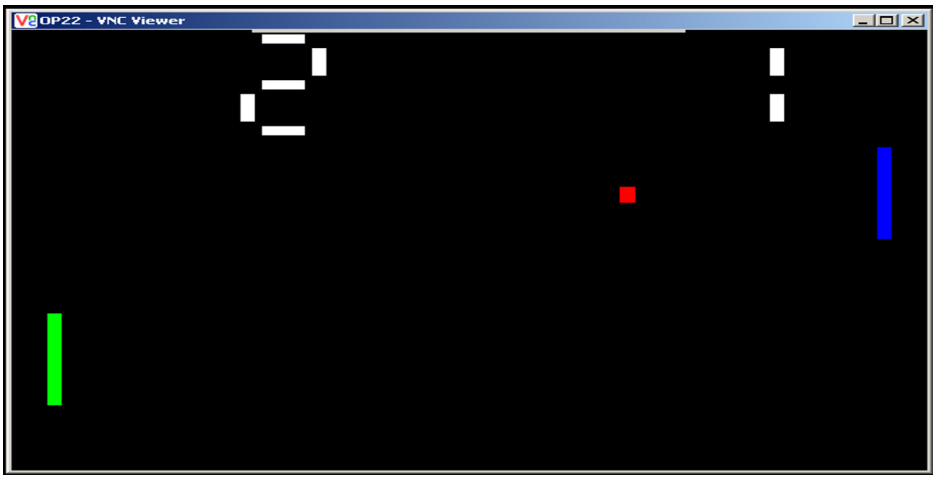
VIDEO JUEGO
EN PAC
USANDO PROTOCOLO RFB



PROTOCOLO RFB

El protocolo RFB o "Remote Frame Buffer" por sus siglas en Ingles, fue creado por por el Laboratorio de Investigacion de Olivetti (ORL) para manejo remoto de monitores. La idea era que el protocolo fuera lo mas simple posible para que cualquier cliente ligero, sin muchas prestaciones pudiera interpretarlo. Entre sus funciones mas simples se tiene las de dibujar rectangulos, especificando basicamente 5 parametros posicion x, posicion y, ancho , alto y color. De esta forma una imagen, se puede descomponer en rectangulos, enviar unicamente los 5 parametros de cada rectangulo, y no tener que transmitir pixel, por pixel la imagen del escritorio del servidor RFB al cliente ligero, ahorrando ancho de banda..

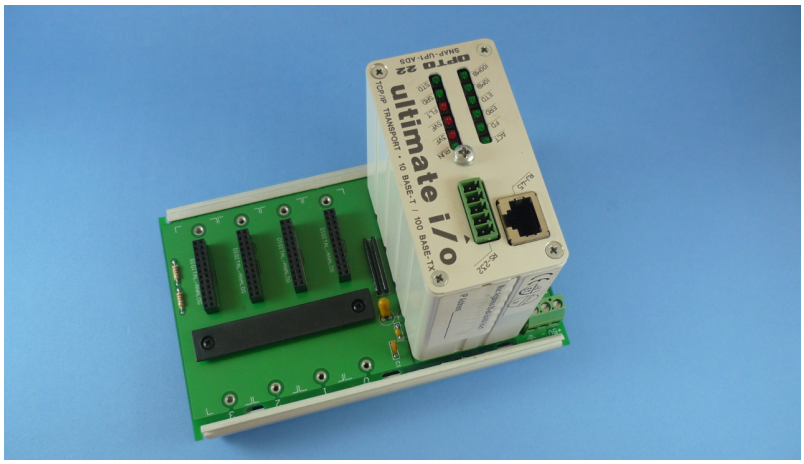
SERVIDOR RFB

La principal utilidad en la actualidad de un servidor RFB es tomar la imagen actual del escritorio (ventanas, menus, iconos,aplicaciones) y enviarla al cliente remoto, asi como capturar las acciones del cliente remoto (movimientos del raton, pulsacion de teclas) y enviarlas al escritorio. Sin embargo, puesto que el protocolo es independiente de cualquier sistema operativo se puede utilizar este para generar contenido en el cliente que no necesariamente existe de forma grafica en el servidor. Un ejemplo clasico de ello es un servidor de juegos, en donde en el servidor no hay pantalla, ni tarjeta grafica, ni controles, ni teclado, solamente procesamiento y un canal de comunicaciones, es en el lado del cliente (jugador) donde se encuentra la tarjeta grafica y monitor asi como teclado y controles.

IMPLEMENTACION EN PAC

Para desarrollar un servidor de aplicacion RFB es necesario contar con algun tipo de conexion de red, y algun medio para realizar programas que puedan enviar y recibir paquetes de red. Existen ejemplos de aplicaciones RFB en este caso un contador que puede ser compilado en cualquier computador con un compilador de C, hasta casos un poco mas extremos en donde un modulo ESP8266 es usado como servidor RFB.

Un PAC (en esta ocasion el SNAP UP1 ADS de Opto22) posee estas caracteristicas, Si no se posee un PAC fisico, se puede recurrir al simulador gratuito PAC Sim.

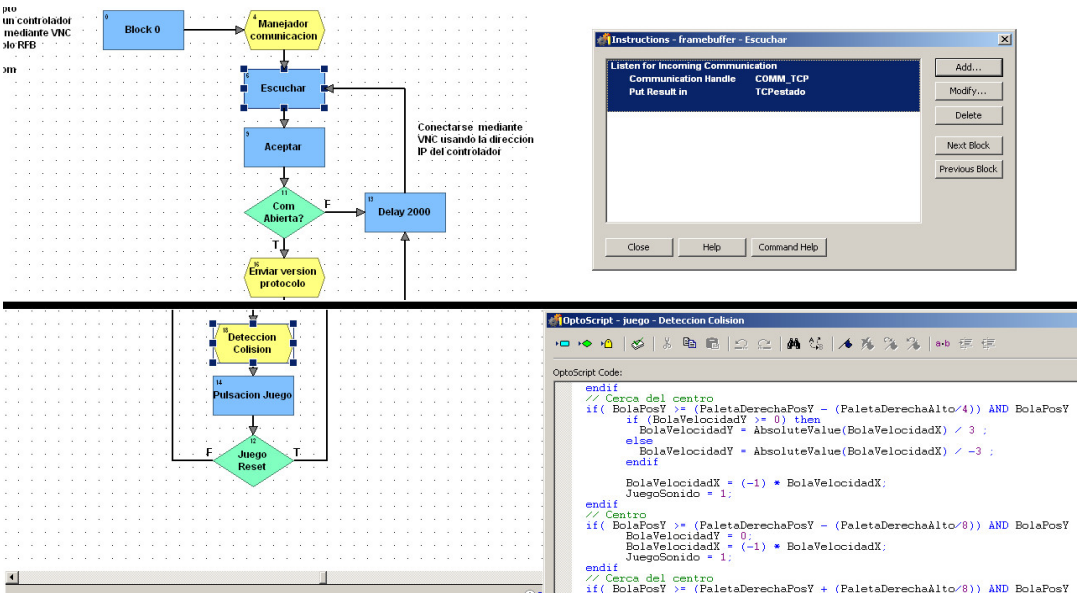


PROGRAMACION OPTOSCRIPT Y FLOWCHART

ELa programacion se realizo mediante la suite gratuita PAC Control para Windows. No existe aun una herramienta oficial para programar un PAC Opto22 mediante Linux, sin embargo, dado que la estrategia realizada de control que se envia al PAC es un archivo de texto cuyo contenido esta codificado en FORTH no deberia ser muy dificil desarrollar una alternativa. El esquema de programacion es mediante "flowcharts" o diagramas de flujo. Se pueden crear varios de estos diagramas y ser ejecutados "en paralelo", de esta forma se puede facilmente separar la logica del juego de la implementacion del protocolo RFB. Una de las grandes ventajas de los PAC versus PLC (aunque las diferencias cada vez son menores) esta en la capacidad de utilizar primitivas basicas de red, que pueden servir para construir protocolos que no vienen implementados de fabrica como el RFB, en este caso las primitivas para crear sockets tanto en modo cliente como en modo servidor.

