21 Supervisory Control and Data Acquisition

21.1 INTRODUCTION

Complex and dispersed power systems necessitate large manpower resources for control, maintenance and management functions. Such resources may be reduced or employed more efficiently with the help of computer systems. This chapter describes the basic interfaces, software and hardware necessary for transmission and distribution power system supervisory control and data acquisition (SCADA). Programmable logic controllers (PLCs) and substation bay controllers may be used for local automatic control functions. These controllers are described together with a practical interlocking application example. This chapter also introduces traditional power line carrier communication and signalling methods. Communication via fibre optic links is mentioned separately in Chapter 12. This chapter goes on to describe a centralized power transmission network control system and covers the very important subject of software management. Such management is essential if software development is to be achieved within quality, time and budget constraints.

21.2 PROGRAMMABLE LOGIC CONTROLLERS (PLCs)

21.2.1 Functions

Programmable logic controllers (PLCs) were initially developed for discrete control applications in machine and materials handling production engineering environments. The on-going development of PLCs for the control and monitoring of industrial systems has increased their capabilities from simple hard wired logic elements (NAND, NOR gates) to advanced functions using software-controlled microprocessors for piping and instrumentation diagram (P&ID)
algorithms, floating point arithmetic, network communication and multiple processor configurations for parallel processing. Modern PLCs are capable of handling power system local control automation requirements. IEC 61131 is rapidly becoming the internationally recognized standard for configuring PLCs. PLCs evolved to become bay controllers, which are basically items of relay grade hardware that are directly connected to instrument transformers (CT/VT) in a substation and have built in binary input and output modules together with a large logic function library that can then be programmed like a PLC. The significant difference between a PLC and a bay controller is the fact that bay controllers are built with substation requirements in mind such as EMC, high making and breaking capacities, added relaying functionality, etc.

21.2.2 PLC selection

21.2.2.1 Control and monitoring specifications

The development of a control system may be divided into various stages as shown in the project development life cycle diagram (Fig. 21.1). A management decision, based on timing and resource availability, is made as to the best stage to obtain competitive tenders for remaining design, supply and installation work from specialist contractors. The first step is to carefully detail the system to be controlled together with possible future expansion requirements. This initial description must carefully detail the hardware and software interfaces and addresses such questions as the physical location of devices, supervisory control connections, motor or actuator loads and physical enclosure protection.

The second step is to define the operational control requirements in a concise and accurate descriptive form. At this second stage, it is essential that full consultation is made with operatives and maintenance crews as well as the engineers in order to ensure the correctness of the descriptions and definitions of the user's wishes. These descriptive control requirements are then converted in a particular format as a sequence of logical events.

A specification (sometimes termed Functional Design Specification or FDS) is next prepared for both the hardware and software. The hardware specification should cover the following points:

- conformity requirements with any existing systems
- communication gateway (RS232/485, etc. or fibre optics glass/plastic) and associated ‘protocols’ (the protocol is the transmitting/receiving data exchange rules which govern the message format, timing and error checking)
- the input and output devices to be connected either directly or via interposing accessories
- power supply requirements
- codes and applicable standards
- installation environment (enclosure protection, temperature, humidity, etc.)
- Functional requirement
- Performance
- Interface (hard and software)
- Environment and system requirement
- Hardware architecture definition
- PLC choice
- Hardware specification (module, interface, etc.)
- Detailed soft design (algorithm and sequential logic)
- Loop and connecting diagram, equipment drawing, etc.

Basic requirements user (or process needs)

Possible early tender stage

Functional analysis and hardware requirement

Tender stage often used

Detailed design hardware and software

- Performance and system tests
- Site
- Factory

System acceptance

Functional test

- Software validation test
- Hardware test with soft in factory

Limit test

- Basic module software test
- Electrical testing on hardware

Software development and hardware building (wiring and cabling control panel)

Figure 21.1 PLC system development lifecycle
Supervisory Control and Data Acquisition 767

- factory and site testing requirements
- documentation and quality assurance (QA) requirements.

The software specification provides a complete and definitive statement of what the control system has to do but not, at this stage, how to do it. It provides the basis for the system design and implementation, and includes both the descriptive and logical sequences and functions taking into account:

- functions to be implemented (Boolean or sequential, P&ID functions, maths, etc.)
- data exchanges (type of information per actuator or motor, analogue values, commands to be exchanged, etc.)
- complete input/output listing
- system software (redundancy or self-diagnostic)
- support structure (programming tool giving access to PLC for on or off-line testing and diagnostics)
- factory and site testing requirements
- documentation and QA requirements.

A search is next made for the PLC system that is best fitted to these carefully defined needs at the most competitive price. The size of the PLC system is determined from what tasks it is required to perform by defining the input/output requirements, the memory size and spare capacity. Other requirements include the piping and instrumentation (P&ID) loop control, floating point maths to perform the calculations and special functions such as time delays. This is the stage for selection from the various options for the most appropriate technical solution.

21.2.2.2 Technical solutions

The options will comprise of a list of hardware and software components which will implement the functions specified in the system specifications. Once the choice is made detailed design of the PLC system follows. The various algorithm and logical sequences, time delays, fault treatments and data exchange tables used have to be validated through the detailed software design document. Such a document may use:

- logic blocks or ladder logic in line with ISA Standards;
- organograms if development is in the specific pseudo software language used by a particular PLC supplier;
- IEC 61131–3, programming languages for programmable controllers.

The ‘response time’ of the PLC is the time it takes to translate a change on an input to effect an output. This is not the same as the ‘scan time’ which is
only one of the response time elements involved. The response time takes into account:

- the input/output update times
- the times to process counters, timers and mathematical instructions
- communication times if the PLC is part of a network control system.

A typical example would be a PLC scan of 1000 instructions in 10 ms and a response time of 35 ms per 1000 instructions.

