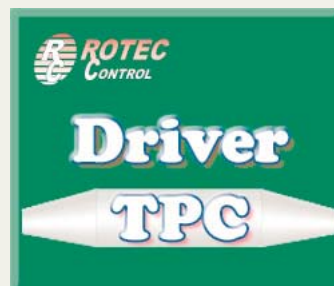
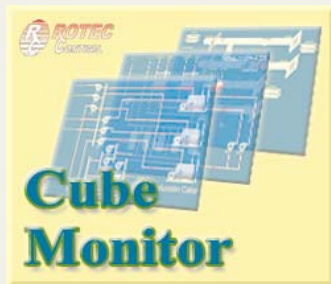
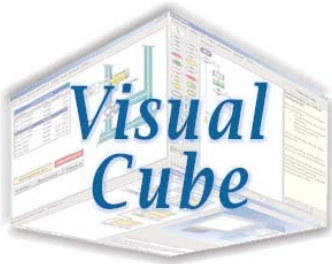


Scada Visual Cube



Sistemas de control
Elementos para la automatización
Reguladores de proceso



El programa scada **Visual Cube** es una herramienta integral creada específicamente para el producto ROTEC CONTROL y que permite desarrollar por completo un control de una instalación de forma gráfica y sencilla.

La principal característica de VISUAL CUBE que le diferencia de las demás herramientas de desarrollo de aplicaciones de control, es que su coste para Usted es de **0 €**. **Es gratuito y de libre distribución.**

El SCADA funciona sobre el sistema operativo Windows versión 98, ME, NT, 2000 y XP y requiere de unas necesidades mínimas de equipo para funcionar adecuadamente.

Las características mínimas recomendadas son:

- PC PIII o superior
- 128 Mb de RAM (recomendado 256)
- 180 Mb de espacio libre en disco duro.
- Tarjeta gráfica de 800 x 600 de resolución con 16 bits.
- Teclado y ratón
- CD Rom o conexión a Internet para la instalación del programa.
- Tarjeta de red o acceso telefónico a redes con protocolo TCP/IP instalado.

El programa VISUAL CUBE consta de tres apartados dentro del programa, que se resumen a continuación:

1.- Distribución:

La distribución permite seleccionar, colocar y auto-conectar, sobre la pantalla, los equipos de que va a constar la instalación. (Imagen 1)

Puede comenzar una distribución mediante un PC, un RC700 o un RC7 REG y agregar después del equipo de control las terminales que éste controlará.

La configuración de los equipos de control es automática en el 90% y sólo las terminales RC7 TIR requieren de una configuración detallada de los módulos y descripciones de cada uno de los puntos de entrada/salida.

Al configurar las terminales RC7 TIR, también puede configurar las sondas que se van a conectar en cada uno de los puntos. Para ello dispone, dentro del programa, de una base de datos de las sondas ROTEC donde, además, puede ir añadiendo sus propias sondas que utilice en las instalaciones. (Imagen 2)

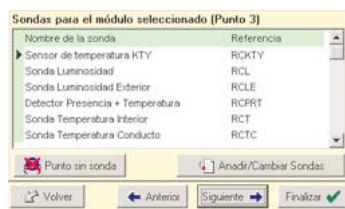


Imagen 2 - Base de datos de sondas.

La configuración de las terminales RC7 TIR se desarrolla paso a paso con explicaciones y una ayuda permanente de los pasos y opciones que debe ir realizando para configurar correctamente las terminales. (Imagen 3)

La distribución dispone de las opciones necesarias para acceder a los demás apartados del programa SCADA y para poder enviar los programas y configuraciones directamente a los equipos programados de la instalación.

Con pocos clicks de ratón y varias opciones fáciles de interpretar, podrá transmitir la programación y la configuración a todos los equipos que consta la instalación.

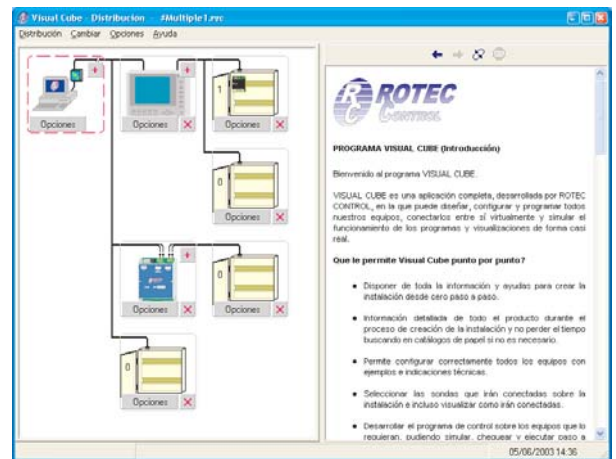


Imagen 1 - Pantalla principal de la distribución.