La logica del juego, ubicacion de las paletas, movimiento de la bola, deteccion de colisiones, puntajes, son realizados por un "chart" o una tarea independiente, de forma tal que el juego podria ser accedido por otros medios y o protocolos como mediante entradas digitales, o usando un SCADA mediante modbus, etc. El juego es para un solo jugador remoto y existe un pequeño algoritmo "inteligente" que hace las veces de computadora oponente.

En la implementacion del protocolo RFB se utilizaron muchos "atajos", muchas respuestas y peticiones del cliente son ignoradas, se configuro una paleta de colores simple, ademas se uso un tipo de codificacion de rectangulos que no permite graficos muy avanzados.



CONCLUSIONES

- Es posbile implementar en un PAC Opto22 el protocolo RFB. Un posible uso para ello seria desarrollar aplicaciones menos "divertidas" como SCADA o HMI de acceso remoto mediante VNC, como una alternativa diferente para aplicaciones de Internet De Las Cosas.

- El juego solo permite la conexion de un unico cliente, sin embargo existe la posibilidad de ampliarlo para que dos o mas jugadores se puedan conectar remotamente.

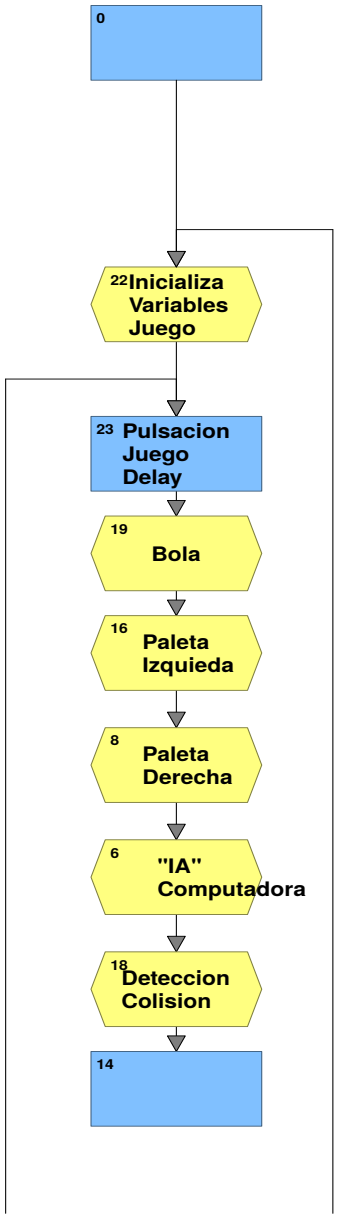
- Los juegos basados en rectangulos y de pantallas que en general no de desplazan ni cambian mucho (p.e Pong, Tetris, Breakout)son los mas facilmente implementables en RFB

- La estrategia funciona tanto el en PAC fisico como en el simulador PAC Sim

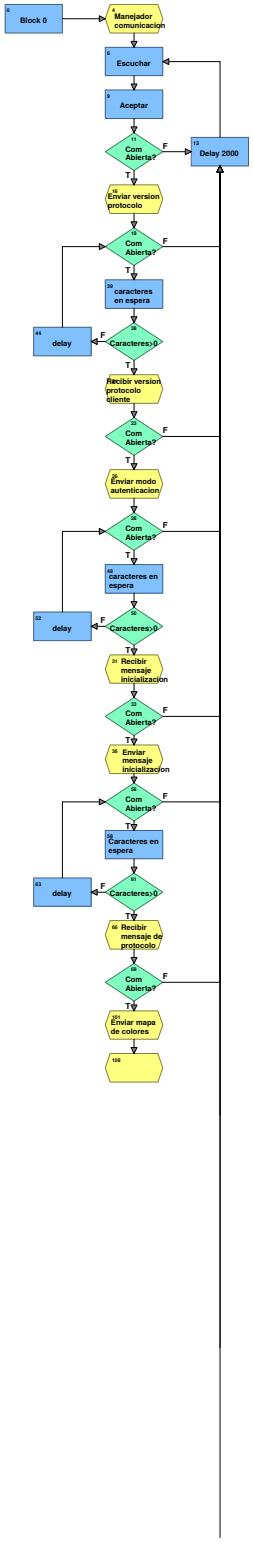
ENLACES

Video de prueba de conexion VNC con PAC y simulador: **https://youtu.be/MyuwvGJnlDw**
Cliente delgado en H/W para conexion remota: **http://s.click.aliexpress.com/e/Q3buZ3ZzN.**

CARTA JUEGO



CARTA FRAMEBUFFER



CODIGO CARTA JUEGO

TITLE: Chart Block Instructions
STRATEGY: RFB
CHART: juego
DATE: 03/29/16 TIME: 07:05:06

ACTIONS

Action Block: Block 0 (Id: 0)
Exit to: Inicializa Variables Juego (Id: 22)

There are no instructions in this action block.

Action Block: Pulsacion Juego (Id: 14)
Exit to: Juego Reset (Id: 12)

Move
From 1
To JuegoPulsacion

Action Block: Pulsacion Juego Delay (Id: 23)
Exit to: Bola (Id: 19)

Delay (mSec)
JuegoPulsacionDelay

SCRIPTS

OptoScript Block: "IA" Computadora (Id: 6)
Exit to: Deteccion Colision (Id: 18)

```
// No muy buena inteligencia artificial de la computadora!!!

if (BolaPosY > PaletaDerechaPosY) then
  PaletaDerechaTecla = 1;
endif

if (BolaPosY < PaletaDerechaPosY) then
  PaletaDerechaTecla = -1;
endif
```

OptoScript Block: Paleta Derecha (Id: 8)
Exit to: "IA" Computadora (Id: 6)

```
// Y posicion

if ( PaletaDerechaTecla > 0) then

  if ( PaletaDerechaPosY + PaletaDerechaVelocidadY + (PaletaDerechaAlto/2) > CampoAbajo) then
    PaletaDerechaPosY =  CampoAbajo - PaletaDerechaAlto/2;
  else
    PaletaDerechaPosY = PaletaDerechaPosY + PaletaDerechaVelocidadY;
  endif

  PaletaDerechaTecla = 0;

endif
```

```
 if ( PaletaDerechaTecla < 0) then

  if ( PaletaDerechaPosY - PaletaDerechaVelocidadY - (PaletaDerechaAlto/2) < CampoArriba) then
    PaletaDerechaPosY =  CampoArriba + PaletaDerechaAlto/2;
  else
    PaletaDerechaPosY = PaletaDerechaPosY - PaletaDerechaVelocidadY;
  endif

  PaletaDerechaTecla = 0;

 endif
```

OptoScript Block: Paleta Izquieda (Id: 16)
Exit to: Paleta Derecha (Id: 8)

```
 // Y posicion

 if ( PaletaIzquierdaTecla > 0) then

  if ( PaletaIzquierdaPosY + PaletaIzquierdaVelocidadY + (PaletaIzquierdaAlto/2) > CampoAbajo) then
    PaletaIzquierdaPosY =  CampoAbajo - PaletaIzquierdaAlto/2;
  else
    PaletaIzquierdaPosY = PaletaIzquierdaPosY + PaletaIzquierdaVelocidadY;
  endif

  PaletaIzquierdaTecla = 0;

 endif


 if ( PaletaIzquierdaTecla < 0) then

  if ( PaletaIzquierdaPosY - PaletaIzquierdaVelocidadY - (PaletaIzquierdaAlto/2) < CampoArriba) then
    PaletaIzquierdaPosY =  CampoArriba + PaletaIzquierdaAlto/2;
  else
    PaletaIzquierdaPosY = PaletaIzquierdaPosY - PaletaIzquierdaVelocidadY;
  endif

  PaletaIzquierdaTecla = 0;

 endif
```