### 21.2.2.3 Communication links

Local automatic primary substation control will invariably involve more than one PLC. The integrated control system will require data to be passed from the switchgear to the associated PLC, from one PLC to another and also to the overall supervisory management system. Correct communication links are therefore the key to the fully automated system and will be the source of problems at the commissioning stage if not properly defined. In the past manufacturers have introduced their own communications protocols and formats such that a variety of software/hardware communication standards exist. It was due to this the standard organizations in Europe and North America started working on open standards. This work has culminated in a set of standards called IEC 61850. It is therefore essential for the user to define the communication standards to be used at the outset (or refer to the open standard) when preparing the specification and enquiry documents. There are three principal communications network arrangements as shown in Fig. 21.2.

The ring and bus/optic star arrangements are the most widely used. Twisted pair copper, coaxial and fibre optic cables link the network together with the choice depending upon transmission protocol used, quantity and rate of

![Figure 21.2 Communication network arrangements](image-url)
information being exchanged, length of circuits and cost for the particular application. Other special considerations for the system specification include:

- system resilience requirements
- alarm reporting
- operator/graphics station
- electromagnetic compatibility
- documentation
- interface requirements.

Figure 21.3 shows a typical switchgear control, metering and alarm interface. In this example, a separate marshalling cabinet is proposed with jumpers between the switchgear equipment connections and the SCADA control termination blocks. This arrangement has the advantage of greatly simplifying testing and maintenance by allowing easy access to all the interface points in one cubicle. The disadvantage of the dedicated separate marshalling cubicle is the added expense, the space requirements and the introduction of additional connections into the system.
21.2.3 Application example

21.2.3.1 User requirements

Figure 21.4 is a single line diagram of a switchboard with two incoming circuit breakers, A and B, and a bus section circuit breaker, C. In this example the simplified user requirements are:

- Automatic control for closing and opening circuit breakers A, B and C.
- Remote monitoring of circuit breaker A, B and C positions.
- Display of incoming circuit current and voltage.
- Control modes to be either local manual or local automatic by PLC.

A brief introduction to the basic requirements for a remote control system is also provided in this example.

21.2.3.2 Input and output requirements

The analogue and digital input and output requirements are defined.

(a) Analogue inputs: The analogue inputs are described by standard loop current transducers as shown in Table 21.1.

![Figure 21.4 Single line diagram for PLC application example]

**Table 21.1 Voltage and current data for example**

<table>
<thead>
<tr>
<th>Description</th>
<th>Sense</th>
<th>Range/engineering units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage incoming circuit A</td>
<td>4–20 mA</td>
<td>0–500 V</td>
</tr>
<tr>
<td>Current incoming circuit A</td>
<td>4–20 mA</td>
<td>0–100 A</td>
</tr>
<tr>
<td>Voltage incoming circuit B</td>
<td>4–20 mA</td>
<td>0–500 V</td>
</tr>
<tr>
<td>Current incoming circuit B</td>
<td>4–20 mA</td>
<td>0–100 A</td>
</tr>
</tbody>
</table>
(b) **Digital inputs:** The digital inputs are defined by logical ‘0’ and ‘1’ condition states. Some simplifications have been introduced into this example (e.g. no maintenance or earth positions for the circuit breakers have been introduced) since the purpose is to explain the basic design steps to be followed rather than describe a complex case (see Table 21.2).

(c) **Digital outputs:** While defining digital outputs, it is very important to consider their duty range. If they are expected to act on the process, i.e. operate a circuit breaker, the switching load and breaking loads may have to be considered or suitable interposing relays added externally (see Table 21.3).

### Table 21.2 Digital inputs for example

<table>
<thead>
<tr>
<th>Description</th>
<th>Requirement</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode switch circuit A</td>
<td>Automatic</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker A position</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker A position</td>
<td>Closed</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker A condition</td>
<td>Faulty</td>
<td>1</td>
</tr>
<tr>
<td>Mode switch circuit B</td>
<td>Automatic</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker B position</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker B position</td>
<td>Closed</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker B condition</td>
<td>Faulty</td>
<td>1</td>
</tr>
<tr>
<td>Mode switch circuit C</td>
<td>Automatic</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker C position</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker C position</td>
<td>Closed</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker C condition</td>
<td>Faulty</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 21.3 Digital outputs for example

<table>
<thead>
<tr>
<th>Description</th>
<th>Requirement</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit breaker A</td>
<td>Open command</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker A</td>
<td>Close command</td>
<td>1</td>
</tr>
<tr>
<td>Incomer A voltage display</td>
<td>Binary coded decimal (BCD)</td>
<td>4 digits</td>
</tr>
<tr>
<td>Incomer A current display</td>
<td>Binary coded decimal (BCD)</td>
<td>4 digits</td>
</tr>
<tr>
<td>Circuit breaker B</td>
<td>Open command</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker B</td>
<td>Close command</td>
<td>1</td>
</tr>
<tr>
<td>Incomer B voltage display</td>
<td>Binary coded decimal (BCD)</td>
<td>4 digits</td>
</tr>
<tr>
<td>Incomer B current display</td>
<td>Binary coded decimal (BCD)</td>
<td>4 digits</td>
</tr>
<tr>
<td>Circuit breaker C</td>
<td>Open command</td>
<td>1</td>
</tr>
<tr>
<td>Circuit breaker C</td>
<td>Close command</td>
<td>1</td>
</tr>
</tbody>
</table>
Normal and abnormal operating conditions are defined together with any system operating constraints in conjunction with maintenance, operations and engineering staff:

(a) Normal condition:
- Circuit breaker changeover control is under automatic mode.
- The left-hand side of the switchgear busbar is fed from incomer A with circuit breaker A closed and the bus section circuit breaker C open.
- The right-hand side of the switchgear busbar is fed from incomer B with circuit breaker B closed and the bus section circuit breaker C open.

(b) Abnormal condition:
- Power input failure or circuit breaker A faulty on incomer A. The whole busbar (both right- and left-hand sections) shall be fed from incomer B with the bus section circuit breaker C closed.
- Power input failure or circuit breaker B faulty on incomer B. The whole busbar (both right- and left-hand sections) shall be fed from incomer A with the bus section circuit breaker C closed.

