Implementing Bluetooth LE data pumps

Author(s)	George Charkoftakis, Panagiotis Sifniadis
Revision	**1.0**
Release Date	**30/10/2019**

Dialog Semiconductor

Achilleos 8 and Lamprou Katsoni,
17674, Kallithea, Athens
Greece

Phone: +30 210 93 10 580
Fax: +30 210 93 10 581

Neue Straße 95
73230 Kirchheim/Teck
Germany

Phone: +49 7021 805-0
Fax: +49 7021 805-100

Preface

This document describes a specific group of Bluetooth® Low Energy (LE) use cases, referred to as data pump applications, in which an external microcontroller pushes and pulls data through a BLE-capable device. The key requirement of such applications is to “pump data” through the Bluetooth LE wireless link, independent of the type of data, the profile used or the end application.

In the paper, we present different possibilities for interfacing to the different levels of the Bluetooth software stack. Each design choice for the data pump architecture has a clear set of benefits and disadvantages ranging from the purely technical, such as speed or power consumption, to business-related topics including cost, maintainability and time to market. We address these aspects and inform readers how to make the right choice for their application.

The remainder of the document covers the applicability of Dialog’s DA14531 device for implementing data pump applications with specific use cases in mind. The DA14531 is a cost-optimized solution that can, among other things, be used as a fully configurable BLE Host Controller Interface (HCI) or to run applications implementing data agnostic “wireless serial” ports.

Data pump applications

Figure 1 shows a simplified picture of the generic data pump architecture. It consists of two components, the BLE-capable subsystem and the high-level application subsystem. We will refer to the former as the controller and the latter as the host. Depending on the parts of the Bluetooth LE stack implemented in each subsystem, several configurations are possible as will be shown later in this document.

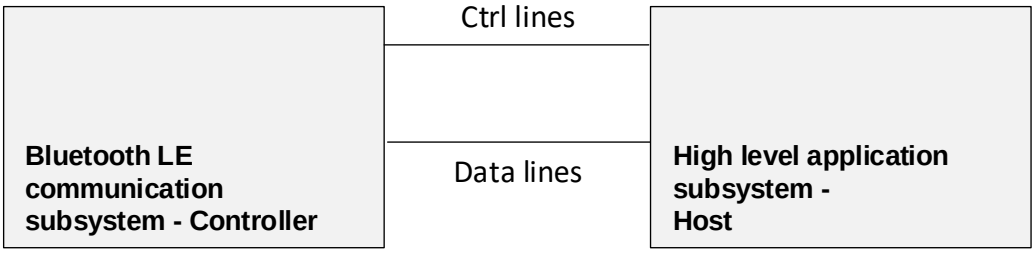

Figure 1 Generic Bluetooth LE data pump architecture.

The two subsystems share data lines to exchange information and control lines to implement functionality such as flow control, sleep control, indication of events, reset and even power gating of the controller. The implementation of the data interface is usually a standard serial protocol such as UART, SPI or I²C. It may follow a standard implementation, as in the case of the UART HCI protocol, or a custom implementation to either save pins (I²C) or increase the link speed (SPI). The rest of the control-related features are implemented using additional general-purpose input / outputs (GPIOs). It is also possible to multiplex part of the control requirements with the data lines or to use in-band control signaling. The right choice will usually depend on the available GPIOs in the controller and host as well as the available hardware interfaces and their capabilities.

The system architecture shown in Figure 1 is quite generic. It exists in most embedded systems and engineers are already trained to understand it. This makes it a good starting point to help in migrating existing non-BLE systems into the Bluetooth world. The Bluetooth LE subsystem becomes a peripheral of the application subsystem which can be interfaced to the application using standard serial interfaces. The Bluetooth LE standard defines one such design option: the HCI. This is the most powerful version of the architecture providing the application subsystem the greatest control over the Bluetooth LE controller. However, such a design option cannot universally be considered the best choice. The learning curve to start working with the HCI is not negligible and Bluetooth LE expertise is essential. An HCI solution will also require significant memory and CPU resources by the application subsystem. Can we simplify this approach? Yes, and many different solutions have been developed to address this problem.

In the following section, we will look at the requirements and challenges when building a system using the data pump approach.

Interface between the host and the controller

The UART is the most commonly used interface to connect a Bluetooth LE data pump controller and its host. It has the advantage of being full-duplex and requiring only two lines in its simplest form. In principle, communication can be initiated by both devices at any time. Data rates up to 921 kbps can be achieved. UART interfacing is considered a standard for the Bluetooth LE HCI implementation and for the different data pump architectures we will examine later. One major drawback is that the UART is only used between two devices and does not inheritably support a bus architecture. It is also an expensive resource in the sense that there are never enough UART interfaces available in a system.

Another candidate is the SPI. It is also full-duplex but can achieve higher data throughput in excess of 20 Mbps and does support a bus architecture. It requires at least four signals. However, the bus capability introduces a certain drawback – all transmissions are controlled and initiated by an SPI master on the bus. The SPI slave cannot initiate a transaction and must request the master to do so. The SPI master grants the SPI slave request and reads the data. Hence a protocol with handshaking and priority is required. Handling full-duplex communication over such a protocol is complex and usually the actual data transfer is half-duplex, in the sense that only one of the two devices will transfer useful information over the SPI link at any given point of time.