OptoScript Block: Deteccion Colision (Id: 18)
Exit to: Pulsacion Juego (Id: 14)

```
 // Rebotes del campo

 /*
 if ( BolaVelocidadX > 0) then
  if ( BolaPosX + (BolaAncho/2) >= CampoDerecha) then
     BolaVelocidadX = (-1) * BolaVelocidadX;
     BolaPosX = BolaPosX -1;
  endif

 endif
```

```
if ( BolaVelocidadX < 0) then
  if ( BolaPosX - (BolaAncho/2) <= CampoIzquierda) then
     BolaVelocidadX = (-1) * BolaVelocidadX ;
     BolaPosX = BolaPosX +1;
  endif
endif
*/

if ( BolaVelocidadY > 0) then
  if ( BolaPosY + (BolaAlto/2) >= CampoAbajo) then
     BolaVelocidadY = (-1) * BolaVelocidadY ;
     BolaPosY = BolaPosY -1;
     JuegoSonido = 1;
  endif
endif

if ( BolaVelocidadY < 0) then
  if ( BolaPosY - (BolaAlto/2) <= CampoArriba) then
     BolaVelocidadY = (-1) * BolaVelocidadY ;
     BolaPosY = BolaPosY +1;
     JuegoSonido = 1;
  endif
endif



// Angulo de rebote de la pelota en la paleta  (BolaVelocidadY)

// Extremo           = 1.0 * BolaVelocidadX
// Cerca del extremo  = 0.7 * BolaVelocidadX
// Cerca del centro   = 0.3 * BolaVelocidadX
// Centro            = 0.0 * BolaVelocidadX
// Centro            = 0.0 * BolaVelocidadX
// Cerca del centro   = 0.3 * BolaVelocidadX
// Cerca del extremo  = 0.7 * BolaVelocidadX
// Extremo           = 1.0 * BolaVelocidadX



// Paleta izquierda X zona
if ( (BolaPosX - (BolaAncho /2) ) <= (PaletaIzquierdaPosX + (PaletaIzquierdaAncho/2))   and BolaVelocidadX < 1) then

  // extremo
  if( BolaPosY >= (PaletaIzquierdaPosY - (PaletaIzquierdaAlto /2) ) AND BolaPosY <= (PaletaIzquierdaPosY - ((PaletaIzquierdaAlto *
3) /8) ))then

     if (BolaVelocidadY >= 0) then
       BolaVelocidadY =  AbsoluteValue(BolaVelocidadX)  ;
     else
       BolaVelocidadY =  AbsoluteValue(BolaVelocidadX) * -1  ;
     endif

     BolaVelocidadX = (-1) * BolaVelocidadX;
     JuegoSonido = 1;
  endif

  // cerca del extremo
  if( BolaPosY >= (PaletaIzquierdaPosY - (PaletaIzquierdaAlto/4) - (PaletaIzquierdaAlto/8)) AND BolaPosY <= (PaletaIzquierdaPosY -
(PaletaIzquierdaAlto/4)  ))then
     if (BolaVelocidadY >= 0) then
       BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * 2 ) /3 ;
     else
       BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * -2 ) /3 ;
     endif


     BolaVelocidadX = (-1) * BolaVelocidadX;
     JuegoSonido = 1;
  endif
  // Cerca del centro
```

```
        if( BolaPosY >= (PaletaIzquierdaPosY - (PaletaIzquierdaAlto/4)) AND BolaPosY <= (PaletaIzquierdaPosY -
(PaletaIzquierdaAlto/8)))then
            if (BolaVelocidadY >= 0) then
              BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / 3 ;
            else
              BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / -3 ;
            endif

            BolaVelocidadX = (-1) * BolaVelocidadX;
            JuegoSonido = 1;
        endif
        // Centro
        if( BolaPosY >= (PaletaIzquierdaPosY - (PaletaIzquierdaAlto/8)) AND BolaPosY <= (PaletaIzquierdaPosY +
(PaletaIzquierdaAlto/8)))then
            BolaVelocidadY = 0;
            BolaVelocidadX = (-1) * BolaVelocidadX;
            JuegoSonido = 1;
        endif
        // Cerca del centro
        if( BolaPosY >= (PaletaIzquierdaPosY + (PaletaIzquierdaAlto/8)) AND BolaPosY <= (PaletaIzquierdaPosY +
(PaletaIzquierdaAlto/4)))then
            if (BolaVelocidadY >= 0) then
              BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / 3 ;
            else
              BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / -3 ;
            endif


            BolaVelocidadX = (-1) * BolaVelocidadX;
            JuegoSonido = 1;
        endif
        // Cerca del extremo
        if( BolaPosY >= (PaletaIzquierdaPosY + (PaletaIzquierdaAlto/4)) AND BolaPosY <= (PaletaIzquierdaPosY + (PaletaIzquierdaAlto/4) +
(PaletaIzquierdaAlto/8)))then

            if (BolaVelocidadY >= 0) then
              BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * 2 ) /3 ;
            else
              BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * -2 ) /3 ;
            endif

            BolaVelocidadX = (-1) * BolaVelocidadX;
            JuegoSonido = 1;
        endif
        // Extremo
        if( BolaPosY >= (PaletaIzquierdaPosY + ( (PaletaIzquierdaAlto * 3 ) /8)) AND BolaPosY <= (PaletaIzquierdaPosY +
(PaletaIzquierdaAlto/2) ))then

            if (BolaVelocidadY >= 0) then
              BolaVelocidadY =  AbsoluteValue(BolaVelocidadX)  ;
            else
              BolaVelocidadY =  AbsoluteValue(BolaVelocidadX) * -1  ;
            endif

            BolaVelocidadX = (-1) * BolaVelocidadX;
            JuegoSonido = 1;
        endif


      endif

      // Paleta derecha X zona
      if ( (BolaPosX + (BolaAncho/2) ) >= (PaletaDerechaPosX - (PaletaDerachaAncho/2)) and BolaVelocidadX > 1) then

        // Extremo
        if( BolaPosY >= (PaletaDerechaPosY - (PaletaDerechaAlto /2) ) AND BolaPosY <= (PaletaDerechaPosY - ((PaletaDerechaAlto * 3) /8)
))then

            if (BolaVelocidadY >= 0) then
              BolaVelocidadY =  AbsoluteValue(BolaVelocidadX)  ;
            else
```

```
            BolaVelocidadY =  AbsoluteValue(BolaVelocidadX) * -1  ;
          endif

          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif

        // Cerca del extremo
        if( BolaPosY >= (PaletaDerechaPosY - (PaletaDerechaAlto/4) - (PaletaDerechaAlto/8)) AND BolaPosY <= (PaletaDerechaPosY -
(PaletaDerechaAlto/4)  ))then
          if (BolaVelocidadY >= 0) then
            BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * 2 ) /3 ;
          else
            BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * -2 ) /3 ;
          endif


          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif
        // Cerca del centro
        if( BolaPosY >= (PaletaDerechaPosY - (PaletaDerechaAlto/4)) AND BolaPosY <= (PaletaDerechaPosY - (PaletaDerechaAlto/8)))then
          if (BolaVelocidadY >= 0) then
            BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / 3 ;
          else
            BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / -3 ;
          endif

          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif
        // Centro
        if( BolaPosY >= (PaletaDerechaPosY - (PaletaDerechaAlto/8)) AND BolaPosY <= (PaletaDerechaPosY + (PaletaDerechaAlto/8)))then
          BolaVelocidadY = 0;
          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif
        // Cerca del centro
        if( BolaPosY >= (PaletaDerechaPosY + (PaletaDerechaAlto/8)) AND BolaPosY <= (PaletaDerechaPosY + (PaletaDerechaAlto/4)))then
          if (BolaVelocidadY >= 0) then
            BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / 3 ;
          else
            BolaVelocidadY = AbsoluteValue(BolaVelocidadX) / -3 ;
          endif


          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif
        // Cerca del extremo
        if( BolaPosY >= (PaletaDerechaPosY + (PaletaDerechaAlto/4)) AND BolaPosY <= (PaletaDerechaPosY + (PaletaDerechaAlto/4) +
(PaletaDerechaAlto/8)))then

          if (BolaVelocidadY >= 0) then
            BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * 2 ) /3 ;
          else
            BolaVelocidadY = ( AbsoluteValue(BolaVelocidadX) * -2 ) /3 ;
          endif

          BolaVelocidadX = (-1) * BolaVelocidadX;
          JuegoSonido = 1;
        endif
        // Extremo
        if( BolaPosY >= (PaletaDerechaPosY + ( (PaletaDerechaAlto * 3 ) /8)) AND BolaPosY <= (PaletaDerechaPosY + (PaletaDerechaAlto/2)
))then

          if (BolaVelocidadY >= 0) then
            BolaVelocidadY =  AbsoluteValue(BolaVelocidadX)  ;
          else
            BolaVelocidadY =  AbsoluteValue(BolaVelocidadX) * -1  ;
          endif
```

```
        BolaVelocidadX = (-1) * BolaVelocidadX;
        JuegoSonido = 1;
    endif


  endif


  // Puntajes.

  // Paleta izquierda X zona
  if ( (BolaPosX - (BolaAncho /2) ) <= PaletaIzquierdaPosX   ) then
    PaletaDerechaPuntaje  = PaletaDerechaPuntaje + 1;
    BolaPosX = CampoIzquierda + ( (CampoDerecha - CampoIzquierda) / 2 ) ;
    BolaPosY = CampoArriba + ( (CampoAbajo - CampoArriba) / 2 ) ;
    BolaVelocidadX  = AbsoluteValue(BolaVelocidadX) * -1 ;
    BolaVelocidadY = 0 ;
    JuegoRedibujaPuntajes = 1 ;
    JuegoSonido = 1;
  endif


  // Paleta derecha X zona
  if ( ( BolaPosX + (BolaAncho/2) ) >= PaletaDerechaPosX  ) then
    PaletaIzquierdaPuntaje  = PaletaIzquierdaPuntaje + 1;
    BolaPosX = CampoIzquierda + ( (CampoDerecha - CampoIzquierda) / 2 ) ;
    BolaPosY = CampoArriba + ( (CampoAbajo - CampoArriba) / 2 ) ;
    BolaVelocidadX  = AbsoluteValue(BolaVelocidadX)  ;
    BolaVelocidadY = 0 ;
    JuegoRedibujaPuntajes = 1 ;
    JuegoSonido = 1;
  endif

  if(PaletaIzquierdaPuntaje > 9 or PaletaDerechaPuntaje > 9 )then
    PaletaIzquierdaPuntaje = 0 ;
    PaletaDerechaPuntaje = 0;
  endif

OptoScript Block: Bola (Id: 19)
Exit to: Paleta Izquieda (Id: 16)


  // X posicion
  if ( BolaVelocidadX > 0) then

    if ( BolaPosX + BolaVelocidadX + (BolaAncho/2) > CampoDerecha) then
      BolaPosX =  CampoDerecha - BolaAncho/2;
    else
      BolaPosX = BolaPosX + BolaVelocidadX;
    endif

  else

    if ( BolaPosX + BolaVelocidadX - (BolaAncho/2) < CampoIzquierda) then
      BolaPosX =  CampoIzquierda + BolaAncho/2;
    else
      BolaPosX = BolaPosX + BolaVelocidadX;
    endif

  endif

  // Y posicion

  if ( BolaVelocidadY > 0) then

    if ( BolaPosY + BolaVelocidadY + (BolaAlto/2) > CampoAbajo) then
      BolaPosY =  CampoAbajo - BolaAlto/2;
    else
      BolaPosY = BolaPosY + BolaVelocidadY;
    endif
```

```
  else

   if ( BolaPosY + BolaVelocidadY - (BolaAlto/2) < CampoArriba) then
     BolaPosY =  CampoArriba + BolaAlto/2;
   else
     BolaPosY = BolaPosY + BolaVelocidadY;
   endif

  endif
```

