implement such component changes can be reduced by building in software flexibility through frameworks and abstraction from the specifics of the target platform.

III. Benefits of Abstraction Based Connectivity

Implementing abstraction based solutions is an efficient way of meeting the challenges of component shortages. Abstracting all or parts of the system software, such as the connectivity components from the hardware and Real Time Operating System (RTOS) layer, allows current software to be reused in the event of a component shortage, and relieves part of the rework burden.

Reuse in software is an area much talked about but well under achieved, and using abstraction is an effective way to ensure that software can be reused when adapting to new situations. Whether due to supply chain disruptions, or simply demand for another member of a product family, the same software libraries and application layers can run across dozens of different target platforms without the need for a full porting effort. This in combination with a reduced recertification workload allows project timelines to be kept to a minimum, enabling companies to move forward with their designs in a timely and cost-effective way, despite unforeseen circumstances.

IV. Abstraction Based Connectivity in Detail

Many factors must be considered when implementing an abstraction based approach. These include abstraction of processor architecture, microcontroller/microprocessor unit (MCU/MPU), OS/RTOS, Bluetooth or Wi-Fi chipset, hardware interface, TCP/IP network stack, permanent storage and debugging software.

To illustrate how an abstraction based connectivity solution may be implemented, we use the example of ClarinoxSoftFrame, developed by Clarinox. This software is designed to allow various levels of abstraction, providing sufficient internal functions to enable deterministic behaviour across a wide range of possible components. A block diagram is provided in Fig. 2, showing the connectivity software and the connection to other elements of the software architecture.

A. *Semiconductor Abstraction*

When dealing with semiconductor shortages, it is important to consider incorporating flexibility in processor architecture and MCU/MPU. This flexibility allows engineers to reuse portions of existing software, making the change from one supported semiconductor to another a significantly reduced effort as compared to a traditional porting. By using a modular approach, new chipsets can be supported by the abstraction layer quickly and efficiently while keeping sufficient functionality within the framework. The bulk of the engineering time can then be spent on testing and performance tuning.

Supported processor architectures and MCU/MPUs for the example of ClarinoxSoftFrame are shown in Tables I and II.

TABLE I. Processor Architectures Supported by ClarinoxSoftFrame

CPU Architectures	Variant
ARM	7/9/11, Cortex-M, Cortex-R, Cortex-A (32Bit/64 Bit)
Analog Devices Blackfin	ADSP-x
Intel x86	All x86
Infineon Aurix	TC2xx/TC3xx
MIPS	FPGA based
PowerPC	FPGA based
Renesas RH850	RH850
RISC-V	FPGA based
SPARC-LEON	FPGA based

TABLE II. MCU/MPUs Supported by ClarinoxSoftFrame

Semiconductor Vendor	Chips
Atmel/Microchip	SAM 4Sxx
Analog Devices	Blackfin ADxxx
Dialog Semiconductor	DA14195/DA14196
Infineon	XMC 4700/AurixTC2xx/TC3xx
Intel	Pentium, Atom, Quark
NXP	LPC18xx/43xx/54xxx; i.MX6, i.MX8, i.MX28, i.MX31; i.MX RT 1xxx, Kinetis K6x / K7x
Renesas	Synergy S5/S7, SHx, R-Car M3/H3, RH850, RA, RE
ST Microelectronics	STM32F4xx, STM32WB55, STM32MP157, STM32H7xx, STM32F7xx, STM32L5xx, STM32U5xx
TI	DSP 5xxx, OMAP, Sitara 3xxx, Tiva TM4C12x, MSP432

B. *Operating System Abstraction*

Component shortages may also necessitate a change in the OS/RTOS. Again, abstraction of this layer significantly increases productivity by removing the need to perform porting when changing OS/RTOS. This OS/RTOS abstraction can be achieved through a combination of several mechanisms.

Many operating systems differ in their scheduling algorithms and the features that they provide. To avoid behaviour differences due to underlying OS/RTOS changes, an abstraction based architecture needs to enable a smooth transition from one OS to other. This can be done by controlling the scheduling instead of relying on the underlying RTOS. The behaviour of the application therefore will not be affected by use of a different RTOS or other parameters such as processor speed.

