Communicating Efficiently between QorIQ Cores in Medical Applications

Mentor Graphics Contributed Article

Introduction

Multicore devices are now widespread and are used in many different market areas. They are not new, of course—they have been around for decades as single cores per board which were then put into racks, through multiple individual cores on a board, to multiple cores on a die. What has changed in recent years is the rise—and tight integration—of homogeneous general purpose processors. Homogeneous systems are ones where identical processors are implemented. With the Freescale QorIQ P series of processors, ranging from the single e500 core P1010 to eight e500 cores in the P4080, there is a vast range of programming methods.

This article will look at different aspects of programming the multicore versions of the QorIQ processor family with specific emphasis on asymmetric multiprocessing (AMP), and present a few examples of where this can be a hard requirement with the P1022 processor as an example.

System Software

Many system designers will use these processors in symmetric multiprocessing (SMP) mode. In an SMP system, one copy of a single operating system (OS) runs across multiple identical cores. Application threads are scheduled across these cores by the OS to make the best use of available horsepower.

In an AMP system, each core executes its own instance of an operating system as if it were running in a single-core environment.

A system can be a combination of SMP and AMP, executing the same instance of an operating system across some of the multiple cores, and a separate instance of one or more operating systems on additional cores. Such a hybrid system is beyond the scope of this article.

Within a multicore system, the processor cores usually communicate with each other and pass data back and forth. This is done using some form of inter process communication (IPC).

Symmetric Multiprocessing (SMP)

Mentor® Embedded Linux® and all QorIQ multicore products support SMP mode. This software reference architecture from Mentor Graphics is one of the most basic software architectures for Linux OS. Mentor Embedded Linux programmers can, with simple configuration files, instantly utilize an effective SMP Linux configuration which will utilize the entire CPU in the Freescale system-on-chip (SoC) with a single operating system running across the cores. Unless there are specific affinity requirements within the programmers view, the operating system takes care of all the housekeeping tasks and allocates individual tasks and threads to an appropriate core during run time. SMP systems are used to boost the processing power of a system and also have the flexibility to load share across the processors, allowing maximum advantage of the power-saving features to be utilized when appropriate. Figure 1 shows a simple diagram of an SMP system.

Asymmetric Multiprocessing (AMP)

In an AMP system, each core executes its own instance of an operating system as if it were running within a single core environment. The cores may be homogeneous (as in the case of P1022), but an AMP system can equally cope with heterogeneous processors where the processor architectures may be diverse to meet the demands of the overall system. An example of this is the QorIQ P1022 processor running Android™ OS on one core and Mentor Embedded Linux OS on another core. Freescale's QorIQ dual- and multicore processors all support this mode of operation.

Figure 1: A Simple SMP System

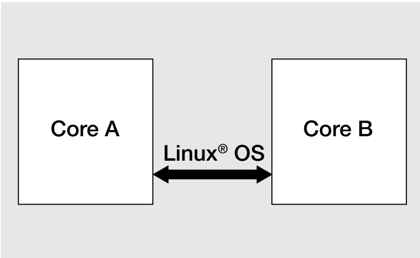
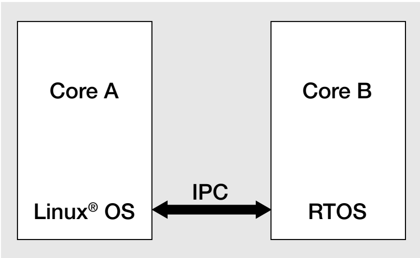


Figure 2: A Simple AMP System



The choice of OS for each core is determined by the function and performance that each core has to perform. It's highly likely that multiple operating systems may be selected. There could be a requirement for rich OS on one core and OS with hard real time and determinism capability on another core. Security and certification also play a part in which OS runs on which core. Figure 2 shows a simple diagram of an AMP system with IPC between core A and core B.

Inter Process Communication (IPC)

For SMP systems, IPC is handled by the OS. In AMP, although each core executes independently, at some point applications running on the different cores will need to exchange data. An example of this scenario is a system in which one core executes Linux OS on the control plane to provide a sophisticated user interface (UI) to capture user input while the other core executes an embedded RTOS on the data plane for deterministic activities. The Linux core passes user input to the RTOS core for processing, which then passes back the result of some execution. It is crucial to select an IPC suitable for the final system design that is supported by each OS.

The IPC, therefore, should be able to operate on different cores and also on different operating systems. There are multiple ways of implementing IPC. Roll your own with shared memory is popular so that the IPC exactly matches both the hardware and software requirements of the system, but this can suffer from scalability limitations. TIPC, Linx, TCP/IP and RPC are other methods which have their pros and cons, however here we will look in a bit more detail at multicore communications API (MCAPI). This is an API defined by a consortium of interested parties including silicon vendors such as Freescale, and software vendors.

