Digital Coil Protection System I/O and Data Subsystem for NSTX-U*

Gregory J. Tchilinguirian[1], Keith G. Erickson[2], Ronald E. Hatcher
Princeton Plasma Physics Laboratory, Princeton NJ, USA
Corresponding author e-mail: gtchilin@pppl.gov

***Abstract*— The Digital Coil Protection System (DCPS) for NSTX-U is a system that evaluates operational parameters in real time and prevents physical damage to the reactor and its coils. The system is built upon commercial products: a real-time Linux operating system and analog and digital I/O boards. Signals from various digital and analog sources are captured, conditioned, validated and stored before being passed to a system of parallel algorithms for processing. Incremental data must be recorded during processing for post-shot analysis. Additionally, algorithm parameter data must be stored in a fashion that prevents both inadvertent and malicious tampering which could result in exposing the machine to the risk of severe damage.**

This paper will discuss the object oriented design and implementation used to address these challenges. Details of how real time Linux tools were used to ensure predictable and stable operation will also be discussed.

Keywords—NSTX; Control; Protection; Real-time; Data

I. Introduction

The Digital Coil Protection System (DCPS) is part of an overall plan to bolster the National Spherical Torus eXperiment (NSTX) [1] for post upgrade operational limits beyond what it was originally constructed to endure [2]. One aspect of this plan includes mechanical upgrades to the physical structure of NSTX, which is beyond the scope of this paper. The other aspect, DCPS, entails the creation of a flexible real-time computer system capable of rapidly signaling a shutdown of the equipment used to power the NSTX Upgrade (NSTX-U) [3,4,5] coil systems through an existing coil fault mechanism. To accomplish this DCPS will use a number of force calculations performed in parallel using coil currents and checked against pre-determined limits to prevent exposing NSTX-U to forces it cannot withstand.

An instance of DCPS, with slightly lower limits, will be created on a similar computer running the Power Supply Real Time Control (PSRTC) process, which is used to command the power supplies as well. This instantiation of the DCPS core will be used to generate more machine friendly shut down regimes in the event of forces exceeding the operational envelope of NSTX-U.

II. Software System Design

The DCPS Core is comprised of a number of subsystems that represent an occupational division executed modularly. To properly allocate the division of labor and determine what functionality would be needed, a state machine diagram was generated and used as the foundation of the software design for this project [6] as shown in Fig 1. It defines the overall capabilities of the system in ten states with four relevant to real time operation

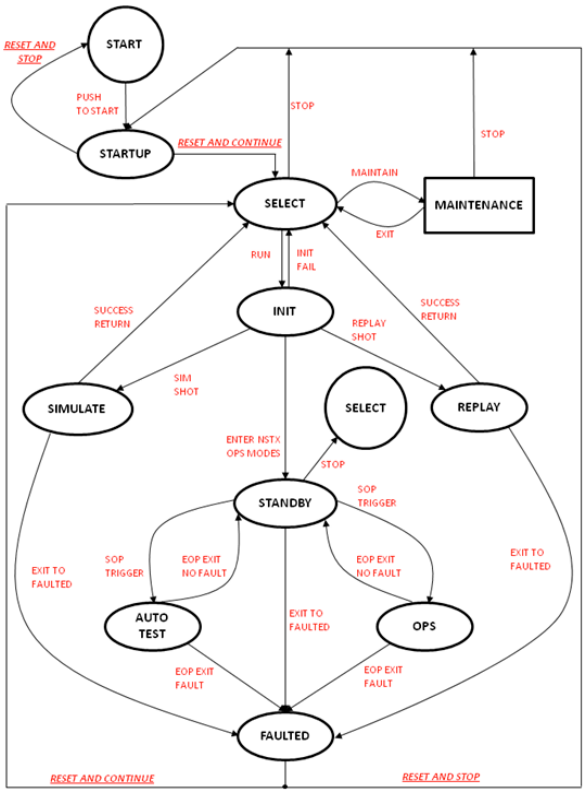

Fig. 1. DCPS State Machine Diagram

It was determined that DCPS should have a set of different states that could be transitioned to based on what the different users of the system would require and its function overall. The choice of an Object Oriented design seemed appropriate in this case as each state required a combination of different, many times overlapping, functions that could be reused and easily interfaced with. Each subsystem behaved like an object. This also naturally leant itself to the parallelization of tasks that would be crucial to achieving our goal of a 200 microsecond rate.