The third commonly used synchronous communication interface is I²C. Its main benefits are that it needs only two signals to be shared between the communicating devices and that it can form a bus with in-band addressing. The maximum data rate ranges from 100 kbps to 1 Mbps. I²C is strictly half-duplex and suffers from the same drawback as the SPI; the slave cannot initiate a transmission, therefore a protocol with handshaking is required.

Low power management

Applications that are battery-powered necessitate power consumption optimization. A power-sensitive application requires that both parts of the system (the host and the controller) can enter a lower power state, usually referred to as sleep, whenever possible. A Bluetooth LE data pump scenario is no exception to this rule and must seamlessly support sleep management. In a scenario where a host-controller pair exchanges data, the controller must be able to wake up and notify the host that data is available for processing and vice versa. The controller and host should both be able to buffer small amounts of data to compensate for the time it takes the other to wake up. Systems with long periods of inactivity may also require the ability to put the controller into a very low power mode or even power gating of the controller when the communication interface is not used. Sleep management is implemented using dedicated GPIO pins or multiplexed with the flow control or data line pins.

Bandwidth

The data pump architecture can be used to serve different data throughput requirements: from occasionally transferring small data chunks to continuously streaming large amounts of data. The data pump must be able to handle all these scenarios. In the case of continuous data streaming, a dedicated binary mode should be available to allow transparent transfer of raw data through the common interface at the highest possible rate, without delays or overhead from headers and in-band signaling.

On the other hand, for sporadic data transfer, the protocol used could support any notification mechanism with headers to transport data as part of a message. In this case, the handshaking mechanism and headers will impose only minimal burden on the system.

Power supply options

Above, we briefly discussed power consumption and sleep management. Designers must also consider the power supply itself when selecting a controller – especially if it is intended to be integrated into existing designs. An existing host subsystem can be powered from a variety of energy sources including batteries of different nominal voltages and capacities, solar panels or the grid. A controller should be able to work from a variety of voltages that can be typically available in modern microcontroller subsystem.

Additional requirements

There are several other parameters to consider. Below is a short list of requests we have seen that might influence the selection of a data pump controller.

- *Size:* Embedded systems often have significant size constraints. The small size and number of components of a modern Bluetooth LE data pump controller facilitates easy interfacing even in highly space-constrained designs.
- *Boot latency:* The boot time is the time required for the controller to become accessible to the external host after a reset. It may directly affect overall power consumption and responsiveness.
- *Transmit output power control*: This is required to fine tune the power consumption or implement near-field mode functionality.
- *Maximum output power*: To increase the radio link throughput especially when streaming.
- *Cost*: Cost is always an important consideration. The lower the overall cost of a data pump controller the larger the group of existing non-connected applications that may decide to join the connected world.

Implementing data pumps over Bluetooth LE

Before describing the different flavors of Bluetooth LE data pump architecture, let's review the Bluetooth stack architecture (Figure 2). The Bluetooth LE stack consists of several layers. The lower layers include the radio, the baseband and the link layers. These handle the basic operations of the radio, the scheduling of radio events, the establishment and management of the connection and the transfer of data.

The standard protocol used to interface the lower layers and the higher layers of the stack is the HCI protocol. This standard allows the host stack or controller IC to be replaced with minimal modifications if any.

Above the HCI, the information is split into two distinct paths.

a) The orange path is mostly concerned with data-driven operations and includes the L2CAP, ATT and GATT layers. It may also include the implementation of the standard profile and services which sit on top of the GATT layer.
b) The blue path is concerned mostly with the control of the connection and radio operations. The GAPM layer handles most of these operations, while the SM layer is mainly responsible for security-related operations.

Control is provided to the application layer through the GAP-available APIs and data through the available profiles and services or directly by calling GATT APIs.

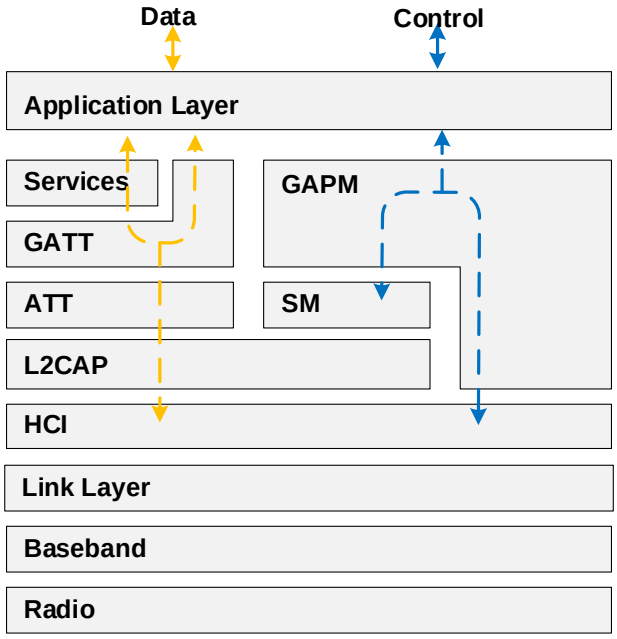

Figure 2 The Bluetooth LE stack

Data pump variations

Figure 3 shows four possible ways to divide the Bluetooth LE stack to end up with a host-controller configuration. In each case, a cut (Cx) splits the Bluetooth LE stack into a lower part (the controller) and an upper part (the host). The data pump basically serializes the messages that go through the cut to deliver them to the host which is usually located an external subsystem. The functionality of the layers missing from the Bluetooth LE stack must be implemented in the host device.

