ISBN: 978-3-95450-222-6 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2020-TUPP12

AUTOMATED MANAGEMENT OF LIBERA SPARK MODULE IOCs IN SPEAR3*

F. Toufexis†, S. Condamoor, D. A. Morataya Campos, C. S. Ramirez, J. J. Sebek, C. Wermelskirchen, J. Corbett, SLAC, Menlo Park, CA 94025, USA
P. Leban, M. Znidarcic, Instrumentation Technologies, Solkan, Slovenia

Abstract

We are actively upgrading BPM processors in the SPEAR3 accelerator complex as several of the existing systems are reaching end-of-life. To consolidate the resources required for development and maintenance we have evaluated and installed several processors from the Libera SPARK hardware series. We found that two common deployment methods typically used with these modules, micro-SD card and network boot, are either hard to maintain or lack flexibility. Instead we have developed an automated method based on a network boot scheme where an external EPICS soft IOC manages the assignment of specific SPARK modules to physical BPMs in the accelerator. Each module queries the soft IOC at boot time to determine which BPM it is assigned to and then starts its IOC with the appropriate BPM prefix for the PV names. This deployment method allows for quick, seamless swapping of SPARK modules by machine operators or physicists. It addition, it allows us to bring additional modules online for testing, or to move modules to different locations with a different PV prefix for the new location. This method is applicable to other EPICS-enabled devices where the device hardware also hosts an IOC.

INTRODUCTION

SPEAR3 is a 3 GeV, 500 mA, 3rd generation synchrotron light source, commissioned in 2004 [1]. It operates with beam current distributed in four bunch trains and a single isolated timing bunch for pump-probe experiments. Top-up occurs at 5-minute intervals. Each top-up event requires about 50 single-bunch charge pulses into targeted SPEAR3 buckets at a 10 Hz rate. The SPEAR3 storage ring contains 18 lattice cells each with 6 button-style Beam Position Monitors (BPMs). Three BPMs per cell are connected to Bergoz processors for fast orbit control and beam inter-lock purposes. Several more BPMs are connected to the Echotek processors [2] to provide Turn-by-Turn (TbT) orbit information at discrete locations. The Echotek processors were developed in-house and produced commercially when SPEAR3 was commissioned; they are used for accelerator physics programs and not for operations. The Echotek have reached their and of life and we have evaluated commercial alternatives for replacement. A Libera Brilliance+ was first tested in SPEAR3 in 2017 [3]. Since TbT studies for accelerator physics programs do not require the long-term stability capability of the Brilliance+, and additionally the fast orbit feedback is implemented in the Bergoz BPM system, a SPARK-ERXR processor [4] was purchased and installed for accelerator physics applications. Additionally we are in the process of installing and testing Beam Loss Monitors (BLMs) for Libera that have the same base hardware and software infrastructure as the SPARKs.

The SPEAR3 injector was commissioned in 1990 [5], and includes the 120 MeV linac injector with a thermionic RF gun [6], the booster synchrotron [7] and the Transport Lines. The entire injector, including the Transport Lines, is equipped with stripline-style BPMs. The original booster synchrotron BPM electronics used a commercial multiplexer to switch between several BPM signals into an in-house built analog BPM processor [8]. In the Linac-To-Booster (LTB) and Booster-To-SPEAR (BTS) Transport Lines 1990's-era Bergoz BPM processors have provided reliable shot-by-shot single-pass data at 10 Hz with limited resolution [8]. As an upgrade to the original Transport Line BPM processors, two smaller-diameter stripline BPMs connected to two SLAC-built uTCA-based BPM processors replaced the last two BPMs at the end of the BTS in 2015 (BTS BPMs 8 and 9). This system has proven hard to maintain and we have evaluated single-pass SPARK-EL processors as a replacement. The units were tested in the BTS and LTB Transport Lines demonstrating comparable position resolution to the uTCA processors at the small-diameter striplines, as well as substantially improved resolution at the large-diameter striplines.