*Work supported by U.S.DOE Contract No. ***DE-AC02-09CH11466***

A. Subsystems

After careful analysis and many long whiteboard sessions a set of subsystems were designated to handle the following tasks:

- A Graphical User Interface component which is the primary front end for most users of DCPS. It provides a means to authenticate, view and modify algorithm parameter data, run simulations, view archived data, change the system state and review logs.
- A Monitoring component that records system state, system status changes, system errors, software errors, coil faults and user actions. It also populates user informational displays via EPICS.
- A Data Management component which archives all forms of DCPS data, initializes and controls hardware I/O devices, outputs system heartbeat, creates and manages system data types, and provides parameter data to other subsystems.
- An Algorithm Management component that will calculate and fault [9,10] if forces that violate definable thresholds are (or will be) present with voltages provided by Halmar signal conditioners to DCPS.
- A Security component that handles user authentication and system privilege assignments based in part on UNIX group assignments.
- A System Manager component that coordinates the overall behavior of the system and alerts other subsystems to changes in state and system status.

By far the most complex subsystem is the one discussed in this paper. It is the primary means by which data is acquired into the real-time system, mainly from hardware inputs, but also from the many MDSplus trees associated with different user types and system functions. Secure, atomic tree creation is a requirement for this project as parameter data contains limits that directly affect the operational envelope of NSTX-U.

All outputs from the DCPS are handled by the data subsystem as well, ranging from physical cable status to coil faults. The collection and conditioning of data is also tasked to the data subsystem and includes an auctioneer system to decide which of two analog input signals is to be used in force calculations.

B. Design Methodology

To further solidify our design and evaluate our intentions, a series of Unified Modeling Language (UML) [7] diagrams were created based on use cases for each user type. These defined interactions were mapped to functions within each subsystem that were directly related to intended use and behavior in each state.

Each subsystem was further defined in functional scope and an Application Programming Interface (API) was created to define the methods used to communicate between subsystems. This also allows complex internal mechanisms to be shrouded behind a simple interface which holds a substantial benefit when multiple developers are working on the same project concurrently, as is the case with DCPS.

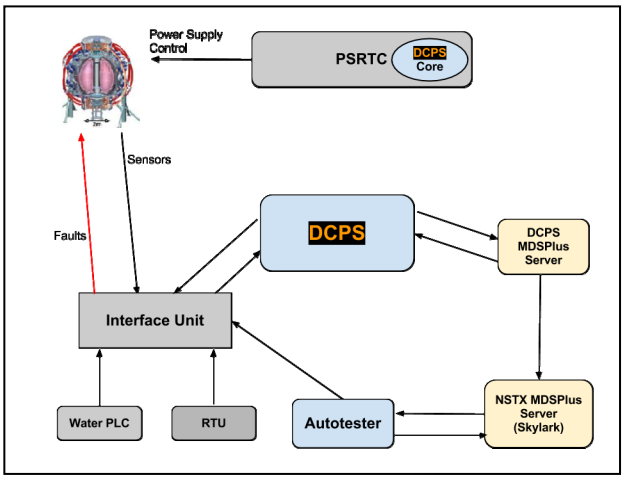

Fig. 2. DCPS System Overview

III. Hardware Considerations

The rigorous demands of a complex system such as DCPS with large amounts of concurrent processing and rapid responses require hardware capable of parallelizing many protection algorithm instances and responding within a tight tolerance. To achieve this, a Commercial Off The Shelf (COTS) product consisting of a Concurrent Computer Corporation (CCUR) Linux based iHawk AMD-based multiprocessor computer was selected. It has 32 cores available and provides a number of tools that simplify the complex task of tuning and coordinating the system[8]. A number of PCI-express based I/O cards were selected to acquire both digital and analog signals, as well as interrupts. Output is only digital in nature and its function is shared with the same card that accepts digital input. Gigabit Ethernet and an Intelligent Platform Management Interface (IPMI) are also included as part of the iHawk system.

