IPCL Editor

Editor for IPCL Programs

IPCL is a process control language developed by Schneider Electric to tailor the application program in an RPU (Remote Processing Unit), to the desired application. Monitoring and control of the plant, as well as calculations and regulation, are handled with great capacity and flexibility.

IPCL is a high-level language that is used to define the relations in an RPU between inputs and outputs, between control logic, controllers and time control. The language is easy to program, read, document and test. Errors can easily be found and corrected.

Programming Language

The language consists of a number of instructions and rules for their use. The programming language is English.

The control code can be stored as a unique code for a certain RPU or as a standard (default) code for a particular type of RPU.

Editor

The program is edited as a text file in the IPCL editor.

The IPCL editor works like a text processor with additional functions for converting and transferring program code.

By using windowing techniques it is possible to have several files and programs open at the same time. Text and pieces of text can be copied from other files or programs using conventional editing tools. This makes it possible to re-use program code from other plants and also to use any text processor, such as Microsoft Word.

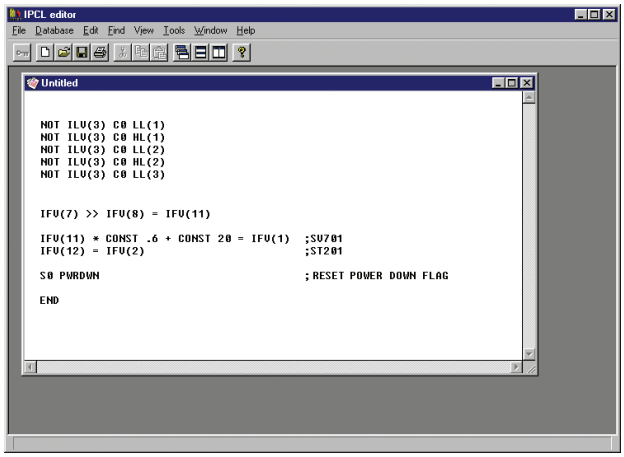

Figure 1: IPCL Editor

Functions

- Editing is done in a text file, using, for instance, the find/replace and cut/paste tools. The fast windowing functions provided by Microsoft Windows are also used.
- Conversion of text files to executable program code for RPUs.
- Program transfer; that is, conversion followed by transmission of the program code to the RPU that is configured as the receiver.

TECHNICAL DATA

Operating System

See requirements in TAC Vista Workstation data sheet, 03-00022-06 or later.

Hardware Requirements

PC	Intel Pentium or compatible
Minimum	2 GHz 32-bit (x86)
Recommended primary memory	3 GB
Minimum primary memory	1 GB
Required hard disk space after installation	> 300 Mbyte

Part Numbers

IPCL Editor	000882251
IPCL Editor – Upgrade	000883251
IPCL Editor – 1 year subscription	000884250

PROGRAM INSTRUCTIONS

Examples of program instructions are written below. They vary for every type of RPU in the system.

Logical Instructions

AND	Logical AND
OR	Logical OR
NOT	Logical NOT
NE	Logical NOT EQUAL (XOR)
SET	Logical assignment
NET	Inverted assignment
TVT	Time variable timer
TVA	Alternative timer
S1	Unconditional setting to 1
S0	Unconditional setting to 0

Logical Variables

IN	Digital inputs
LL	Low alarm
HL	High alarm
OUT	Digital outputs
FI	Fictive variables
ILV	Internal logical variables
TVS	Time variable status
TVAS	Alternative control status

Special Logic Functions

TIME	Time assignment
PWRDWN	Set when the power returns after a power failure
OFFLINE	Communications check
CLARM	Set if there is/are alarm/s at central unit
LLARM	Set if there are alarms in the alarm stack

Forcing

F1	Unconditional forcing ON
F0	Unconditional forcing OFF
AU	Return to normal, non-forced status, AUTO
C1	Conditional forcing ON if LAC*=1, otherwise AUTO
C0	Conditional forcing OFF if LAC*=1, otherwise AUTO

Arithmetic Functions

RT	Running time
CNT	Counter
MV	Measured value table
SV	Set value table
RV	Controller variable
IFV	Table for internal floating-point number variables
AOUT	Table for analog outputs
SQRT	Square root; stored in FLAC**
+	Addition
-	Subtraction
/	Division
*	Multiplication
>>	Max
<<	Min
>	Arithmetical comparison with logical result
>=	Arithmetical comparison with logical result
<	Arithmetical comparison with logical result
<=	Arithmetical comparison with logical result
=	Arithmetical assignment
CONST	Constant: integer or floating-point number

Conditions and Jumps

J xxx	Unconditional jump to xxx
CJ xxx	Unconditional jump to xxx if LAC*=1, otherwise proceed to next instruction
NJ xxx	Unconditional jump to xxx if LAC*=0, otherwise proceed to next instruction
JSR xxx	Jump to subroutine with address xxx
RTN	Return jump from subroutine
IF	Start instruction
THEN	Followed by the instructions to be performed if the condition is true
ELSE	Followed by the instructions to be performed if the condition is false
ENDIF	End instruction

Other Instructions

AL	Alarm condition
PU	Pulse condition
R1 (n)	Activate controller (n)
R0 (n)	De-activate controller (n)
MI (n)	Drive output signal from controller REG (n) to minimum
MA (n)	Drive output signal from controller REG (n) to maximum
DON	ON delay
DOFF	OFF delay
DOFFON (n)	ON/OFF delay
D v	Deviation; update on deviation of at least v units
T n	Update every n seconds
DL (n) y	Delay on jump to y, delay time as per table n
RPU (n)	RPU unit number n; n=0: global to all RPUs
ZC (n)	Zone controller unit number n
ZG (n)	Zone controller group number n
STIM hh:mm	Set clock
YEAR	Taken from real-time clock in RPU
MONTH	Taken from real-time clock in RPU
DATE	Taken from real-time clock in RPU
HOUR	Taken from real-time clock in RPU
MIN	Taken from real-time clock in RPU
SEC	Taken from real-time clock in RPU
WDAY	Day of week; 1=Monday … 7=Sunday
END	Program end

*) LAC = Logical accumulator. LAC is an auxiliary variable which serves as a storage location for the last read or calculated logic states.

**) FLAC = Floating-point accumulator. FLAC is an auxiliary variable which serves as a storage location for the last variable read or the last floating-point number calculated.

03-00017-05-en Jan 2011

For more information visit

www.schneider-electric.com/buildings

Schneider Electric