The different flavors of data pump architecture, together with some basic properties and the protocol used, are summarized in Table 1.

Table 1 The different flavors of the data pump architecture

Cut	Communication protocol	Control over the controller	Comments
C1	HCI	Full	Standard HCI host-controller implementation.
C2	DGTL	Full	Dialog GTL host-controller implementation. Serializes all the stack messages intended for the application layer and vice versa.
C3	DGTL – hybrid	Selected	Dialog GTL host-controller implementation with ability to select which application layer functionality is implemented in the controller and which is exposed to the host.
C4	Application defined protocol.	Limited	Complete application running on the controller providing limited (i.e. CodeLess AT) or no control over the Bluetooth link (i.e. DSPS) and exposing a data pipe to the external host.

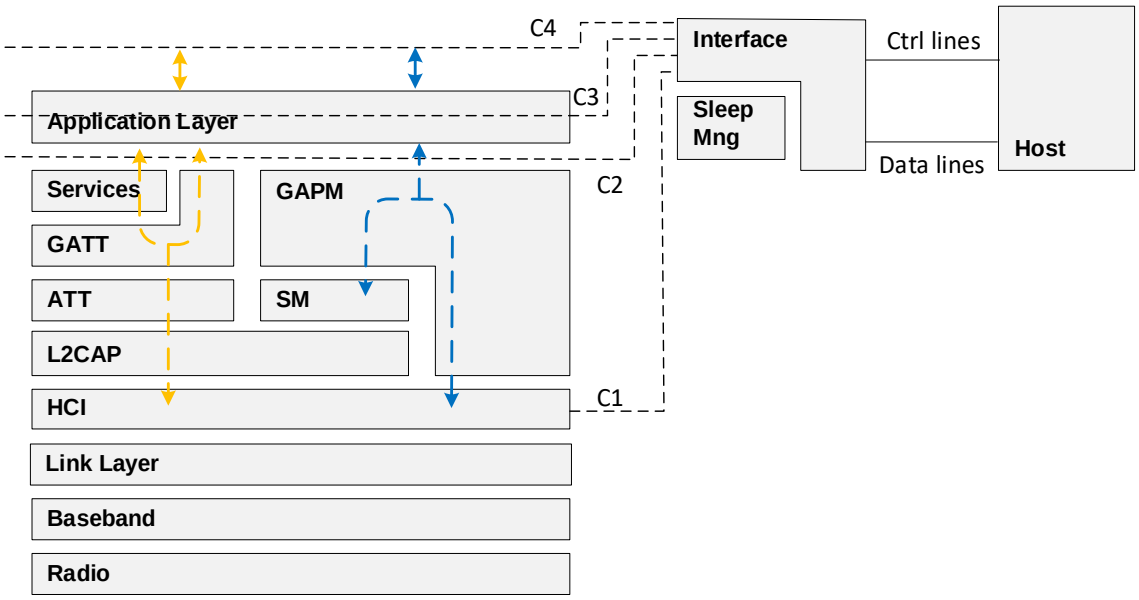

Figure 3 Possible controller - host configurations

HCI host-controller architecture

The C1 cut above gives rise to an HCI configuration. This is a Bluetooth® SIG standard protocol, supported by a wide number of vendors. The Dialog HCI supports a standard UART implementation but also provides proven solutions where the HCI messages have been serialized over proprietary protocols using I²C or SPI.

An HCI controller implementation is based on a Bluetooth SIG-certified controller. This allows developers to select a Bluetooth LE host stack from any of a wide number of vendors and still ensure a high level of interoperability.

The host has full control over the capabilities of the controller with no limitation. Any feature supported by the controller can be invoked and controlled.

There are drawbacks to this approach. For example, the upper layers of the Bluetooth LE stack or similar functionality must be implemented in the host. This usually implies extra cost for a certified host stack implementation and requires Bluetooth LE expertise. Still a Bluetooth certified controller ensures minimal interoperability issues.

This architecture is not power optimized either. The controller needs to wake-up to serve radio events and exchange information with the upper layers of the Bluetooth LE stack for both application and non-application related events. This means that the host has to wake up more frequently and thus consumes more power.

The memory resources for the Bluetooth upper layers are also not negligible. Requirements of 50 Kbytes for code and 10 Kbytes for data should be expected.