(c) Operating constraints:
- Bus section circuit breaker C must not be closed when circuit breakers A and B are both in the closed position. Such a constraint may typically be due to fault level restrictions on the switchgear with the two incoming supplies paralleled.
- When circuit breaker A or B or C is under local mode, the automatic control is disabled. This is a safety constraint.
- When power supply is restored after failure on incomer circuits A or B the system should remain in its current configuration awaiting operator intervention. This ensures positive action and status acknowledgement by operations personnel.

(d) Communications requirements between PLC and communications controller:
- Master/slave configuration where the PLC/bay controller local to the switchgear is the slave and the communications controller, which has intelligence and interfaces between the communications network and the PLC, is the master for remote control purposes.
- Define the communications network protocol (e.g. ‘Modbus, IEC 61850’).
- Define the serial link format (e.g. RS 485, Optic bus glass-plastic).

(e) Remote control operation:
Usually, the central control centre (CCC) is comprised of duplicated mini-computers as the central processor associated with various man/machine interfaces. These interfaces include such items as a hand-dressed (Fig. 21.5)
Figure 21.5  Hand dressed conventional mimic panel. The panel consists of individual engraved tiles to display the substation single line diagram, switch position and key metering.
or automatically updated (Fig. 21.6) mimic displays, operator consoles/ keyboards, visual display units (VDUs), data loggers, event recorders, telephone, public address and radio speech communications. The CCC acquires information from the communications controller or remote telemetry units (or ‘remote terminal units’) (RTUs) associated with interrogation scan. Each RTU has a unique address code and is accessed in turn for a given period of time when information requests or control signals may be sent and information received. In order to avoid large amounts of data overloading the system during a fault (e.g. a busbar fault would create a multitude of circuit breaker status changes, network load flow alterations, metering and alarm indications) information is prioritized. Further ‘front-end processors’ are used for data acquisition in order to free the main computer for data processing.

21.2.3.4 Detail design to tender enquiry stage

From the foregoing a logic block diagram is next prepared as shown in Fig. 21.7. This should then be fully detailed into an overall descriptive and
technical specification such that enquiries may be launched with different manufacturers to supply equipment for the particular application. Figure 21.8 shows the PLC control cubicle for the Channel Tunnel 21 kV network automatic interlocking. This is designed to ensure that the UK and French unsynchronized Grid supplies are not paralleled inadvertently by incorrect switching sequences. The PLCs are in the top left-hand corner of the cubicle.
21.3 POWER LINE CARRIER COMMUNICATION LINKS

21.3.1 Introduction

The use of transmission lines as communications channels has obvious advantages to the electrical supply utility since it saves investing in additional
dedicated communications radio, hard wire or fibre optic cable links. System Control and Data Acquisition (SCADA) requires a communications network to transmit the information back to a central control centre, (CCC). There is a fundamental relationship and trade-off between the amount of information that may be transmitted over a given communications circuit, the speed of transmission and the bandwidth of the communications channel involved. The larger the bandwidth the faster a greater amount of information may be transmitted. Hence the bandwidth is the limiting factor for the signalling speed upon which the telecontrol system response times are based. Power line carrier circuits operate at only a few hundred kHz carrier frequency with signalling speeds restricted as a consequence to approximately 600 Baud when a single (4 kHz) channel is shared with speech, and up to 9600 Baud (analog) or 28 kbs (digital) on an equivalent data only channel. With modern digital equipment it is possible to attain signalling speeds >64 Kbit/s using a good quality line (e.g. better than 30 dB signal-to-noise ratio).

The subject is well covered by IEC Standards as listed in Table 21.4. The data transmission is performed by modulating the carrier frequency using audio frequency shift keying with modem (modulator/demodulator) interface units. Higher carrier frequencies (and hence larger bandwidths and signalling speeds) are not possible because of the stray capacitance (and hence high losses and attenuation) involved in overhead line power circuits. Telecontrol systems designed around power line carrier communication links are therefore specified with rather slow five second response times. The response time here is defined as the time between a change of state occurring at an outlying substation and it being announced at the CCC. This is one of the major reasons why fibre optic cable communications links are taking over from power line carrier-based systems. The other key reason is the immunity of fibre optic links from electromagnetic interference.

### 21.3.2 Power line carrier communication principles

#### 21.3.2.1 Modulation

Power line carrier systems amplitude modulate the carrier frequency. Full amplitude modulation (AM) has a frequency spectrum of sidebands symmetrical about the carrier frequency as shown in Fig. 21.9. These sidebands contain all the information being transmitted and the carrier frequency is only the bearer of the

<table>
<thead>
<tr>
<th>Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 60353</td>
<td>Line traps for AC power systems</td>
</tr>
<tr>
<td>IEC 60481</td>
<td>Coupling devices for power line carrier systems</td>
</tr>
<tr>
<td>IEC 60495</td>
<td>Recommended values for characteristic input and output quantities of single sideband power line carrier terminals</td>
</tr>
<tr>
<td>IEC 60663</td>
<td>Planning of (single sideband) power line carrier systems</td>
</tr>
</tbody>
</table>
messages. It is therefore possible to achieve savings in transmitter power without degrading signal performance by reducing the power of the carrier frequency and by deleting one of the sidebands. This is known as single sideband (SSB) transmission. The carrier is not fully suppressed because it is used to synchronize the remote end receiver with the corresponding transmitter. The transmitter and receiver circuits therefore tend to be slightly more complex than those used for normal broadcast AM transmission because of the filtering and accurate synchronizing involved. The lower ranges (~30 kHz to 200 kHz) of carrier frequencies are used on long transmission lines and the higher ranges (~200 kHz to 500 kHz) on shorter lines. This helps to offset the attenuation effects of long lines with high frequency transmission. Computer simulations may be used to optimize the best frequency band to employ on a given overhead line taking into account interference from any adjacent circuits.