A. Timing and Syncronization

In order to properly synchronize DCPS with the rest of NSTX-U, a number of outside time-related signals to ultimately trigger software interrupts were needed. Coordination of the data subsystem acquisition cycle is handled by the CCUR Realtime Clock and Interrupt Module (RCIM). This module is capable of handling twelve external edge triggered interrupt signals and can output as many. Internally it has 12 interrupts that synchronize application code running across multiple CPUs. A 64bit (2x 32bit address space) uninterruptable POSIX clock is included as well . Hardware is clocked internally at 20 Mhz. In addition the RCIM has the capability to synchronize its clock with other RCIMs via a fiber optic daisy chain. Ultimately this

should be used to connect to the NSTX-U Control System (NCS) real time computer running PSRTC.

DCPS requires a number of signals to trigger different types of operation. When a plasma is not being attempted the system is actively acquiring data and processing at a reduced rate. The expectation is that the Halmars will report zero voltage at all times when NSTX-U is idle, and therefore a non-zero value should trigger a fault during that type of operation [14]. In the event of such a condition, the data is archived and stored for review, though this type of data is not considered to be part of the normal NSTX-U shot cycle and is not included in that dataset. Conversely, when the machine is in Plasma Operations mode, transitioning to a plasma attempt at the start of cycle clock event (SOC) prepares DCPS for not only transitioning to real time protection but also triggers the loading and evaluation of parameter data and the creation of MDSPlus shot trees. Depending on the operational state of DCPS, a variety of tree types are available. This is discussed further in section V.

Due to the strong ties between the state of the NSTX-U shot cycle and the envelope that DCPS operates within, it is of the utmost importance to keep the system in lockstep with the outside world. To that end, our design calls for a programmable timing unit driven by the NSTX-U facility clock to send NSTX-U standard clock events like Start of Pulse (SOP) and End of Pulse (EOP) to DCPS to mark changes in operation. It was also determined that an additional non-standard clock signal, designated T-(n), is required. It signals the start of real-time operation sampling rates and causes the last 100 samples to be recorded to a buffer for baseline subtraction calculations. This removed the need for complex hardware configurations such as pre-trigger sampling and allowed for a larger selection of hardware I/O products to be compatible with our design. This was important to our selection of a digital Input/Output card as it allowed a simple, inexpensive card to be used for that purpose.

Response to these clock signals is handled exclusively by the RCIM which is driven directly by the NSTX-U facility clock line. This 5KHz signal directly drives one of the twelve edge triggered interrupts on the RCIM which in turn triggers acquisition from the Analog and Digital boards. This is achieved via the API call “getdata()” to each thread independently. Each of these boards has such a thread running on a dedicated core and thus can operate concurrently. This circumstance is where the features provided by Concurrent RedHawk operating system and Nightstar tools are extremely useful to coordinate and time system functions in a deterministic way. Each 200-microsecond cycle requires a number of actions to be completed, in a very repeatable way with a very tight tolerance. Incomplete cycles are considered a fault of the DCPS and cause NSTX-U plasma attempts to cease. Monitoring this cycle is done externally in a number of ways, the most obvious of which is a system heartbeat which must be generated when the system is active.

B. Digital Output

Digital output is initiated prior to acquiring new input data for each 200 microsecond cycle. Digital I/O is handled by an Adlink 7296 PCI-express card with 96 channels available for input or output. All output from the DCPS system is digital and emanates from a subset channels on the 7296 which has the capability of splitting input or output into banks of 24. Digital output includes the real-time system heartbeat, the lack of which causes a system fault. Additional digital outputs include fault lines, system status and performance information:

- Four coil fault lines exist to differentiate between faults detected in different sets of coils. A line exists for each of the OH, TF PF1 and PF2 coils. These lines are directly tied to algorithms used to calculate the forces acting upon them.
- A system generated loop bit which alternates between a high and low state each time a 200 microsecond cycle is complete. This signal also serves a secondary role as a diagnostic tool used to externally verify system performance. This signal is acquired during plasma attempts by a Dtacq Solutions ACQ196CPCI 96 channel 500 kSPS simultaneous transient digitizer for review by DCPS developers and system administrators in the event of a DCPS triggered fault.
- A watchdog timer to signal to external monitoring equipment (located in the hardware interface box seen in Fig. 2) in the event of a computer or broader software failure. Any disruption to normal DCPS operation (an Operating System lockup, for example) causes this timer to cease and thus inhibit further operation of NSTX-U in any capacity.
- Multiple System Status bits each provide a dedicated line to signal what state the system is in, (see state diagram Fig. 1 for various states), whether I/O cables are connected (cable status) and what operational mode the system is currently in. Further information about the interconnection box is available in Section IV. These signals are also archived externally by the same Dtacq ACQ196 that records the loop bit.