Since these BPM processors, as well as the BLMs, have the same base hardware and software, we wanted a unified architecture of deploying and managing these modules. We aim for an architecture that allows for easy swaps in the event of a module failure, but additionally gives us the flexibility to bring modules online for testing or to relocate modules and change the EPICS Process Variable (PV) prefix they publish appropriately. All these requirements come down to having the ability to dynamically change the EPICS PV prefixes. We perform this through an external EPICS soft Input/Output Controller (IOC) that manages the information for the prefixes and settings. Each module runs a shell script that acquires all this information and starts its IOC with the appropriate prefix and settings. This work is organized as follows: we begin by explaining our considerations for choosing this deployment architecture and then give a high-level description of our solution. Subsequently we describe the infrastructure that is needed to implement this architecture and the details of how a module boots and acquires its PV prefixes. Finally, in the appendix we show certain important code snippets.

* Work sponsored by US Department of Energy Contract DE-AC02-76SF00515.
† ftouf@slac.stanford.edu

ISBN: 978-3-95450-222-6 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2020-TUPP12

CONSIDERATIONS FOR DEPLOYMENT

SPEAR3 operates with a very small accelerator team; typically only one person is an expert on any given system, with a second person having some knowledge about it. As a result in deploying new electronics systems we are aiming for architectures that if a module fails, an operator can swap the failed module with a spare with only minimal guidance.

When a SPARK BPM processor boots up, it requires the following information in order to operate inside the control system: the PV prefixes that identify the physical BPM it is assigned to and settings that are specific to that BPM (e.g. gains and offsets). There are two straightforward approaches in deploying the SPARKs: using the micro-SD (uSD) card or perform a network boot.

Using the uSD card involves hard-coding the PV prefixes and settings in the card. Note that the uSD card can itself become corrupted. Therefore for each physical BPM we need to maintain two uSD cards. The cards themselves are small and labeling them is very hard, thus making management difficult. In addition, it is time consuming to upgrade the firmware in the event of a software update from Libera; we would need to update all the uSD cards including the spares. Alternatively we could have one uSD card template where someone would have to clone during a module swap and change the settings. Most operators, however, do not have the skills to perform this task. Additionally in the BTS the different BPMs are physically different and need different configuration, thus increasing the chances of an error during a module swap.

The alternative to the uSD card is a network boot. In this case a linux boot image is stored in the Trivial File Transfer Protocol (TFTP) server and is supplied to the module during boot time. The IOC files are stored in the Network File System (NFS) server. The Dynamic Host Configuration Protocol (DHCP) server assigns a hostname based on the Media Access Control (MAC) address that is linked to the physical BPM the specific module is connected to, i.e. `BTS-BPM08`. This allows for easy firmware upgrade as all the files are located in a central server; all that is required to happen is update them and reboot the modules. However, in the event of a module swap the Network Group needs to update the entry in the DHCP server with the MAC address of the spare module and restart the DHCP server that could cause other issues in the network. To handle restoring the settings for a physical BPM, either EPICS `autosave` or simply a MATLAB script could be used.

SOLUTION OVERVIEW

To solve the aforementioned issues, we developed a new deployment architecture based on the network boot with the addition of a database that manages assignment of BPM processors to physical BPMs. In our architecture the network hostnames for the SPARK modules are based simply on the module model and some identification number, i.e. `spark-el-04`; the hostname has no connection to a physical BPM. The mapping of a BPM processor to a physical BPM is handled by a soft IOC that resides in central server and an Extensible Display Manager (EDM) panel is used to enter the settings. For example this soft IOC will maintain the information that module `spark-el-04` is connected to `BTS-BPM08`. When module `spark-el-04` boots up, before starting the IOC, it will query the soft IOC "which BPM am I assigned?" based on its hostname. The result of this query will be `BTS-BPM08` and the module will start the BPM IOC with `BTS-BPM08` as the prefix for its PVs.

In the event of a module failure the following steps need to be taken to recover:

1. The Operator assigns one of the spare BPM processors to the physical BPM the failed module was connected to in the soft IOC EDM panel.

2. The Operator physically swaps the modules and powers up the new module.

3. The Network Group updates a repaired module information in DHCP during normal business hours when convenient.

An advantage of this approach is that we can bring spare modules online in the control system to test them without affecting operations, i.e conflicting PV names. Additionally we can move modules around. This is important as in SPEAR we only plan to have a few SPARK TbT processors and want to be able to move them to different BPMs as needed. The same would apply to BLMs.

