Nota de Aplicación: CAN-105
Título: **Display Color Proculus P48272J43C-T01**
Autor: Ing. Iván C. Sierra

Revisiones	Fecha	Comentarios
0	18/03/20	

En la presente nota de aplicación interiorizaremos en el desarrollo de una simple interface de Usuario para el display inteligente P48272J43C-T01 de la empresa Proculus. Con este ejemplo se pretende demostrar la facilidad de uso y la versatilidad del mismo a la hora de llevar a cabo sus proyectos.

En esta nota de aplicación se da una muy breve descripción de como usar el programa UnicView AD y se recomienda la lectura UnicView AD User Guide.

Índice

Descripción del ejemplo

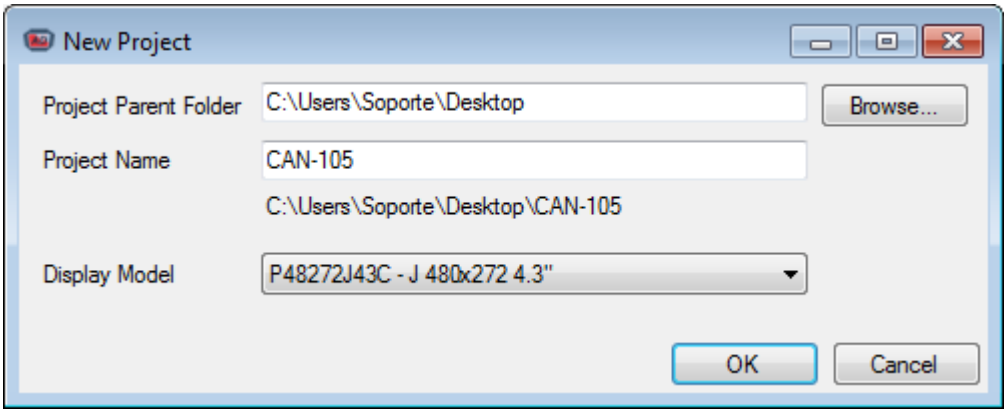
El ejemplo a desarrollar contará con una animación que se mostrara al energizar el display y, finalizada ésta, se mostrara un menú que contendrá 3 botones. Cada uno de estos botones permite acceder a configurar funciones y/o mostrar determinados datos, es decir, se crearan 3 pantallas adicionales.

La primer pantalla es la que permite configurar el RTC del display y se llega pulsando el botón con el icono del reloj. La segunda pantalla permite controlar el brillo y el encendido/apagado de una supuesta lampara conectada al sistema y a la cual se llega pulsando el botón con el icono de un sol. Por ultimo, la tercer pantalla permite verificar la temperatura de un determinado proceso y configurar una alarma y se llega a ésta pulsando el botón con el icono del termómetro.

Cada una de las pantallas indicadas antes tiene un botón que permite volver al menú principal.

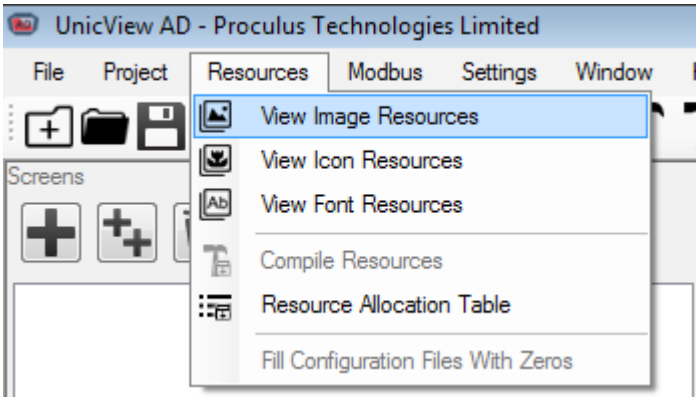
Desarrollo del proyecto.

Lo primero es abrir el programa UnicView AD y crear un nuevo proyecto, al cual hay que definirle un nombre, ubicación y tipo de display a utilizar.

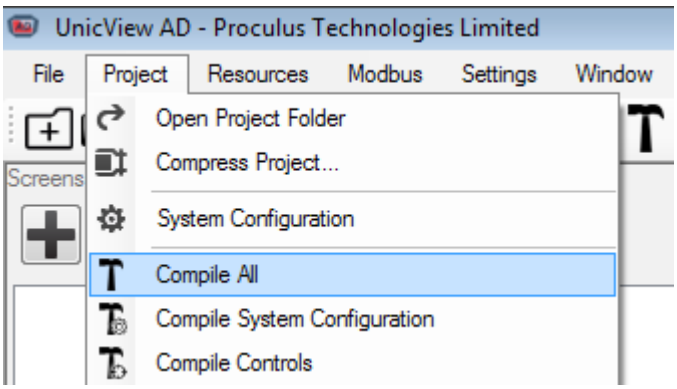


Carga de recursos e imágenes en el proyecto

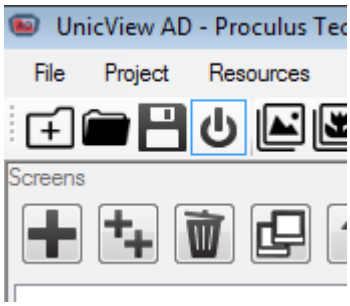
Con el proyecto ya creado se procede, en primer lugar, a cargar los recursos que se utilizaran en cada una de las pantallas. Para ello vamos a **Resources** y seleccionamos los recursos que se desea agregar. Los recursos pueden ser imágenes, iconos y fuentes (en caso de ser necesario). El programa tiene asignada una fuente por defecto, pero si el Usuario desea utilizar una fuente determinada deberá agregarla como un recurso.



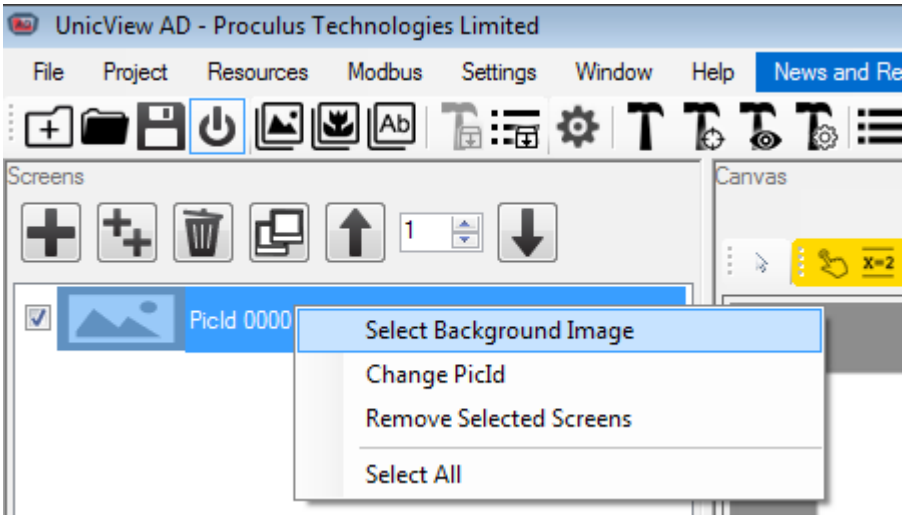
Una vez finalizado la carga de los recursos se debe compilar el proyecto pulsando el botón **Compile All** para que los recursos seleccionados estén disponibles para ser utilizados.