Con el sistema de elementos gráficos de la distribución puede definir fácilmente cómo irán distribuidas las RC7 TIR dentro de los armarios y el programa le calculará automáticamente qué material es el más adecuado para realizar esa configuración.

Dentro de los armarios puede añadir o quitar railes donde se colocarán las terminales y a medida que las vaya colocando, el programa las irá enumerando.

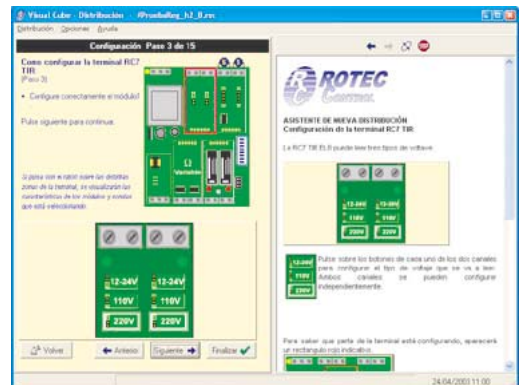


Imagen 3 - Configuración de las rc7 TIR

2.- Programación:

Toda instalación que se pretende controlar de forma automática, necesita de una serie de normas o instrucciones que le transmiten como y cuando debe funcionar cada elemento.

La facilidad y la rapidez en el desarrollo de programas de control siempre ha sido una prioridad para nosotros. Hasta el momento los programas SCADA existentes en el mercado obligaban a tener conocimientos extensos en la materia de programación para desarrollar aplicaciones de control de instalaciones. Con nuestro sistema, si usted puede desarrollar en un diagrama de flujo u organigrama de cómo debe funcionar la instalación, **ya tiene el programa hecho!**

En este apartado dispone de 30 macro-instrucciones gráficas muy potentes que, combinadas entre sí y colocadas sobre la pantalla en forma de diagrama de flujo, permiten desarrollar cualquier programa de control simple o complejo. (Imagen 4)

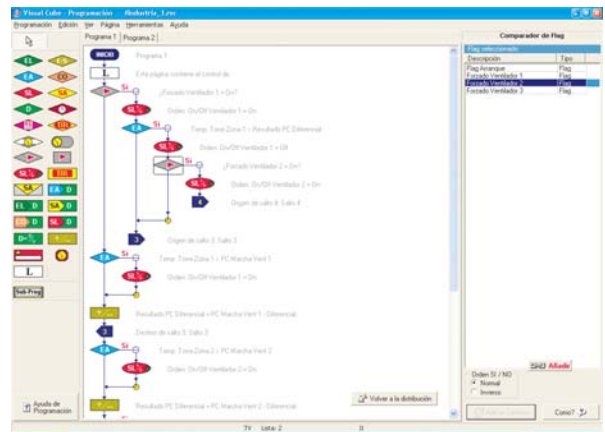


Imagen 4 . Instrucciones y método de programación mediante un organigrama de flujo

Los tipos de datos que se utilizan para ejecutar una regulación son: Señales de entrada/salida, datos con coma flotante, flags, horarios, calendarios, temporizadores, mensajes de alarma, parámetros y estados de terminal.

Las instrucciones se colocan y añaden con la misma facilidad que si de un programa de dibujo se tratara. Seleccionando cualquier instrucción del panel de instrucciones, mueva el ratón sobre el diagrama de flujo del programa y pulsando el botón del ratón la instrucción se coloca y conecta en el sitio indicado. Sólo queda seleccionarla y configurarla mediante las opciones disponibles sobre la misma pantalla.

El sistema se flexibiliza todavía más pudiendo crear distintas páginas de programa, facilitando el desarrollo por zonas, lo que simplificará la simulación real de que dispone este apartado de programación. También puede crear o importar subprogramas que, a partir de las mismas 30 macro-instrucciones, asignando a los parámetros de las instrucciones las señales y datos del programa principal, permite simplificar el control repetitivo de máquinas en un solo bucle de ejecución.

