Sādhanā Vol. 33, Part 5, October 2008, pp. 643–661. © Printed in India

Development of a laboratory model of SSSC using RTAI on Linux platform

ANJU MEGHWANI[1] and A M KULKARNI[2]

[1]Emerson Network Power (India) Pvt Limited, Mumbai 400 604
[2]Department Electrical Engineering, Indian Institute of Technology, Mumbai 400 076
e-mail: anju.meghwani@emerson.com; anil@ee.iitb.ac.in

Abstract. This paper presents the implementation of Static Synchronous Series Compensator (SSSC) controller on Real Time Application Interface (RTAI) for Linux Operating System (OS). RTAI provides real-time capability to Linux General Purpose Operating System (GPOS) over and above the capabilities of non real-time Linux environment, e.g. access to TCP/IP, graphical display and windowing systems, file and database systems. Both Type II controllers, DC voltage and current scheduling controllers, are implemented in RTAI. To create a user friendly environment, Graphical User Interface (GUI) is developed in Linux OS in user space (non real-time) using a software available from Quasar Technologies (Qt). The controller is tested on a small scale laboratory model of a Voltage Source Converter (VSC) connected in series with a transmission line. The real time controller performs well in both inductive and capacitive regions.

Keywords. SSSC; RTAI; VSC; Hard real time; GPOS; Type II controller.

1. Introduction

Power transfer capability of long transmission lines is limited by stability considerations. Reducing the effective reactance of lines by series compensation is a direct approach to increase transmission capability; often, it is the most economical solution. Re-routing of power flows in meshed networks is also possible using series compensation devices. The Static Synchronous Series Compensator (SSSC) (Hingorani & Gyugyi 2001) is based on Voltage Source Converter (VSC) (figures 1 and 2) with self commutating devices like IGBTs. It injects variable reactive voltage (voltage in *quadrature* with current) in series with the line. This ensures that no active power exchange takes place with the device except for the power drawn from the system to compensate power losses. This implies that the DC voltage of VSC can be maintained by a capacitor.

While in steady state, reactive voltage injection by SSSC is equivalent to having a capacitor or inductor in series with the line, the SSSC dynamic behaviour is different because of its voltage (reactive) source characteristics. The SSSC has only one degree of freedom, viz. reactive voltage control (unless an energy source is connected on the DC side of VSC which will allow for real power exchange).

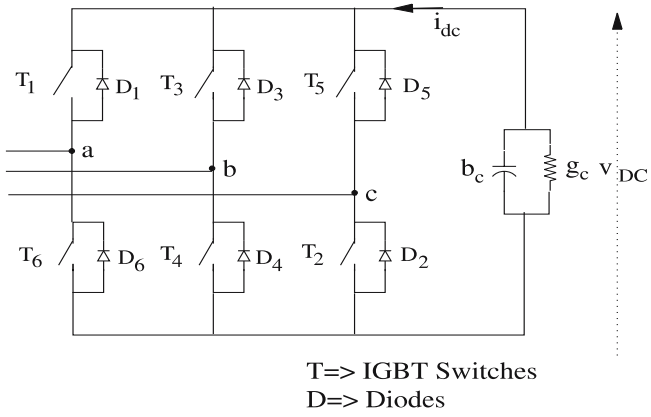

Figure 1. IGBT based VSC.

In this paper, we present a digital control system implemented for a laboratory model of SSSC. In this controller scheme, the modulation index of the converter is kept constant and the capacitor voltage is unregulated (it varies depending on current requirement). This constant modulation index control strategy has the following advantages:

(i) The number of switchings can be kept to a minimum. This is because a degree of freedom in the switching strategy is not lost due to magnitude control requirements. This results in lesser switching losses.
(ii) Converter voltage harmonics are not operating point dependent since modulation index is constant.
(iii) Maximum possible ac voltage can be obtained from a dc voltage by keeping the modulation index at the maximum, resulting in better switch utilization.
(iv) It is ideally suited for multi-pulse and Selective Harmonic Elimination Modulation (SHEM) strategies.

This control strategy has been proposed for static compensator (STATCOM (Schauder & Mehta 1993)) and SSSC (see descriptions in (Guygyi *et al* 1997) and (Sunil Kumar & Ghosh 1999)) and has been termed as Type II controller.

SSSC controller implementation requires two components: a Computer Aided Control System Design (CACSD) and a dedicated hardware with a hard real-time OS. An example of a controller prototyping environment is Matlab/Simulink/Real-time Workshop software, which can be used to generate and compile codes for different targets e.g. Digital Signal Processors (DSPs) and microprocessors (Dong *et al* 2004). The main disadvantage of this solution is the cost of the needed software.

The software described in this paper, to implement SSSC controller, is based on Linux RTAI, a hard real-time extension of GNU/Linux OS developed by Departimento di Ingegneria Aerospaziale del Politecnico di Milano (DIAPM) (Linux). This environment allows to quickly create a real-time controller for real plants by generating and compiling the full controller application from Qt software.

The generated application runs as a normal kernel space hard real-time application on standard x86 computer using drivers, implemented by a programmer to interface with data acquisition boards. The generated hard real-time task can be locally or remotely monitored using an external Graphical User Interface (GUI) application. Qt software (Trolltech) is used to develop GUI for this project.

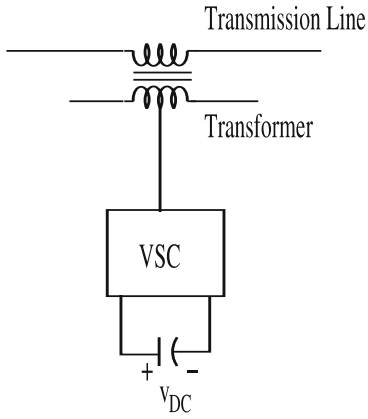

Figure 2. Schematic of SSSC.

The paper is organized as follows. Section 2 describes the mathematical model of SSSC and the control strategy. RTAI architecture and its features are described in § 3. The device driver and GUI application are explained in § 4. In § 5, the results of RTAI controller with hardware-in-loop are discussed. Table 1 provides a list of acronyms used in the paper.

2. SSSC Model and Controller

2.1 *SSSC model*

It is convenient to model the SSSC by converting the 3 phase voltages and currents of the system into D-Q variables using the transformation defined as follows:

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\omega_o t) & \sin(\omega_o t) & \frac{1}{\sqrt{2}} \\ \cos(\omega_o t - \frac{2\pi}{3}) & \sin(\omega_o t - \frac{2\pi}{3}) & \frac{1}{\sqrt{2}} \\ \cos(\omega_o t - \frac{4\pi}{3}) & \sin(\omega_o t - \frac{4\pi}{3}) & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} f_D \\ f_Q \\ f_o \end{bmatrix}, \tag{1}$$

Table 1. Glossary of acronyms.