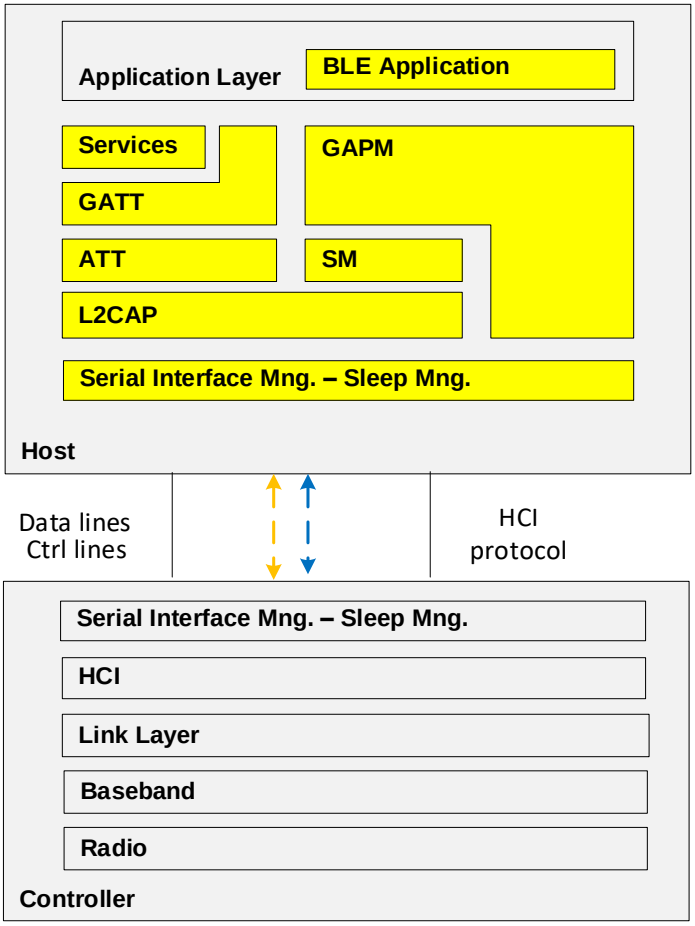

Figure 4 The HCI Host Controller Architecture

GTL host-controller architecture

The GTL architecture includes a complete Bluetooth stack in the controller. GTL is a dedicated protocol that exchanges all application-related messages over a serial link. The interface used can be UART, I²C or SPI. The GTL operates on similar principles to the HCI protocol. It provides an interface to all events and commands targeted to the Bluetooth LE application. GTL though is more power efficient than HCI since the external host should handle only application-related information.

Moreover, this architecture requires less code to be developed for the external host as the Bluetooth stack is implemented in the controller. However, the complexity remains significant and a good understanding of all the operations of the controller vendor stack is needed.

Dialog's own GTL implementation, the DGTL data pump architecture, is very versatile with two possibilities. The pure version (given by cut C2 in Figure 3) follows the standard GTL approach. Meanwhile, in the DGTL – hybrid partitioning (C3 in Figure 3), the designer can move part of the application code into the controller and expose a subset of the functionality to the external host. This offers the potential to optimize further on power and reduce the complexity of the host's software. A typical example is to select the part of the system (host or controller) where the services are implemented. Services that contain static information, like device ID, are best implemented on the controller so that the external host does not need to be woken each time the remote peer wants to read from this service. Conversely, services containing dynamic information

can be implemented in the external host to simplify the validation of the information and reduce the amount of data exchanged over the serial interface.

For new applications in particular, DGTL is a better choice than HCI. Not only is it more power efficient, it can also achieve higher throughput as less information needs to be exchanged between host and controller. Moreover, it provides full control over the stack, eliminates interoperability issues as the specification is derived from the vendor's controller implementation and incurs no additional cost to acquire Bluetooth software components for the host.