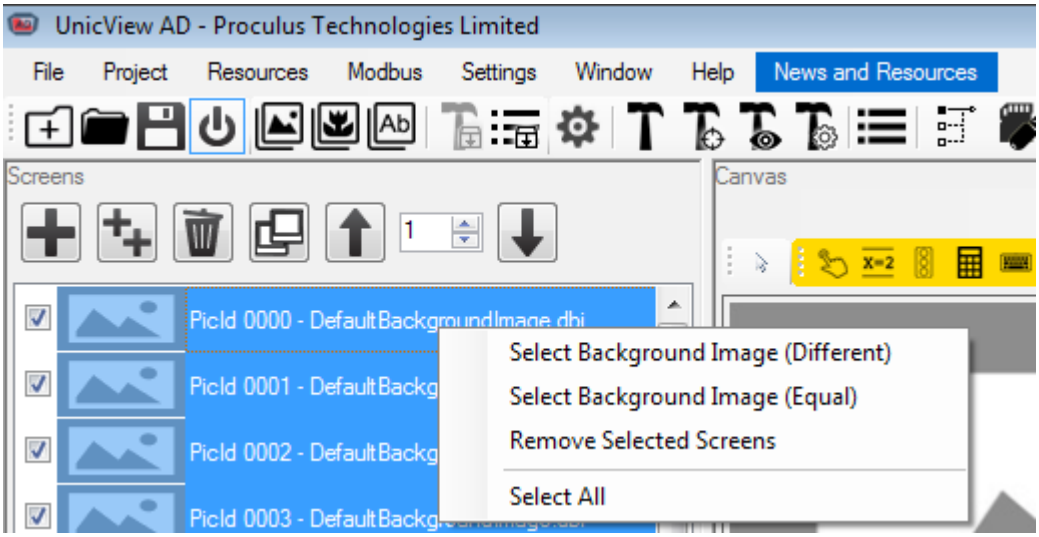


Con los recursos ya asignados se puede proceder a generar las imágenes que formaran parte del proyecto. Hay 2 formas de hacerlo. Una es generar las imágenes en forma individual usando el botón con el símbolo + y la otra es generar todas las imágenes usando el botón con el símbolo ++. En el primer caso se agrega una única imagen, mientras que en el otro se pueden generar el numero deseado de imágenes.



Una vez definida las imágenes se le debe asignar un recurso a cada una de ellas. Para ello se pulsa el botón derecho del mouse sobre la imagen y se selecciona “Select Background Image”. Se abrirá una ventana donde se muestran los recursos de imágenes, cargados previamente, para seleccionar la imagen de fondo deseada. Se debe realizar este procedimiento con cada una de las imágenes generadas. También puede seleccionarse mas de una imagen a la vez y cargar distintos recursos a las diferentes imágenes en un solo paso. Esto último resulta muy útil en el caso de querer generar una animación.

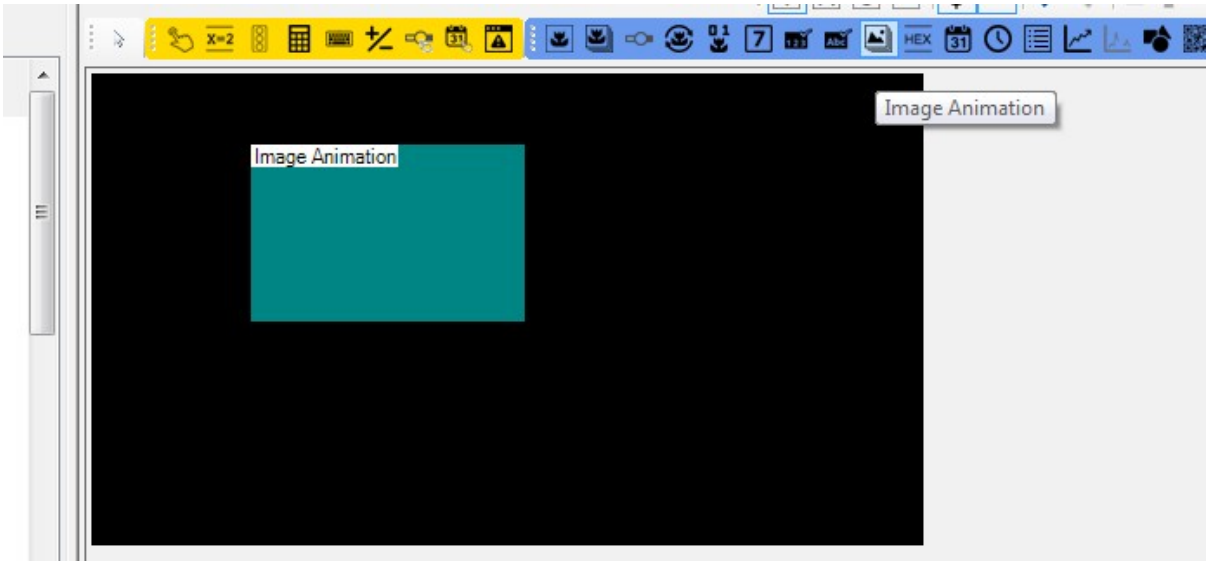




Con los recursos ya cargados y las imágenes ya definidas se procede generar distintas las pantallas (animación, menú principal, configuración del RTC, control de iluminación y control de temperatura).

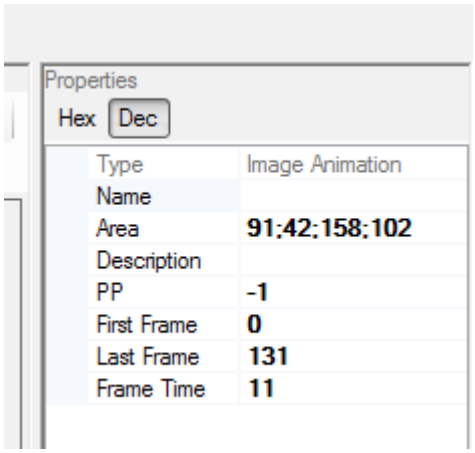
Animación

Como se comento, al encender el display el mismo muestra una animación antes de mostrar el menú principal. Para lograr esto se le debe agregar la variable **Image Animation**, que muestra una secuencia de imágenes con un intervalo de tiempo determinado, a la primera imagen que muestra el display al encenderlo (**PicId 0000**). En este caso la primera imagen se corresponde con la primara imagen de la animación, por lo tanto, la seleccionamos desde el panel **Screens** y en el panel **Canvas** se procede a colocar la variable **Image Animation** pulsando el icono correspondiente. El tamaño de la variable agregada y su posición, en este caso, no son importantes.



Si se selecciona la variable agregada a la imagen, se pueden ver las propiedades de la misma en el panel **Properties**. En la variable **Image Animation** se debe definir el ID de la primera y última imagen de la secuencia y el intervalo de tiempo entre las imágenes, que es múltiplo de 8mseg. La animación que se va a mostrar consta de 131 (**PicId 0000** – **PicId 0130**) imágenes, pero si se seleccionara la imagen **PicId 0130** como imagen donde finaliza al animación el display terminaría mostrando está imagen y no la del menú principal como se pretende hacer. Por lo tanto, es importante que la imagen a mostrar luego de la animación este justo a continuación de la secuencia de imágenes de ésta.

La configuración las propiedades de la variable **Image Animation** se muestra a continuación.



Menú principal

Finalizada la animación el display quedará mostrando la pantalla del menú principal, como se menciono antes. Esta pantalla cuenta con 3 botones. Para asignarle una función a cada uno de los botones del menú se les debe agregar un control del tipo **Basic Touch** a cada uno de ellos. En éste caso es importante la ubicación y el tamaño del control, ya que cada uno de ellos debe estar sobre los iconos, que harán de botones, y ocupar el área que sea necesaria para interpretar que el Usuario presiono ese icono. En este caso el tamaño de los iconos son suficientemente grandes para que el Usuario pueda tocarlos sin problemas y se tomo el tamaño del control igual al de los iconos.