SSSC	Static synchronous series compensator
PC	Personal computer
PI	Proportional-integral
VSC	Voltage source converter
SHEM	Selective harmonic elimination modulation
GPOS	General purpose operating system
RTAI	Real-time application interface
DAS	Data acquisition system
SCC	Scheduled current controller
GUI	Graphical user interface
PCI	Peripheral component interconnect
API	Application programming interface
CACSD	Computer aided control system design
Qt	Quasar technologies
ADC	Analog to digital converter
IGBT	Insulated gate bipolar transistor

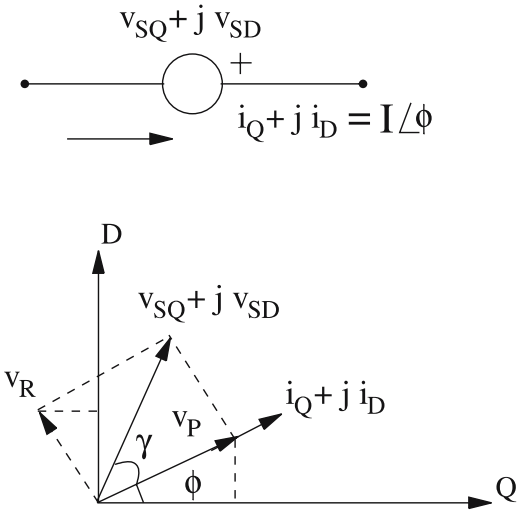

Figure 3. Phasor diagram.

where ω_o is the system frequency (which is assumed to be equal to the base frequency ω_b and is constant). Also zero sequence quantities (f_o) are assumed to be zero. The transformation is power invariant. The injected reactive and real voltage of SSSC are defined in terms of injected voltages in D-Q frame (v_{SD}, v_{SQ}) as follows (figure 3),

$$v_R = v_{SD}\cos(\phi) - v_{SQ}\sin(\phi) \tag{2}$$

$$v_P = v_{SD}\sin(\phi) + v_{SQ}\cos(\phi), \tag{3}$$

where ϕ is the angle made by current space vector ($\phi = \tan^{-1}\frac{i_D}{i_Q}$) as shown in figure 3. In this paper, a positive v_R implies that SSSC injects capacitive voltage and a positive v_P implies that it supplies real power.

Ideally, v_P is zero, i.e. the SSSC does not exchange any real power with the system. However, some power is drawn to compensate for losses. The DC side capacitor is described by the dynamical equation,

$$\frac{dv_{DC}}{dt} = -\frac{g_c\omega_b}{b_c}v_{DC} - i_{dc}\frac{\omega_b}{b_c}, \tag{4}$$

where g_c and b_c are the conductance ($\frac{1}{R}$) and susceptance (ωC) of the DC capacitor, respectively, as shown in figure 1. i_{dc} is the current drawn from the capacitor; it is related to the real voltage (assuming converter is lossless) as follows:

$$i_{dc} = \frac{v_{SD}i_D + v_{SQ}i_Q}{v_{DC}} = \frac{v_P I}{v_{DC}}. \tag{5}$$

Where, magnitude of transmission line current $I = \sqrt{i_D^2 + {\imath_Q}^2}$.

The magnitude of converter output voltage is proportional to the capacitor voltage. The constant of proportionality (modulation index k_m) depends on the switching strategy employed.

Therefore,

$$v_P = k_m v_{DC} \cos\gamma \tag{6}$$

$$v_R = k_m v_{DC} \sin\gamma, \tag{7}$$

where, γ is the angle of injected voltage space vector with respect to the line current vector as shown in figure 3.

2.2 *SSSC controllers*

There are two controller structures, Type I and Type II for SSSC. In Type I controller, both magnitude (modulation index k_m) and phase of converter output voltage (γ) are controlled (Tripathi & Kulkarni 2000). The capacitor is maintained at a constant voltage by controlling the component of injected voltage in phase with line current (V_{Pref}). In Type II controller, reactive voltage can be controlled by phase angle of converter, while k_m is kept constant (Sen 1998). The capacitor voltage is not regulated at fixed value. This can be understood by considering a simple system, consisting of a transmission line connecting two voltage sources as shown in figure 4.

The dynamical equations describing the line current in D-Q variables are given by:

$$\frac{di_D}{dt} = -\frac{r\omega_b}{x} i_D - \omega_b i_Q + \frac{\omega_b}{x}(v_{D1} - v_{D2} + v_{SD}) \tag{8}$$

$$\frac{di_Q}{dt} = -\frac{r\omega_b}{x} i_Q + \omega_b i_D + \frac{\omega_b}{x}(v_{Q1} - v_{Q2} + v_{SQ}), \tag{9}$$

where, r, x are the line resistance and reactance (which includes coupling transformer leakage reactance), respectively.

The steady state behaviour is obtained by setting the Left Hand Side (LHS) of Equations 4, 8 and 9 equal to zero, and solving for v_{SD}, v_{SQ} and v_{DC}. By substituting v_{SD}, v_{SQ} and v_{DC} expressions in Equations 2, 3, 6 and 7 for different values of γ, the following equations for steady value of current and capacitor voltage can be obtained:

$$(r + jx + Z)(i_Q + ji_D) = (v_{Q1} - v_{Q2}) + j(v_{D1} - v_{D2}) \tag{10}$$

$$v_{DC} = \frac{k_m}{g_c} I \cos\gamma, \tag{11}$$

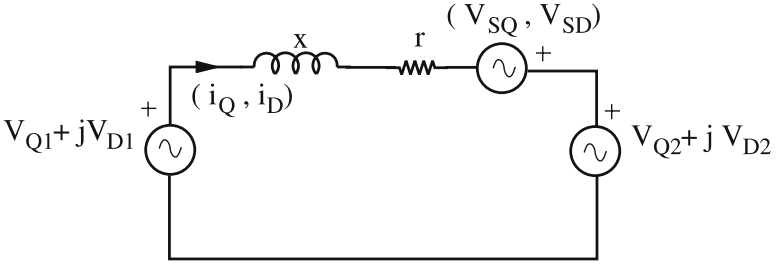

Figure 4. SSSC compensated system.