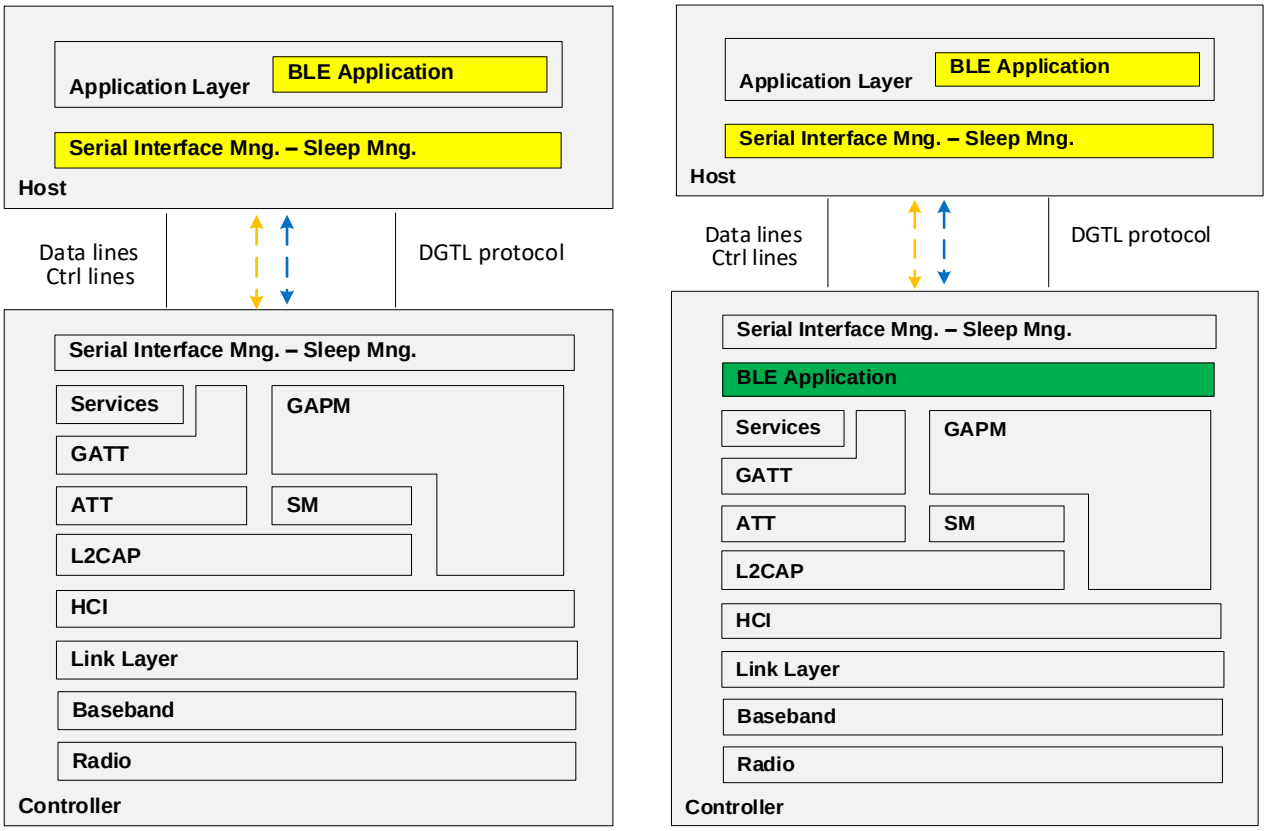

Figure 5 The DGTL (left) and DGTL-Hybrid (right) architectures. Notice that in the DGTL-hybrid case part of the Bluetooth LE application is located within the controller (shown in green)

However, there are drawbacks. A DGTL implementation tightly couples the external host implementation to the vendor of the controller. Working with this architecture requires significant Bluetooth expertise and good understanding of how things are implemented in the controller.

Application defined protocols – DSPS / CodeLess AT

The highest cut in Figure 3 (C4) involves implementing a complete Bluetooth LE application in the controller. The application handles all the Bluetooth-related functionality and exposes only selected data, commands and events going through the stack to the external host over a custom protocol.

DSPS host-controller architecture

One such application is based on the Dialog Serial Port Service (DSPS) reference design, which acts as a UART cable emulator. Two implementations are provided in the reference design: one for the central device and one for the peripheral. In real life scenarios the central device is usually an Android / iOS mobile phone.

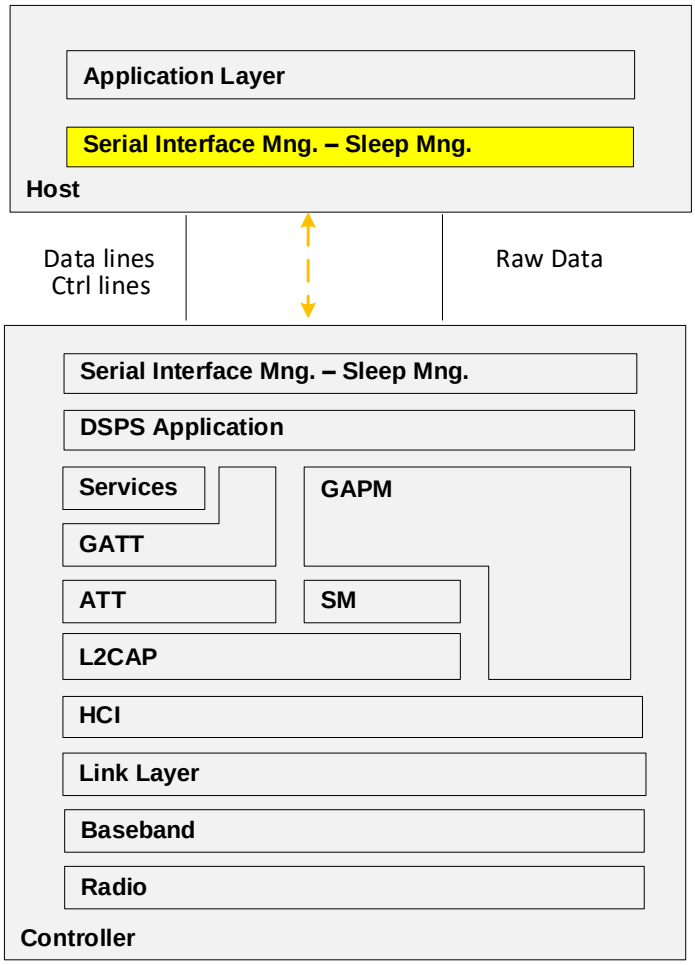

Figure 6 DSPS Host Controller architecture

The DSPS is designed with simplicity and performance in mind. There is no control over the Bluetooth link. The connection parameters are hardcoded, and the connection is handled by the controller. Using the Android / iOS DSPS software, the end user has the option to select the DSPS device to connect to. The DSPS may be thought of as a serial port replacement or a generic method of transferring serialized data to / from the host device. The core of the DSPS application is a custom Bluetooth LE service using just three characteristics; two for RX/TX and one for flow control. For the serial interface between the controller and the host, the standard flow control through RTS and CTS signals is implemented where both ends support this feature.

The DSPS is optimized to maximize data throughput using internal ring buffers, DMAs and flow control mechanisms. For specific setups, the throughput can exceed 600 Kbit/sec. DSPS is one of the most power-optimized solutions. The controller and host have to wake each other only for application data-related activities. All other operations are handled by the controller.