Al igual que la variable **Image Animation**, el control **Basic Touch** tiene unas propiedades que definen su comportamiento. En este caso se le debe indicar a cual imagen debe pasar al pulsar sobre esa área, con la propiedad **JumpId,** y cual imagen mostrar mientras esa área es pulsada, con la propiedad **FxId**. Esto último es para dar el efecto visual que se presiono el botón. La configuración para los 3 botones es la siguiente:

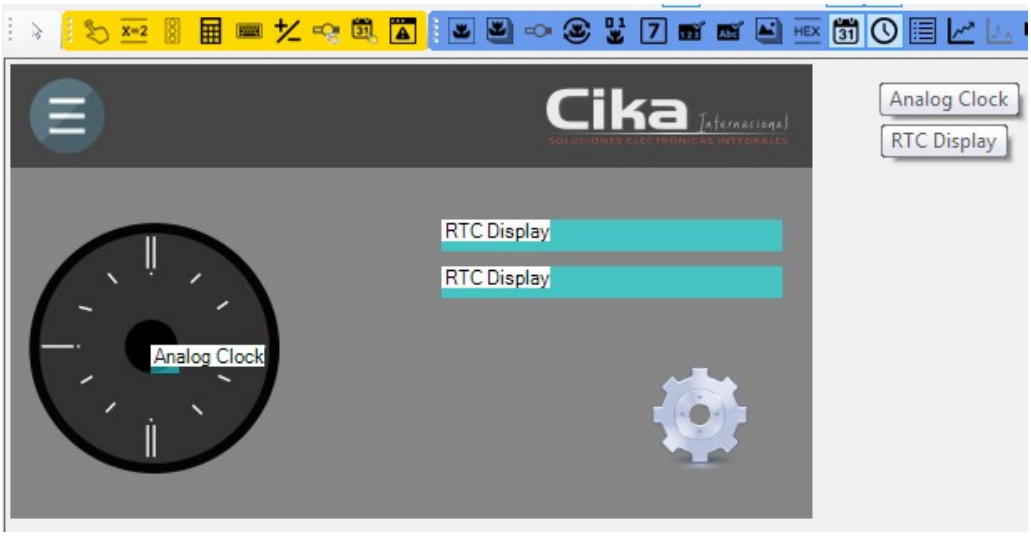
- Visualizar y configurar RTC:
  - **JumpId** = PicId 0133
  - **FxId** = PicId 0132
- Control de iluminación:
  - **JumpId** = PicId 0137
  - **FxId** = PicId 0132
- Control de temperatura:
  - **JumpId** = PicId 0139
  - **FxId** = PicId 0132

Visualizar y configurar RTC

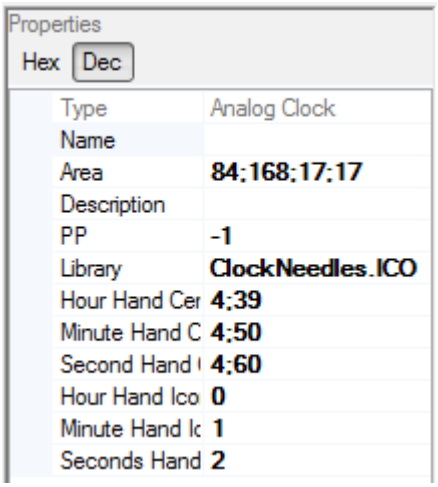
En esta pantalla se mostrara 2 formas de ver la información del RTC en el display. Una de ellas es por medio de la representación de un reloj de agujas (analógico) y la otra es por medio de texto.

La pantalla tiene información fija, que es la imagen de fondo cargada (**PicId 0133**), e información dinámica que se agrega en forma de icono y texto.

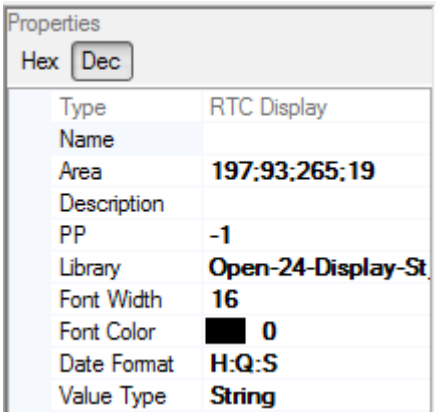
Para mostrar información relacionada con el RTC se debe agregar sobre la imagen las variables **Analog Clock** y **RTC Display**. Estos controles permiten mostrar la información del RTC en formato analógico (agujas) o en formato de texto, respectivamente.

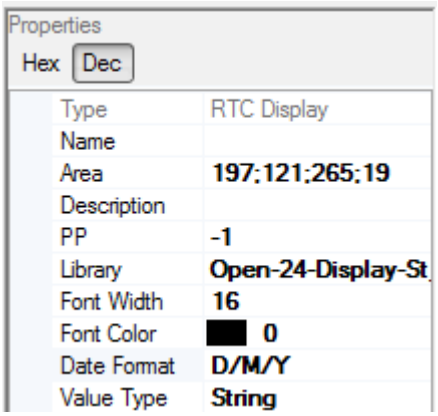


A la variable **Analog Clock** hay que definirle, principalmente, la librería donde están los iconos a usar para las agujas, la ubicación de cada uno de los iconos y el Id del icono correspondiente a cada una de las agujas (hora, minuto y segundos). Por lo tanto, las propiedades de está variable se configuran como se muestra a continuación.

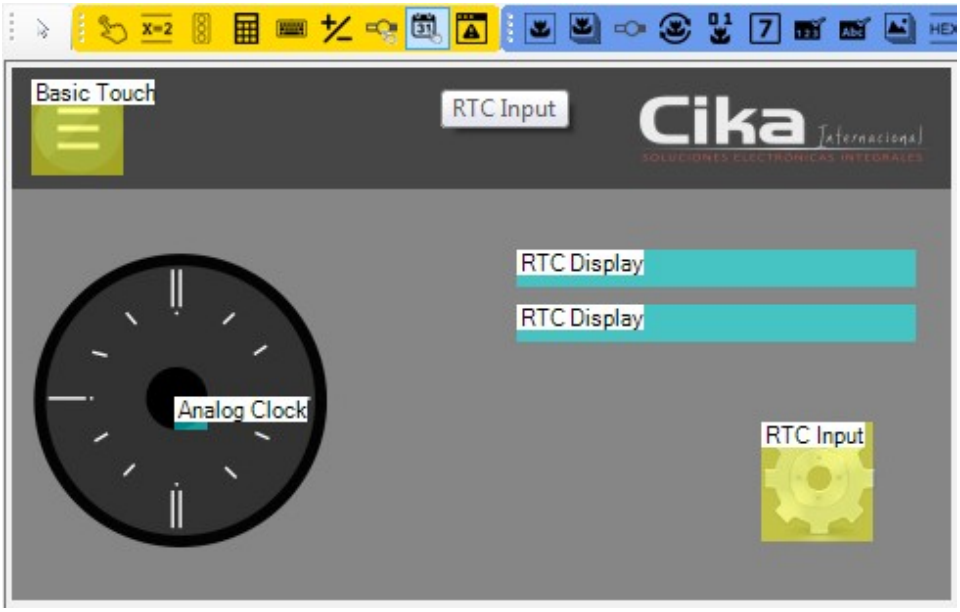


A las variables **RTC Display** hay que definirle, principalmente, la fuente a utilizar, el ancho y color de la misma y el formato de la información a mostrar. El formato define que información, relaciona con el RTC, se debe mostrar. En uno de los **RTC Display** se mostrará la hora con el formato **H:Q:S** y en el otro, se mostrara la fecha con el formato **D/M/Y**. Por lo tanto, las propiedades de está variable se configuran como se muestra a continuación.