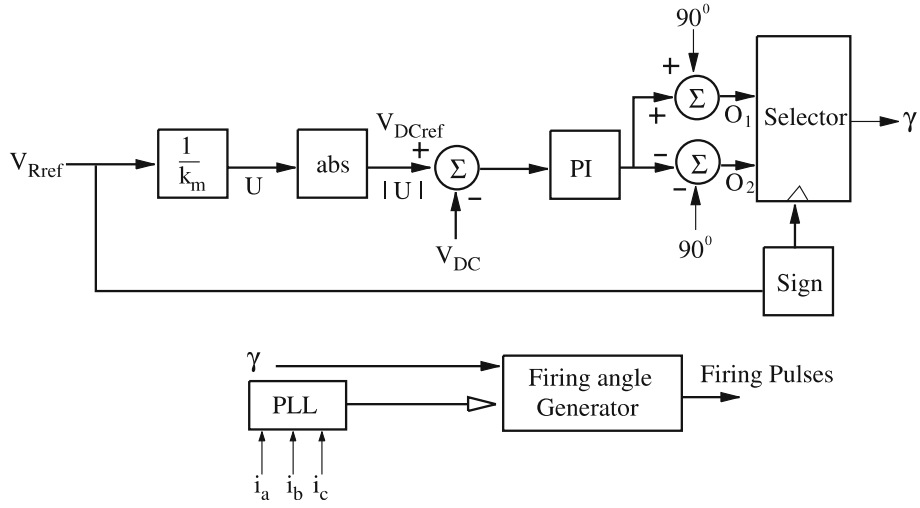

Figure 5. Type II controller.

where

$$Z = \frac{k_m^2}{g_c} \cos\gamma(\cos\gamma + j\sin\gamma). \tag{12}$$

For a fixed value of γ, it is apparent, that in steady state, the SSSC behaves like an impedance Z. Viable operation is possible only for γ slightly smaller than -90° and greater than 90°, i.e. $-90^\circ < \gamma < 90^\circ$.

The angle γ can be controlled to be close to $\pm 90^\circ$ depending on whether reactive voltage reference, V_{Rref}, is positive or negative (figure 5). For positive V_{Rref}, selector input will be O_1 and for negative V_{Rref} it will be O_2. To change reactive voltage, the phase angle varies transiently so as to change the voltage of capacitor. Since, real voltage is very small in steady state for γ near $\pm 90^\circ$, reactive voltage is practically proportional to capacitor voltage by the factor k_m. A closed loop control scheme shown in figure 5 can be used to achieve reactive voltage control. The following limits can be easily incorporated in the controller itself.

2.2a *Voltage limits on SSSC:* This can be easily implemented by disallowing the reactive voltage reference (set by higher level controllers) from exceeding limits.

2.2b *Firing angle limit:* At small values of transmission line current angle it may not be possible to achieve desired reactive voltage, especially *inductive* reactive voltage. This is an inherent limitation due to the fact that capacitor voltage is a function of current magnitude (11) and cannot be remedied by an increase in firing angle.

Another limitation is the problem of over-current. This can be remedied by decreasing the reactive voltage reference when current exceeds the limit. A current controller of the form shown in figure 6 can be used for scheduling current in the line, and is termed as *Scheduled Current Controller* (SCC). The setting of the current reference is done according to steady state power flow requirement. Alternatively, we can regulate the power flow using a power flow regulator instead of current controller. The output of this controller sets the reactive voltage reference.

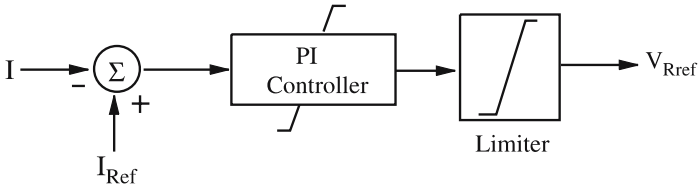

Figure 6. Scheduled current controller.

3. RTAI Linux

3.1 *Background*

Linux is an open source or community effort, multi-tasking operating system. However, Linux does not guarantee response time for its processes, which is the essential requirement of a real-time system. There are application areas which require real-time response, e.g. robotics, computer used in health care and military applications. There are several commercial hard real-time operating systems available, e.g. QNX, AMX. RTAI developers have pointed out that when there are applications of hard real-time, it would be useful to have device drivers, networking protocols and other features offered by Linux along with the real-time features. RTAI allows a user to write 'Hard Real-Time' and 'Soft Real-Time' applications in Linux environment. RTAI is chosen over other alternatives of real-time OS because of following reasons (Seul).

3.1a *Open source:* Linux is an open and free source. In one sense, for the entire base system, which includes the kernel, the GNU tools, and all the basic utilities, as programmers and users have access to the source code as well as the right to modify it (Linux). An important aspect of open software is the ability to write kernel extensions and drivers as needed, which is done in this project. In another sense, acquiring Linux does not require any cash outlay at all.

3.1b *Modular and configurable:* Linux is modular enough to allow the particular service to be cycled without shutting down the entire computer. One can easily reconfigure the Linux operating system to only include those services needed for application. This reduces memory requirements, and may improve performance.

3.1c *Compatible:* Linux has support for older hardware. When old hardware is rendered obsolete by the latest version of Windows (or MacOS, or other OSes), it can most likely still run enough of Linux to be perfectly useful. Linux is backward compatible.

3.1d *Flexible and reusable:* Linux is flexible enough to modify source code to suit needs. In a laboratory environment, the code can be extensively modified and reused.

A real-time system can be defined as a system capable of guaranteeing timing requirements of the processes under its control. It must be fast and predictable. Fast implies, it has a low latency, i.e. it responds to external, asynchronous events in a short time. Lower the latency, better the system will respond to events which require immediate attention. Predictable means, that it is able to determine completion time for the task with certainty (Abbot 2006).

RTAI is a real-time extension for the Linux kernel which allows one to write applications with strict timing constraints for Linux. Like Linux, RTAI is an open source or community effort. RTAI offers the same services as the Linux kernel core, adding features of an industrial

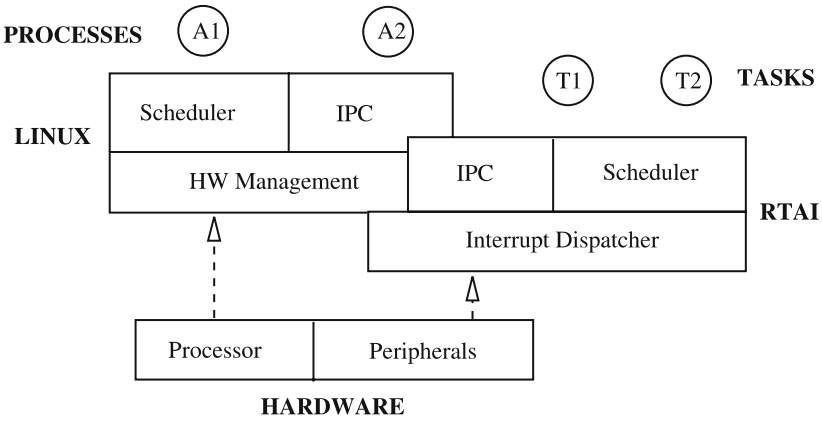

Figure 7. The RTAI architecture.

real-time OS. It basically consists of an *Interrupt Dispatcher*. RTAI mainly traps the peripheral interrupts and if necessary re-routes them to Linux. It is not an intrusive modification of the kernel; it uses the concept of *Hardware Abstraction Layer* (HAL) to get information from Linux and to trap some fundamental functions. RTAI considers Linux as a background task running when no real-time activity occurs.