El simulador real del programa de regulación, permite modificar todos los valores de señales, datos, horarios, calendarios, flags, etc. del programa y ver cómo actuará la regulación sobre la instalación. (Imagen 5)

Dentro del apartado de programación también puede configurar qué señales o datos se tienen que capturar en forma de registro histórico para que, durante la ejecución de la programación, se almacenen los valores reales de la instalación con la frecuencia y duración indicados en la configuración.

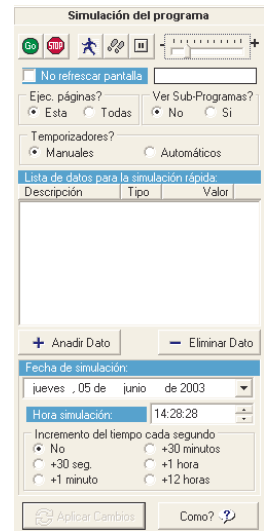


Imagen 5. Herramientas de simulación de los programas.



Imagen 6. Configuración de registros históricos

Los tiempos que permite la captura son:

MUESTREO: Desde 5 segundos hasta 12 horas configurables independientemente para cada una de las capturas.

DURACIÓN: Desde 1 día hasta varios años (dependiendo de la capacidad de memoria del equipo que muestrea).

VALORES: Valor real, Máximo y Mínimo, Media de valores dentro del muestreo, Cambios de valor y horas de funcionamiento.

Todos los registros históricos capturados por el equipo de regulación son rotativos dentro del mismo fichero y cuando el registro definido ha sobrepasado la duración en tiempo de los registros, vuelve a empezar desde el principio del mismo y sobre escribe los valores anteriormente recogidos. (Imágenes 6 y 7)

El sistema está pensado para que el programa ejecutor capture valores durante un tiempo establecido y, antes de que este periodo se cumpla, se descarguen, desde el programa CUBE MONITOR o cualquier otro programa de monitorización, al disco duro, donde se almacenarán de forma definitiva en ficheros independientes por señal y por días en el caso de CUBE MONITOR.

Se pueden definir tantos registros históricos como capacidad disponga el equipo que tiene que almacenarlos temporalmente.

Una vez descargados los registros históricos sobre el programa CUBE MONITOR, éste dispone de opciones y pantallas para poder mostrar gráficamente los valores de las señales capturadas en formato gráfica de línea. También dispone de opciones para exportar estos valores a ficheros, en formato separado por comas, para su posterior importación en hojas de cálculo o bases de datos.

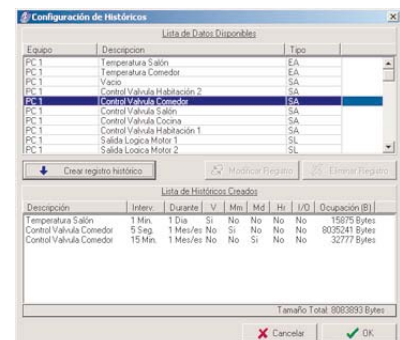


Imagen 7. Lista de registros históricos

Todas las instrucciones de Visual Cube son gráficas y fáciles de identificar de un solo vistazo. La ventaja del sistema de instrucciones gráficas y colocadas en forma de organigrama es que permite interpretar un programa realizado por cualquier persona rápidamente.

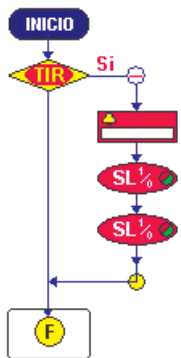
Cada una de las macro-instrucciones se comporta como una instrucción a la vista del usuario, pero internamente ejecutan una gran cantidad de instrucciones que, desarrolladas en un lenguaje de alto nivel se convertirían en muchas instrucciones por cada una de éstas.