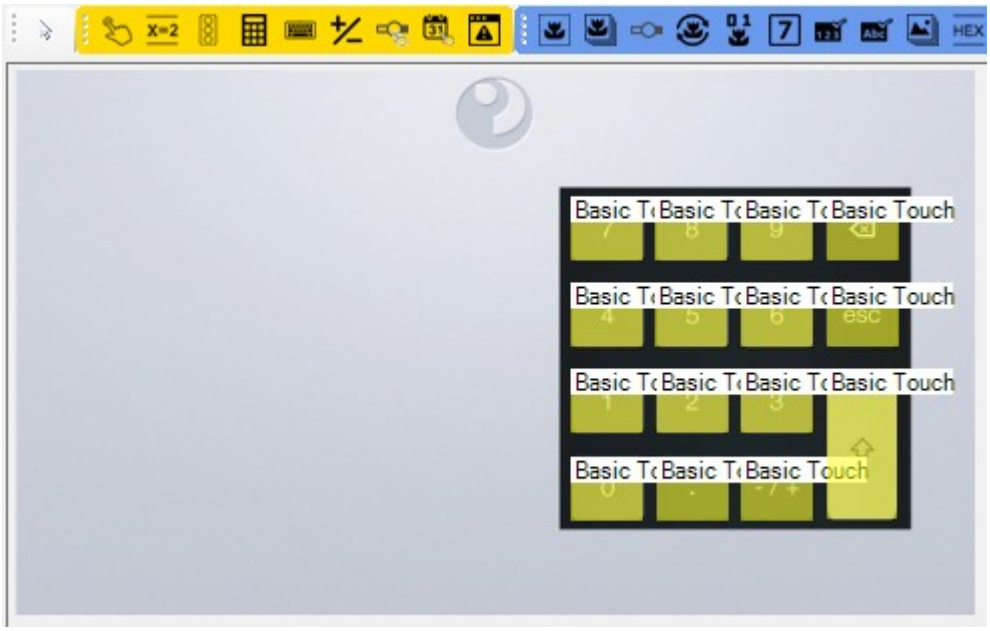




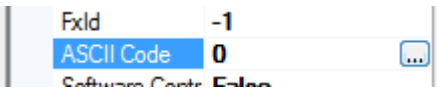
Ya se definió como y donde mostrar la información del RTC, ahora falta definir de que forma se puede modificar valor del RTC. Para ello haremos uso del control **RTC Input**, el cual permite superponer una imagen de un teclado a la pantalla principal y usarlo para ingresar la configuración. Este control lo ubicaremos sobre la imagen del engranaje, de manera que ocupe toda la superficie de la misma.



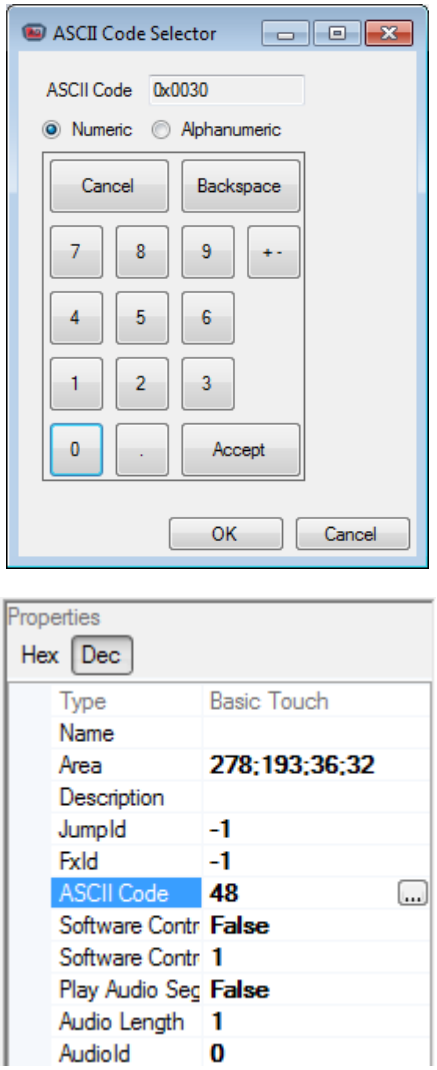
La imagen del teclado ya esta cargada en el proyecto y corresponde al **PicId 0135**. Antes de poder usar esta imagen como teclado numérico se debe asignar a cada una de las teclas un valor ASCII. Para ello agregamos sobre cada uno de los botones del teclado numérico un control **Basic Touch**, al igual que se hizo en el menú de principal, pero en este caso se dejara la propiedad **JumpId** en -1 y se configurara el **ASCII Code** de cada una de las teclas. Ademas se debe configurar la propiedad **FxId**. En este caso se configura la propiedad **FxId** con la imagen **PicId 0136**.



La asignación del código ASCII se puede hacer en forma manual o por medio del **ASCII Code Selector** que se muestra al pulsar el botón con los 3 puntos.



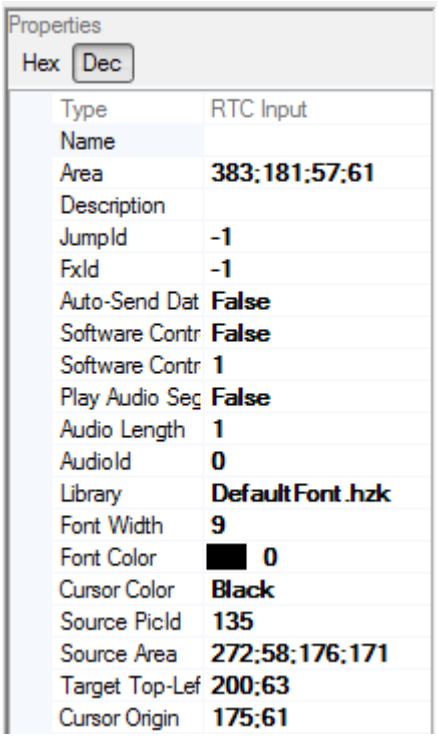
El **ASCII Code Selector** permite definir el código de cada uno de los botones del teclado numérico. Una vez seleccionado el número correspondiente al botón del teclado numérico se pulsa **OK** y el mismo es cargado en el campo **ASCII Code** del **Basic Touch**. Este procedimiento se debe repetir para todos los botones del teclado numérico.



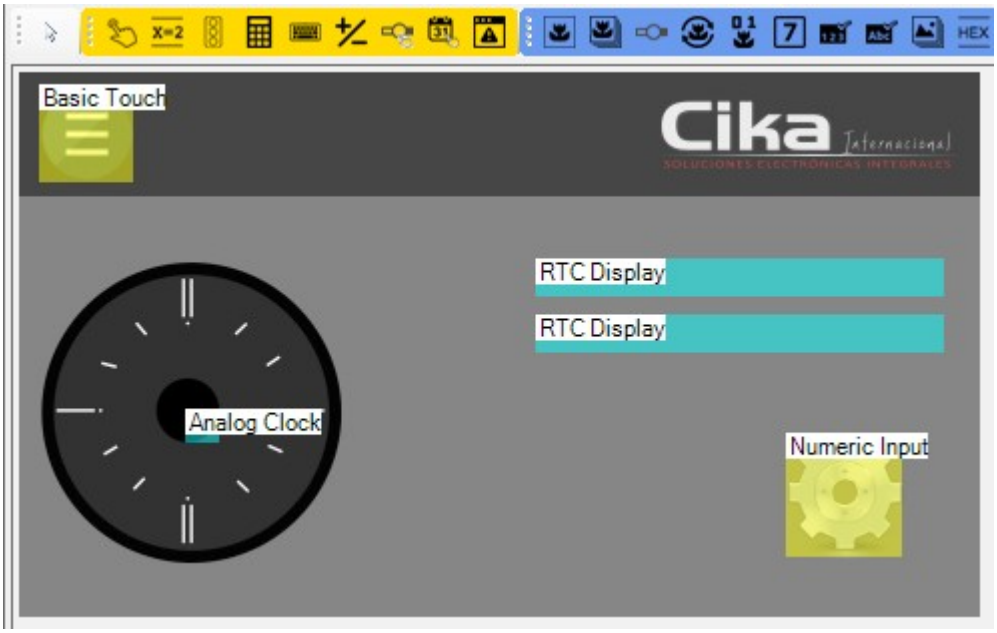
Con el teclado numérico ya configurado volvemos al **PicId 0133** y configuramos el **RTC Input**. Hay varias propiedades que se puede configurar en este control, pero las que nos interesan ahora son:

- **Library**: permite seleccionar una de las fuentes cargadas como recurso.
- **Font With**: define el ancho de la fuente.
- **Source PicId**: define la imagen a usar como teclado.
- **Source Area**: define el área de la imagen que se corresponder con el teclado
- **Target Top-Left Point:** Selecciona donde mostrar los datos ingresados.
- **Cursor Origen**: Selecciona donde aparecerá el cursor.