OptoScript Block: Inicializa Variables Juego (Id: 22)
Exit to: Pulsacion Juego Delay (Id: 23)

```
  JuegoReset = 0;
  JuegoPulsacionDelay = 100;

  CampoColor = 0;
  CampoArriba = 120;
  CampoAbajo = 480;
  CampoIzquierda = 0;
  CampoDerecha = 640;


  BolaColor = 1;
  BolaAlto = 11;
  BolaAncho = 11;


  PaletaIzquierdaColor = 2;
  PaletaIzquierdaAlto = 100;
  PaletaIzquierdaAncho = 10;
  PaletaIzquierdaPosX = 30;
  PaletaIzquierdaPosY = CampoArriba + ( (CampoAbajo - CampoArriba) / 2 ) ;
  PaletaIzquierdaVelocidadY = 7;

  PaletaDerechaColor = 3;
  PaletaDerechaAlto = 100;
  PaletaDerachaAncho = 10;
  PaletaDerechaPosX = 610;
  PaletaDerechaPosY = CampoArriba + ( (CampoAbajo - CampoArriba) / 2 ) ;
  PaletaDerechaVelocidadY = 7;


  BolaVelocidadX = 10;
  BolaVelocidadY = 0;

  BolaPosX = CampoIzquierda + ( (CampoDerecha - CampoIzquierda) / 2 ) ;
  BolaPosY = CampoArriba + ( (CampoAbajo - CampoArriba) / 2 ) ;



  Juego7SegColor = 4;
  Juego7SegAlto = 30;
  Juego7SegAncho = 10;

  JuegoIzquierda7SegAposX = 175;
  JuegoIzquierda7SegAposY = 5;

  JuegoIzquierda7SegBposX = 210;
  JuegoIzquierda7SegBposY = 20;

  JuegoIzquierda7SegCposX = 210;
  JuegoIzquierda7SegCposY = 70;
```

JuegoIzquierda7SegDposX = 175;
JuegoIzquierda7SegDposY = 105;

JuegoIzquierda7SegEposX = 160;
JuegoIzquierda7SegEposY = 70;

JuegoIzquierda7SegFposX = 160;
JuegoIzquierda7SegFposY = 20;

JuegoIzquierda7SegGposX = 175;
JuegoIzquierda7SegGposY = 55;

JuegoDerecha7SegAposX = JuegoIzquierda7SegAposX+320;
JuegoDerecha7SegAposY = JuegoIzquierda7SegAposY;

JuegoDerecha7SegBposX = JuegoIzquierda7SegBposX+320;
JuegoDerecha7SegBposY = JuegoIzquierda7SegBposY;

JuegoDerecha7SegCposX = JuegoIzquierda7SegCposX+320;
JuegoDerecha7SegCposY = JuegoIzquierda7SegCposY;

JuegoDerecha7SegDposX = JuegoIzquierda7SegDposX+320;
JuegoDerecha7SegDposY = JuegoIzquierda7SegDposY;

JuegoDerecha7SegEposX = JuegoIzquierda7SegEposX+320;
JuegoDerecha7SegEposY = JuegoIzquierda7SegEposY;

JuegoDerecha7SegFposX = JuegoIzquierda7SegFposX+320;
JuegoDerecha7SegFposY = JuegoIzquierda7SegFposY;

JuegoDerecha7SegGposX = JuegoIzquierda7SegGposX+320;
JuegoDerecha7SegGposY = JuegoIzquierda7SegGposY;

PaletaIzquierdaPuntaje = 0 ;
PaletaDerechaPuntaje = 0 ;

CONDITIONS

Condition Block: Juego Reset (Id: 12)
Operator Type: AND
TRUE Exit to: Inicializa Variables Juego (Id: 22)
FALSE Exit to: Pulsacion Juego Delay (Id: 23)

Is	JuegoReset
Equal?	
To	1

CONTINUE BLOCKS

There are no continue blocks in this flowchart.

CODIGO CARTA FRAMEBUFFER

TITLE: Chart Block Instructions
STRATEGY: RFB
CHART: framebuffer
DATE: 03/29/16 TIME: 07:06:18

ACTIONS

Action Block: Block 0 (Id: 0)
Exit to: Manejador comunicacion (Id: 4)

There are no instructions in this action block.