3.2 *RTAI architecture*

Figure 7 shows the basic architecture of RTAI (Sarolahti 2001). The interrupts are originated from processor and peripherals, out of which the processor interrupts (mainly error signal such as division error) are still handled by the standard Linux kernel. However, interrupts originating from the peripherals (e.g. timer) are handled by RTAI's *Interrupt Dispatcher*. RTAI forwards an interrupt to the standard Linux kernel handler when real-time task is not active. If the interrupts are disabled in standard kernel, RTAI queues the interrupts and delivers it after the Linux kernel has enabled the interrupts again.

Figure 7 also shows the interprocess communication mechanisms and scheduler, which are implemented separately for the Linux and for the RTAI. The scheduler exists separately for Linux side and RTAI side.

3.3 *RTAI features*

3.3a *Communication with linux processes:* RTAI supports two mechanisms to provide communication between kernel space and user space Linux processes. These are:

- Real-Time - First In First Out (RT-FIFO): RT-FIFO is a point-to-point link connecting one real-time task to one Linux process (like Linux pipe). User space treats RT-FIFO as a character device. A process opens a FIFO for reading and writing, and then uses *read* and *write* on file descriptor to transfer data.
- Shared Memory: In situations where large amount of data, like video frame buffers, have to be moved quickly between a real-time task in kernel space and one or more processes in user space, shared memory is preferred over RT-FIFO.
  In RT-FIFO, data is transferred in a serial order. Delay in transferring the data depends upon size of FIFO and type of data to be transfered while in 'Shared Memory', the data

is simultaneously transferred in parallel order. Therefore, in SSSC controller application 'Shared Memory' is used to transfer data from GUI to kernel space real-time program.

3.3b *Timer tick interrupt:* RTAI supports two modes of timer tick interrupt.

- Periodic mode: In this mode the timer is set to interrupt at the pre-specified period. When the timer counter overflows and generates the interrupt, it is automatically reloaded with the correct starting number and started again. It is a timer free run.
- One-shot mode: Periodic mode lets the timer free run, interrupting at the specific interval. While one-shot mode reprograms the timer every time it is interrupted. At every tick interrupt, time for next 'event' is computed and programmed into the timer for that interval. The resolution of the timing interval is now determined by the clock driving the timer, and not the periodic interval of timer interrupts.

3.4 *Software specifications*

RTAI itself is not a complete OS. It is an add on to Linux which provides real-time capability to GPOS. The work presented in this paper is implemented on RTAI version RTAI-3·5 installed on Linux kernel 2·6·19. It is not possible to install RTAI on any kernel version. Its compatibility with kernel version is mentioned in RTAI repository itself. For example, RTAI-3·5 can be installed on kernel 2·6·19, 2·6·17, 2·4·32 and other kernels as mentioned in RTAI installation directory (RTAI). The code developed on RTAI is in C language which includes RTAI and Linux kernel libraries (installed as a part of RTAI). The code is compiled using *MAKEFILE*, available on site (RTAI Examples), which uses *gcc* compiler. To execute the code in hard real-time, the executable file is inserted into kernel space using *insmod* command.

A device driver is a low level language kernel program in which any hardware device can be accessed by its peripheral address. Some high level language device drivers are available (peripheral address not required) e.g. Comedi (Comedi). In this project device driver code is developed (for Advantech 1710 HG (Advantech)) because of time criticality issues. It is observed that, execution time taken by Comedi device driver is of 100 μs while developed code takes only 3 μs.

The GUI is developed on Qt Designer. It is developed to display data on terminal and to provide control signals to user. The user can select the mode of controller in GUI and simultaneously set V_{Rref} or I_{ref} for respective controllers. The output of controller is displayed on GUI. RTAI shared memory feature is used to store kernel space data in memory, which is also accessed by user space (GUI). A typical process flow for developing RTAI code along with Qt is shown in figure 8.

4. Simulation set-up

4.1 *The physical model*

The controller implemented on RTAI is tested on the scaled model of the VSC module connected in series with the line as shown in figure 9. The 3-phase bypass switches are connected across the transformer and bypass switch is opened. Before opening of bypass switch, pulses should be given to IGBT's gate. When the pulses are given, VSC starts injecting the voltage in transmission line. The VSC employs six IGBTs connected in anti parallel with diodes. Semikron SKHI 22 A IGBT driver chips are used to drive the IGBTs. The Pentium IV processor based Personal Computer (PC) is used for implementation of real-time controller

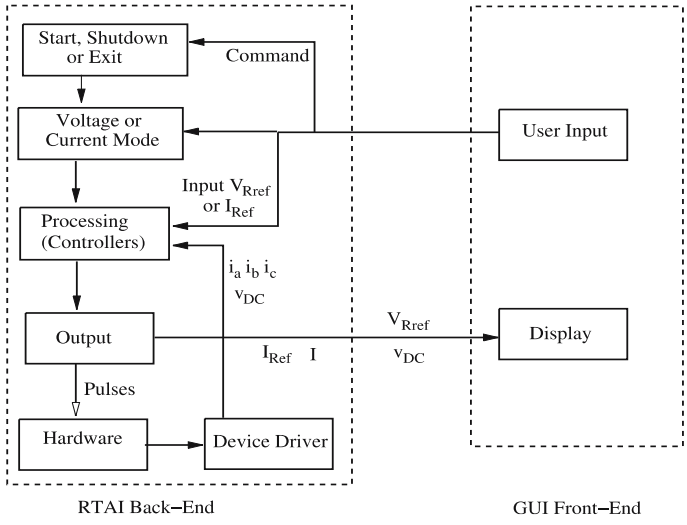

Figure 8. Process of RTAI and GUI.

on RTAI. The hardware set-up of SSSC along with the transmission line model is shown in figure 10.