C. Digital Input

In addition to a GUI, a physical Human Machine Interface (HMI) is provided for Fire Control Power Control (FCPC) technicians in the field to perform a number of tasks required for operating the Digital Coil Protection System in the event of a fault. These include the following

- A line to reset and halt DCPS operation.
- A line to reset and continue DCPS operation.
- An override in the event of a TF coil fault
- An override in the event of an OH coil fault
- An override in the event of a PF1 coil fault
- An override in the event of a PF2 coil fault

D. Analog Input

The input of analog signals consisting of Halmar conditioned voltages, Water system status and is acquired by a pair of General Standards AI64 cards which, of course, acquire signals in parallel and are synced by a shared clock line. These input signals are duplicated upstream at the Halmar level and are compared as part of the data conditioning process before they are made available to other subsystems for processing or instrumentation.

E. Input Data Conditioning

Analog data acquired by the data subsystem requires a few steps to be groomed for consumption by other subsystems. Initially, each Halmar signal is duplicated and sent to two different analog input cards for acquisition. Upon calling the external "getData()" function, raw data is acquired from the cards and buffered. Next, a 100-sample moving average from each channel are used to perform baseline subtraction and test the resulting value against a limit value defined for the signal. The pair of signals of similar origin are subsequently compared by a simple "auctioneer" function that selects the channel with the greatest magnitude (see Fig. 3). The channel selected is stored as a boolean value in the same buffered array containing raw channel data, but as a different dimension with the same index and stride.

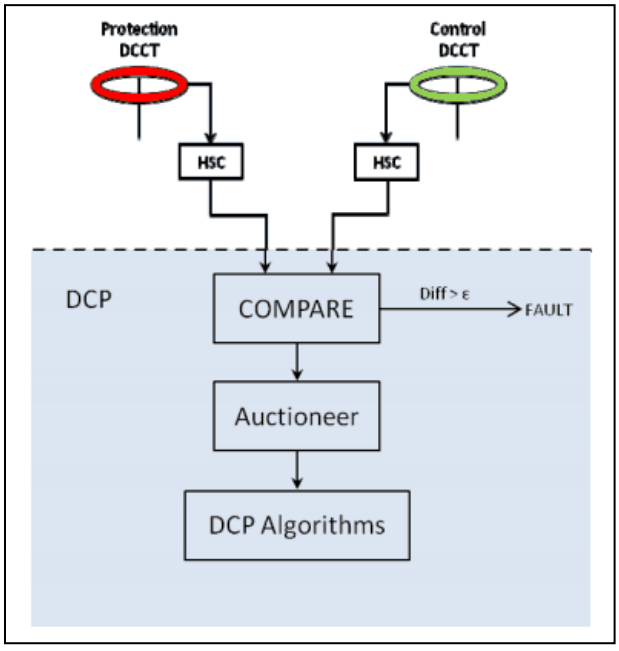

Fig. 3. Analog Data Auctioneer

Once data is conditioned, it is made available for consumption by other subsystems. Under most circumstances, its primary consumer of this data is the Algorithm subsystem. Values are read directly from the buffer and used in calculating physical forces acting upon the machine. Most of these algorithms have a single operation that is performed for each 200-microsecond cycle. A few are dependent upon the results of a prior algorithm execution. As a result it becomes important for diagnostic purposes to save these intermediate values for evaluation in the event of a fault or for instrumentation Algorithm behavior during simulation or operation.

IV. Interconnection & Autotester

All interconnection between physical hardware and the DCPS computer is managed by a rack mounted interface unit. Beyond simply acting like a switch the unit also monitors the Digital Coil Protection System for faults while conditioning and buffering signals being acquired. This unit also provides the ability to change inputs from physical NSTX-U signals to generated signals from the Autotester.

The Autotester is a secondary, stand alone computer system used to simulate a physical connection to NSTX-U. The advantages of such a system are many in that a generated simulation or playback of actual shot data is possible for testing and tuning the DCPS. Being a stand alone system but requiring much of the same functionality as DCPS necessitates the selection of a similar hardware and software infrastructure. The fundamental physical difference between both systems is the type of I/O being performed.