Host			Android	iOS	DA14585
Connection parameters	MTU	octets	>130	>130	>130
	Connection Interval	ms	12.5	30	12.5
	Host max. write command size	B	128	128	231
Throughput measurements	Half Duplex				
	Central Tx	packets/conn event	2	4	2
		kB/s	11.0	11.6	11.2
		kbit/s	88.0	92.8	89.6
	Peripheral Tx	packets/conn event	2	5	2
		kB/s	11.5	10.0	11.2
		kbit/s	92.0	80.3	89.6
	Full Duplex				
	Central Tx	packets/conn event	3	2	3
		kB/s	10.9	7.1	11.2
		kbit/s	87.4	56.8	89.6
	Peripheral Tx	packets/conn event	3	2	3
		kB/s	11.5	9.5	11.2
		kbit/s	92.0	76.0	89.6

Figure 7: Typical DSPS performance table.

As a data pump, the DSPS architecture can transfer large chunks of data with minimal host intervention.

But what about the control over the Bluetooth link? DSPS is not a good choice when the application requires control of the connection and its parameters. Of course, the DSPS implementation is provided in source code and the end user may optimize the connection parameters and add control over the link by modifying the code to compile their own custom DSPS. This will require DSPS software knowledge, understanding of the internal architecture and the ability to develop in the controller environment.

A DSPS-like data pump is a perfect option for devices that want to connect to a mobile phone when the existing DSPS functionality adequately meets the application requirements. Almost no Bluetooth knowledge is required to work with the controller. Code development for the host is reduced to handling a serial interface for data transfers and one can establish a self-managed virtual UART wireless link with little effort. However, if some of the application requirements are not met, the designer will face the dilemma of either investing in customized DSPS solutions or picking up another architecture that provides more flexibility.

CodeLess AT Commands data pump

Another application that implements a similar approach to the DSPS data pump is the CodeLess AT Commands data pump. The CodeLess AT Commands application provides a data pipe but also a limited control path over the Bluetooth connection making it a good fit for a larger set of use cases than the DSPS.

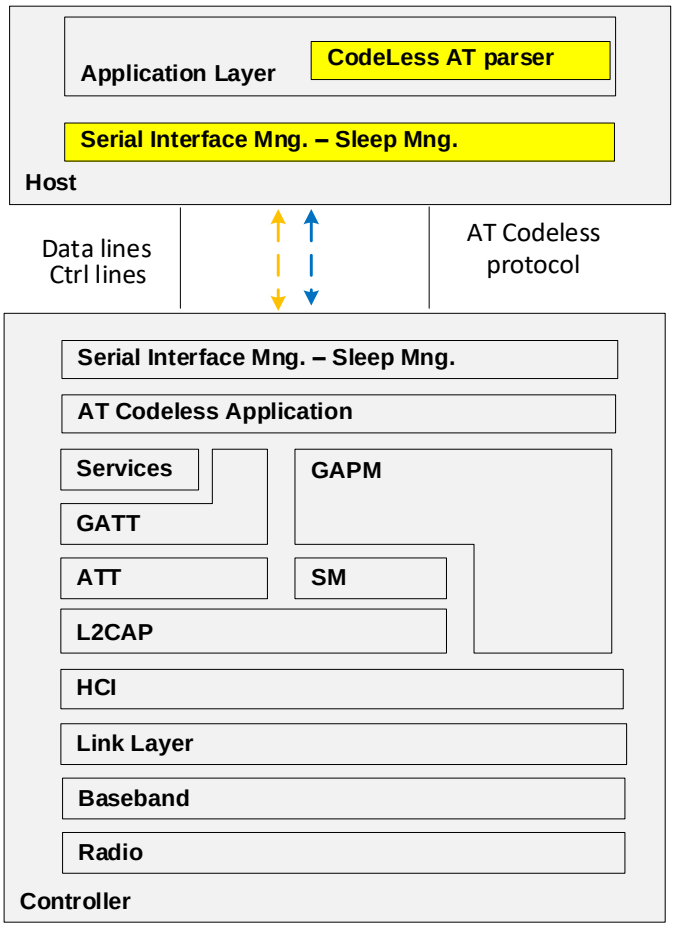

Figure 8 AT Codeless Host Controller application

Compared to the HCI and DGTL data pump solutions described earlier, the CodeLess AT Commands host implementation is significantly simpler and requires little competency in the Bluetooth LE protocol. Therefore, solutions based on this approach can substantially reduce the time to market for a Bluetooth LE-enabled product.

Dialog Semiconductor has introduced a CodeLess AT Commands solution. Such solutions are commonly found in the market. As the name implies those solutions are based on AT commands and try to hide the Bluetooth LE implementation details from the user. AT commands are short, simple, human-readable text commands and can be issued over any of the available communication interfaces (UART, SPI, I²C). Every command instructs the CodeLess AT controller to execute one or more predefined functions. Depending on the outcome of the function, the CodeLess AT controller returns an appropriate response.

Typical CodeLess AT Commands support scanning for other Bluetooth LE devices, establishing new connections and basic connection management. The host development is built around issuing the AT commands at appropriate times and responding to error codes as well as being the data source or data sink.

This approach comes with several advantages over more traditional ways of interacting with the controller. It is power efficient since the messages going over the controller-host link are mainly application related. In addition, the CodeLess AT implementation provides simplified Bluetooth LE connection management and data transfer control. Therefore, the user does not need to be familiar with low-level Bluetooth LE details.

Developing the final application in the host requires minimal effort. AT commands are simple to remember, and all the messages are in human-readable form, allowing a designer to directly interact or experiment with the controller. No cryptic codes and obscure messages are exchanged.