4.2 *Data acquisition system (DAS)*

The main task of DAS is to acquire and preprocess 8 or 16-channels analog signals that are measured through voltage and current sensors. The inputs to Peripheral Component

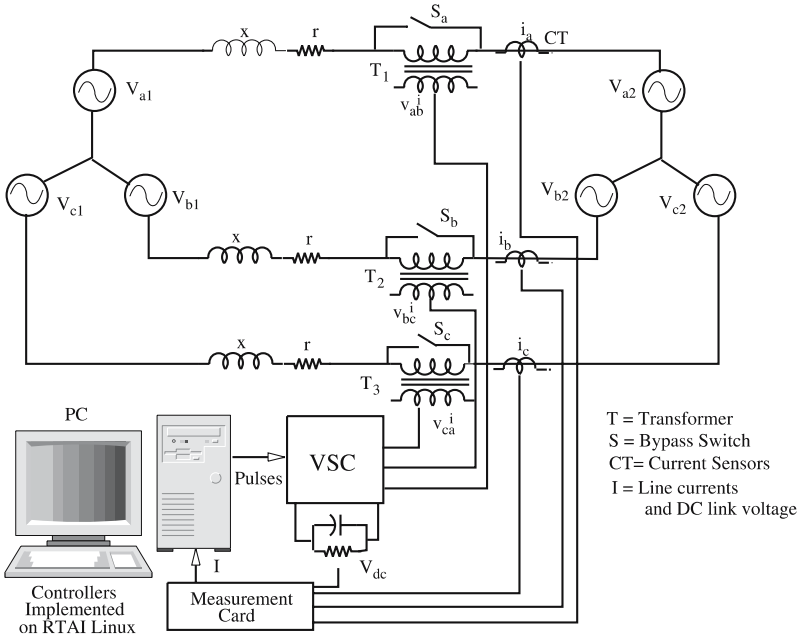

Figure 9. Project set-up.

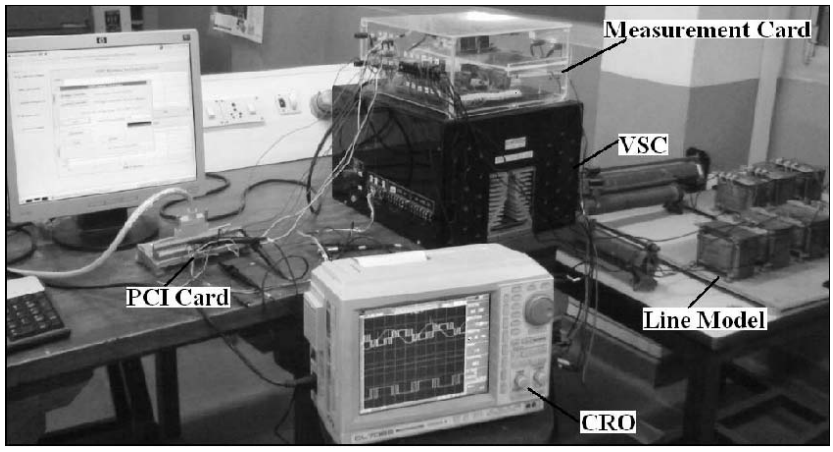

Figure 10. SSSC with transmission line model and GUI.

Interconnect (PCI) card (which is connected to PCI slot of PC) are 3-phase line currents (i_a, i_b, i_c), 3-phase voltages at VSC terminal ($v^i_{ab}, v^i_{bc}, v^i_{ca}$), and dc link voltage ($v_{DC}$). These data are measured by the *measurement board* as shown in figure 9. In the board, galvanic isolation is provided between the high voltage circuit and secondary output electronic circuit using ISO122 chips. These inputs are given to PC via PCI card (consist of ADC, DAC channels). PCI card *Advantech 1710 HG* is connected to the PC by using the available PCI slot. The analog channels of ADC are multiplexed and each channel is scanned at every 40 μs. Therefore, a given variable will be sampled every 280 μs.

The output of controller (γ) is given to a pulse generator which is also a real-time task. 3-phase pulses are given to VSC via digital output channels. Since, the signals are too weak to drive VSC, these are given to *buffer board*. The board consists of Operational Amplifiers (Op-Amps) for isolation, amplification and error detection. The error detection circuits block firing pulses to IGBTs if there is any short circuit in dc bus or low voltage supply to IGBT driver board.

4.3 *Firing controls for VSC*

Firing pulses are synchronized with ac system in quadrature with line current using a PLL which is a second order closed loop system. The PLL shown in figure 11 is essentially the same as presented in Gole *et al* (1989). PLL is based on the fact that the dot product of ac quantities, which are in quadrature, is zero. U_1 is input to PLL and U_2 is output which is in quadrature with input. The error e $(U_1 \cdot U_2)$ is given to PI compensator. If it is given a high loop gain, it tends not only to follow ac system phase with small transient error, but also responds to noise introduced into its input. If the PLL loop gain is reduced, it follows variations in ac system frequency with a slower response time but is less sensitive to noise. Therefore, there is a trade off involved between speed of response and noise immunity. The Voltage Controlled Oscillator (VCO) is used to generate a sinusoidal and co-sinusoidal waveform. In steady state, the PLL tracks ac system phase with zero error.

The output of PLL along with γ is given to firing angle generator block to generate firing pulses for IGBT using SHEM. SHEM (Rashid 1993) is used to generate the pulses for VSC. SHEM offers a tight control of harmonics spectrum of a given voltage waveform generated by

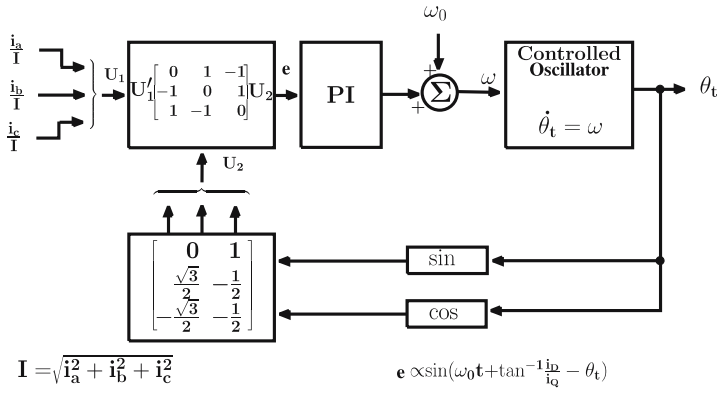

Figure 11. Phase locked loop.

power electronics converters along with minimum number of switching transitions (Patel & Hoft 1974). This eliminates 5^{th}, 7^{th} and 11^{th} order harmonics present in the output voltage of converter. This ensures that the line current is practically sinusoidal since the transmission line inductance serves to reduce the higher order harmonics.

4.4 *RTAI modules*

Controllers are programmed in *C* which can be easily implemented and modified. The real-time modules are dynamically loaded in the form of kernel modules in RTAI using *insmod* (Bovet *et al* 2002) command and removed by using *rmmod*. The PC onboard timer is used to manage the data-acquisition sampling time. The time step selected for the PI compensator for Type II controller is 20 μs. In other words, PLL gives the pulses to the converter every 20 μs. Since the PI compensator gives output every 20 μs, but data is sampled at an interval of 280 μs, one can go for either zero order hold (data remains same for next 280 μs) or give the approximate data input to PLL every 20 μs using linear extrapolation. The data is linearly extrapolated using past two samples of data as shown in figures 12 and 13.