La configuración para el **Numeric Input** es la siguiente:



Para volver al menú principal se asignara un control del tipo **Basic Touch** al icono de menú, en la parte superior izquierda de la imagen, y se configurara el **JumpId** con el Id **PicId 0131** y el **FxId** con el **PicId 0134**.



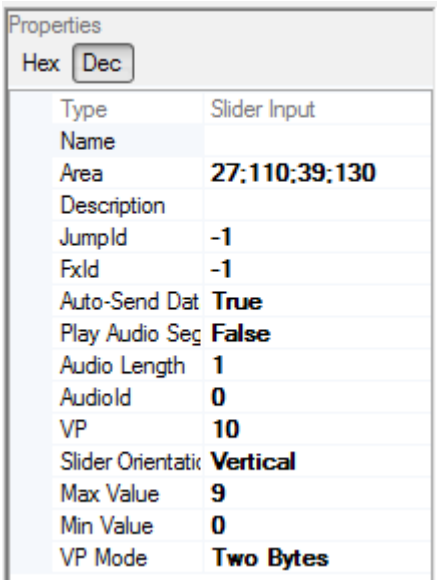
Control de iluminación

Al igual que en la pantalla de Configuración del RTC, en ésta pantalla habrá información fija, que es la imagen de fondo cargada (**PicId 0137**), pero se tendrá información que se modificara por interacción del Usuario con el display.

La iluminación se controla mediante 2 tipos de controles. Uno permite cambiar la intensidad de la iluminación (control deslizante) y el otra encender o apagar la luz (control On-Off).

Para implementar el control de intensidad se agrega al **PicId 0137** un control **Slider Input**, al cual se le define el valor máximo y mínimo que puede tomar mediante **Max Value** y **Min Value**, respectivamente. El valor seleccionado en el control se guarda en una posición de memoria **VP**[1] configurable por el Usuario. Para el proyecto que se esta implementando solo se permiten 10 niveles de intensidad y se define la posición de memoria **VP** = 10 (0x000A), quedando el control configurado de la siguiente manera:

1 Las posiciones de memoria VP pueden ser escritas por un control y leídas por una variable, pero ademas, pueden ser leídas o escritas mediante comando por el puerto serie



La propiedad **Auto-Send Data** en **True**, indica que el display enviara por el puerto serie el dato guardado en la posición de memoria **VP** asignada al control, cada vez que el Usuario actué sobre éste. Los datos enviados esta en hexadecimal y tienen el siguiente formato:

<FHH><FHL> <BC> 0x83 <VP><VP> <LEN> <VL1><VL1>

Donde:

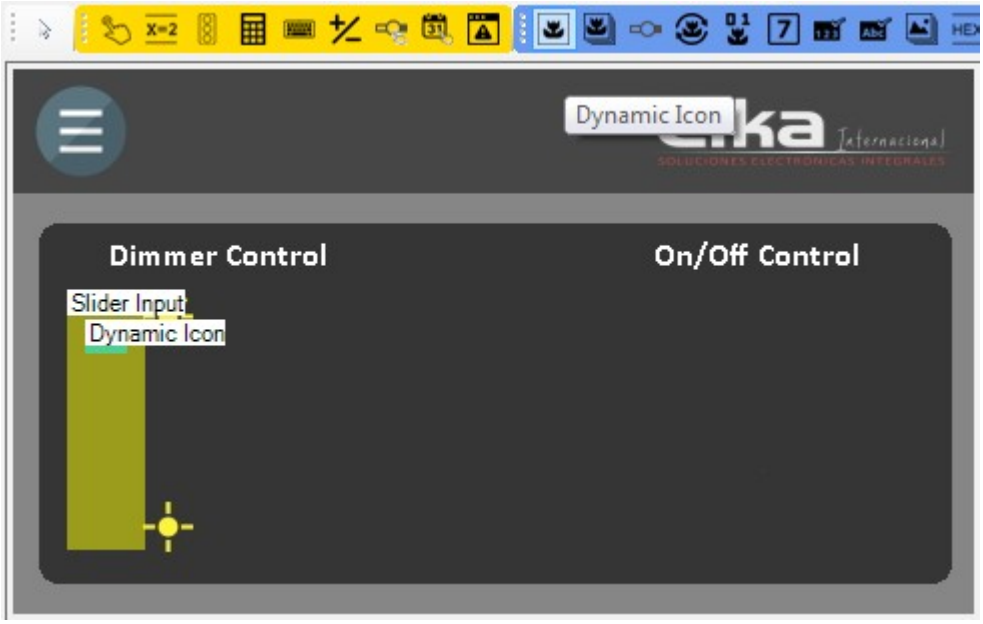
- **<FHH><FHL>**: es el Header (5AA5).
- **<BC>**: es la cantidad de bytes.
- **0x83**: Indica que es la lectura de un word (16bits) en RAM.
- **<VP><VP>**: dirección en la memoria RAM.
- **<LEN>**: número de VPs (Variable Pointer) a leer.
- **<VL1><VL1>**: dato (16bits).

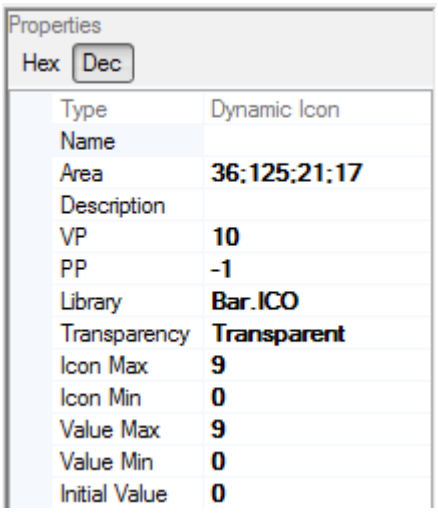
Por ejemplo, el control enviara los siguientes datos si se configuro una iluminación del %40.

5A A5 06 83 00 0A 01 00 04

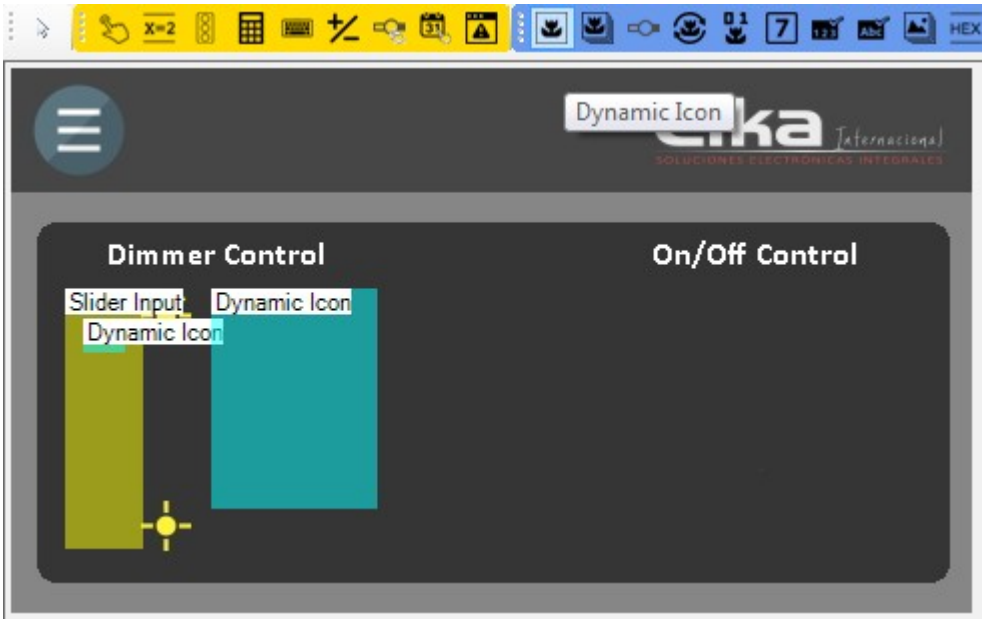
Para mas información al respecto consulte el UnicView AD Development Guide.

