Chapter 1

OVERVIEW

Edition 1

Contents

INTRODUCTION

A Function Block is a concept formalised by the international standard, IEC 1131/3, Programming Languages for Programmable Controllers. A Function Block contains an encapsulated algorithm which the user or system programmer can control by means of a number of parameters.

The PC3000 Function Block Library provides a set of Function Blocks which may be regarded as a set of high level 'Software Instruments,' that are analogous to discrete instruments that can be connected together by software to create a control strategy for a particular application.e.g. PID Controller, Timer, Counter etc.

Most function blocks have **Inputs** which may be written to from the Programming or Operator Station, from a Supervisory system or from the PC3000 itself by using 'soft wiring' or from action statements within a Sequential Function Chart.

Most function blocks have **Outputs** which are calculated by the function block.

Many function blocks have **Configuration Parameters** which are set on start up and never change.

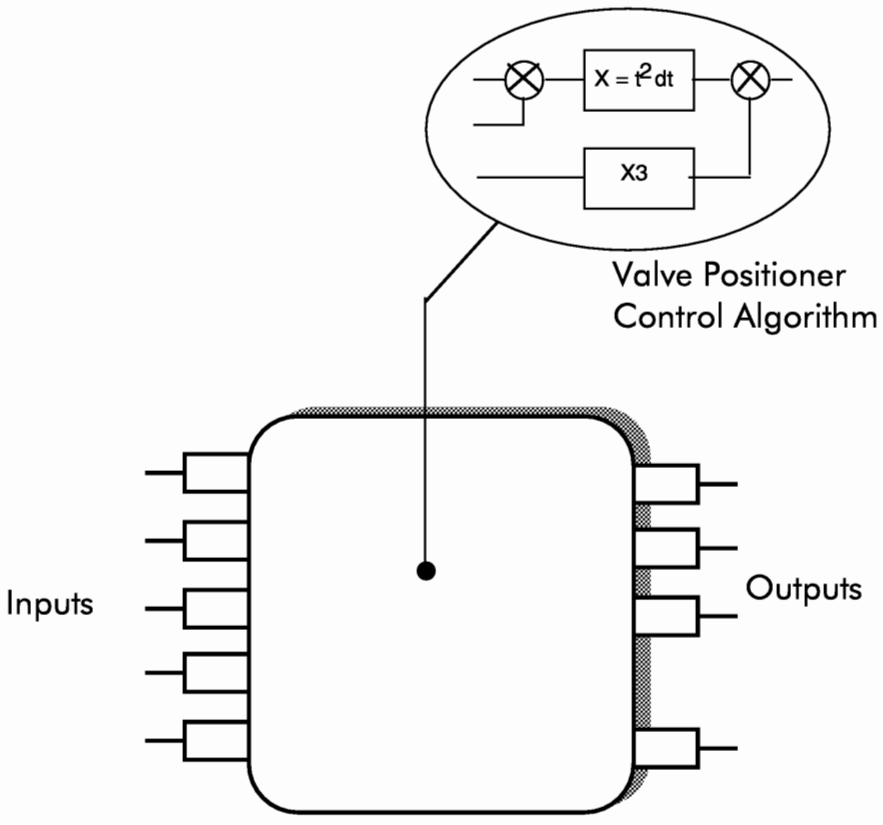

Figure 1-1- A Function Block

The inputs of a Function Block can be initialised to a 'cold start' value on the first execution of the block. All parameters have default cold start which may be overridden by the user program.

Function Blocks can be inter-connected by means of 'soft wiring ' to construct complex control strategies. Wiring takes the form of textual expressions which determine the relationship between the output of one block and the input of the next. The connections are made by language statements constructed using the Structured Text (ST) language or using graphical wiring. This wiring may be point-to-point or may include complex arithmetic or logical expressions.

Numerical Functions such as ABS_REAL are represented in a similar diagramatic form to Function Blocks in the PC3000 documentation. However, a Function may have one or many inputs but only has **one** output. In contrast, Function Blocks have multiple inputs and multiple outputs.

It is also important to note the differences between the execution of the PC3000 Functions and the execution of Function Blocks. In the PC3000 the execution of a numerical, or other, function is a transient event. A Function is executed during the evaluation of an expression, the Function is not executed again until it is next encountered in an expression.