For the higher order current controller, which requires a slower response time, the time step chosen is 1 ms. *Forward Euler's* method is used to numerically solve the integrator of PI compensator. The method is stable for small time steps.

In table 2, different tasks, their functions, execution time, periodicity and priority are listed. These tasks are implemented in kernel space. The static priorities are assigned to each task according to their functionality and sequence of execution. For example, output of PLL is given to controller and then pulses will be given to VSC as per the dc voltage reference. To generate pulses, PLL should complete its task before Type II controller. The task with lowest periodicity is assigned the highest priority. Zero priority in RTAI is ranked as the highest

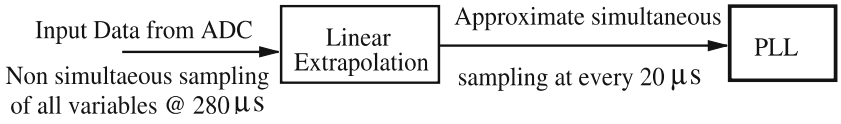

Figure 12. Flow chart of data transfer from ADC to PLL and controller.

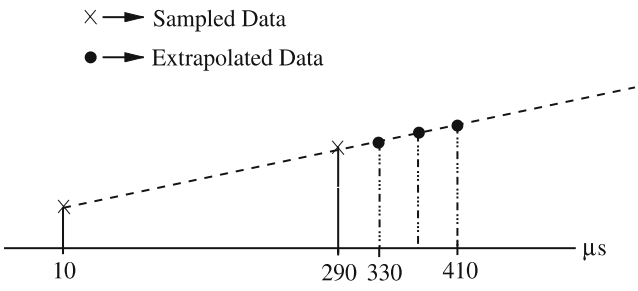

Figure 13. Linear extrapolation of previous sampled data.

priority. If two or more tasks are in ready queue, then scheduler and resources will be assigned to the task which has the highest priority.

Since the execution time of task is much less than its period, the remaining time, $t = \textit{Periodicity} - \textit{Execution Time}$, is utilized by the Linux kernel. This is because Linux is scheduled as a lowest priority task in RTAI scheduler. Therefore, one can easily access other facilities of Linux General Purpose Operating System (GPOS).

4.5 *Issues in implementation*

There are some issues involved in implementing the controllers on RTAI. These are:

- SSSC controller is implemented on RTAI which is installed on Pentium-4 processor of 3·2 GHz. The minimum tick period observed for the processor is 20 μs with $3 - 4\,\mu$s of jitter. CPU resources (processor, memory, timer, etc.) are shared by real-time tasks on the basis of time. In a dual core PC, two processors are available. This implies that two real-time tasks can be run simultaneously, sharing the same memory. The SSSC application is also tested on Intel Core 2 Duo PC of 1·86 GHz processors.
- The latency of the processor can be reduced by selecting few options while installing RTAI on PC. These options are:
  - Disable DMA
  - Disable video mode
  - Disable power management support
  - Disable USB support in PC BIOS.

  By disabling all supports, the jitter can be reduced to 1–2 μs and tick period of 15–18 μs can be achieved.

Table 2. Tasks and their functions on RTAI.

Tasks	Processes	Execution time (μs)	Periodicity (μs)	Priority
Task_1	PLL, Type II PI and SHEM	5	20	0
Task_2	Data scanning and extrapolation	5	40	1
Task_3	Current controller PI	3	1000	2

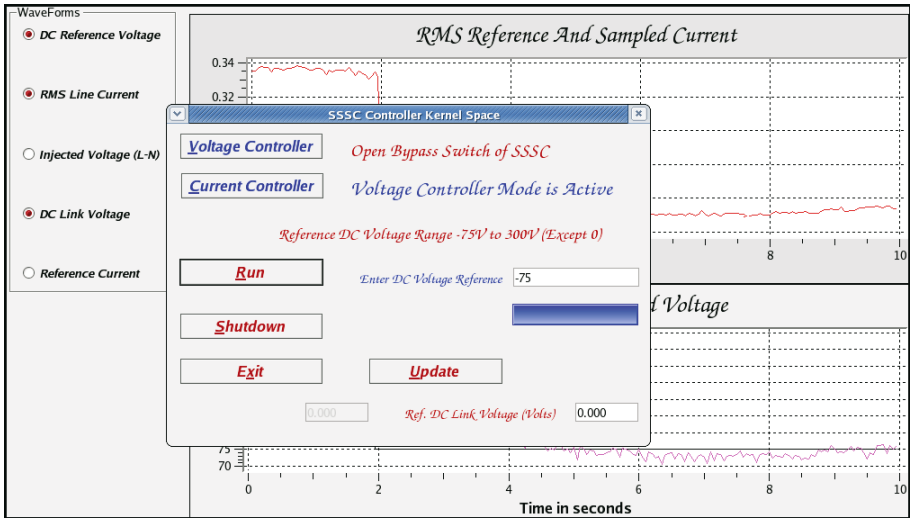

Figure 14. Screen capture of GUI showing waveform recorder window and reference window.

- The risk involved in developing the application code at kernel level is that the critical memory area is unprotected. A bug in a program can cause a corruption in internal data and a system crash. To avoid this, program is first written in user space. After successful compilation and execution of program, it can be changed to kernel space by changing some Application Programming Interfaces (API)(RTAI Examples).
- To transfer data from real world to PC, PCI card is used as interface. Device driver for PCI card is written in Linux and dynamically inserted into kernel space.

4.6 *SSSC GUI*

GUI is created in *Qt-3·3 Designer* for real-time SSSC controller application. It is a C++ toolkit for multi-platform GUI and application development. Qt provides single source portability across MS Windows, Mac OS X, Linux and all major commercial Unix variants. Qt is a Trolltech product (Trolltech). Here, the user level program, which interacts with user for reference value to Type II and current controller, is implemented. This reference input is given to real-time task running in kernel space. The communication between user space task and kernel space task is done via shared memory. A chunk of memory is allocated to transfer data from real to non real-time system. The GUI is shown in figure 14. GUI has following features.

4.6a *System initialization:* The ADC and device driver are first initialized. Therefore, while booting the PC, device driver for ADC card is inserted in kernel space to find base address of ADC for input/output. SSSC controller real-time task is also inserted into kernel when this routine executed. The PLL and rest of the program is run and pulses are given to the IGBTs before the bypass switch is opened.