V. Data Archiving and MDSplus Security

Data integrity, in particular parameter and limit data integrity is of the utmost importance to DCPS. The selection of MDSplus to archive nearly all of the data required to operate DCPS was not without its controversy. By its nature, MDSplus was not designed to be a secure mechanism to protect data. It does provide some rudimentary methods to prevent uncontrolled access to, and editing of, trees and shot files based primarily on UNIX file permissions and inbound connection account mapping. While this is sufficient under most circumstances, due to the nature of what DCPS does and how it functions, a higher level of security is required. This was achieved in a number of ways.

A. Passkey Protected Atomic Tree creation

During NCS software development and testing efforts, an idea was brought forth for a method to atomically create a single shot outside of the normal client server model provided by MDSplus. Ideally, this system would prevent users from corrupting or completely overwriting existing shots inadvertently. To accomplish this, a server process was created that handles shot creation for the user and simply returns a valid empty shot number that can be used immediately. To further secure this process and prevent requests from unauthorized users, a passkey is supplied by the client to the server process (Fig. 4) that must match one on record. This also allows a great deal of flexibility in that tree ownership can be set based on a particular passkey, thus allowing multiple users to create "exclusive" sets of trees that are effectively read only. This method also eliminates the need for mapping incoming connections to UNIX accounts at the time of tree creation, allowing processes that

requests trees to run as any user on any system without necessitating a direct map to a UNIX account.

This also removes the requirement that users of the system manually configure system environment variables to define tree locations as this now handled by the DCPS Core itself.

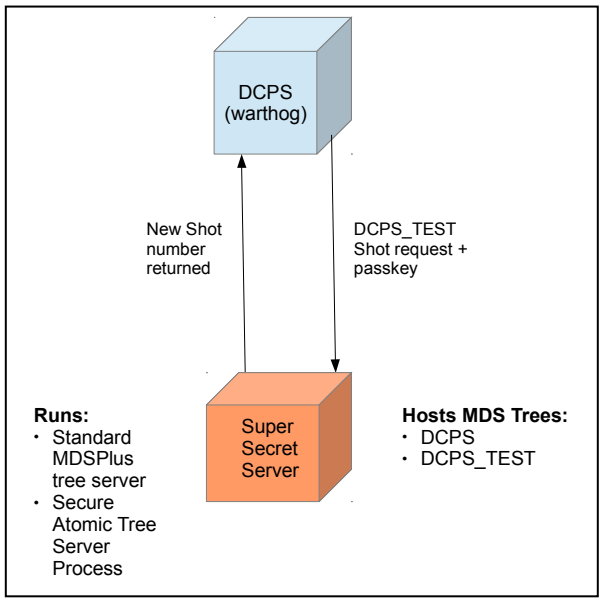

Fig. 4. Atomic Tree Creation Mechanism

B. *Hidden Servers*

MDSplus includes a mechanism to map incoming connections from specified network ports through either a persistent daemon or a UNIX xinetd spawned "mdsip" process mapped and owned by a user account. This requires administration of a "mdsip.hosts" map file that allows both user and machine level mappings to local and user accounts. Usually the UNIX "nobody" account is utilized as a catch-all to prevent unmapped user from viewing or modifying trees that they are not authorized to. This method has many advantages, namely its simplicity and its ability to use UNIX groups to regulate file access permissions. It also has a number of disadvantages:

- Poorly administrated mapping files can allow users to inadvertently modify trees.
- Incoming connections from a machine that has system level mapping to a privileged remote account allows all users of the system to utilize the same account for access. This is not an uncommon scenario.
- Users added to privileged groups for other purposes can gain undesirable access to tree files.

To prevent such issues, a secondary system level authentication mechanism was created and direct connections to the MDSplus server hosting these important trees are prevented. Non-standard "mdsip" ports are used and firewall rules created to further obfuscate access to these trees. Finally a mapping file, fully managed by DCPS administrators, is maintained under revision control with all changes recorded. This file is very simple and only allows the DCPS Data Subsystem process write access to these important tree files.

General access from the NSTX-U MDSplus data-server, Skylark, is read-only and allows users to view and copy (for simulation) DCPS trees but modification and shot creation is not allowed. New trees are instantiated via the atomic creation mechanism and that passkey protected process is fully managed by the DCPS Core computer.