A CodeLess AT Commands application may be used as an introduction to Bluetooth LE, reducing the learning curve and allowing the user to familiarize themselves with several Bluetooth LE concepts before diving into more complex solutions. As such, it can accelerate development significantly.

There are also disadvantages related to this approach. The data throughput of the CodeLess AT implementation is not optimized. Although powerful, CodeLess AT Commands application only supports a limited set of the Bluetooth LE stack capabilities. Thus, during development, the designer may again face the dilemma of either extending the capabilities of the CodeLess AT controller or moving to other solutions that provide full flexibility.

The DA14531 in data pump applications

DA14531 key features

The DA14531 is the latest addition to the DA145xx family of Bluetooth LE devices. The device is both cost and power optimized aiming to serve simple, disposable and data pump applications. It can serve any of the data pump architectures presented in this document, functioning as an HCI, GTL, DSPS or CodeLess AT controller.

Addressing data pump requirements

As explained above, among the key requirements for a data pump application are data throughput, interface options between the host and controller, low power consumption, flexibility in power supply, small size and low boot latency. The DA14531 has been designed from the beginning to address the basic requirements of a data pump controller.

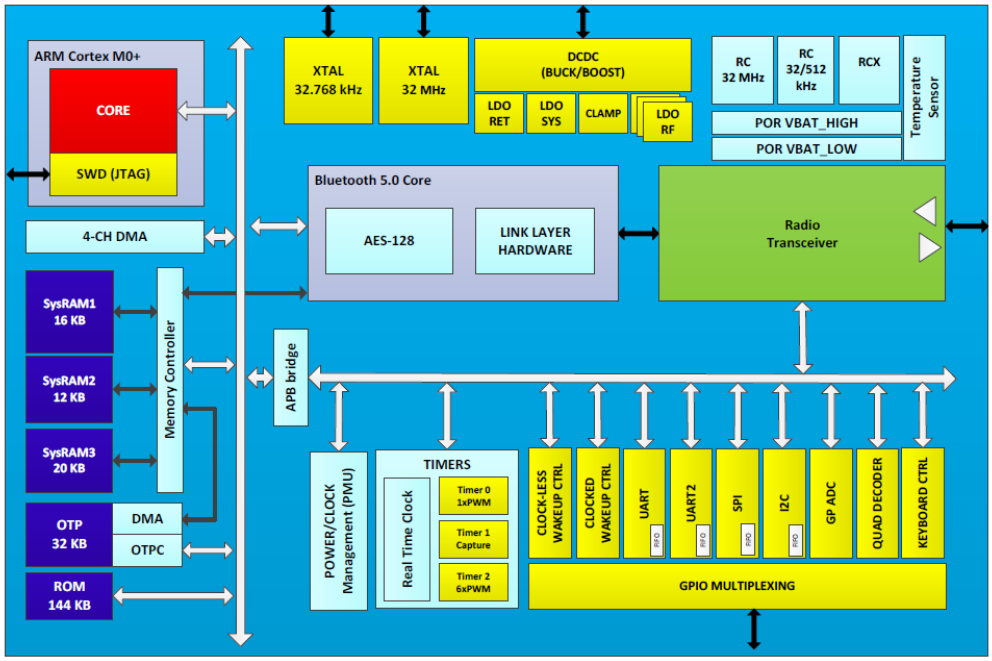

Figure 9 Block diagram of DA14531.

As well as complying with the Bluetooth 5.1 specification, it provides all three types of serial interfaces (UART, SPI and I²C), allowing any kind of serial communication available in the host subsystem. The DA14531's power consumption has been optimized thanks to a configurable transmit power and a radio receiver current below 2.2 mA for a buck configuration powered from 3. 3V

The power management circuitry is built around 2 rails, VBAT_HIGH and VBAT_LOW. Between the two rails, there is a DC/DC converter, an LDO and an internal switch. This setup allows the DA14531 to work from whichever voltage is available in the host subsystem. For example, if a voltage below 1.8 V is available, this can be provided at VBAT_LOW and boosted to VBAT_HIGH using the internal DC/DC converter. Voltages between 1.8 and 3.3 V can be provided to VBAT_HIGH. In this situation, various options are possible. The voltage can be bucked to VBAT_LOW with the internal DC/DC converter to optimize power consumption. For cost-down solutions, both rails can be externally shorted to save the cost of the DC-DC inductor. Alternatively, the internal LDO or switch can be used instead of the buck DC/DC configuration to extend VBAT _HIGH operation below 1.8 V to drain the battery.

The DA14531 is available in a choice of package. The FCGQFN24 version measures 2.2 mm x 3.0 mm and provides 12 available GPIOs. Meanwhile the WLCSP17-package version is just 1.694 mm x 2.032 mm, yet still has 6 available GPIOs, minimizing the area required to enable in Bluetooth LE connectivity.

The device cold boots to fully operational in less than 30 ms if code is read from external Flash memory. If code is stored in the internal OTP memory, boot time falls to just 1.5 ms.

Use cases

We will finish our review of Bluetooth LE data pumps by describing a few examples of such systems and how to select the best architecture in each case.