Multicore Communications API (MCAPI)

First released in 2008, MCAPI was developed by a group of industry professionals under the umbrella of The Multicore Association™. MCAPI is not a protocol, but an API only. It was designed for IPC within tightly coupled systems as a lightweight alternative to more complex solutions. MCAPI partitions communicating units into nodes and further groups nodes into domains. The definition of a node and scope of a domain are implementation-defined to provide greater flexibility to the system designer. Nodes and domains are assigned unique identifiers by the application at initialization. Nodes communicate across ports, using handles called endpoints which are assigned by MCAPI, and are unique to the node. Endpoints are addressed hierarchically via <domain, node, port>.

MCAPI offers a lot of flexibility for the transfer of data. Broadly, there are three options:

- Messages are datagrams—chunks of data—sent from one endpoint to another. No connection needs to be established to send a message. This is the most flexible form of communication, like User Datagram Protocol (UDP) in networking, where senders and receivers may be changing along with priorities.
- A packet channel is a first-in, first-out, unidirectional stream of variable-sized data packets, sent from one endpoint to another, after a connection has been established.
- A scalar channel is similar to a packet channel, but processes single words of data, where a word may be 8, 16, 32, or 64 bits of data.

MCAPI does not define the protocol, this is left to the implementer. As a result, there is no expectation that one vendor's MCAPI implementation will interoperate with another's, even if they both fully comply with the MCAPI specification. To address this issue, OpenMCAPI was created. This is a complete, open source implementation of MCAPI for Linux OS, which is designed for easy portability to other systems. It may be obtained from **openmcapi.org** at no charge.

Figure 3: Aggregator for Medical Applications Using the QorIQ P1022 Processor

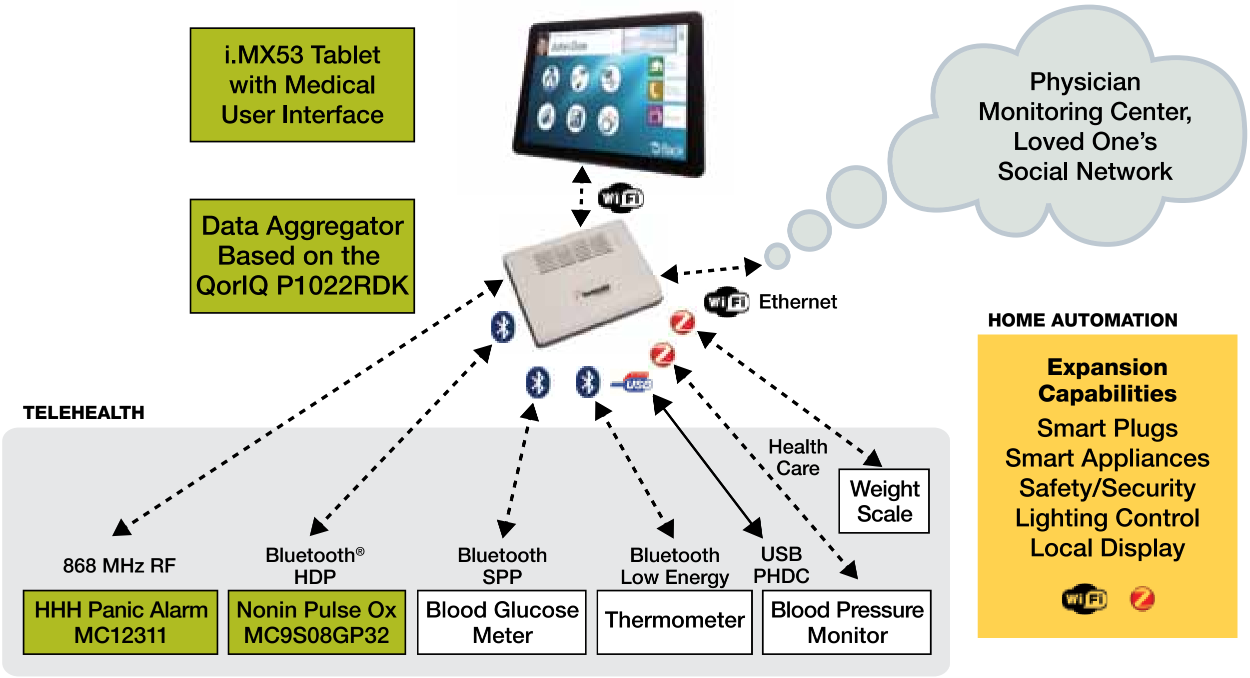
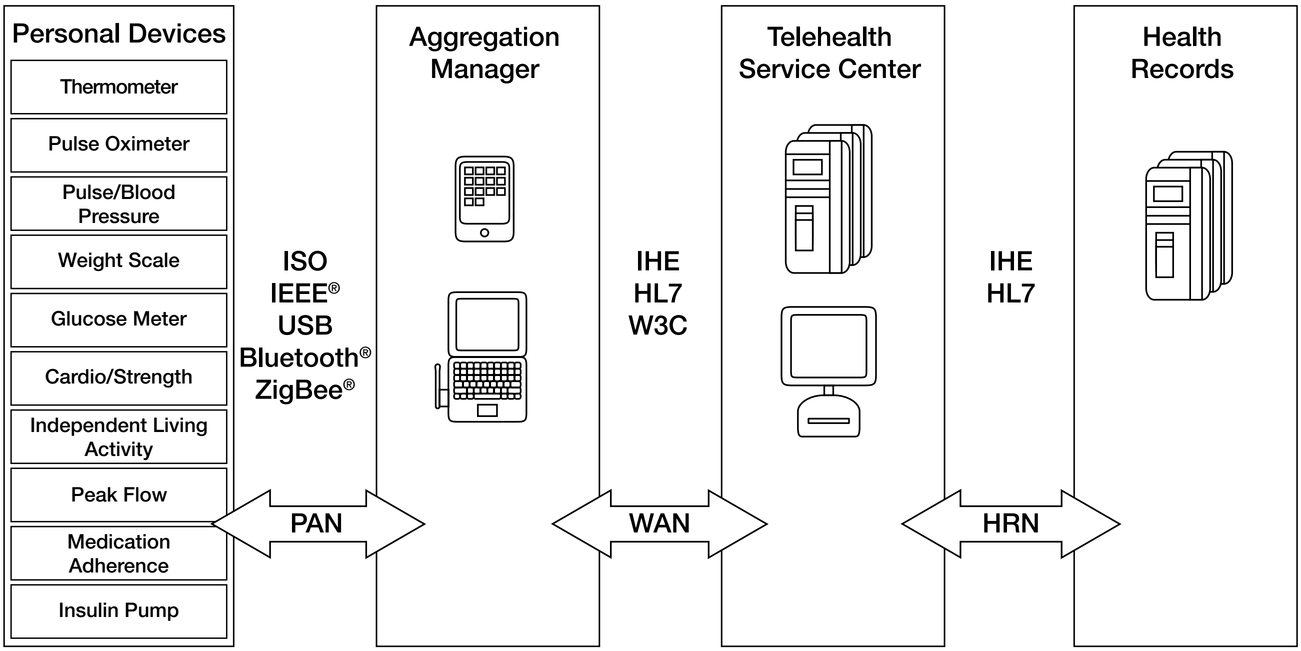