Each power line carrier path can carry one audio frequency (AF) channel. This requires a minimum bandwidth of some 4 kHz. The lower ~2 kHz end of this base band is often reserved for speech which requires a bandwidth from approximately 300 kHz to 2000 Hz. It is possible to use the speech band for teleprotection, thereby freeing up bandwidth for data transmission and the channel is often used as a telephone system for the electrical supply utility. Dialing pulses may be transmitted by shifting a pilot signal frequency and detecting the shift pattern at the receiving end. An override facility is normally also provided for emergency/maintenance purposes whereby the telephones are connected directly via a front panel jack socket into the speech circuits.
The remainder of the channel bandwidth, 2000 Hz to \( \sim 3480 \) Hz is available for telecontrol, teleprotection and telegraph transmission using frequency shift keying (FSK). This form of modulation has many advantages over on-off keying of the carrier frequency and provided that the wanted signal (mark or space) is slightly stronger than any interfering signals the information will be correctly received. The main difficulty is that the use of automatic gain control is very limited and the time constant must be short. This is because the mark and space (logical ‘0’ and ‘1’) frequencies are only separated by a few tens of Hertz and may fade independently of one another. A strong mark may be followed by a weak space especially under power fault conditions. A pilot signal, added in the spectrum outside the audio base band (e.g. at 3600 Hz), is therefore used for supervision of the power line carrier channel and regulation of the receiver automatic gain control (agc). The Baud is the shortest single signal unit in a signalling code and may be expressed as the reciprocal of the time of the shortest signal element. For example if the shortest signal element were 20 ms in length then the data transmission speed would be \( 1/0.02 = 50 \) Bauds. The bandwidth of the telecontrol channel is determined by the frequency shift speed. A 200 Baud telecontrol channel shifting \( \pm 90 \) Hz occupies 360 Hz of bandwidth and the frequencies used are selected from CCITT standard recommended channels as described in IEC 60481 and 60663.

Power line carrier schemes are used in conjunction with overhead line distance protection direct intertripping/blocking or permissive intertripping/blocking as described in Chapter 10. It is, of course, essential that such signals are correctly transmitted and received over the very transmission line that the protection scheme is attempting to protect from the consequences of a prolonged fault. During the fault noise will be generated that could degrade the teleprotection signal. Therefore the power line carrier teleprotection signal is boosted to maximum power and all other signals may be disconnected (speech and telecontrol) in order to improve the reliability under fault conditions.

21.3.2.2 Circuit configurations

It is not usual to find power line carrier installations on distribution lines at voltages less than 36 kV. This is because such lines tend to have many tee-off points which would attenuate the signals and necessitate the installation of many power frequency rated filters or ‘line traps’. Also short power lines may employ pilot wire protection and the telecontrol system requirements may be able to use spare pilot cable cores.

The high frequency carrier signal is coupled to the overhead transmission line via high voltage coupling capacitors of value around 5000 pf. These act as a low impedance (few hundred ohms) at carrier frequencies but as an open circuit at power frequency (\( \sim 0.6 \) M\( \Omega \) @ 50 Hz) thus isolating the radio equipment from the power equipment. In addition coupling filters and transformers are necessary to match the power line carrier transmitter output impedance to the overhead line and thereby ensure maximum power transfer.
The carrier frequencies must not be effectively short circuited to ground through earthing switches at substations or through the neutrals of power transformers. Each power line carrier overhead line transmission circuit must therefore be effectively isolated at radio frequency from the substation busbars, transformers and switchgear.

This is achieved by ‘line traps’ which are parallel inductance and capacitance (L, C) tuned circuits. These line traps are inserted in series with, and at the end of, the transmission line to act as high impedance at the carrier frequency and prevent such frequencies entering the substation busbars. The line trap coil has a low impedance at 50 Hz in order to minimize power frequency losses. Surge diveters are connected across the tuned circuit to prevent damage against surges. The traps are specified to carry rated current and to withstand short circuit conditions.

Figure 21.10 shows phase-to-earth and phase-to-phase coupling arrangements between the power line carrier radio frequency equipment and the power frequency overhead line. Phase-to-earth coupling requires only half the equipment necessary for the phase-to-phase method. If a power system fault occurs on the phase being also used as a teleprotection or telecontrol channel the power line carrier signal will be considerably degraded and an assessment has to be made as to the security of the system under these conditions. For double circuit transmission lines it is possible to arrange the power line carrier protection intertripping for one circuit to be transmitted over the adjacent circuit. In this way the teleprotection channel does not signal over the actual line it is protecting. A diagrammatic representation of this commonly used arrangement is shown in Fig. 21.11a. Figure 21.11b shows a Middle East 145 kV substation overhead line bay with the incoming gantry and Power Line Carrier line traps mounted on CVTs associated with the distance protection scheme. The CVTs are used to couple the power line carrier signal to the overhead line.

21.4 SUPERVISORY CONTROL AND DATA ACQUISITION (SCADA)

21.4.1 Introduction

The term Supervisory Control and Data Acquisition (SCADA) refers to the network of computer processors that provide control and monitoring of a remote mechanical or electrical operation (e.g. management of a power distribution grid or control of mechanical processes in a manufacturing plant). Typically in the past a SCADA-based system would encompass the computers and network links which manage the remote operation via a set of field located programmable logic controllers (PLCs) and remote telemetry units (RTUs). The PLCs or RTUs would be connected to field transmitters and actuators and would convert analogue field data into digital form for transmission over the network.
In many instances today, with respect to high voltage substations or even medium voltage substations, the substation control devices are not necessarily RTUs or PLC's, but IEDs (intelligent electronic devices) that serve the purpose of protection, local and remote control. These IEDs provide a means to
acquire and transmit the analogue and binary input data to the control system via communication links.

SCADA systems are in essence a real-time operating database that represent both the current and past values or status of the field input/output points (tags) used to monitor and control the operation.

Relationships can be set up within the database to enable functional (or computed) elements to be represented which provide operators with a logical representation of the remote operation. This representation enables the whole operation to be monitored and controlled through a central point of command whereby concise information is available in a clear schematic and textual form typically on graphics workstations.

The supervisory functions of SCADA systems present plant operators with a representation of the current and historical states by means of hierarchical graphic schematics, event logs and summaries. These screens also identify all abnormal conditions and equipment failures which require operator acknowledgement and remedial actions. The control functions enable specified items of plant to be controlled by issuing direct commands, by instigating predetermined control sequences, or by automatically making a programmed response to a particular event or status change.