C. *Archiving*

As with the majority of data generated by NSTX-U, DCPS shot data will be archived in an MDSplus shot tree. The reasons for this are many fold:

- Robust visualization and analysis tools already exist
- User familiarity
- Existing administrative support
- Enables creation of simple data sets with the key field being a common shot number
- Acceptable performance
- Compatible with many data types (signals, scalars, strings) with support for a broad range of precision
- Segmented records support large data sets which is important for the increased pulse length of NSTX-U

DCPS will utilize its own private server for archiving DCPS model and shot trees using the atomic method for creation of the former. Both types of trees will be available for viewing from NSTX-U's primary data server Skylark will provide a means for accessing these trees.

VI. Conclusions and Future Work

Creating a system to protect NSTX-U from physical damage is an important step towards its new scientific mission. Exploring its new operating envelope is a required step to achieving that goal. To that end, a new flexible real-time system was needed to test and explore these new regimes [12]. DCPS accomplishes this and works in conjunction with the physical reinforcements made to the structure of the machine. A robust simulation system including both software based and physical signal introduction provides a method for physicists and engineers to test parameters ahead of NSTX-U operation. This expands to include experimental parameters once NSTX-U is commissioned.

Future plans include migration of NCS to the same architecture as DCPS to enable rapid inter-program communication. Additional refinements such as PSRTC and DCPS clock synchronization utilizing the advanced features of the RCIM cards to drive both systems are possible.

ACKNOWLEDGEMENTS

Charles Neumeyer created the top level system requirements document for the overall NSTX-U Digital Coil Protection System [11]. Ronald Hatcher created the software requirements document [6] from which this design derives.

Paul Sichta contributed to both the DCPS hardware interface design and the Reconfigurable Timing Unit development with Steve DeLuca [13].

REFERENCES

[1] Ono, M., Kaye, S., Peng, Y., Barnes, G., Blanchard, W., Carter, M., et al. (2000). Exploration of spherical torus physics in the NSTX device. *Nuclear Fusion*, *40*(3Y), 557.

[2] Dudek, L., Chrzanowski, J., Heitzenroeder, P., Mangra, D., Neumeyer, C., Smith, M., et al. (2012). Progress on NSTX center stack upgrade. *Fusion Engineering and Design*.

[3] Neumeyer, C., Avasarala, S., Chrzanowski, J., Dudek, L., Fan, H., Hatcher, R., ... & Zhan, H. (2009, June). National Spherical Torus Experiment (NSTX) Center Stack Upgrade. In *Fusion Engineering, 2009. SOFE 2009. 23rd IEEE/NPSS Symposium on* (pp. 1-4). IEEE.

[4] Menard, J. E., Gerhardt, S., Bell, M., Bialek, J., Brooks, A., Canik, J., ... & Zolfaghari, A. (2012). Overview of the physics and engineering design of NSTX upgrade. *Nuclear Fusion*, *52*(8), 083015.

[5] Menard, J., Menard, J., Canik, J., Covele, B., Kaye, S., Kessel, C., et al. (2010). Physics design of the NSTX-U. *27th EPS Conf. on Plasma Physics P*.

[6] R.E. Hatcher, "Digital Coil Protection System Software Requirements Document," NSTX-SRD-13-163-0, unpublished.

[7] UML, O. (2011). *2.4. 1 superstructure specification*. document formal/2011-08-06. Technical report, OMG.

[8] Baietto, J., Korty, J., Blackwood, J., & Houston, J. (2008). Real-Time linux: The RedHawk Approach. *Concurrent Computer Corporation White Paper (Sep)*.

[9] Woolley, R., Titus, P., Neumeyer, C., & Hatcher, R. (2011). Digital Coil Protection System (DCPS) algorithms for the NSTX centerstack upgrade. *Fusion Engineering (SOFE), 2011 IEEE/NPSS 24th Symposium on*. IEEE.

[10] Titus, P. H., Woolley, R., & Hatcher, R. (2011). Stress multipliers for the NSTX upgrade digital coil protection system. *Fusion Engineering (SOFE), 2011 IEEE/NPSS 24th Symposium on*. IEEE.

[11] C. Neumeyer, "Coil Protection System Requirements Document," NSTX_CSU-RQMT-CPS-159, unpublished.

[12] Menard, J., & Neumeyer, C. (2009). NSTX Upgrade Scientific Motivation and Project Requirements. *318*, 15-16.

[13] S. Deluca, P. Sichta, & G. Tchilinguirian. (2013, June). Reconfigurable Timing Module for NSTX-U. In *25th Symposium on Fusion Engineering, 2013. Proceedings,* unpublished.