Action Block: Escuchar (Id: 6)
Exit to: Aceptar (Id: 9)

Listen for Incoming Communication
Communication HandleCOMM_TCP
Put Result in TCPestado

Action Block: Aceptar (Id: 9)
Exit to: Com Abierta? (Id: 11)

Accept Incoming Communication
Communication HandleCOMM_TCP
Put Result in TCPestado

Action Block: Delay 2000 (Id: 13)
Exit to: Escuchar (Id: 6)

Delay (mSec)
2000

Action Block: caracteres en espera (Id: 39)
Exit to: Caracteres>0 (Id: 38)

Get Number of Characters Waiting
Communication HandleCOMM_TCP
Put in caractersenespera

Action Block: delay (Id: 44)
Exit to: Com Abierta? (Id: 18)

Delay (mSec)
10

Action Block: caracteres en espera (Id: 48)
Exit to: Caracteres>0 (Id: 50)

Get Number of Characters Waiting
Communication HandleCOMM_TCP
Put in caractersenespera

Action Block: delay (Id: 52)

Exit to: Com Abierta? (Id: 28)

Delay (mSec)
10

Action Block: Caracteres en espera (Id: 58)
Exit to: Caracteres>0 (Id: 61)

Get Number of Characters Waiting
Communication Handle COMM_TCP
Put in caractersenespera

Action Block: delay (Id: 63)
Exit to: Com Abierta? (Id: 56)

Delay (mSec)
10

Action Block: Caracteres en espera (Id: 75)
Exit to: Com Abierta? (Id: 138)

Get Number of Characters Waiting
Communication Handle COMM_TCP
Put in caractersenespera

Action Block: delay 1 ms (Id: 111)
Exit to: Pulsacion Juego (Id: 156)

Delay (mSec)
1

Action Block: Juego Reset (Id: 170)
Exit to: Caracteres en espera (Id: 75)

Move
From 1
To JuegoReset

SCRIPTS

OptoScript Block: Manejador comunicacion (Id: 4)
Exit to: Escuchar (Id: 6)

```
// Inicializa el manejador de la comunicacion
SetCommunicationHandleValue("tcp:5900", COMM_TCP);
```

OptoScript Block: Enviar version protocolo (Id: 16)
Exit to: Com Abierta? (Id: 18)

```
TCPcadenaescritura = "";
TCPcadenaescritura = "RFB 003.003"+ chr(10);
TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Recibir version protocolo cliente (Id: 21)

Exit to: Com Abierta? (Id: 23)

```
TCPestado = ReceiveNChars(TCPcadenalectura, caractersenespera, COMM_TCP);
```

OptoScript Block: Enviar modo autenticacion (Id: 26)
Exit to: Com Abierta? (Id: 28)

```
TCPcadenaescritura = "";
for tmp0 =0 to 3 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next

TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 3);

TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Recibir mensaje inicializacion cliente (Id: 31)
Exit to: Com Abierta? (Id: 33)

```
TCPestado = ReceiveNChars(TCPcadenalectura, caractersenespera, COMM_TCP);
```

OptoScript Block: Enviar mensaje inicializacion (Id: 36)
Exit to: Com Abierta? (Id: 56)

```
TCPcadenaescritura = "";

TCPcadenaescritura = "";
for tmp0 =0 to 27 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next


// Framebuffer Ancho
// --------------------
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(128, TCPcadenaescritura, 1);

// Framebuffer Alto
// --------------------
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(224, TCPcadenaescritura, 3);

// Format pixel del servidor
// --------------------

// bits-por-pixel
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 4);
// profundidad
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 5);
// bandera big endian
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 6);
// bandera color verdadero
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 7);
// rojo max
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(7, TCPcadenaescritura, 9);
// verde max
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(7, TCPcadenaescritura, 11);
// azul max
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(3, TCPcadenaescritura, 13);
// desplazamiento del rojo
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
```

```
  // desplazamiento del verde
  TCPestado = SetNthCharacter(3, TCPcadenaescritura, 15);
  // desplazamiento del azul
  TCPestado = SetNthCharacter(6, TCPcadenaescritura, 16);
  // bits de relleno
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

  // Longitud del nombre
  // --------------------
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 20);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 21);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 22);
  TCPestado = SetNthCharacter(4, TCPcadenaescritura, 23);

  // cadena del nombre
  // --------------------
  TCPestado = SetNthCharacter(79, TCPcadenaescritura, 24);
  TCPestado = SetNthCharacter(80, TCPcadenaescritura, 25);
  TCPestado = SetNthCharacter(50, TCPcadenaescritura, 26);
  TCPestado = SetNthCharacter(50, TCPcadenaescritura, 27);



  TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);

OptoScript Block: Recibir mensaje de protocolo cliente (Id: 66)
Exit to: Com Abierta? (Id: 69)


  TCPestado = ReceiveNChars(TCPcadenalectura, caractersenespera, COMM_TCP);

OptoScript Block: Recibir peticiones cliente (Id: 78)
Exit to: Buscando Mensajes de Teclado (Id: 165)


  TCPestado = ReceiveNChars(TCPcadenalectura, caractersenespera, COMM_TCP);

OptoScript Block: Enviar mapa de colores (Id: 101)
Exit to: Pantalla negra inicial (Id: 108)


  TCPcadenaescritura = "";

  for tmp0 =0 to 35 step 1
    TCPcadenaescritura = TCPcadenaescritura + "X";
  next



  // Mapa de colores ( 5 colores Neg,Roj,Ver,Azu,Bla) facil y corto
  // ------------------------------------------------
  TCPestado = SetNthCharacter(1, TCPcadenaescritura, 0);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);

  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 3);

  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 4);
  TCPestado = SetNthCharacter(5, TCPcadenaescritura, 5);


  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 6);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 7);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 8);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 9);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 10);
  TCPestado = SetNthCharacter(0, TCPcadenaescritura, 11);
```

```
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 15);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);

TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 20);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 22);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 23);

TCPestado = SetNthCharacter(0, TCPcadenaescritura, 24);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 26);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 28);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 29);


TCPestado = SetNthCharacter(255, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 31);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 32);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(255, TCPcadenaescritura, 35);



TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Pantalla negra inicial (Id: 108)
Exit to: Com Abierta? (Id: 73)

```
TCPcadenaescritura = "";

for tmp0 =0 to 20 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next


// Actualizacion Framebuffer
// --------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 3);


// X
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 5);

// Y
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 7);


// Ancho
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(128, TCPcadenaescritura, 9);


// Alto
TCPestado = SetNthCharacter(1, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(224, TCPcadenaescritura, 11);
```

```
// Codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// datos del color del pixel
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 20);



TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Redibujar Bola & Paletas (Id: 155)
Exit to: sonido (Id: 185)

```
JuegoPulsacion = 0;

TCPcadenaescritura = "";

for tmp0 =0 to 105 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next


// La primera vez poner valores no indeterminados
if(zFBtmp1 == 0 )then
  zFBtmp1 = BolaPosX;
endif
if(zFBtmp2 == 0 )then
  zFBtmp2 = BolaPosY;
endif
if(zFBtmp3 == 0 )then
  zFBtmp3 = PaletaIzquierdaPosX;
endif
if(zFBtmp4 == 0 )then
  zFBtmp4 = PaletaIzquierdaPosY;
endif
if(zFBtmp5 == 0 )then
  zFBtmp5 = PaletaDerechaPosX;
endif
if(zFBtmp6 == 0 )then
  zFBtmp6 = PaletaDerechaPosY;
endif




// Actualizacion Framebuffer ( bola, paletas )
// -----------------------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(6, TCPcadenaescritura, 3);


// Borrando la BOLA anterior
```

```
// X
int8H = ( zFBtmp1 - (BolaAncho/2) ) >> 8;
int8L = 0x000000FF bitand ( zFBtmp1 - (BolaAncho/2) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 5);

// Y
int8H = ( zFBtmp2 - (BolaAlto/2) ) >> 8;
int8L = 0x000000FF bitand ( zFBtmp2 - (BolaAlto/2));
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 7);

// ancho
int8H = BolaAncho >> 8;
int8L = 0x000000FF bitand BolaAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 9);

// alto
int8H = BolaAlto >> 8;
int8L = 0x000000FF bitand BolaAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 11);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// datos de color de pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 20);


// Dibujando la BOLA actual

// X
int8H = ( BolaPosX - (BolaAncho/2) ) >> 8;
int8L = 0x000000FF bitand ( BolaPosX - (BolaAncho/2) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura,22);

// Y
int8H = ( BolaPosY - (BolaAlto/2) ) >> 8;
int8L = 0x000000FF bitand ( BolaPosY - (BolaAlto/2));
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 23);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 24);

// ancho
int8H = BolaAncho >> 8;
int8L = 0x000000FF bitand BolaAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 26);

// alto
int8H = BolaAlto >> 8;
int8L = 0x000000FF bitand BolaAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 28);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 29);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 31);
```

```
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 32);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 35);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 36);

// datos de color de pixel
TCPestado = SetNthCharacter(BolaColor, TCPcadenaescritura, 37);


// Borrando la PALETA izquierda anterior

// X
int8H = ( CampoIzquierda ) >> 8;
int8L = 0x000000FF bitand ( CampoIzquierda );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 38);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 39);

// Y
int8H = ( CampoArriba )  >> 8;
int8L = 0x000000FF bitand ( CampoArriba );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 40);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 41);

// ancho
int8H = ( ( PaletaIzquierdaPosX + (PaletaIzquierdaAncho/2 ) ) - CampoIzquierda )>> 8;
int8L = 0x000000FF bitand ( ( PaletaIzquierdaPosX + (PaletaIzquierdaAncho/2 ) ) - CampoIzquierda );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 42);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 43);

// alto
int8H = ( CampoAbajo - CampoArriba ) >> 8;
int8L = 0x000000FF bitand ( CampoAbajo - CampoArriba );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 44);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 45);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 46);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 47);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 48);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 49);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 50);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 51);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 52);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 53);

// datos de color de pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 54);


// dibujando la PALETA izquierda actual

// X
int8H = ( PaletaIzquierdaPosX - (PaletaIzquierdaAncho/2) ) >> 8;
int8L = 0x000000FF bitand ( PaletaIzquierdaPosX - (PaletaIzquierdaAncho/2) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 55);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura,56);

// Y
int8H = ( PaletaIzquierdaPosY - (PaletaIzquierdaAlto/2) ) >> 8;
int8L = 0x000000FF bitand ( PaletaIzquierdaPosY - (PaletaIzquierdaAlto/2));
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 57);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 58);

// ancho
```

```
int8H = PaletaIzquierdaAncho >> 8;
int8L = 0x000000FF bitand PaletaIzquierdaAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 59);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 60);

// alto
int8H = PaletaIzquierdaAlto >> 8;
int8L = 0x000000FF bitand PaletaIzquierdaAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 61);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 62);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 63);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 64);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 65);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 66);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 67);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 68);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 69);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 70);

// datos de color de pixel
TCPestado = SetNthCharacter(PaletaIzquierdaColor, TCPcadenaescritura, 71);


// Borrando la PALETA derecha anterior

// X
int8H = ( PaletaDerechaPosX - (PaletaDerachaAncho/2 )) >> 8;
int8L = 0x000000FF bitand ( PaletaDerechaPosX - (PaletaDerachaAncho/2 ) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 72);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 73);

// Y
int8H = ( CampoArriba )  >> 8;
int8L = 0x000000FF bitand ( CampoArriba );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 74);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 75);

// ancho
int8H = ( CampoDerecha - ( PaletaDerechaPosX - (PaletaDerachaAncho/2 ) ) ) >> 8;
int8L = 0x000000FF bitand ( CampoDerecha - ( PaletaDerechaPosX - (PaletaDerachaAncho/2 ) ) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 76);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 77);

// alto
int8H = ( CampoAbajo - CampoArriba ) >> 8;
int8L = 0x000000FF bitand ( CampoAbajo - CampoArriba );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 78);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 79);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 80);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 81);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 82);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 83);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 84);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 85);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 86);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 87);

// datos de color de pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 88);
```

```
// Dibujar PALETA derecha actual

// X
int8H = ( PaletaDerechaPosX - (PaletaDerachaAncho/2) ) >> 8;
int8L = 0x000000FF bitand ( PaletaDerechaPosX - (PaletaDerachaAncho/2) );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 89);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura,90);

// Y
int8H = ( PaletaDerechaPosY - (PaletaDerechaAlto/2) ) >> 8;
int8L = 0x000000FF bitand ( PaletaDerechaPosY - (PaletaDerechaAlto/2));
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 91);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 92);

// ancho
int8H = PaletaDerachaAncho >> 8;
int8L = 0x000000FF bitand PaletaDerachaAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 93);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 94);

// alto
int8H = PaletaDerechaAlto >> 8;
int8L = 0x000000FF bitand PaletaDerechaAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 95);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 96);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 97);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 98);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 99);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 100);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 101);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 102);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 103);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 104);

// datos de color de pixel
TCPestado = SetNthCharacter(PaletaDerechaColor, TCPcadenaescritura, 105);



// Transmitir por socket RFB
TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);


// Actualizando las posiciones anteriores
zFBtmp1 = BolaPosX;
zFBtmp2 = BolaPosY;
zFBtmp3 = PaletaIzquierdaPosX;
zFBtmp4 = PaletaIzquierdaPosY;
zFBtmp5 = PaletaDerechaPosX;
zFBtmp6 = PaletaDerechaPosY;
```

OptoScript Block: Buscando Mensajes de Teclado (Id: 165)
Exit to: Com Abierta? (Id: 85)

```
// Ignorando TODAS las peticiones del cliente excepto acciones del teclado

// tecla w presionada 0x04 0xZZ 0xYY 0xYY 0x00 0x00 0x00 0x77
// tecla s presionada 0x04 0xZZ 0xYY 0xYY 0x00 0x00 0x00 0x73

// 0xZZ NO cero!
// 0xYY bits de relleno


tmp1 = 0;
```

```
tmp4 = 1;


while ( tmp4 )

  tmp0 = FindCharacterInString(4, tmp1, TCPcadenalectura);

  if (tmp0 > -1 ) then

    tmp2 = GetNthCharacter(TCPcadenalectura, tmp0+1);

    if(tmp2 > 0) then

      tmp3 = GetNthCharacter(TCPcadenalectura, tmp0+7);

      if(tmp3 == 0x77) then
        PaletaIzquierdaTecla = -1;
      endif

      if(tmp3 == 0x73) then
        PaletaIzquierdaTecla = 1;
      endif

      if(tmp3 < 0) then
        tmp4 = 0;
      endif

    else
      tmp4 = 0;
    endif

  else
    tmp4 = 0;
  endif

  tmp1=tmp0+1;

wend
```

OptoScript Block: enviar campana (Id: 182)
Exit to: Puntajes (Id: 173)

```
  // Efecto de sonido 5.1 Canales 3D
  JuegoSonido = 0;

  TCPcadenaescritura = "";
  TCPcadenaescritura = "X";



  // Campana
  // --------------------
  TCPestado = SetNthCharacter(2, TCPcadenaescritura, 0);


  TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Redibujar puntajes derecha (Id: 196)
Exit to: Juego Reset (Id: 170)

```
  JuegoRedibujaPuntajes = 0;

  JuegoPulsacion = 0;
```

```
TCPcadenaescritura = "";

for tmp0 =0 to 139 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next

// mascara para display 7 segmentos a dibujar
// Bit 0 seg A,
// Bit 1 seg B
//
// Bit 7 seg G

if (PaletaDerechaPuntaje == 0) then
  JuegoDerecha7SegMascara = 0xB00111111;
endif
if (PaletaDerechaPuntaje == 1) then
  JuegoDerecha7SegMascara = 0xB00000110;
endif
if (PaletaDerechaPuntaje == 2) then
  JuegoDerecha7SegMascara = 0xB01011011;
endif
if (PaletaDerechaPuntaje == 3) then
  JuegoDerecha7SegMascara = 0xB01001111;
endif
if (PaletaDerechaPuntaje == 4) then
  JuegoDerecha7SegMascara = 0xB01100110;
endif
if (PaletaDerechaPuntaje == 5) then
  JuegoDerecha7SegMascara = 0xB01101101;
endif
if (PaletaDerechaPuntaje == 6) then
  JuegoDerecha7SegMascara = 0xB01111101;
endif
if (PaletaDerechaPuntaje == 7) then
  JuegoDerecha7SegMascara = 0xB00000111;
endif
if (PaletaDerechaPuntaje == 8) then
  JuegoDerecha7SegMascara = 0xB01111111;
endif
if (PaletaDerechaPuntaje == 9) then
  JuegoDerecha7SegMascara = 0xB01101111;
endif


// Actualizacion Framebuffer ( puntaje  )
// -------------------------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 3);