In contrast, Function Blocks and their associated wiring statements are executed continuously; they execute at a rate determined by the *Task* with which they are associated. Time-variant aspects such as manipulation of a configuration parameter, are handled via the sequence program. A Function Block generally contains internal storage to hold values between executions. In most cases these values are retained when power is off and are used to allow the Function Block to be 'warm started'.The *PC3000 User Guide* provides an overview of Function Block wiring and the sequence program.

Function Blocks execute in a strict order; for example with blocks which are associated with physical inputs, the input values are collected at a fixed point in time, the body of the function block is then executed. For output function blocks the reverse applies, the body is executed before the output value is sent to the physical output.

The wiring between blocks is executed as if it was part of the Function Block which contains the destination parameter. Execution rate is independent of the type of the source or destination parameter. If a Function Block refers to more than one output of another block, the PC3000 system ensures that all outputs are produced by the same execution.

For a full description of the execution of Function Blocks, I/O latency and other issues reference should be made to the PC3000 Real Time Operating System Reference (HA022918).

GENERAL FORM

Depending on the Programming Station type, the Function Block may be represented textually or graphically. Refer to the relevant User Guide for details on the type of representation used.

Function blocks are represented by diagrams in this text. All function block diagrams adopt the standard of **inputs** enter from the left, **outputs** exit from the right. The data type for each parameter is indicated adjacent to the input or output 'pin'. Some input parameters can be modified by the Function Block when it executes, such parameters also appear as outputs and are termed **input/output** parameters.

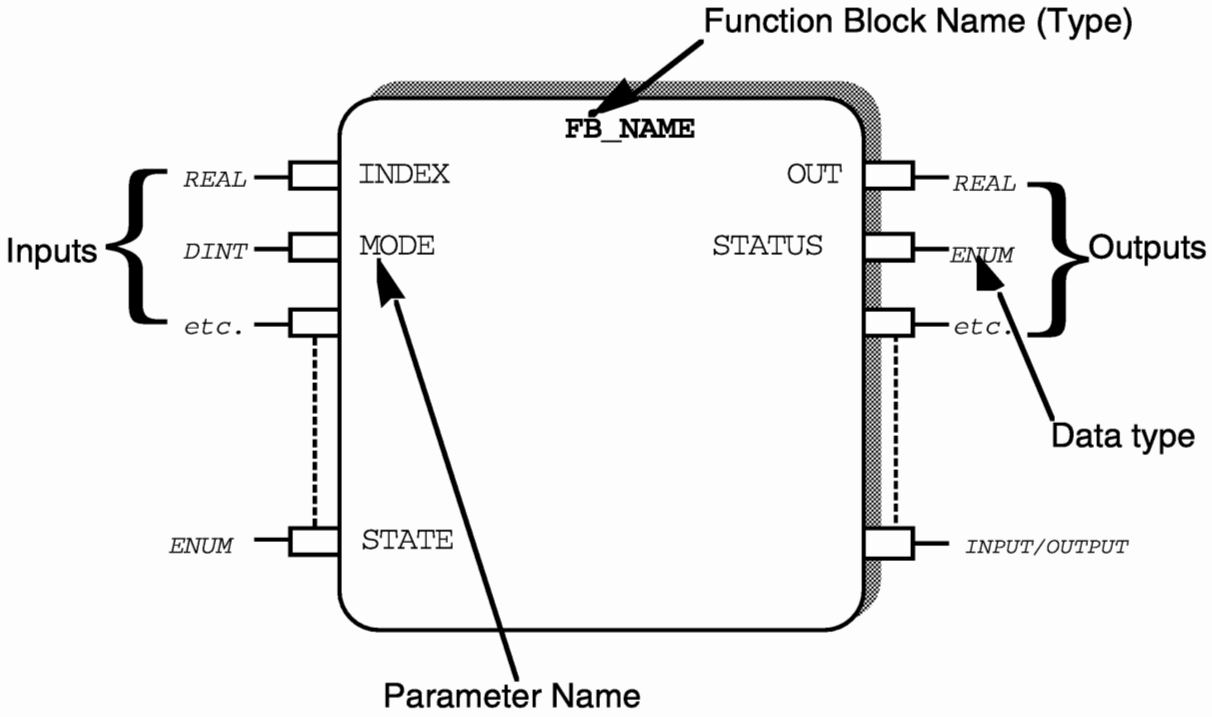

Figure 1-2 General Form

All Function Block names are restricted to 11 characters.