Smart white goods

Most mid- and high-end white good appliances already include a microcontroller unit to enable some of their high-end features such as elaborate controls and advanced user interfaces. However, they still lack connectivity mainly due to the cost and complexity. Bringing connectivity into such devices will enable a range of new possibilities that allow consumers to interact with the appliance through their smartphones. This could include simple use cases like exchanging valuable usage statistics (such as operating time and power consumption), and service or error codes logs to remotely diagnose issues. In addition, appliances could be configured or personalized via the extremely powerful interface that a smartphone provides. In such cases Bluetooth LE security will be an important factor to consider. Dialog offers tools (SDK, reference designs) that facilitate the development of such applications.

Since there are currently no Bluetooth LE standard services defined for most white goods, a DSPS-like data pump is an excellent starting point to bring connectivity to these devices. There are no interoperability requirements, and a proprietary command-response protocol able to sustain the sudden disconnections that can happen in any wireless connection is the only thing that needs to be developed on top of the wireless serial interface provided by DSPS. With just that protocol implementation the appliance would be able to exchange information and commands without spending time making Bluetooth LE-related decisions.

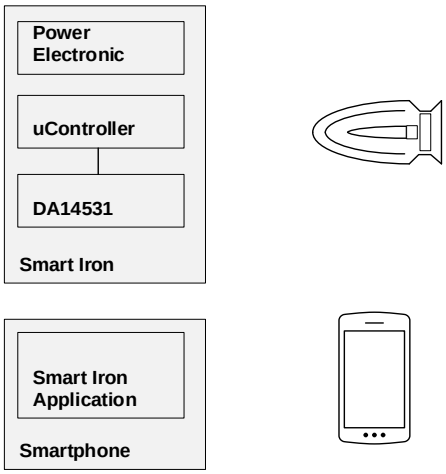

Figure 10 The "Smart" Iron use case

The appliance would advertise its existence when powered and be able to connect to nearby smartphones. It is easy to think of the smart dish washer or the smart electric stove but the cost and size of the DA14531 enables connectivity even for small and cost-sensitive applications like a smart iron or the smart hairdryer. Many of the small devices that already have a microcontroller and can spare a couple of pins, can leverage a huge range of capabilities by connecting to a smartphone. Imagine, for example, Alexa selecting the best setting for ironing your favorite dress from a database in the internet.

Battery-operated tools

Connectivity will bring a similar set of benefits to all kind of small devices. However, for battery-operated devices such as cordless drills, more control over the Bluetooth link is required. To save power, the device should not advertise its existence all the time. For the same reason, it may need to disconnect if there is no action from the user. Such functionality can be easily implemented using the CodeLess AT Commands approach. The CodeLess AT Commands application gives enough control as well as information to understand the state of the Bluetooth LE link.

Selecting the right architecture

Where there are defined Bluetooth LE services that relate to the specific use case, the GTL architecture may be the best way forward. This will enable the services to be exposed to other interoperable applications or even allow the target device to access services provided by a smartphone.

A thermometer, a glucose meter, a blood pressure meter, a weight scale are all examples of devices that could benefit from standard Bluetooth LE services. The GTL architecture provides all the flexibility of a Bluetooth stack through a serial port at the expense of the increased development cost of a Bluetooth application in the host. A typical code footprint for a simple application is 5-10 Kbits.

In summary, the advantages and disadvantages of all communication protocols can be found in the table below.

Table 2 – Advantages and disadvantages of each communication protocol

Communication Protocol	Advantages	Disadvantages
HCI	• Standard interface supported by a wide number of vendors • High performance (throughput) • Full control over Bluetooth	• Advanced Bluetooth LE knowledge is required • A Bluetooth LE stack is necessary implying extra cost • Significant complexity

GTL	• Full control over Bluetooth • Most power efficient	• Not standard • Simpler than HCI but still complex • Requires Bluetooth knowledge
DGTL/DGTL – Hybrid	• Can select which application layer functionality is implemented in the controller and which is exposed to the host • Full control over Bluetooth • Most power efficient	• Not standard • Simpler than HCI but still complex • Requires Bluetooth knowledge and
DSPS / CodeLess AT	• No prior Bluetooth LE knowledge required • Simple to use – minimal development required • Fast time to market • Source code provided by Dialog	• Limited (AT codeless) or no control (DSPS) over the Bluetooth link • Can be difficult to add new features since recompilation may be necessary

Conclusions

In this paper a practical approach to the different BLE data pump architectures was undertaken. In particular different data pump architectures were examined, from solutions where considerable expertise and Bluetooth knowledge are necessary to much simpler ones. Elaborate solutions provide unquestionably better control of the data pumping process whereas simpler ones allow the creation of an application with minimal effort and in a short amount of time. In addition to the BLE data pumps the DA14531 was presented, an IC from Dialog Semiconductor ideally suited to data pump applications. Finally, several use cases were provided along with guidelines to select the most appropriate architecture for each use case.

Terms and Definitions

ATT	Attribute profile
BLE	Bluetooth Low Energy
DSPS	Dialog Serial Power Service
GATT	Generic Attribute Profile
GPIO	General Purpose Input / Output
GTL	GAP-like protocol which is natively supported by DA14531
HCI	Host Controller Interface
I²C	Inter – Integrated Circuit
L2CAP	Logical Link Control and Adaptation Protocol
LDO	Low - Dropout
MTU	Maximum Transmission Unit
OTP	One – Time Programmable
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver – Transmitter

References

[1] DA14531, Datasheet, Dialog Semiconductor
[2] Bluetooth SIG, Bluetooth Core Specification v5.0