Ahora hay que asignarle una variable **Dynamic Icon** al control **Slider Input** para mostrar la barra del control deslizante. La variable **Dynamic Icon** cambia la imagen que muestra según el valor guardado en una posición de memoria **VP**. Para que la imagen cambie según el valor seleccionado en el **Slider Input**, hay que configurar la propiedad **VP** con el mismo valor de **VP** asignado al **Slider Input**, es decir, 10. Por ejemplo, si en **VP** esta almacenado el valor 8, el **Dynamic Icon** mostrará el icono con el Id 0x0008 que esta cargado dentro del recurso. El conjunto de iconos a usar por la variable se seleccionan, de los recursos cargados previamente, con la propiedad **Library**. Para este caso se utiliza el recurso **Bar.ico** que contiene las barras utilizadas para mostrar el control deslizante. Ademas de la propiedad **Library**, se debe configurar las propiedades **Icon Max**, **Icon Min**, **Value Max** y **Value Min**. **Icon Max e Icon Min** indican el Id máximo y mínimo para los iconos a usar, mientras que **Value Max** y **Value Min** definen el intervalo que puede tomar el valor almacenada en la posición de memoria **VP** para que la variable **Dynamic Icon** actué. Así, el **Dynamic Icon** queda configurado de la siguiente manera:





Ademas del control deslizante para indicar el nivel de intensidad de la iluminación, se mostrar la imagen de una lampara que aumenta su brillo al aumentar el valor del nivel de intensidad. Para lograr esto se debe agregar, también, una nueva variable **Dynamic Icon y** referenciada a la posición de memoria **VP** del control **Slider Input**.



Properties

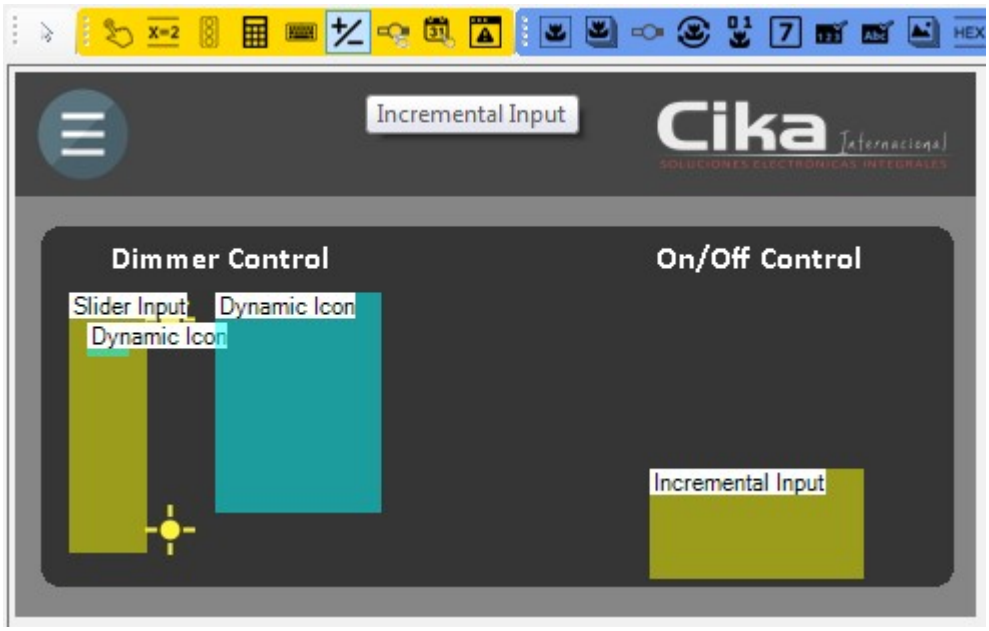
Hex | Dec

Type	Dynamic Icon
Name	
Area	**100;110;83;110**
Description	
VP	**10**
PP	**-1**
Library	**Led.ICO**
Transparency	**Transparent**
Icon Max	**9**
Icon Min	**0**
Value Max	**9**
Value Min	**0**
Initial Value	**0**

Para generar el control On-Off de la lampara utilizaremos un control **Incremental Input** y 2 variables **Dynamic Icon**. La variable **Dynamic Icon** ya se utilizo y no se dará mayor detalle de su implementación, solo se indica que se utilizan para animar el botón de On-Off y imagen de la lampara. Nos centraremos, en cambio, en la utilización del control **Incremental Input**. Este control permite incrementar o decrementar una posición de memoria **VP**. En este caso en particular se utilizara para incrementar una posición de memoria entre 0 y 1 con incrementos de 1 y con la propiedad **Loop** en **True**. La propiedad **Loop** permite cargar el valor mínimo o máximo definido para el control si el incremento supera uno de estos limites, es decir, que si el valor de **VP** es 1 y se vuelve a realizar un nuevo incremento, su valor pasará a ser 0, ya que el máximo valor permitido es 1.

Al igual que se hizo en el control **Slider Input**, se configuro la propiedad **Auto-Send Data** en **True**, para enviar el dato almacenado en la posición de memoria **VP** asignada al control, por el puerto serie.

Agregamos el control **Incremental Input** y lo configuramos como se muestra a continuación.

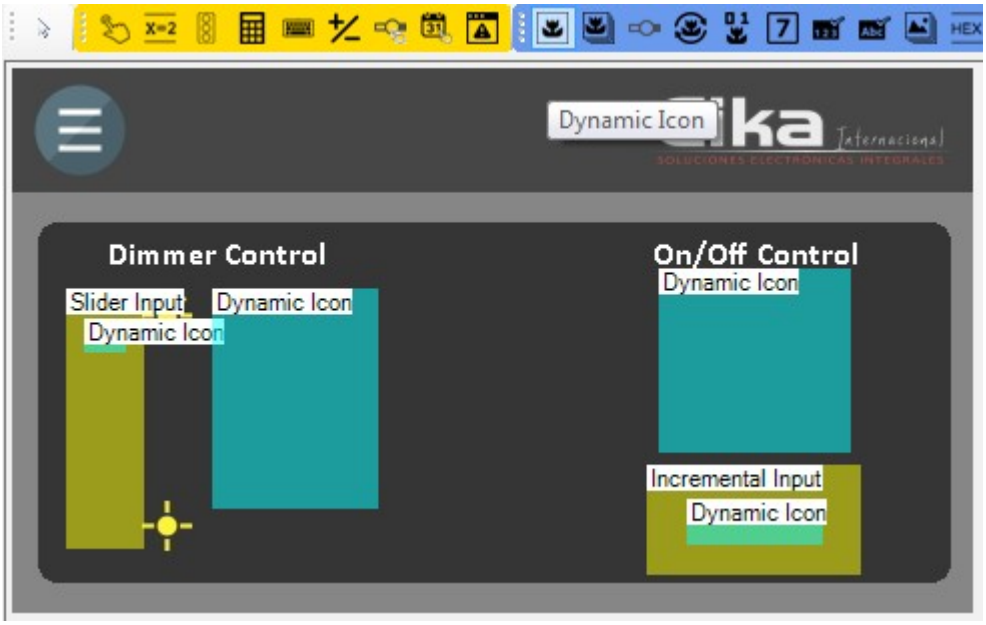


Properties

Hex | Dec

Type	Incremental Input
Name	
Area	**317;198;107;55**
Description	
JumpId	**-1**
FxId	**-1**
Auto-Send Dat	**True**
Software Contr	**False**
Software Contr	**1**
Play Audio Seg	**False**
Audio Length	**1**
AudioId	**0**
VP	**11**
Increment Sign	**+**
Increment Valu	**1**
Limit Value Rar	**False**
Max Value	**1**
Min Value	**0**
Continuous Inc	**False**
Loop	**True**
VP Mode	**Two Bytes**