SCADA systems do not usually handle the collation of statistical data for management information purposes. However, a SCADA system usually exists in an integrated computer hierarchy of control and as such interfaces usually exist to other computer-based systems.
21.4.2 Typical characteristics

It is convenient to describe SCADA systems by considering their typical characteristics in relation to input/output, modes of control and interfaces with operating personnel.

21.4.2.1 Plant input/output

Typically a SCADA system interfaces with plant over a wide geographical area via PLCs and other RTU/bay control equipment local to the plant. The number and types of the input/output points, the nature of the local equipment connected. There are two basic modes of capture of input data which may be used by the central processing facility of the SCADA. These are:

(i) scheduled capture, whereby the local units are polled on a regular basis and all input data is transferred; or

Figure 21.11(b) Qaboos 132 kV substation, Oman-OHL incomers with surge arresters, CVTs/line traps and cable sealing ends
Supervisory Control and Data Acquisition

(ii) change of state capture, whereby only input data which has changed is transferred.

The input and output data are held centrally in a real time database. By holding historical as well as current data it is possible to provide facilities for analysis and reporting of trends. This facility is often particularly important for systems where most of the data is analogue rather than discrete.

In the database input and output data is usually grouped into functional units or elements. For example several input/output points might be grouped to provide the complete representation of an electrical circuit breaker. Frequently such groups of plant input points are transformed to calculate computed points which are also stored in the database. For example a single computed point might represent the status of a number of associated circuit breakers.

A typical use of such computed points is in the management of alarms. In many cases alarms are categorized at least into major and minor alarms. Each alarm is itself likely to be a computed point usually computed from an input value and a trip level. Some major alarms may also be computed from combinations of minor alarms. Complex strategies for predicting alarm conditions may also be used.

21.4.2.2 Control modes

Control of plant associated with a SCADA system may be either local or remote. Local control exercised automatically, for example by a local PLC, or may be by local mechanical or electrical controls (automatically or manually operated).

Remote control via the SCADA may be instigated by an operator or may be automatic. Automatic controls can be initiated by time (scheduled control) or events (change of state control). In both cases control frequently involves initiating a pre-programmed sequence of actions which are then automatically carried out.

One advantage of distributed control with the control systems located locally in substations and reporting to a higher level SCADA system is the lower dependence on troublesome communication links. In modern computer-controlled stations, the SCADA system sends out a simple remote control command. Any sequential control sequence is executed locally.

21.4.2.3 Operator interface

The operator interface for a modern SCADA system should be designed to provide the maximum support to the operator in his role of monitoring and controlling the plant. In order to achieve this considerable use is made of sophisticated real time graphics to display current and retrospective input output values and trends.

A well-designed operator interface can provide considerable support in alarm management. Where there is a potential for large number of alarms it is
particularly important that they are grouped, classified and displayed in a coherent fashion which enables the operator to concentrate on the more important alarms. Often the facility to filter out minor or consequential alarms or to acknowledge them in groups for later response can be valuable on its own.

Nowadays graphical displays are usually Windows based. There is an increasing trend to using a mouse or a touch sensitive screen to supplement a keyboard for most operator input. Graphical displays should incorporate a hierarchy of displays from high level overall plant schematics to tables of associated input/output points at the lowest levels. Banner display of important information about key events is often used. In order to supply all this functionality it is common to use multiple screens at single operator positions.

An important security facility which is required in many SCADA systems is the definition of different classes of user with access to different functions or facilities. The distinction may simply be between supervisor and operator or between supervisor terminal and operator terminal or may involve several levels of access. Most systems provide the facility to set up or configure the SCADA database. This is usually necessary for the installation and commissioning of the system but should not be available to the ordinary operator.

21.4.3 Design issues

The throughput of the system is one of three main issues in any design. This will be dictated by:

(i) the magnitude and number of the field input/outputs;
(ii) the data capture time required, which is usually dictated by the time constraints of the process being monitored and/or the time taken to respond to an event;
(iii) whether any sophisticated schemes are used for data compression;
(iv) whether a deterministic communications protocol is required guaranteeing a response in a specified time;
(v) what level of integrity is expected of the data communications;
(vi) what physical media for data transmission is acceptable in the particular application.

Certain key plant input/output and associated operation may be deemed as being high integrity. For such input/output redundancy needs to be considered in one or more areas of a SCADA system to minimize the effects of failure. Typical redundancy may include:

- dual links for plant input/output to two or more PLC or RTUs;
- dual communication links handling dialogues with the main supervisory processors;
- redundancy in the main supervisory processors provided by either employing a standby fault tolerant processor or by having two or more processors.
providing a shadowing function. In this case a ‘standby’ processor would shadow all operations of a ‘normal’ processor and watchdog mechanisms would enable a switchover to occur if any communication failure or data integrity errors are detected.

As the loading in terms of the number of plant input/output points increase the processing power required increases. The central architecture of a SCADA system may require several processors each dedicated to specific operations. A typical partitioning would include:

- **Front End Processors (FEP).** FEP are dedicated to handling data acquisition from field RTU and PLC equipment;
- **Graphics Workstations.** Where a number of operator positions are required a distributed client-server-based architecture spreading the load between a main supervisory processor and two or more graphics workstations should be provided.
- **Main Supervisory Processor (MSP).** One or more MSPs provide centralized control and representation of the field input/output (plant status) by means of one or more databases. The central processor will perform functions such as data logging, handling of control sequences, maintenance of logical (functional) equipment states.

