Advanced Device Serialization Using an External Serialization Server

A BPM Microsystems White Paper

By Nader Shehad

May 2013

BPM Microsystems
5373 West Sam Houston Pkwy N. Ste. 250
Houston, Texas 77041 USA
Telephone: +1 713 688 4600
www.bpmmicro.com

Contents

Overview

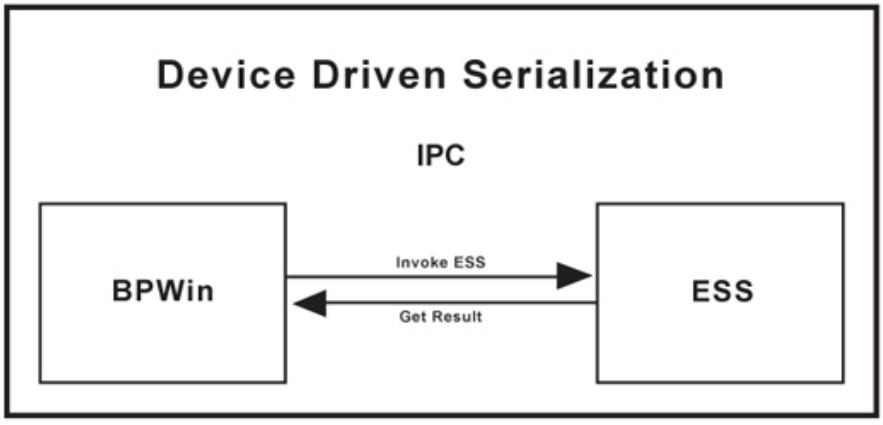
In the device programming industry, serialization is the process of writing unique data to each programmed device. It can be used to program basic numeric serial numbers to a single device address and can also be used to program more complex data such as MAC addresses, encryption keys, GUIDs and randomization seeds to several device addresses on each device.

With modern device programming solutions, serialization data is generated by a customer-written application called an External Serialization Server (ESS). The ESS generates serialization data algorithmically, retrieves serialization information from a database, or uses more sophisticated techniques to create device-specific data for programming into each Device Under Test (DUT). It is invoked by the programming application as needed to retrieve the required serialization data during a job session for each DUT.

Historically serialization applications generate data in a vacuum without any input from the device itself. Serialization data is either pre-computed (e.g. a list of MAC IDs) or generated based on fixed inputs that are device-independent (e.g. timestamps, firmware revision codes, etc). While this approach is effective for simple cases, it fails to address the complicated requirements of modern serialization. A common use case requires generating a serialization payload based on information already stored in each device. For instance, the contents of a device hardware register must be read and passed to the serialization application to feed a serialization algorithm or used as a lookup key into a database of serialization data. In such scenarios, there must exist a communication framework by which the programming application, which is capable of reading device-specific data, can communicate bi-directionally with the ESS to facilitate device programming. Since each device/ESS is unique, the framework must be flexible and extensible allowing each device algorithm to communicate the required information to the ESS.

Device-Driven Serialization

To address the needs of modern serialization use cases, BPM Microsystems has developed a communication framework called Device-Driven Serialization (DDS). DDS allows an ESS to communicate bi-directionally with a device-specific programming algorithm running as part of the programming application. Based on customer specifications, the programming algorithm is designed to read all the appropriate information needed by the ESS and pass that information to the ESS at runtime, just in time. The communication protocol between the algorithm and the ESS can be as simple as a single unidirectional data transfer or may consist of several data transfers and logic to alter operation flow based on data read from the device or returned from the ESS.



Using this approach, the ESS actually becomes part of the programming algorithm. The algorithm is written according to customer specifications and is designed to communicate with the ESS in a pre-defined manner, sending and retrieving data at the appropriate control/decision points. The ESS has the opportunity to monitor any aspect of the operation, read any relevant information from the DUT, present appropriate messages to the user, and even choose to fail the device operation if any retrieved data fails validation.

More than Just Serialization

The power of the bi-directional communication framework provided by DDS goes far beyond what is typically considered serialization. For instance, the ESS can be notified each time the device is powered on or off, or it can read device-specific data for logging purposes or details about the programming mode. Instead of just reporting a device failure, a more detailed failure analysis can be provided to indicate exactly which phase of the program and/or verify operations failed.

Because the ESS is a full-fledged Windows application, this information can be used in a myriad of ways, which include storing it in databases for later retrieval, sending e-mail alerts to key personnel when certain trigger conditions occur, or integrating with other customer systems.

ESS Considerations

The ESS is a standalone application that can be written in any language, such as C++, JavaScript or Visual Basic. The ESS project could consist of one or many source code files allowing for applications ranging from very simple to very complex. To communicate with the programming application, the ESS will define an object that exposes a list of agreed-upon functions that are invoked using Interprocess Communication Techniques (IPC). The details of IPC are invisible to the developer and are handled by the DDS framework. The programming algorithm will reference and invoke these ESS functions as part of the operation flow, passing the appropriate data as parameters. Each function will then take the appropriate action and return any relevant data to the algorithm. The algorithm will then make conditional decisions based on the returned data and potentially provide additional data to the ESS in the form of a subsequent function call or modify the operation flow as a result. The entire process is customer defined both in the ESS and the programming algorithm, giving the customer full control of the operation.

Conclusion

Serialization is a difficult problem that many original equipment manufacturers and programming centers face. Traditional approaches to serialization have provided the framework needed to program static device-independent serialization data, but modern serialization requirements demand a more sophisticated approach that can generate serialization data based on device-specific information already stored in each device. Device-Driven Serialization leverages a flexible, secure, bi-directional communication framework between the programming algorithm and the serialization application, allowing for a truly custom process flow. Through DDS, the ESS is granted intimate knowledge of any arbitrary data on the DUT and can directly impact program operation flow. DDS is a breakthrough in the serialization process that facilitates today's demanding serialization requirements.