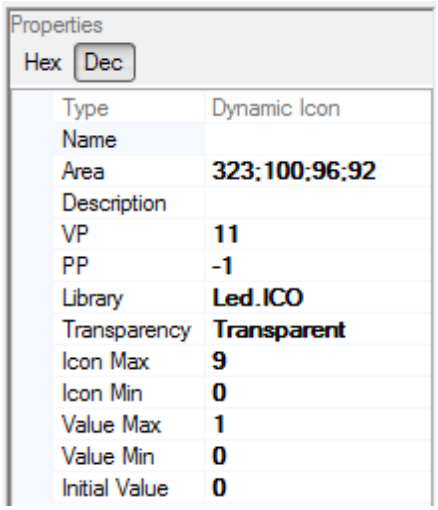
Como se dijo, se utilizan 2 variables **Dynamic Icon**. La configuración de cada una es la siguiente:



Properties

Hex | Dec

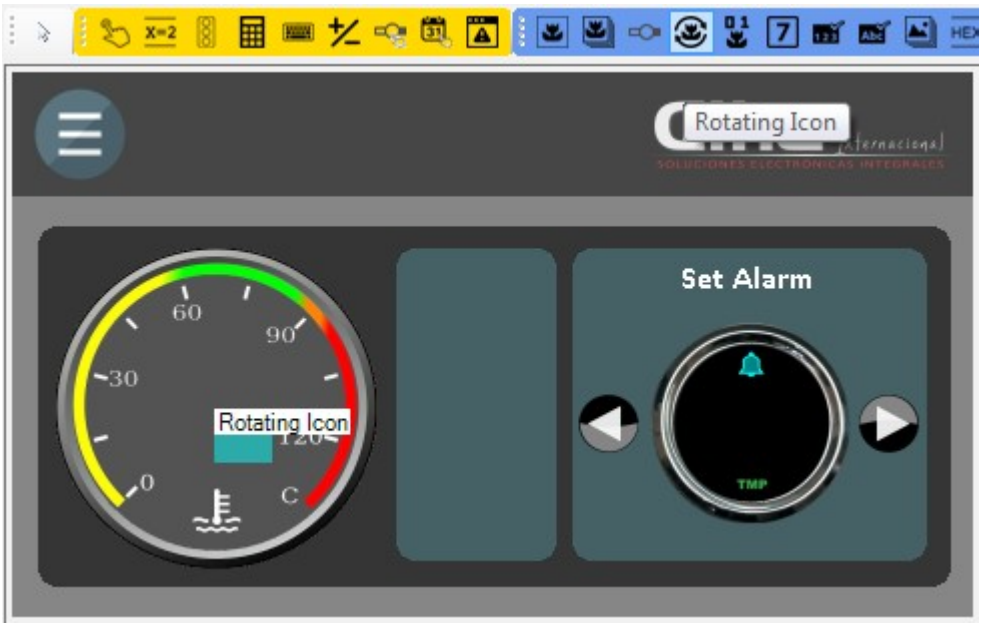
Type	Dynamic Icon
Name	
Area	**337;215;68;23**
Description	
VP	**11**
PP	**-1**
Library	**OnOff.ICO**
Transparency	**Opaque**
Icon Max	**1**
Icon Min	**0**
Value Max	**1**
Value Min	**0**
Initial Value	**0**

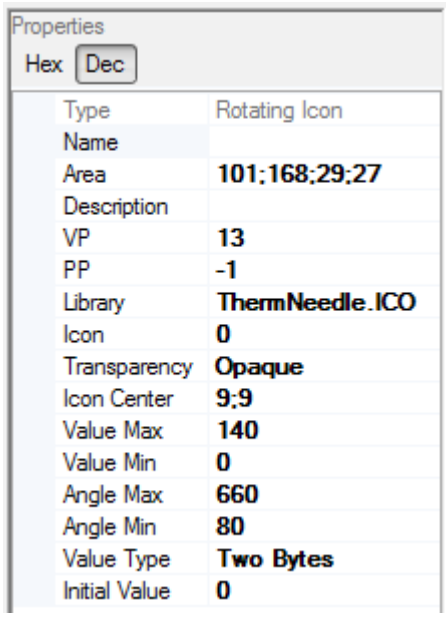


Control de temperatura

En esta pantalla se va a implementar la visualización de la temperatura de un proceso en 2 formatos, uno analógico (aguja) y otro digital, y la configuración de una alarma. Para esto se implementaran algunas variables y controles usados anteriormente, pero se agregaran 2 nuevas variables que son **Rotating Icon** y **Numeric Display**.

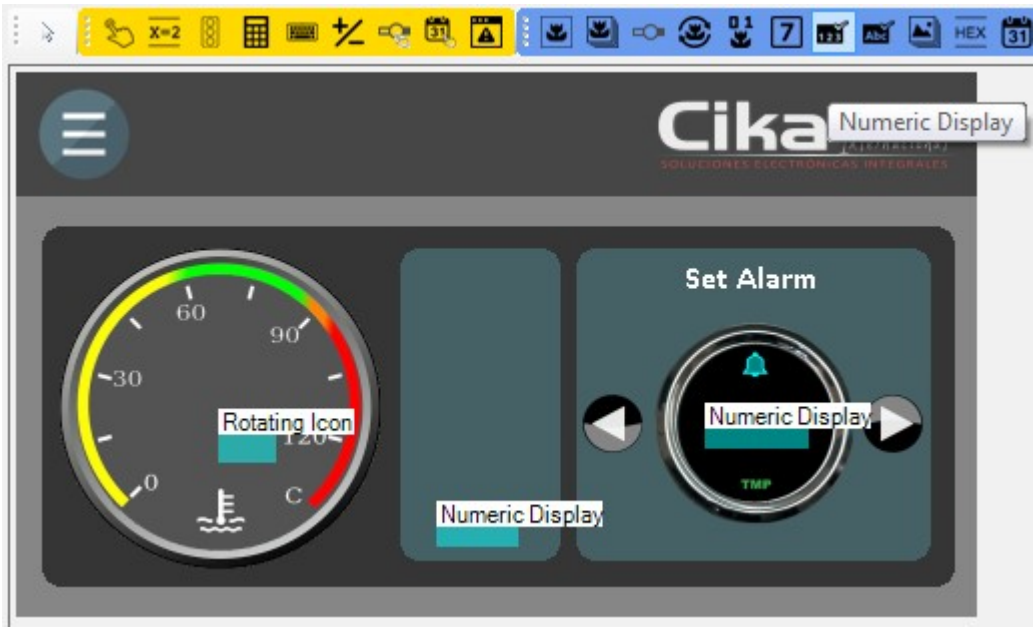
La variable **Rotating Icon** permite hacer rotar un icono sobre un punto. Esto nos va a resultar útil para representar la aguja en el medidor de temperatura analógico. El ángulo que rotara el icono depende del contenido de una posición de memoria **VP** que se le asigna a la variable. El valor de esa posición de memoria se modifica por medio del envió de un comando por puerto serie, en este caso. Ademas de asignarle una posición de memoria **VP** a la variable, hay que configurarle la librería donde está el icono a usar para la aguja, la ubicación del centro del icono que se toma como pivote, valores máximos y mínimos para la variable, angulo de giro máximo y mínimo y valor inicial. Por lo tanto las propiedades de está variable se configuran como se muestra a continuación.





Según el cuadrante, la aguja debería moverse entre los 40º y los 330º (fuera de escala), pero como la variable **Rotating Icon** permite pasos de 0,5º se duplico el ángulo para pasar a estar definido entre los 80 y 660.