INFRASTRUCTURE

Figure 1 summarizes the infrastructure required for this deployment scheme: On the network side a number of services are needed: A TFTP server is used to supply the boot image to the SPARK modules when they are powered on. DHCP is used to assign IP addresses and hostnames to the modules. Each module's MAC address (including spares) needs to be entered to the DHCP server database along with a specific hostname for that MAC address. The DHCP server additionally points the module to the TFTP server to download the boot image and the NFS server for file storage. The NFS server is used to supply a file system that the module mounts. The SPARK IOC files reside in the NFS space. Additionally we create directories for each SPARK module and for each physical BPM. The module mounts these directories once it has its hostname and BPM assignment, and uses them for log or temporary files.

A soft IOC is needed to hold the mapping between each SPARK module and the physical BPM it is assigned. This soft IOC resides in one of our servers. For each SPARK module we create a PV; the hostname is part of that PV name, while the PV value is the physical BPM assignment. The PV value can additionally be `UNASSIGNED` to denote that the module is not connected to a BPM, or `TEST` in the case we want to bring a module online for testing. In our case for the Transport Lines we use two PVs: one for the Transport Line, `LTB` or `BTS`, and a second for the

ISBN: 978-3-95450-222-6 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2020-TUPP12

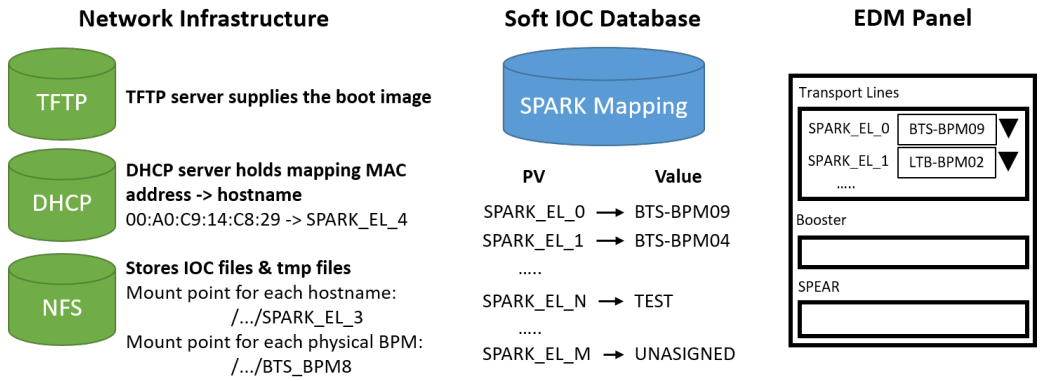

Figure 1: Overview of the Infrastructure needed to perform Network Boot with a Hostname/PV prefix Soft IOC database.

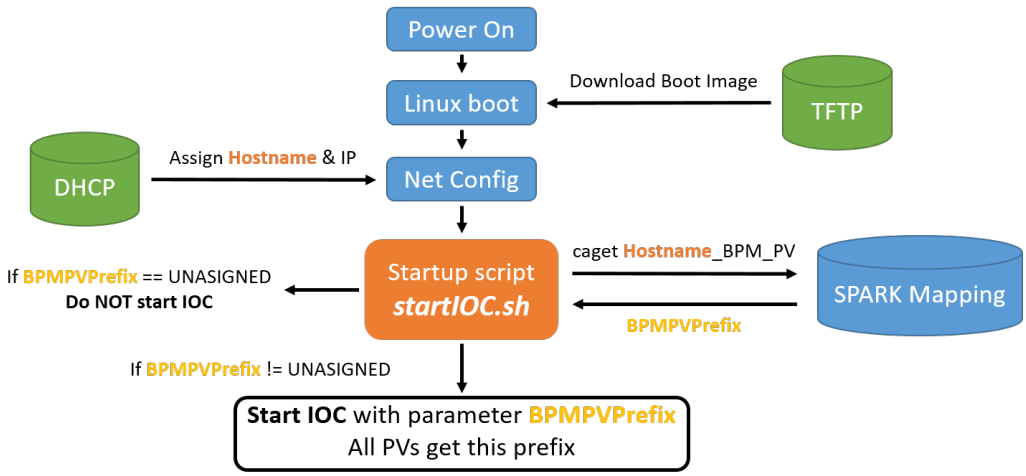

Figure 2: Simplified boot sequence of a SPARK module.