### 21.4.4 Example (Channel Tunnel)

The Channel Tunnel Engineering Management System (EMS) employs a SCADA system configured to manage remote equipment via 26000 direct input/output points and a further 7000 computed points. The equipment under the EMS control is the Fixed Equipment located in the two terminals in Folkestone in the UK and Coquelles in France and in the three tunnels (Running Tunnel-North, Running Tunnel-South and the Service Tunnel). The Fixed Equipment manages the following:

- **Electrical Distribution:**
  - Connections to National Grids (225 kV and 132 kV)
  - Supply to 25 kV Overhead Catenary System
  - Tunnel distribution of 21 kV and 3.3 kV supplies
  - Terminal and Tunnel Lighting.
- **Mechanical Systems:**
  - Normal and Supplementary ventilation systems
  - Tunnel Cooling
  - Pumping
  - Fire Fighting equipment.

Figure 21.12 shows the RTU equipment located in the 178 equipment rooms located between the service and running tunnels. These handle the data
Figure 21.12  Arrangement of RTUs in Channel Tunnel SCADA system
acquisition and control of the 26 000 input/output points via 600 PLCs. When an input/output point changes state the new status is sent to both the French and UK control centres using a drop insert connection to the RTUs. The input/output states are handled simultaneously by main EMS processors (MEPs) in both UK and French control centres. The MEPs are DEC VAX processors running identical SCADA application software. The machines operate in a normal/standby mode. The normal machine is the master and handles all operator dialogues from both the UK and French operator positions. The standby processor whilst maintaining data compatibility with the normal processor also monitors the health of the normal processor, site networks and through tunnel point-to-point links. If any failures are detected then a switchover will occur and the standby machine will move to a normal status.

Three dedicated FEPs are provided in each terminal. Two of these FEPs handle communications with RTUs. The other four FEPs (two in each terminal) provide dual redundant links to a number of external systems such as fire detection and access control.

Data integrity is provided as follows:

- Certain plant input/output has links to two different RTU processors.
- All RTUs communicate with both the French and UK control centres. In addition input/output states received in the French control centre are routed to the UK control centre by the through tunnel links. Similarly the UK control centre transmits input/output states to the French control centre. In a full availability operation each MEP receives two identical messages which are filtered accordingly.
- Redundant on site networks.

Dedicated operator servers (OPS) provide five operator positions in the UK and four in France. In normal operation these provide for a supervisory position and two or more operating positions. The UK control centre also has a major incident control centre (MICC) with a dedicated OPS.

EMS operations are possible simultaneously in both the UK and French control centres. However, only one control centre can have an active status which determines the nature of the possible operation.

21.5 SOFTWARE MANAGEMENT

PLCs, power distribution systems and SCADA systems all make use of software. In many cases, the software components can be seen as the main contributors to the systems’ functionality. This use of software has many advantages but it also poses many problems which need to be addressed carefully if they are not to threaten project success.
21.5.1 Software – a special case

The use of software in control systems offers the engineer increased flexibility in the design and operation of systems. Often software allows system to provide functionality which could not otherwise be provided in a cost effective way. However, software development projects are renowned for being late, over budget and not meeting the requirements of the customer. The key to understanding why software development projects frequently possess these unfortunate characteristics is to look at how software development differs from other branches of engineering.

The problems presented by software are many and somewhat fundamental in their nature. The still maturing discipline of software engineering attempts to address these problems.

21.5.1.1 Software is complex

Software is a highly complex dynamic object, with even a simple programme having a large number of possible behaviour patterns. For most non-trivial software it is impossible to exhaustively test its behaviour or prove that it will always behave as its specification requires. The difficulty of proving that a software system meets its specification is compounded by the lack of fundamental laws that can be applied to software. The mathematics underlying software engineering is still in its infancy compared with other branches of engineering.

21.5.1.2 Software is discontinuous

The discontinuous nature of software means that small changes in input values can result in large unexpected changes in the software and system behaviour. Small changes in the software itself can have similar results. As a result meaningful testing is much less straightforward than for analogue systems. Testing of a completed software system does have a place in providing confidence that it performs its functions correctly but more is required. Considerable effort needs to be expended on managing and assessing the software development process.

21.5.1.3 Software changes present difficulties

The range of functions a software system can perform and the apparent ease with which new software can be added makes software very attractive to engineers. However this is deceptive, once software is built it is difficult to change with confidence. Even minor changes can have dramatic and unforeseen effects on often unrelated parts of a system. Furthermore, as more changes are made the software architecture will tend to become increasingly complex and fragmented. Changes become increasingly difficult to implement satisfactorily. This fact should be borne in mind when requesting modifications to completed systems.
21.5.1.4 *Software is insubstantial*

The intangible nature of software means that you cannot see, touch or feel software. As a result, a software system is very difficult to appreciate until the very end of a development when the component parts are integrated. Unfortunately, by this stage a high proportion of project resources will have been expended making any corrective actions expensive to say the least. Furthermore, testing at this stage is only of limited use in providing confidence in the software.

21.5.1.5 *Software requirements are often unclear*

Software systems usually perform a very large number of diverse functions which can interact with each other in complex and subtle ways. It is very difficult for a customer to describe these functions precisely and this leads to unclear and changing requirements. This problem is made worse by the culture gap that frequently exists between customers and software developers. In other branches of engineering, the specifier of a product will usually be experienced in the engineering discipline required to build that product. This situation rarely exists with software systems. As a result, software systems are often specified in narrative English because the notations of software engineering are unfamiliar to the customer. The use of English (or other natural languages) can lead to ambiguities and inconsistencies in the specification which are then fed into the development process and only discovered late in the project when they are difficult and costly to correct.

21.5.2 **Software life cycle**

At the highest level, a software development project should be managed in the same manner as any other engineering project. Thus, a software development should follow a software project life cycle similar to that shown in Fig. 21.1.\(^2\) Such a lifecycle has clearly defined phases, with each phase having defined inputs and outputs. The project should have recognized review points to aid control. Normally, review points would occur at least at the end of each phase. The whole software and system development process should take place within a quality assurance system such as the ISO 9000 series. The lifecycle shown in Fig. 21.1 and described below is a lifecycle for software development, which should be integrated into the overall project lifecycle.

21.5.2.1 *Requirements specification phase*

The objective of the requirements specification phase is to produce a clear, complete, unambiguous, non-contradictory description of what a system is to do. The requirements specification should be fully understandable to both the
customers and the developers. There may be a separate software requirements specification, but if not, the software requirements should be clearly separated and identifiable within the overall requirements specification. Errors at the requirements specification phase can have very serious consequences and therefore the developers should make a major effort to confirm its correctness.