Another source of behaviour differences may be too many RTOS thread switching events. High-speed connectivity applications will be especially affected by such a design. To avoid too many thread switching events during a connectivity stack receiving packets from the physical interface, passing them to various protocol levels and then eventually to the user application, a cooperative multitasking architecture will help the processor to allocate resources to this connectivity stack in an uninterrupted manner.

The RTOS can be used to separate user application from the connectivity stack and physical interface drivers. Connectivity data packets will be passed from application to connectivity stack, and then to physical layer or vice versa. If the connectivity stack has multiple layers, cooperative tasking avoids excessive thread switches which would reduce performance.

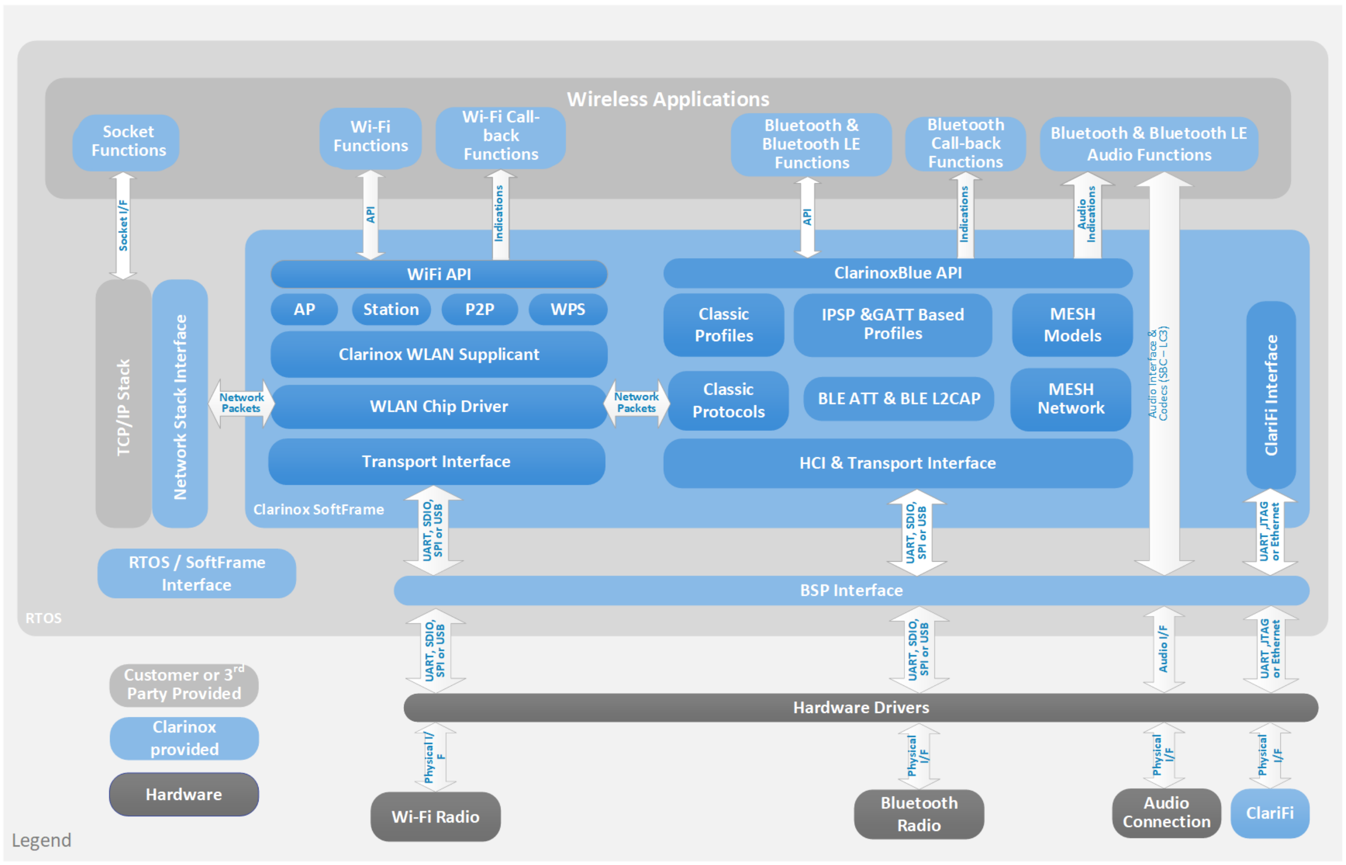

Fig. 2. Architecture of connectivity solution within abstraction based architecture

Table III shows the current OS/RTOS options that are supported by the ClarinoxSoftFrame architecture. Changing from one RTOS to another can be performed without the need to change the wireless protocol stack, wireless application, or functionality of the end product.