BPM number within the Transport Line, i.e. `BPM08`. In our implementation these two PVs for module with hostname `$UNITNAME` are `SPARKMngr:$UNITNAME:BPM:TL` and `SPARKMngr:$UNITNAME:BPM:BPM`. The options for `SPARKMngr:$UNITNAME:BPM:TL` are `LTB`, `BTS`, `TEST`, and `UNASSIGNED`, while for `SPARKMngr:$UNITNAME:BPM:BPMN` the options are `BPM01` through `BPM09`. At boot time a script in the module will querry these two PVs and determine the BPM prefix, i.e. `SPARK-BTS-BTS08`. We further use this soft IOC to hold the settings for each BPM, such as gains and offsets. We do this with another set of PVs per BPM, i.e. `SPARKMngr:SPARK-BTS-BTS08:Kx`.

Finally a set of EDM panels are needed as an interface to this system. One EDM panel shows the list of SPARK modules and for each module the assignment of Transport Line and BPM number within that Transport Line, and similarly for the Booster and SPEAR3. Another EDM panel holds the settings for each physical BPM.

MODULE BOOT SEQUENCE

Figure 2 illustrates the boot sequence of each SPARK module. When the unit powers on it announces itself to the network and TFTP supplies the linux Boot Image. Then linux starts running and it gets from DCHP the hostname and IP based on its MAC address.

Eventually the `startIOC.sh` scripts executes. We have modified this script from what was supplied from Libera to perform the following steps:

1. Query PVs on the soft IOC based on its hostname and calculate the PV prefix for the module's IOC.

2. Perform a series of checks that this PV prefix is valid in the accelerator, and additionally that there is no other IOC on the network running with it. This PV prefix is written in a temporary file read by the `st.cmd`.

3. Query PVs on the soft IOC based on the PV prefix to obtain the settings for the physical BPM, i.e. gains and offsets. These settings are written in another temporary file as EPICS `dbpf` commands read by the `st.cmd` after the `iocInit()`.

4. Start the IOC with `procServ` and a modified `st.cmd` to read the PV prefix and settings.

ISBN: 978-3-95450-222-6 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2020-TUPP12

SUMMARY

We have presented an automated method for deploying Libera SPARK modules other EPICS-enabled devices where the device hardware also hosts an IOC. This method is based on a network boot scheme where an external EPICS soft IOC manages the assignment of specific SPARK modules to physical BPMs in the accelerator. Each module queries the soft IOC at boot time to determine which BPM it is assigned to and then starts its IOC with the appropriate BPM prefix for the PV names. This deployment method allows for quick, seamless swapping of SPARK modules by machine operators or physicists. It addition, it allows us to bring additional modules online for testing, or to move modules to different locations with a different PV prefix for the new location. Most importantly though, since a script automates the process and performs various checks for errors, this method minimizes the chances of making a mistake when bringing a module online, whether that is using different settings or having the same PV prefix as an already running IOC.

ACKNOWLEDGMENTS

The authors would like to thank the SPEAR3 operations staff, Scott Wallters, and Kevin Helms for technical assistance.

APPENDIX

Here we show some code snippets from the `startIOC.sh` script for the Transport Lines. To assemble the PV prefix for the Transport Lines we have two PVs, one to assign the Transport Line (`LTB`, `BTS`, `TEST`, `UNASSIGNED`), and one for the BPM number within the Transport Line. We start by getting the hostname:

```
UNITNAME=`hostname`
```

This will return `spark-el-01`. Then we query the PV `SPARKMngr:spark-el-01:BPM:TL` in the soft IOC to get the Transport Line assignment and write the result to a temporary file:

```
exec $CAGET -t SPARKMngr:$UNITNAME:
    BPM:TL |& tee $IOC_CONFIG/
    transportLineName
```