4.6b *Mode selection:* The SSSC may be operated in a SCC mode wherein V_{Rref} is set by SCC in order to obtain desired current, or in voltage control mode, wherein V_{Rref} or equivalent V_{DCref}, as shown in figure 5, is kept constant and directly set by the operator. Voltage and current controllers are implemented in the same routine but with different task

priorities and periodicity as explained in table 2. Current controller is much slower than the voltage controller. Switching from one to another control mode may cause transients which have to be taken care of while programming. If a user switches from current to voltage control mode, V_{Rref} will remain at the value set by the current controller until the new value is given by the operator. All the initial values of PI compensator of current controller are initialized to zero. While switching from voltage to current control mode, rms current reference set by voltage controller is kept constant at the value it was at before the change of mode is given.

4.6c *Data display:* In this GUI, online wave recorder is also implemented. The radio buttons are provided to select different waveforms (actual dc voltage and line rms current). Since the GUI is non real-time, there is a time delay in data display on screen with respect to data in kernel space.

4.6d *System shutdown:* It is possible that VSC injects its maximum capacity of reactive power for compensation, so it is advisable that before exit from routine the system is properly shutdown. For this, low dc voltage reference is given to voltage controller. After this, all upper IGBTs are turned on while all lower IGBT switches are kept off. This means that before closing bypass switches, the operator has to inject zero voltage in series of transmission line. In the end, the user quits from subroutine by pressing an exit push button.

Real-time applications are often required to be run in kernel space in-order to bypass Linux kernel and its standard interrupt handlers. By doing this, the real-time task becomes predictable, and the response time can be bounded. However, GUI is non real-time in user space. To share common data between these two, shared memory has been used.

5. RTAI controller results

For the following ratings of hardware devices, the PI controller for PLL, Type II controller and current controller are tuned on MATLAB and then implemented on RTAI. The devices and their ratings are as follows:

1. Fundamental switching frequency: 50 Hz
2. Semikron IGBT bridge inverter: B6CI 750/415 30 F
3. Line inductance: 663·2 mH per phase
4. Line resistance: 20 Ω per phase
5. DC shunt capacitance: 100 μF 450 V DC
6. Shunt resistance: 27 kΩ
7. PCI Card: Advantech 1710 HG
8. Receiving end L-L rms voltage: 230 V$\angle 0^\circ$
9. Sending end L-L rms voltage: 230$\angle 30^\circ$
10. Nominal reactive voltage V_{Rref} range: 78 V (capacitive) to $-19{\cdot}5$ V (inductive)
11. Maximum line current: 0·6 A

Note: The VSC and capacitor rating are not optimized for the rating of SSSC. However, they are sufficient for the same.

5.1 *Response of type II controller*

The response of Type II controller for a step change in V_{Rref} is shown in figure 15. The outer scheduled current controller is disabled because no current reference (which is displayed

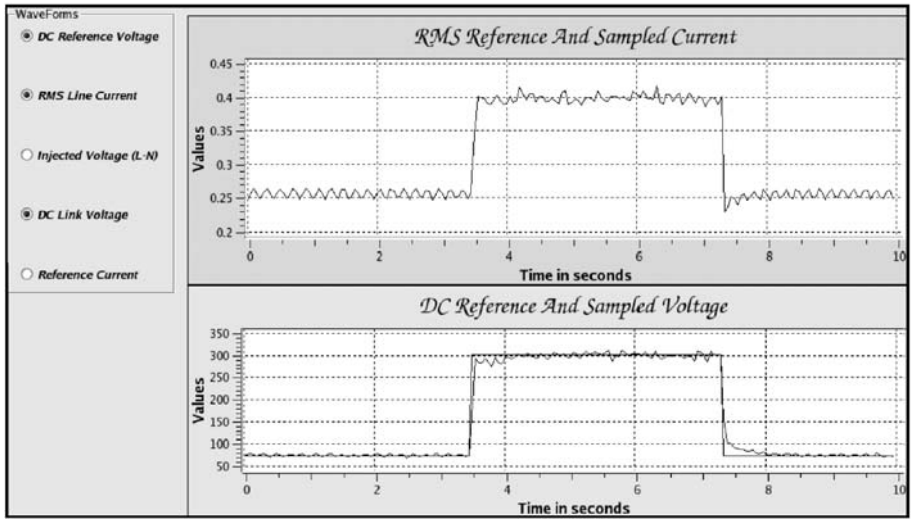

Figure 15. Type II controller response: v_{DC}, v_{DCref} and I are displayed on GUI (Voltage Control Mode).

in figure 16) is displayed in window. The data displayed in 'The Reference Current And Sampled Current' window is the actual rms line current. The rms value of line current is calculated in RTAI task using the expression $\sqrt{\frac{i_a^2+i_b^2+i_c^2}{3}}$, where i_a, i_b and i_c are instantaneous sampled currents in three phase (figure 9). The actual response time is of the order of one cycle. In figure 15, it is of several seconds. It is deliberately made slow to capture the profile of currents and voltage in transient conditions. Note the change in voltage reference causes a corresponding change in line current.

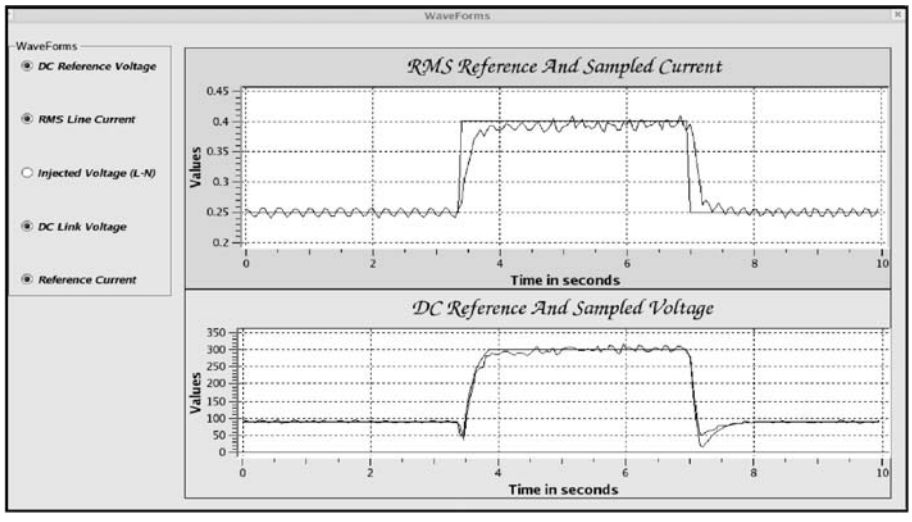

Figure 16. Scheduled current controller response: v_{DC}, v_{DCref}, I and I_{ref} are displayed on GUI (Current control mode).