When the requirements specification has been agreed a requirements test specification (often called an acceptance test specification) should be drawn up. This document should state those tests a system must pass for it to be acceptable to the customer. Should a system fail any of the acceptance tests the customer has the right for the problem to be fixed and re-tests performed. However, these tests cannot on their own ensure that the software is correct.

21.5.2.2 Software design phase

Using the requirements specification the developers will begin designing the software. As with any engineering discipline this is an essentially creative process which can be done in many different ways.

The objective of the software design phase is to decompose the software into a coherent set of self-contained modules which will each have their own specification and which can each be tested separately. The software design phase will often see the software development process disappear into a tunnel as far as the customer is concerned. Some time later a fully working system will emerge from the other side at the software validation phase. The work carried out within this tunnel is vitally important and it is well worth the customer understanding and monitoring what occurs.

A structured top down approach should be taken to this high level design of the software, producing a hierarchy of modules at different levels. A variety of techniques, often supported by automated tools, may be used during the design. Typical techniques include data flow diagrams, state transition diagrams, object-orientated notations and entity relationship diagrams. Any of these techniques should be supplemented by English language descriptions.

21.5.2.3 Software module design phase

The objective of the software module design phase is to perform the detailed design of exactly how each module will carry out its required task. The means by which this detailed design is expressed will vary depending on the type of system being developed and the tools used by the supplier. Typical approaches are to use logic diagrams, flow charts, pseudo code (programming language like statements), formal mathematical notation or decision tables. Alternatively the techniques used during the software design phase may continue to be used. Often a combination of such methods, supplemented by English language description is best. It is essential that the required inputs and outputs, their meaning and possible values are clearly identified for each module.
During the detailed design of a module the developer should produce a test specification detailing those tests that need to be carried out to confirm the correct functions of a module once coded.

21.5.2.4 Code phase

The objective of the code phase is to transform the software design specification and software module design specifications into a coherent computer programme or programmes.

It is important to ensure that the code produced is understandable to persons other than the author. In order to achieve this, project standards should be set up and adhered to code structures, format and commenting. The code produced should also be reviewed and changes to approved code are strictly controlled.

In principle the programming language or languages to be used could be selected at this stage. In practice it is likely that design constraints considered earlier will already have determined the language. Such considerations might be dictated by availability, experience or processor used in addition to the merits of a particular language. If possible a high level, structured language should always be preferred to using assembler.

21.5.2.5 Software testing phase

The software testing phase covers the testing of the software from individual modules to the complete software system. The phase therefore involves much more than testing against the acceptance test specification. The objective of the phase is to ensure that the software functions correctly, in so far as this can be achieved by testing. In order to satisfactorily test the individual modules it is likely to be necessary for much of this testing to take place in parallel with the coding phase, though conceptually it occurs afterwards.

Records should be kept of the testing of each individual module, of each group of modules as software integration proceeds and of the complete integrated software. These records should be considered part of the documentation of the software and should be retained either by the supplier or the customer. The customer should ensure that the testing process is monitored either directly or by a third party.

21.5.2.6 Software/hardware integration phase

The objective of the software/hardware integration phase is to combine the software and hardware into a coherent whole. The integration process involves further testing of the software and system, with further changes being made to the software to resolve any problems which arise. Frequently part or all of this phase must take place at the customer’s site. It is essential that the activities of
this phase particularly software changes and their testing are adequately controlled and recorded.

### 21.5.2.7 Software validation phase

The software validation phase occurs when the software is complete.

The objective of the phase is to ensure that the completed software complies with the software requirements specification. A variety of methods may be used including software and system testing and various levels of review of the software and system documentation.

The relationship between software validation and acceptance testing may vary depending on the type of project, the function of the software and the customer requirements. In some cases software validation is required as part of the acceptance testing before software installation on site. In other cases validation may be required after all commissioning adjustments have been implemented.

### 21.5.2.8 Software maintenance phase

Software maintenance differs from other maintenance activities in that it necessarily involves modifications to the software. These modifications may correct errors in the software, add facilities which should have been included originally or add new facilities. Often software maintenance involves upgrading to a new operating system and modifying existing software so that it works within the environment.

Because software maintenance always involves new changes to the software it requires careful control and regulation. For example the benefits or otherwise of each proposed change and any safety implications (e.g. IEC61508 compliance) should be carefully considered and analysed before the change is authorized.

### 21.5.3 Software implementation practice

The process of software development described in Section 21.5.2 provides a theoretical framework for the activities which an engineer can expect to see taking place. The concept of a software lifecycle and its associated documentation are well understood and accepted but interpretations vary. In particular, phase and document titles may not match those presented in Section 21.5.2. Nevertheless it should be possible to identify all the key features in any software development process (see Section 21.5.3.1).

In practice there are a variety of tools, methods and techniques which suppliers can and should use during the software lifecycle. There can be clear rules about which should be used and the choice may well affect the lifecycle and documentation set associated with the software. The important point is that none of the tools, methods or techniques is sufficient on its own. They should be
used as part of an approach based on a coherent justifiable lifecycle and associated with comprehensive documentation and software project management techniques (see Section 21.5.4).

21.5.3.1 Key lifecycle features

The key features which should be evident in any software lifecycle are:

- a clear specification of the software identifies those requirements separately from the system requirements and separately from the software design;
- a software design which is recorded and goes through two or more stages of increasing detail before coding;
- software testing which is clearly specified, and which covers each stage of the code being developed, integrated and installed;
- final validation of the completed code against the requirements;
- control of changes to the complete software; and
- formal design and quality reviews at appropriate points in the life cycle.