We need to check that `caget` did indeed return the value of the PV by measuring the number of words in the temporary file; if there is only one word then that word is the value we asked, otherwise the soft IOC was not accessible and the script should exit without starting the IOC:

```
cat $IOC_CONFIG/transportLineName |
    wc -w > $IOC_CONFIG/
    transportLineNameWC

if [ ! "$(head -n 1 $IOC_CONFIG/
    transportLineNameWC)" -eq 1 ]
then
        echo "SPARKManager soft IOC
            not accessible" |& tee -a
            $STARTUP_ERROR
        exit 1
fi
```

Assuming no error we get the Transport Line assignment:

```
TRANSPORTLINE="$(head -n 1
    $IOC_CONFIG/transportLineName)"
```

that is `BTS`. We use the same procedure and query another PV to get the BPM number within that Transport Line:

```
BPMNUMBER="$(head -n 1 $IOC_CONFIG/
    bpmName)"
```

that is `BPM08`. We can now calculate the PV Prefix:

```
BPM_PREFIX="SPARK-$TRANSPORTLINE-
    $BPMNUMBER"
```

that is `SPARK-BTS-BPM08`. Now we need to perform a series of checks to ensure this PV prefix is valid; for example there is no `BPM08` in our `LTB`:

```
if [ "$BPM_PREFIX" == "SPARK-LTB-
    BPM08" ]
then
        CAPUT_EXIT_STATUS=$(exec
            $CAPUT -t SPARKMngr:
            $UNITNAME:STATUS.VAL "
            Error: SPARK-LTB-BPM08
            doesn't exist")
        exit 1
fi
```

If there is an error at this point we log it to a status PV. Assuming the PV prefix is valid we need to check that no other IOC already started on the network with this BPM prefix by checking a PV that the SPARK should publish:

```
exec $CAGET -t $BPM_PREFIX:input:
    max_adc |& tee -a $IOC_CONFIG/
    iocAlreadyRunning
```

If `caget` returns more that one word, then then IOC was not accessible, therefore not running.

To handle the settings for a particular physical BPM we use the same procedure: the soft IOC has a number of settings PVs for each physical BPM. The `startIOC.sh` continues to query the values of these PVs and writes a set of EPICs `dbpf` commands in an temporary file that will be loaded in the `st.cmd` after the `iocInit()` command:

```
echo "dbpf(\"$BPM_PREFIX:dsp:
    pickup_orientation_sp\",\"
    $PICKUPORIENTATION\")" >
/tmp/BPMSettings.env
```

Once all these steps have occurred successfully the `startIOC.sh` starts the module's IOC using `procServ`.

ISBN: 978-3-95450-222-6 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2020-TUPP12

REFERENCES

[1] "SPEAR 3 Design Report," SLAC, CA, USA, Tech. Rep. SLAC-R–609, Dec. 2002.

[2] J. J. Sebek, D. J. Martin, T. Straumann, and J. V. Wachter, "Design and performance of SSRL beam position electronics", in *Proc. 14th Beam Instrumentation Workshop (BIW'10)*, Santa Fe, NM, USA, May 2010, paper TUPSM029, pp. 182–186.

[3] S. Condamoor *et al.*, "Machine studies with Libera instruments at the SLAC SPEAR3 accelerators," in *Proc. 7th Int. Beam Instrumentation Conf. (IBIC'18)*, Shanghai, China, Sep. 2018, pp. 284–288.

[4] P. Leban, "Libera SPARK-ERXR User Manual," Tech. Rep., 2017.

[5] S. Baird and J. Safranek, "Commissioning the SSRL injector," in *Proc. 1991 IEEE Particle Accelerator Conf. (IPAC'91)*, San Francisco, CA, USA, May 1991, vol. 5, pp. 2865–2867.

[6] M. Borland, "A High Brightness Thermionic Microwave Electron Gun," Ph.D. dissertation, Stanford University, CA, USA, Feb. 1991.

[7] H. Wiedemann *et al.*, "The 3 GeV synchrotron injector for SPEAR", in *Proc. 1991 IEEE Particle Accelerator Conf. (IPAC'91)*, San Francisco, CA, USA, May 1991, vol. 5, pp. 2688–2690.

[8] W. Lavender *et al.*, "The SSRL injector beam position monitoring systems," in *Proc. 1991 IEEE Particle Accelerator Conf. (IPAC'91)*, San Francisco, CA, USA, May 1991, vol. 2, pp. 1151–1153.