Aquí le relacionamos la lista de instrucciones disponibles dentro de la programación:

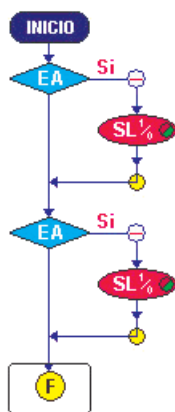
Instrucciones especiales	
	29 - Final de página
	30 - Destino de salto
	31 - Final de condición
	32 - Inicio de Página
	33 - Inicio de Sub-Programa

- | | | | |
|--|------------------------------|--|-----------------------------------|
| | Comparador Entrada Lógica | | Comparador Dos Entradas/Salidas |
| | Comparador Entrada Analógica | | Comparador Contador |
| | Comparador Salida Lógica | | Comparador Salida Analógica |
| | Comparador de Dato | | Comparador de Horario |
| | Comparador de Calendario | | Comparador de Errores de Terminal |
| | Comparador de Temporizador | | Asignador de Temporizador |
| | Comparador de Flag | | Asignador de Flag |
| | Orden de Salida Lógica | | Orden de Salida Analógica |
| | Pasa EA a DATO | | Pasa EL a DATO |
| | Pasa SA a DATO | | Pasa CONTADOR a DATO |
| | Pasa SL a DATO | | Pasa DATO a DATO o valor fijo |
| | Formula | | Alarma |
| | Cronómetro | | Texto libre |
| | Sub-Programa | | |

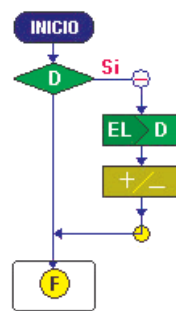
Algunos ejemplos de programación y su significado lógico:



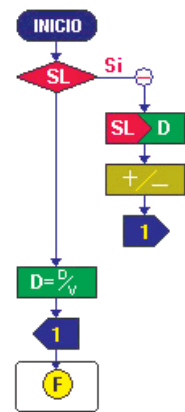
[SI] Error Terminal
 [MENSAJE] Terminal no funciona correctamente
 [DEACTIVA] Salida Lógica 1
 [DEACTIVA] Salida Lógica 2
 [FINAL DE PÁGINA]



[SI] EA >= Punto consigna
 [ACTIVA] Salida lógica
 [SI] EA < Punto consigna2
 [DEACTIVA] Salida lógica
 [FINAL DE PÁGINA]



[SI] Valor > 0
 [COGE VALOR] Valor = EL
 [FORMULA] Veces = Veces + 1
 [FINAL DE PÁGINA]



[SI] SL = ON
 [COGE VALOR] Valor = SL
 [FORMULA] Veces = Veces + 1
 [ORIGEN DE SALTO a 1]
 [ASIGNA VALOR] Veces = 0
 [DESTINO DE SALTO 1]
 [FINAL DE PÁGINA]

OPCIONES ESPECIALES DENTRO DE LA PROGRAMACIÓN:

Una característica importante dentro del apartado de programación es la de poder importar e incluso crear una librería de **sub-programas** con el fin de ir añadiendo facilidades de programación al entorno y disminuir, aún más si cabe, los tiempos de desarrollo de programas de control. También dispondrá en nuestra Web de una zona de descarga de sub-programas que se irán añadiendo a nuestra librería para facilitarle el control de elementos complejos, en los cuales sólo tendrá que configurar en qué puntos de entrada o salida quiere actuar y qué parámetros son los que regirán su funcionamiento.



Para equipos programados en Visual Cube

PROGRAMACIÓN DIRECTA CON LENGUAJE DE ALTO NIVEL:

Entre los programas desarrollados por ROTEC para la comunicación con los equipos de control y las terminales, existe uno que puede ser utilizado por cualquiera que disponga de su propio programa de control de instalaciones. DRIVER TPC es una librería DLL standard de Windows que funciona sobre plataformas, Windows XP, 2000, NT y Vista.

Como usar el Driver TPC:

El Driver TPC DLL dispone de funciones que pueden ser llamadas desde la aplicación desarrollada en cualquier lenguaje de alto nivel. Al tratarse de una librería de funciones que se incluye directamente sobre el código fuente, se usa como una llamada de función del propio lenguaje.

Dichas funciones permiten configurar las comunicaciones de la DLL con las terminales que tiene conectadas en el BUS o en el RS232. Una vez configuradas, la propia DLL funcionará de forma autónoma y realizará una exploración de las terminales, capturando sus valores y traspasándolas a la aplicación de alto nivel.