Figure 4: Example Implementation in AMP Configuration



Example Implementation: Aggregator for Medical Applications

There is an emerging breed of devices in the medical and telehealth markets where there are diverse requirements put upon the system, especially from a software point of view. There are two primary use cases for data aggregators in the medical market. One use case is the clinical or hospital setting, where one might aggregate multiple patient statistics reads to a central location. The other primary use case is the home telehealth gateway, which provides remotely monitored statistics about a patient from the comfort of their home to the primary caretaker or potentially to a doctor's office.

Figure 3 shows a data aggregator without a UI in an AMP configuration using the QorIQ P1022 RDK. This example depicts a hospital-like environment where no patient interaction is required.

Figure 4 shows the Continua Health Alliance view of this area where the aggregation manager is a device with a graphical user interface (GUI) that gathers information from sensors such as weighing scales, glucose meters, etc., stores it and then makes it available on request to a doctor, service provider or the actual user. The home-based aggregators are becoming more like full-featured smart phones from a user perspective, simple enough to be used by anyone, including those with disabilities or even those who are not technically inclined. The device should not appear as a medical device, but rather as a useful tool to be used in other aspects of one's life. Consequently, such a device would have an OS like Linux or Android running on it. A simple-to-navigate GUI based on Android can provide the ease of use needed for this telehealth gateway.

Since vital information is shared over IP networks, security within the system is critical. Freescale's QorIQ processors, along with Mentor Embedded Linux OS, provide a secure connection through which patient statistics can be transmitted. The addition of trust architecture to the QorIQ platforms protects the aggregators from attack or tamper.

Using a dual OS system provides this separation. An open OS is used for the user interface and also the connection to the external network through Wi-Fi®, cable, etc. The other OS needs the size kept to a minimum to make certification through FDA easier and therefore less costly. Security and integrity of data can be guaranteed by having the right IPC communication defined across the OS interfaces.

There are many other diverse application areas, such as printing for example, where the same logic is applicable.

Dual OS systems can be implemented on a single processor core using virtualization. However, using a dual-core device such as the P1022 delivers an optimized approach.

Conclusion

For many applications, it is useful to select a different operating system for each core, depending on the functions that the core is performing. If it is real time, then a conventional RTOS makes sense; for other purposes, Linux or Android may be a good choice. Example applications, where a multi-OS approach is optimal, include numerous medical applications as well as high performance network printers. These systems are characterized by their need for real-time functionality, a user interface and extensive networking capabilities.

The software architecture may be SMP, where a single OS is run across all the cores, or AMP, where each core runs its own OS. When AMP is employed, a key issue is inter-core communication and MCAPI provides a proven, standardized method to address this matter. In either case, Freescale QorIQ devices offer great flexibility to the hardware and software developer.

How to Reach Us:

Home Page:
freescale.com

Power Architecture Portfolio Information:
freescale.com/power

e-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675 2150
LDCForFreescaleSemiconductor@hibbertgroup.com

For more information, visit freescale.com/power

Document Number: PWRARBYNDBITSCE REV 0