C. Wireless Chipset Abstraction

In wireless systems, the wireless chipset is also a major factor. Similar to semiconductor and operating system abstraction, abstraction at this level can facilitate faster changeover should changes need to be made to the wireless chipset. To achieve such abstraction, an intermediate board support package layer needs to be used. Such a layer can be implemented by using a well-defined lightweight interface architecture.

Table IV lists ClarinoxSoftFrame's currently supported wireless chipsets, exemplifying the wide range of Bluetooth and Wi-Fi chipsets that may be substituted should such requirements arise.

D. Hardware Interface Abstraction

Various hardware interfaces can be used to interface to the connectivity devices. Wi-Fi generally uses SDIO, PCIE, USB or SPI interfaces, while Bluetooth more commonly uses UART or USB (but SDIO or SPI are used by some designs). ClarinoxSoftFrame accounts for all these possible interfaces, as shown in Fig. 2. Abstraction at this level provides flexibility should the hardware need to change in projects impacted by chipset shortages.

E. TCP/IP Network Stack Abstraction

The TCP/IP network stack is an important consideration for today's connected embedded devices and there are many open source and commercial choices. In some cases, the TCP/IP stack is provided as part of the RTOS, as in the example of Microsoft providing NetX/NetX-Duo with AzureRTOS (formerly ThreadX RTOS).

Table V shows the TCP/IP stacks supported by ClarinoxSoftFrame architecture. Creating an abstraction for this layer facilitates a simpler process when change is required and therefore assists to prolong product life cycle.

TABLE III. OS/RTOS SUPPORTED BY CLARINOXSOFTFRAME

OS	RTOS
Android	AutoSAR
Automotive Grade Linux (AGL)	CMSIS & RTX
Linux (Ubuntu etc.)	embOS /MQX
Mentor Graphics Linux	FreeRTOS
SYSGO Embedded Linux ELinOS	Nucleus INTEGRITY
Wind River Linux	QNX
Windows 8.1/10/11	ThreadX / Microsoft Azure RTOS
Yocto Linux	TI-RTOS / eCos
	uCos-II/III
	Wind River VxWorks
	Windows CE/Mobile

TABLE IV. WIRELESS CHIPSETS SUPPORTED BY CLARINOXSOFTFRAME

Bluetooth Chipset	Wi-Fi Chipset
Ceva	NXP 88W88xx, 88W89xx, 88Q9098
Cypress	Realtek RTL8723, RTL8821, RTL8822
Dialog Semi	Texas Instruments WL6, WL7, WL8
Marvell	
MediaTek	
Nordic nRF52840, nRF52832	
Qualcomm / CSR	
Realtek	
ST Micro STM32WB55	
Synopsis	
Texas Instruments	
Any HCI Interface SoC	

F. Permanent Storage Abstraction

Another abstraction that is essential is that of the permanent storage. Whether a file system or Flash/EEPROM memory interface is used, an abstraction mechanism avoids application rewrites when a change is necessitated. A connectivity stack may require storage of pairing and bonding information or configuration data. This level of abstraction needs the connectivity stack to utilize generic memory access functions. In the user wireless application, these memory access functions are translated into one of the above-mentioned permanent storage methods.

G. Debugger Abstraction

Once the framework elements are in place, abstracting the debugging mechanism can provide a final layer of flexibility.

Due to connected applications not lending themselves well to static debugging (i.e. stop and step though), relying on the Integrated Development Environment (IDE) to debug such complex products frequently results in overlooked issues and schedule slippage. The ClarinoxSoftFrame abstraction layer includes a debugger, ClariFi, which gives a clear real-time view into the connectivity layer, including protocol specifics, memory state, application fatal errors, warnings and test messages. It also allows for comprehensive postmortem analysis via detailed log files, as well as complex filtering, performance analysis, and automated testing, all of which help to keep development on schedule.