For details of Function Block and I/O execution reference to the PC3000 Real Time Operating System Reference is recommended.

TERMINOLOGY AND NOMENCLATURE:

The following terms are used to describe the PC3000 Function Block Hierarchy:-

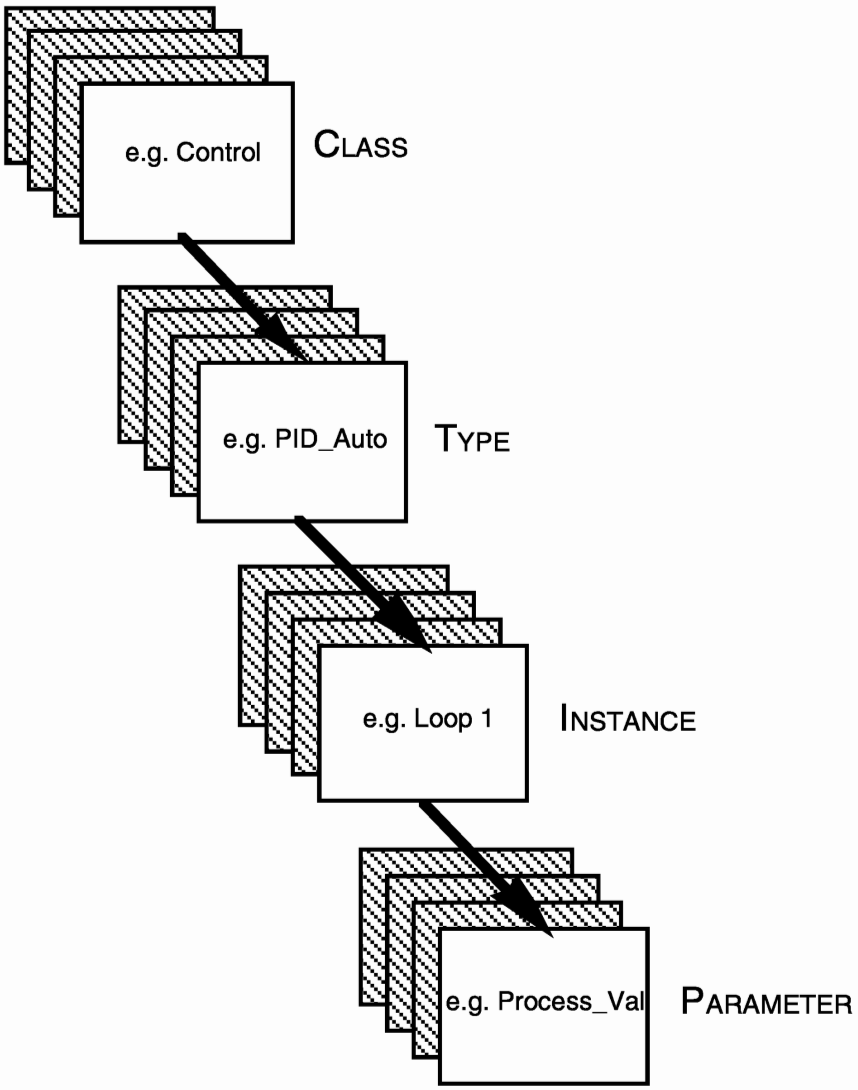

Figure 1-3 Function Block Hierarchy

The following nomenclature and fonts are used in this text:

CONTROL	Denotes Function Block class
PID_Auto	Denotes Function Block Type
Loop1	A Function Block of type **PID_Auto** with a name of Loop1. This is described as the 'Instance' name.
Loop1.PV	A parameter of the function block instance, Loop1, having a mnemonic of PV
REAL	The IEC data type of the parameter

A junction block name which is prefixed with an 'x' denotes an obsolete block. These blocks will not normally appear on the Programming Station class list but if a program developed in an earlier release is loaded, access to these blocks is provided.

This manual provides descriptions for reference purposes only.

FUNCTION BLOCK ATTRIBUTES

All Function Blocks have a set of attributes which are unique to that block. This data is tabulated for each block type.