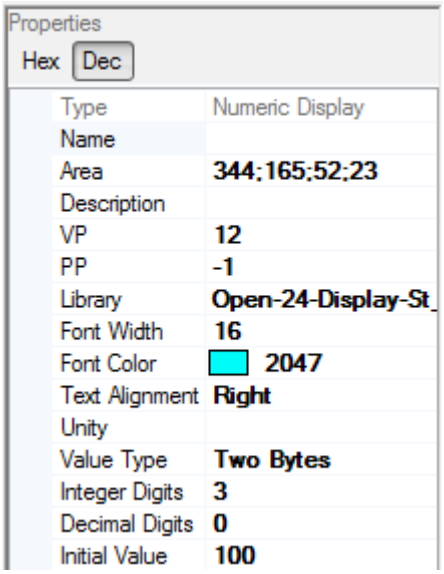
La variable **Numeric Display** permite mostrar un numero en el display y la utilizaremos para mostrar, por un lado, la temperatura en formato digital y por el otro, el valor configurado de la alarma. El valor que muestra la variable **Numeric Display** depende del valor almacenado en la posición de memoria **VP** asignada a la variable. En el caso de la variable **Numeric Display** que muestra la temperatura, se le asigna la misma posición **VP** que a la variable **Rotating Icon,** de esta manera ambas variables muestran la misma información de temperatura de 2 formas distintas. En el caso de la variable **Numeric Display** que muestra el valor configurado de la alarma, se le asigna la posición de memoria **VP** = 12 (0x000C) cuyo contenido se modificará por medio 2 controles **Incremental Input**. Uno de los controles incrementa el valor almacenado en la posición de memoria **VP** y el otro lo decrementa. Ambos controles lo hacen en saltos de 1 en 1. Un cambio en la configuración con respecto a su uso en la pantalla de Control de Iluminación, es que, en este caso si el botón se mantiene presionado se produce un incremento/decremento continuo del dato almacenado en la posición de memoria **VP** asignada al control. Para esto se configuro la propiedad **Continuous Increment** en **True**. Las propiedades de estas variables se configuran como se muestra a continuación.



Properties

Hex | Dec

Type	Numeric Display
Name	
Area	210;214;41;23
Description	
VP	13
PP	-1
Library	Open-24-Display-St_
Font Width	16
Font Color	2047
Text Alignment	Center
Unity	
Value Type	Two Bytes
Integer Digits	3
Decimal Digits	0
Initial Value	0



Como se menciono, la temperatura que muestra el display, tanto en formato analógico como digital, se basa en la información que contenga la posición de memoria **VP** asignada a las variables y que la misma va a ser escrita por medio de comandos enviados por el puerto serie. El comando que permite escribir una posición de memoria es el siguiente:

<FHH><FHL> <BC> 0x82 <VP><VP> <VL1><VL1>

Donde:

- **<FHH><FHL>**: es el Header (5AA5).
- **<BC>**: es la cantidad de bytes.
- **0x82**: Indica que es una escritura de un word (16bits) en RAM.
- **<VP><VP>**: dirección en la memoria RAM.

<VL1><VL1>: dato (16bits).

Por ejemplo, si se quiere escribir la posición de memoria 13 (0x000D) con el valor 300 (0x012C) el comando que se debe enviar en hexadecimal es:

5A A5 05 82 00 0D 01 2C

Para mas información al respecto consulte el UnicView AD Development Guide.

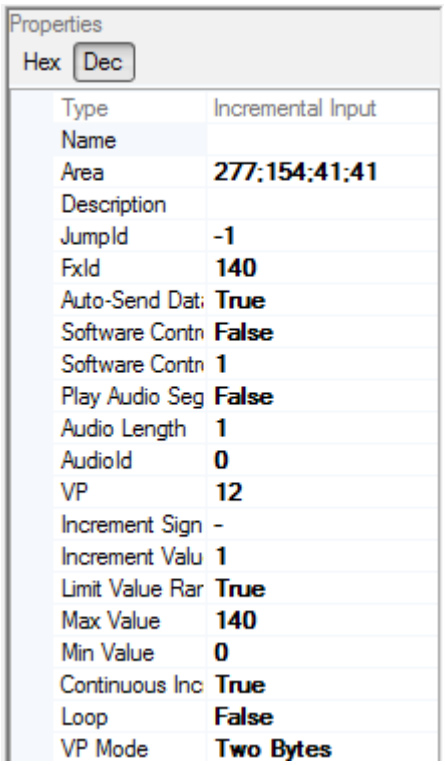
El resto de los controles o variables utilizado en esta pantalla no se explicaran, ya que fueron utilizados en otras pantallas y su implementación es muy similar.

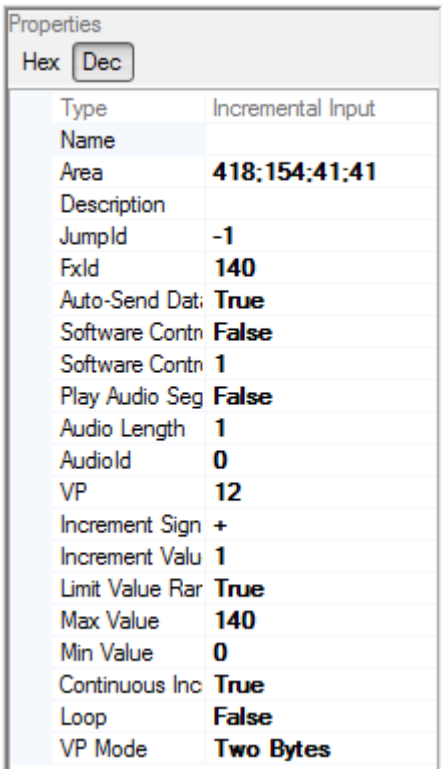
La distribución final de los distintos controles y variables y sus configuraciones son las siguientes.

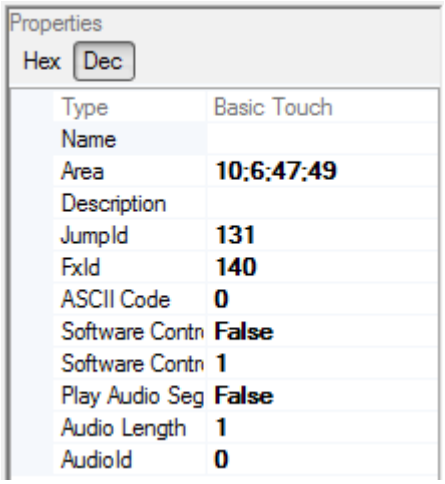
Properties

Hex | Dec

Type	Dynamic Icon
Name	
Area	**202;93;60;107**
Description	
VP	**13**
PP	**-1**
Library	**ThermLevel.ICO**
Transparency	**Opaque**
Icon Max	**14**
Icon Min	**0**
Value Max	**140**
Value Min	**0**
Initial Value	**0**


Properties
Hex
Dec
Type
Incremental Input
Name
Area
277;154;41;41
Description
JumpId
-1
FxId
140
Auto-Send Dat
True
Software Contr
False
Software Contr
1
Play Audio Seg
False
Audio Length
1
AudioId
0
VP
12
Increment Sign
-
Increment Valu
1
Limit Value Rar
True
Max Value
140
Min Value
0
Continuous Inc
True
Loop
False
VP Mode
Two Bytes


Properties
Hex
Dec
Type
Incremental Input
Name
Area
418;154;41;41
Description
JumpId
-1
FxId
140
Auto-Send Dat
True
Software Contr
False
Software Contr
1
Play Audio Seg
False
Audio Length
1
AudioId
0
VP
12
Increment Sign
+
Increment Valu
1
Limit Value Rar
True
Max Value
140
Min Value
0
Continuous Inc
True
Loop
False
VP Mode
Two Bytes



Esto concluye la configuración de todas las pantallas del proyecto. Resta compilar el proyecto mediante **Compile All,** cargarlo a una memoria SD para luego ser grabado en la memoria interna del display y probar su funcionamiento.