// Borrar el puntaje izquierdo

// X
int8H = 320  >> 8;
int8L = 0x000000FF bitand 320 ;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 5);

// Y
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 7);

// ancho
int8H = 320 >> 8;
int8L = 0x000000FF bitand 320;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 9);
```

```
// alto
int8H = CampoArriba >> 8;
int8L = 0x000000FF bitand CampoArriba;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 11);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// datos de color de pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 20);

// Derecha 7 Segmento A

// X
int8H = ( JuegoDerecha7SegAposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegAposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 22);

// Y
int8H = ( JuegoDerecha7SegAposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegAposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 23);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 24);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 26);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 28);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 29);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 31);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 32);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 35);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 36);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000001) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 37);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 37);
endif


// derecha 7 Segmento B
```

```
// X
int8H = ( JuegoDerecha7SegBposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegBposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 38);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 39);

// Y
int8H = ( JuegoDerecha7SegBposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegBposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 40);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 41);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 42);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 43);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 44);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 45);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 46);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 47);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 48);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 49);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 50);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 51);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 52);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 53);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000010) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 54);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 54);
endif


// derecha 7 Segmento C

// X
int8H = ( JuegoDerecha7SegCposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegCposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 55);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 56);

// Y
int8H = ( JuegoDerecha7SegCposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegCposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 57);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 58);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 59);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 60);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 61);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 62);
```

```
// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 63);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 64);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 65);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 66);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 67);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 68);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 69);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 70);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000100) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 71);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 71);
endif


// Derecha 7 Segmento D

// X
int8H = ( JuegoDerecha7SegDposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegDposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 72);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 73);

// Y
int8H = ( JuegoDerecha7SegDposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegDposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 74);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 75);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 76);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 77);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 78);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 79);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 80);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 81);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 82);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 83);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 84);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 85);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 86);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 87);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00001000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 88);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 88);
endif



// Derecha 7 Segmento E
```

```
// X
int8H = ( JuegoDerecha7SegEposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegEposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 89);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 90);

// Y
int8H = ( JuegoDerecha7SegEposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegEposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 91);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 92);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 93);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 94);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 95);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 96);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 97);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 98);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 99);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 100);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 101);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 102);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 103);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 104);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00010000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 105);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 105);
endif



// Derecha 7 Segmento F

// X
int8H = ( JuegoDerecha7SegFposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegFposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 106);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 107);

// Y
int8H = ( JuegoDerecha7SegFposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegFposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 108);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 109);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 110);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 111);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
```

```
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 112);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 113);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 114);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 115);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 116);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 117);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 118);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 119);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 120);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 121);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00100000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 122);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 122);
endif



// Derecho 7 Segmento G

// X
int8H = ( JuegoDerecha7SegGposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegGposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 123);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 124);

// Y
int8H = ( JuegoDerecha7SegGposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegGposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 125);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 126);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 127);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 128);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 129);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 130);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 131);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 132);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 133);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 134);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 135);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 136);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 137);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 138);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB01000000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 139);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 139);
endif
```

```
// Transmitir por socket RFB
TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Redibujar puntajes izquierda (Id: 197)
Exit to: Redibujar puntajes derecha (Id: 196)

```
JuegoRedibujaPuntajes = 0;

JuegoPulsacion = 0;

TCPcadenaescritura = "";

for tmp0 =0 to 139 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next


// mascara para display 7 segmentos a dibujar
// Bit 0 seg A,
// Bit 1 seg B
//
// Bit 7 seg G

if (PaletaIzquierdaPuntaje == 0) then
  JuegoIzquierda7SegMascara = 0xB00111111;
endif
if (PaletaIzquierdaPuntaje == 1) then
  JuegoIzquierda7SegMascara = 0xB00000110;
endif
if (PaletaIzquierdaPuntaje == 2) then
  JuegoIzquierda7SegMascara = 0xB01011011;
endif
if (PaletaIzquierdaPuntaje == 3) then
  JuegoIzquierda7SegMascara = 0xB01001111;
endif
if (PaletaIzquierdaPuntaje == 4) then
  JuegoIzquierda7SegMascara = 0xB01100110;
endif
if (PaletaIzquierdaPuntaje == 5) then
  JuegoIzquierda7SegMascara = 0xB01101101;
endif
if (PaletaIzquierdaPuntaje == 6) then
  JuegoIzquierda7SegMascara = 0xB01111101;
endif
if (PaletaIzquierdaPuntaje == 7) then
  JuegoIzquierda7SegMascara = 0xB00000111;
endif
if (PaletaIzquierdaPuntaje == 8) then
  JuegoIzquierda7SegMascara = 0xB01111111;
endif
if (PaletaIzquierdaPuntaje == 9) then
  JuegoIzquierda7SegMascara = 0xB01101111;
endif



// Actualizacion Framebuffer  ( puntaje  )
// -------------------------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 3);

// Borrar puntaje izquierdo
```

```
// X
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0 );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 5);

// Y
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 7);

// Ancho
int8H = 640 >> 8;
int8L = 0x000000FF bitand 320;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 9);

// Alto
int8H = CampoArriba >> 8;
int8L = 0x000000FF bitand CampoArriba;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 11);

// Codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// dato de color del pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 20);

// Izquierdo 7 Segmento A

// X
int8H = ( JuegoIzquierda7SegAposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegAposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 22);

// Y
int8H = ( JuegoIzquierda7SegAposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegAposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 23);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 24);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 26);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 28);

// Codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 29);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 31);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 32);
```

```
// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 35);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 36);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00000001) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 37);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 37);
endif


// Izquierda 7 Segmento B

// X
int8H = ( JuegoIzquierda7SegBposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegBposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 38);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 39);

// Y
int8H = ( JuegoIzquierda7SegBposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegBposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 40);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 41);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 42);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 43);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 44);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 45);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 46);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 47);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 48);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 49);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 50);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 51);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 52);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 53);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00000010) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 54);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 54);
endif


// Izquierdo 7 Segmento C

// X
int8H = ( JuegoIzquierda7SegCposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegCposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 55);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 56);
```

```
// Y
int8H = ( JuegoIzquierda7SegCposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegCposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 57);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 58);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 59);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 60);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 61);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 62);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 63);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 64);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 65);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 66);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 67);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 68);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 69);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 70);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00000100) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 71);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 71);
endif



// Izquierda 7 Segmento D

// X
int8H = ( JuegoIzquierda7SegDposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegDposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 72);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 73);

// Y
int8H = ( JuegoIzquierda7SegDposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegDposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 74);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 75);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 76);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 77);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 78);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 79);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 80);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 81);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 82);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 83);
```

```
// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 84);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 85);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 86);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 87);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00001000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 88);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 88);
endif


// Izquierda 7 Segmento E

// X
int8H = ( JuegoIzquierda7SegEposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegEposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 89);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 90);

// Y
int8H = ( JuegoIzquierda7SegEposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegEposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 91);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 92);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 93);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 94);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 95);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 96);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 97);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 98);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 99);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 100);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 101);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 102);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 103);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 104);

// dato de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00010000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 105);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 105);
endif


// izquierda 7 Segmento F

// X
int8H = ( JuegoIzquierda7SegFposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegFposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 106);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 107);
```

```
// Y
int8H = ( JuegoIzquierda7SegFposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegFposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 108);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 109);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 110);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 111);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 112);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 113);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 114);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 115);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 116);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 117);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 118);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 119);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 120);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 121);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00100000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 122);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 122);
endif



// Izquerda 7 Segmento G

// X
int8H = ( JuegoIzquierda7SegGposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegGposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 123);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 124);

// Y
int8H = ( JuegoIzquierda7SegGposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegGposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 125);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 126);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 127);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 128);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 129);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 130);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 131);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 132);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 133);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 134);
```

```
 // # subrectangulos
 TCPestado = SetNthCharacter(0, TCPcadenaescritura, 135);
 TCPestado = SetNthCharacter(0, TCPcadenaescritura, 136);
 TCPestado = SetNthCharacter(0, TCPcadenaescritura, 137);
 TCPestado = SetNthCharacter(0, TCPcadenaescritura, 138);

 // datos de color de pixel
 if (JuegoIzquierda7SegMascara bitand 0xB01000000) then
   TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 139);
 else
   TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 139);
 endif




 // Transmitir por socket RFB
 TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Redibujar puntajes izquierda (Id: 213)
Exit to: Redibujar puntajes derecha (Id: 216)