One main aspect of debug abstraction is hardware interface abstraction. By providing a consistent interface for automated testing, workload is reduced when changes are required. For example, ClariFi allows Ethernet, UART or JTAG to be used for connecting the target platform for debugging, and abstracts debug messaging from OS/RTOS, MPU/MCU and hardware interfaces. Such flexibility can be a huge benefit in certification testing, allowing for efficient recertification, and a reduction in the overall product development cycle compared to IDE debugging alone, even in the case of a chipset change.

TABLE V. TCP/IP NETWORK STACKS SUPPORTED BY CLARINOXSOFTFRAME

TCP/IP Stack
Android
AutoSAR TCP/IP
HCC TCP/IP
Linux TCP/IP
LWIP
NDK TCP/IP
NetX/Netx-Duo
NORTi uITRON TCP/IP
QNX
MQX RTCS
Segger emNet
VxWorks
Microsoft Windows 8.1/10/11

V. CASE STUDIES IN ABSTRACTION BASED CONNECTIVITY

Abstraction based solutions have been applied to assist customers in the current crisis from a variety of industries including home automation, radio communications and office equipment sectors. Here we look at several cases that have benefitted from abstraction based approaches.

A long-standing vendor within the radio communications space was disrupted by the components shortages and struggled to obtain sufficient components to meet customer orders. A decision was made in early 2021 to move all products to a target agnostic approach to provide certainty that platform changes could be handled with minimum impact when needed. This move was made by the end of 2021, protecting these products from future uncertainty.

In mid-2021 a major player in the office equipment sector was also affected by the semiconductor shortages. The choice was either to not meet product demand or change both MPU and wireless chipset to enable sufficient production. The inability to get sufficient chipset supply to fulfill customer orders forced the need to change chipset vendors. All embedded software needed to be rewritten which introduced a time lag, but the use of the abstraction based approach minimized the effort. Going forward with the abstraction based approach will prevent this situation recurring.

Even without the current component shortages, the abstraction based software approach is beneficial to enable the maintenance of a single application layer across product family members. In 2015, a Fortune 100 corporation in the home automation industry realized that the maintenance load of different software across the different targets used for the one product family was causing inefficiencies. A decision was made to consolidate the connectivity component of all family members to abstraction based solutions. This relieved the maintenance workload and enabled the company to move forward with their designs with the knowledge that the software could be reused into the future as hardware was upgraded. Additionally, such an architecture gives a longer life cycle to any product by allowing MCU/MPU, wireless hardware and RTOS changes.

The presentation of this paper will include a discussion of specific customer examples.

VI. CONCLUSION

Abstraction based approaches to software design are particularly relevant in the context of component shortages, which have had a widespread impact on embedded software development over the past two years. This approach aligns with the golden rule of software engineering – reuse. Abstraction provides the ability to reuse current software when needing to switch components, minimizing the redesign workload and keeping projects on schedule.

Multiple layers of hardware and software based abstraction are advisable, including abstraction of the MCU/MPU, Bluetooth or Wi-Fi chipset, OS/RTOS, TCP/IP network stack, permanent storage and debugging software. ClarinoxSoftFrame is an example of a software package that facilitates implementation of abstraction at these various levels.

Despite the challenges presented by component shortages, there are many examples of projects that have successfully adapted to this situation using abstraction based models. Engineers should be aware of these options at the beginning of the development process to avoid the inconvenience of reworking entire projects when forced to switch platforms due to supply disruptions.

REFERENCES

[1] K.C. Quah, 'What is ahead for semiconductor shortages,' *Gartner*. [Online]. Available: https://www.gartner.com.au/en/articles/what-is-ahead-for-semiconductor-shortages

[2] 'The chip shortage: current challenges, predictions, and potential solutions,' *FS Community.* [Online]. Available: https://community.fs.com/blog/the-chip-shortage-current-challenges-predictions-and-potential-solutions.html

[3] Bloomberg, 'China stockpiles chips, chip-making machines to resist U.S.,' *Yahoo.* [Online] Available: https://au.finance.yahoo.com/news/china-stockpiles-chips-chip-making-210000407.html