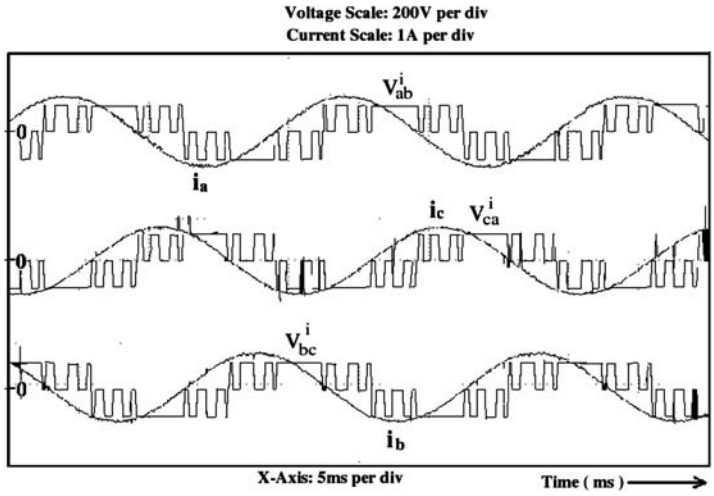

Figure 17. Internal controller: Three phase L-L voltages at VSC terminals and line currents in capacitive mode (captured on oscilloscope).

5.2 *Response of scheduled current controller*

The response of the controller, when current is increased from about 0·85 times to 1·35 times of its uncompensated value, is shown in figure 16. The actual rms line current follows the step change given to controller reference as shown in 'The Reference Current And Sampled Current' window. The scheduled current controller sets the reactive voltage reference. If the line current is to be reduced below the uncompensated level, then V_{Rref} is made inductive. Similarly, if it is to be increased above the uncompensated level, then V_{Rref} is made capacitive.

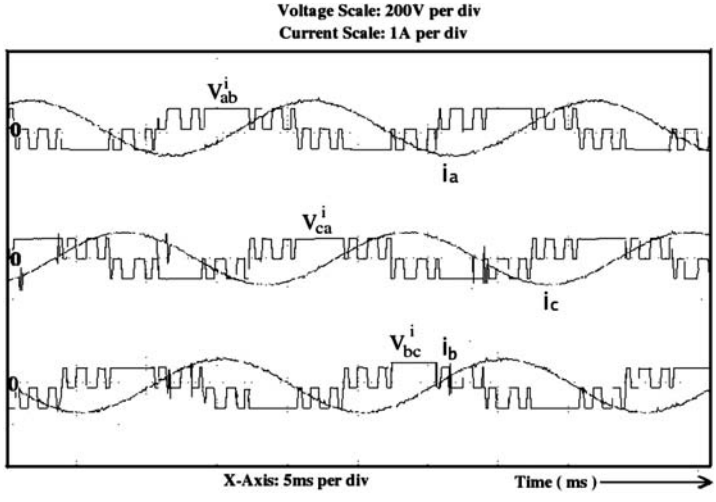

Figure 18. Internal controller: Three phase L-L voltages at VSC terminals and line currents in inductive mode (captured on oscilloscope).

The steady state line-line voltage and current waveforms for capacitive and inductive mode are shown in figures 17 and 18, respectively, where, v^i_{ab}, v^i_{bc} and v^i_{ca} denote the voltages across the VSC terminals.

6. Conclusion

A stable and completely operational controller for the SSSC has been implemented using RTAI of Linux. Results obtained confirm theoretical predictions of the controller capability and functionality. It is observed that while executing real-time applications, one can access non real-time Linux environment e.g. TCP/IP, graphical display, etc. without affecting its performance. The controller is working in its inductive and capacitive regions for both voltage and current control modes. Switching from voltage to current control mode and vice-versa has been made jitterless.

A GUI for SSSC controller has also been implemented on Linux environment. The real-time application are often required to be run in kernel space in order to bypass Linux kernel and its interrupt handlers, because of which the response time of tasks can be bounded. GUI is a user space application, which is non real-time and used to display data of the real-time tasks. To share data between real and non real-time task shared memory has been used, which gives satisfactory results in data transfer.

The financial support received under Research Scheme on Power (RSOP) scheme from Central Power Research Institute (CPRI), Ministry of Power, Government of India is gratefully acknowledged.

References

Abbot Doug 2006 Linux for embedded and real-time application. *Newness Publication* Printed in United States of America

Bovet Daniel P, Cesati Marco, Oram Andy 2002 Understanding the Linux Kernel, Second Edition, *O'Reilly and Associates, Inc., Sebastopol: CA* Prentice Hall Publication

Dong L, Crow M L, Yang Z, Shen C, Zhang L, Atcitty S 2004 A reconfigurable FACTS system for university laboratories. *IEEE TPWRS* 19(1) 120–128

Gole A M, Sood V K, Mootoosamy L 1989 Validation and analysis of grid control system using D-Q-Z transformation for static compensator system. *Canadian Conference on Electrical and Computer Engineering, Montreal PQ, Canada* 745–748

Guygyi L, Schauder C, Sen K 1997 Static synchronous series compensator: A solid state approach to series compensation of transmission line. *IEEE Transaction on Power Delivery* 12(1): 406–417

Hingorani N G, Gyugyi L 2001 Understanding FACTS. *IEEE Press*, Standard Publishers Distributors

Patel Hasmukh S, Hoft Richard G 1974 Generalized techniques of harmonic elimination and voltage control in thyristor inverters: Part II-Voltage control techniques. *IEEE Transaction on Industry Application* 1A–10(5): 666–673

Rashid Muhammad H 1993 Power electronics: Circuits, devices and applications. Prentice Hall Publication

Sarolahti Pasi 2001 Real-Time application interface. University of Helsinki, Department of Computer Science

Schauder C, Mehta H 1993 Vector analysis and control of advanced static var compensator. *IEE Proceeding-C* 140(4): 299–306

Sen K K 1998 SSSC-Static synchronous series compensator: Theory, modelling and application. *IEEE Transaction on Power Delivery* 13(1): 241–246

Sunil Kumar L, Ghosh A 1999 Modelling and control design of static synchronous series compensator. *IEEE Transaction on Power Delivery* 14(4): 1448–1453

Tripathi R H, Kulkarni A M 2000 Phase angle based reactive voltage control of static synchronous series compensator. *National Power Systems Conference (NPSC 2000)*, Bangalore

Seul, Available: http://www.seul.org/docs/whylinux.html

Comedi, Available: http://www.comedi.org/doc/index.html

Advantech, Available: http://www.advantech.com/support

Trolltech, Available: http://www.trolltech.com/qt/

RTAI, Available: http://www.rtai.org

RTAI Examples, Available: http://www.captain.at/programming/rtai/examples.php.