Type	This is a hexadecimal, four digit number, that defines the Function Block type. It refers to the file name used to identify files associated with this Function Block installed on the Programming Station. The number also sets the Class and the order in which the block appears on the Type list on the Programming Station In normal use there is no requirement to make reference to this parameter and it is provided for information only.
Class	The Class that this block is associated with e.g. CONTROL
Default_Task	Every Function Block is assigned a default execution rate. The default PC3000 configuration assigns function blocks to either of two tasks: Task_1 and Task_2 (10ms and 100ms).
Short List	This attribute defines those parameters which appear as part of a 'short list' or summary of the block, when viewed on the Programming Station. The short list provides a method of displaying the most commonly accessed parameters in a compact form so that many instances of the same Function Block type may be viewed on the same screen
Execution Time	This attribute can be used to estimate the time taken by the PC3000 real time system to execute the function block. In the case of a complex block such as PID_Auto, a number of times are provided, covering a range of modes of operation The execution times of trivial blocks which have negligible effect on the system performance, are not given.
Instance Memory Requirement	This provides the amount of RAM required for every 'instance' of a given Function Block type. Figures refer to memory usage in bytes.

Table 1-1 Function Block Attributes

PARAMETER NAMES AND MNEMONICS

All Function Block parameter names given in this reference adopt the convention:

Parameter e.g. **Preset_DT, Stop_Bits, Status** -where <Parameter> is a textual name up to 12 characters long.

The naming convention presents the Function Block parameter in the same way as used on the Programming Station. In general PC3000 Function Block parameters have names which include 'separators' in the form of the underscore character; names always begin with a capital letter.

The underscore is a valid part of the parameter name.

In the sections covering parameter description, or in the case of the more complex blocks, the functional description, parameter names are followed by parenthesis ( ). The contents of the parenthesis ( ) refer to the mnemonics which can be used as an abbreviated name when referencing a parameter within a Structured Text statement. e.g. **SP** can be used in place of the full parameter name **Setpoint.**

PARAMETER ATTRIBUTES

Every Function Block parameter has a set of attributes which characterise its operation e.g input, output, data type etc.

For each block attributes are tabulated at the end of each section. Attributes are:-

Name	The textual name of the parameter.
Type	The data type of the parameter. See Table 1-3 for Data types.
Cold Start	The default value that the parameter will assume following a 'cold start' of the application program. Alternative values may be entered prior to saving the application program
Read Access	Default access level for displaying the parameter of the Programming Station. (Operator, Supervisor or Configuration) designated as Oper, Super and Config respectively.
Write Access	Default access level for changing the parameter on the Programming Station. (Operator, Supervisor or Configuration), designated as Oper, Super and Config respectively.
High Limit	The upper limit used for range checks in the EI Bisync communications handler. It should be noted that the value may be exceed this limit if updated from within the application program; this limit applies to the communications interface only.
Low Limit	The lower limit for range checks in the EI Bisync communications handler..
Senses	Lists the range of options associated with an enumerated parameter or the states of a boolean parameter, e.g. 'UP', 'DOWN'..

Table 1-2 Parameter Attributes

Conventions

In some text, the IEC 1131-3 standard data type is also given to clarify the use of a particular data type. In such cases, the IEC data type is given in parenthesis e.g. floating point (REAL), duration (TIME).

Hexadecimal values are depicted with a lowercase 'h' suffix, e.g. 73h, 0Ah

KEYWORD	DATA TYPE	BITS	RANGE
IO_ADDRESS	Channel address	32	1:01:01 to 8:12:14
BOOL	Boolean	1	0 or 1
REAL	Real Number	32	$\pm 10^{\pm 38}$
SINT	Short Integer	8	-128 to 127
USINT	Unsigned Short Integer	8	0 to 255
INT	Integer	16	-32768 to 32767
UINT	Unsigned Integer	16	0 to 65535
DINT	Double Integer	32	-2147483648 to 2147483647
UDINT	Unsigned Double Integer	32	0 to 42944967296
TIME	Duration	32	Up to 49 days
TIME_OF_DAY	Time of Day	32	00:00:00 to 23:59:59
DATE	Date	32	01-Jan-1970 to 01-Jan-2136
DATE AND TIME	Date AND Time of Day	32	01-Jan-1970-00:00:00 to 01-Jan-2136-23:59:59
STRING	Variable length character string		
ENUM	Enumerated item	32	0 to 2^{32}

Table 1-3 Function Block Parameter Data Types