21.5.3.2 Software safety and reliability

The safety aspects of systems containing software are often not appreciated by engineers. Software can often provide the potential for enhanced safety through enhanced functionality. However, the characteristics of software are such that special care is needed where it is to be used as part of a system which has the potential to harm or is otherwise required to be of high integrity. It is beyond the scope of this section to provide any detailed guidance on the issue, but Table 21.5 lists a few of the emerging draft and final standards and guidelines which

<table>
<thead>
<tr>
<th>Standard</th>
<th>Title</th>
<th>Source</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSE PES</td>
<td>Guidelines for programmable electronic systems</td>
<td>UK Health &amp; Safety Executive</td>
<td>1987</td>
</tr>
<tr>
<td>MoD Interim Def Stan 00–55</td>
<td>The procurement of safety critical software in Defence equipment</td>
<td>UK Ministry of Defence</td>
<td>1991</td>
</tr>
<tr>
<td>MoD Draft Def Stan 00–56</td>
<td>Safety management requirements for defence systems containing programmable electronics</td>
<td>UK Ministry of Defence</td>
<td>1993</td>
</tr>
</tbody>
</table>
Various tools are available to assist with software specification, design, implementation and test. Different methods address different aspects of the software lifecycle and use different approaches. All provide at least some of the framework on which to base a software lifecycle.

Computer aided software engineering (CASE) tools and the methods on which they are based are frequently used as the foundation on which a software development project is planned. Such tools typically assist with specification, design and implementation and provide much of the necessary lifecycle documentation for those phases. The provision of such documentation by an automated tool helps ensure that it is consistent and follows a coherent format. Most importantly traceability of requirements, through to the final design and code, is also ensured. In many cases the methods and tools make extensive use of diagrams which helps make the designs understandable.

Formal methods provide a mathematically based approach to software specification and design. The principal attraction of such methods is that they allow a proof that the mathematical specification is internally consistent and that the completed code correctly implements the specification. At the time of writing these methods are not widely used and the necessary skills for their use are in short supply. In the future the use of such methods can be expected to increase, particularly for high integrity applications.

A proprietary code management tool, to control build configurations should be adopted by system developers. Such tools assist in providing librarian facilities in a multi-developer environment and ensure that all software modifications are recorded and incorporated into new system builds. The tools also provide configuration control facilities by version stamping individual files and enabling current and historic versions of a software system to be recovered and rebuilt.

Other methods and tools are available which are not so readily categorized. Static analysis can be used to analyse the software code and generate metrics which express various characteristics of the code as numbers. Combinations of these metrics can be used to help form a judgment about the quality of the code and its structure. Dynamic analysis can be used to exercise the code and collect data about its behaviour in use.

Configuration management of software systems should be applied during the development and operational life of the software in order to control any changes required and to maintain the software in a known state.
To achieve configuration management the components of a software system are partitioned to form configuration items. These encompass all design and test documentation as well as the constituent software components.

The concept of a baseline is applied to software once the build is in a known state, usually once the software integration phase in the development lifecycle is reached. Thereafter any changes required, resulting from anomalies or functional modifications, are controlled through a predefined change control process. The basic stages of the change control process are:

- identification of need for change;
- identify change implementation, assess impact and approve (or reject) implementation;
- audit change implementation;
- install modified software and update the baseline.

### 21.5.4 Software project management

This section sets out areas in which the problems and techniques of software-based projects differ from those in more traditional manufacturing projects. None the less the basic issues of project management remain valid and to achieve success the following areas need to be addressed:

- definition of work scope;
- risk incurred;
- resources required and
- tasks and phases to be accomplished.

#### 21.5.4.1 Planning and estimating

In common with any project, planning and estimation attempts to quantify what resource is required. Typically this is measured in man–months effort, the chronological duration, and task breakdown and other areas affecting cost. The complexity of software requirements and the difficulty of correctly defining them make resource requirements difficult to estimate. Over recent years a number of estimation techniques have evolved which attempt to quantify the likely costs and durations. The basis for the different techniques, in all cases, is based on past experiences and the function sizing of the whole computer-based system.

Each estimation technique has a number of common attributes:

- project scope;
- software metrics (measurements) forming the basis on which the estimates are to be made
- functional and task decomposition allowing estimation of individual items.
There are two basic categories of estimation techniques, size orientated and function oriented. An example of a size-orientated technique is the constructive cost model (COCOMO). This computes development effort as a function of programme size and produces development effort (cost) and duration.

In contrast function-orientated techniques typically refer to a function point analysis and considers effort associated with the number of user inputs and outputs, enquiries, files and interfaces. Once calculated function points are used to derive productivity, quality and cost measurements.

21.5.4.2 Scheduling

In a small software development a single software engineer may well analyse design, code, test and subsequently install a system. However, as project size and complexity increases more engineers must become involved. In a multi person project team there is a time overhead incurred in communication between team members. In addition when team members join project teams in an attempt to make up lost time, they need to learn the system, most likely from those already working on the project. In summary, as project size and complexity increase then the engineering effort required for implementation increases exponentially. If project development slips (or requires accelerating) adding new effort will typically increase the magnitude of any slippage (at least in the short term).

The basic issue to be considered is that people’s working relationships and structures are essential for project success, but need careful structuring and management.

21.5.4.3 Effort distribution

All software estimation techniques lead to estimates of project duration in effort (typically man months). These assume an effort distribution across the development lifecycle of 40-20-40 (see Fig. 21.1). The 40-20-40 distribution puts the emphasis on the front-end analysis and design tasks and back-end testing.

21.5.4.4 Progress monitoring

The insubstantial nature of software makes progress very difficult to measure. A lot of resource is often required to complete a project which is reported as nearly complete. Typical figures quoted are 50% of resource to complete the last 10% of the project.

By partitioning and reporting on software development activities down to a low level realistic measurement of progress becomes more practical. Because each basic task is small and self-contained it is relatively straightforward to
identify whether it has been completed and thus estimate the progress which
has been made.

REFERENCES

1. Leveson, N.6., Software safety: Why, what, and how, Computing Surveys,
2. IEC, Draft Software for Computers in the Application of Industrial Safety
Related Systems, IEC (Secretariat) 122, International Electrotechnical Com-
3. IEE, Safety-related System-Professional Brief, Issue 1, January 1992, The
Institution Electrical Engineers.