Las declaraciones de las funciones para el lenguaje Delphi son:

Funciones específicas de TIR

```
procedure InicializarListaTIR(TPC: Byte; Canal: Byte); stdcall; external 'DriverTPCdll.dll';
function AnadirTIR(TPC: byte; Canal: byte;Codigo: byte; Placa1: byte; Placa2: byte; Placa3: byte; Placa4: byte; Prioridad: byte): smallint; stdcall; external 'DriverTPCdll.dll';
function PonerValorTIR(TPC: byte; Canal: byte;Codigo: byte; Punto: byte; Valor: single): smallint; stdcall; external 'DriverTPCdll.dll';
function CogerValorTIR(TPC: byte; Canal: byte;Codigo: byte; Punto: byte): single; stdcall; external 'DriverTPCdll.dll';
function MostrarInfo(tpc: Byte; canal: Byte): Smallint; stdcall; external 'DriverTPCdll.dll';
procedure ProcCambiosTIR(proc1: TProcCambioValor; proc2: TProcCambioEstado); stdcall; external 'DriverTPCdll.dll';
function CogerEstadoTIR(TPC: byte; Canal: byte;Codigo: byte): smallint; stdcall; external 'DriverTPCdll.dll';
function CogerResetTIR(TPC: byte; Canal: byte;Codigo: byte): smallint; stdcall; external 'DriverTPCdll.dll';
procedure PonerEstadoTIR(TPC: byte; Canal: byte;Codigo: byte; EnteradoReset: Byte; InhibirSalidas: Byte) stdcall; external 'DriverTPCdll.dll';
procedure PonerConfPunto(TPC: byte; Canal: byte;Codigo: byte; Punto: Byte; Conf: PConfPunto) stdcall; external 'DriverTPCdll.dll';
```

Funciones específicas de RC620-VC

```
function AnadirTRG(TPC: byte; Canal: byte;Codigo: byte): smallint; stdcall; external 'DriverTPCdll.dll';
function MostrarInfoTRG(tpc: byte; canal: byte;Codigo: Byte): Smallint; stdcall; external 'DriverTPCdll.dll';
function VerUsoCanal(tpc: byte; canal: byte): Longint; stdcall; external 'DriverTPCdll.dll';
function CogerDatosHechosTRG(tpc: byte; canal: byte; codigo: byte): Smallint; stdcall; external 'DriverTPCdll.dll';
function QuitarTRG(TPC: byte; Canal: byte;Codigo: byte): smallint; stdcall; external 'DriverTPCdll.dll';
procedure ReiniciarTRG(TPC: byte; Canal: byte;Codigo: byte); stdcall; external 'DriverTPCdll.dll';
function PonerValorTRG(TPC: byte; Canal: byte;Codigo: byte; Tipo: byte; Variable: Word; var Valor: TValorUniv): Smallint; stdcall; external 'DriverTPCdll.dll';
function CogerValorTRG(TPC: byte; Canal: byte;Codigo: byte; Tipo: byte; Variable: Word; var Valor: TValorUniv): Smallint; stdcall; external 'DriverTPCdll.dll';
procedure ProcCambiosTRG(proc1: TProcCambioTRG; proc2: TProcCambioInterno); stdcall; external 'DriverTPCdll.dll';
procedure ModoTestTRG(TPC: byte; Canal: byte;Codigo: byte; test: boolean); stdcall; external 'DriverTPCdll.dll';
function CogerEstadoTRG(tpc: byte; canal: byte; codigo: byte): boolean; stdcall; external 'DriverTPCdll.dll';
```

Funciones generales

```
function EstadoTPC(TPC: Byte): byte; stdcall; external 'DriverTPCdll.dll';
function PedirCommLibre(TPC: Byte; Canal: Byte; BitsTX: Word; BitsRX: Word; Reintentos: Byte; Buffer: PByteArray): smallint; stdcall; external 'DriverTPCdll.dll';
function DevolverCommLibre(var Estado: Smallint; var BitsRX: Word; Buffer: PByteArray): smallint; stdcall; external 'DriverTPCdll.dll';
procedure LibExit; stdcall; external 'DriverTPCdll.dll';
```

TProcCambioValor = procedure(TPC: byte; Canal: byte;Codigo: byte; Punto: Byte; Valor: single); stdcall;

TProcCambioEstado = procedure(TPC: byte; Canal: byte;Codigo: byte; Estado: Smallint; Reset: byte); stdcall;

Toda la documentación sobre los comandos está disponible dentro del DRIVER TPC.PDF que puede encontrar en nuestra dirección www.roteccontrol.com o bien en el paquete de desarrollo de Visual Cube.