```
 JuegoRedibujaPuntajes = 0;

 JuegoPulsacion = 0;

 TCPcadenaescritura = "";

 for tmp0 =0 to 139 step 1
   TCPcadenaescritura = TCPcadenaescritura + "X";
 next


 // mascara para display 7 segmentos a dibujar
 // Bit 0 seg A,
 // Bit 1 seg B
 //
 // Bit 7 seg G

 if (PaletaIzquierdaPuntaje == 0) then
   JuegoIzquierda7SegMascara = 0xB00111111;
 endif
 if (PaletaIzquierdaPuntaje == 1) then
   JuegoIzquierda7SegMascara = 0xB00000110;
 endif
 if (PaletaIzquierdaPuntaje == 2) then
   JuegoIzquierda7SegMascara = 0xB01011011;
 endif
 if (PaletaIzquierdaPuntaje == 3) then
   JuegoIzquierda7SegMascara = 0xB01001111;
 endif
 if (PaletaIzquierdaPuntaje == 4) then
   JuegoIzquierda7SegMascara = 0xB01100110;
 endif
 if (PaletaIzquierdaPuntaje == 5) then
   JuegoIzquierda7SegMascara = 0xB01101101;
 endif
 if (PaletaIzquierdaPuntaje == 6) then
   JuegoIzquierda7SegMascara = 0xB01111101;
 endif
 if (PaletaIzquierdaPuntaje == 7) then
   JuegoIzquierda7SegMascara = 0xB00000111;
 endif
 if (PaletaIzquierdaPuntaje == 8) then
   JuegoIzquierda7SegMascara = 0xB01111111;
 endif
```

```
if (PaletaIzquierdaPuntaje == 9) then
  JuegoIzquierda7SegMascara = 0xB01101111;
endif



// Actualizacion Framebuffer  ( puntaje  )
// -------------------------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 3);

// Borrar puntaje izquierdo

// X
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0 );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 5);

// Y
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 7);

// Ancho
int8H = 640 >> 8;
int8L = 0x000000FF bitand 320;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 9);

// Alto
int8H = CampoArriba >> 8;
int8L = 0x000000FF bitand CampoArriba;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 11);

// Codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// dato de color del pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 20);

// Izquierdo 7 Segmento A

// X
int8H = ( JuegoIzquierda7SegAposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegAposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 22);

// Y
int8H = ( JuegoIzquierda7SegAposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegAposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 23);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 24);

// ancho
int8H = Juego7SegAlto >> 8;
```

```
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 26);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 28);

// Codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 29);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 31);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 32);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 35);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 36);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00000001) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 37);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 37);
endif


// Izquierda 7 Segmento B

// X
int8H = ( JuegoIzquierda7SegBposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegBposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 38);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 39);

// Y
int8H = ( JuegoIzquierda7SegBposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegBposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 40);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 41);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 42);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 43);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 44);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 45);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 46);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 47);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 48);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 49);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 50);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 51);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 52);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 53);

// datos de color de pixel
```

```
if (JuegoIzquierda7SegMascara bitand 0xB00000010) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 54);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 54);
endif


// Izquierdo 7 Segmento C

// X
int8H = ( JuegoIzquierda7SegCposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegCposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 55);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 56);

// Y
int8H = ( JuegoIzquierda7SegCposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegCposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 57);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 58);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 59);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 60);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 61);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 62);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 63);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 64);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 65);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 66);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 67);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 68);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 69);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 70);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00000100) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 71);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 71);
endif



// Izquierda 7 Segmento D

// X
int8H = ( JuegoIzquierda7SegDposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegDposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 72);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 73);

// Y
int8H = ( JuegoIzquierda7SegDposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegDposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 74);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 75);

// ancho
int8H = Juego7SegAlto >> 8;
```

```
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 76);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 77);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 78);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 79);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 80);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 81);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 82);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 83);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 84);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 85);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 86);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 87);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00001000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 88);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 88);
endif


// Izquierda 7 Segmento E

// X
int8H = ( JuegoIzquierda7SegEposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegEposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 89);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 90);

// Y
int8H = ( JuegoIzquierda7SegEposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegEposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 91);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 92);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 93);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 94);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 95);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 96);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 97);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 98);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 99);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 100);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 101);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 102);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 103);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 104);

// dato de color de pixel
```

```
if (JuegoIzquierda7SegMascara bitand 0xB00010000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 105);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 105);
endif


// izquierda 7 Segmento F

// X
int8H = ( JuegoIzquierda7SegFposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegFposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 106);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 107);

// Y
int8H = ( JuegoIzquierda7SegFposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegFposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 108);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 109);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 110);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 111);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 112);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 113);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 114);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 115);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 116);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 117);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 118);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 119);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 120);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 121);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB00100000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 122);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 122);
endif



// Izquerda 7 Segmento G

// X
int8H = ( JuegoIzquierda7SegGposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegGposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 123);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 124);

// Y
int8H = ( JuegoIzquierda7SegGposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoIzquierda7SegGposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 125);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 126);

// ancho
int8H = Juego7SegAlto >> 8;
```

```
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 127);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 128);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 129);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 130);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 131);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 132);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 133);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 134);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 135);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 136);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 137);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 138);

// datos de color de pixel
if (JuegoIzquierda7SegMascara bitand 0xB01000000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 139);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 139);
endif




// Transmitir por socket RFB
TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

OptoScript Block: Redibujar puntajes derecha (Id: 216)
Exit to: Caracteres en espera (Id: 75)

```
JuegoRedibujaPuntajes = 0;

JuegoPulsacion = 0;

TCPcadenaescritura = "";

for tmp0 =0 to 139 step 1
  TCPcadenaescritura = TCPcadenaescritura + "X";
next

// mascara para display 7 segmentos a dibujar
// Bit 0 seg A,
// Bit 1 seg B
//
// Bit 7 seg G

if (PaletaDerechaPuntaje == 0) then
  JuegoDerecha7SegMascara = 0xB00111111;
endif
if (PaletaDerechaPuntaje == 1) then
  JuegoDerecha7SegMascara = 0xB00000110;
endif
if (PaletaDerechaPuntaje == 2) then
  JuegoDerecha7SegMascara = 0xB01011011;
endif
if (PaletaDerechaPuntaje == 3) then
  JuegoDerecha7SegMascara = 0xB01001111;
endif
if (PaletaDerechaPuntaje == 4) then
```

```
  JuegoDerecha7SegMascara = 0xB01100110;
endif
if (PaletaDerechaPuntaje == 5) then
  JuegoDerecha7SegMascara = 0xB01101101;
endif
if (PaletaDerechaPuntaje == 6) then
  JuegoDerecha7SegMascara = 0xB01111101;
endif
if (PaletaDerechaPuntaje == 7) then
  JuegoDerecha7SegMascara = 0xB00000111;
endif
if (PaletaDerechaPuntaje == 8) then
  JuegoDerecha7SegMascara = 0xB01111111;
endif
if (PaletaDerechaPuntaje == 9) then
  JuegoDerecha7SegMascara = 0xB01101111;
endif


// Actualizacion Framebuffer ( puntaje  )
// -------------------------------------
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 0);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 1);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 2);
TCPestado = SetNthCharacter(8, TCPcadenaescritura, 3);

// Borrar el puntaje izquierdo

// X
int8H = 320  >> 8;
int8L = 0x000000FF bitand 320 ;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 4);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 5);

// Y
int8H = ( 0 ) ;
int8L = 0x000000FF bitand ( 0);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 6);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 7);

// ancho
int8H = 320 >> 8;
int8L = 0x000000FF bitand 320;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 8);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 9);

// alto
int8H = CampoArriba >> 8;
int8L = 0x000000FF bitand CampoArriba;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 10);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 11);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 12);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 13);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 14);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 15);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 16);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 17);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 18);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 19);

// datos de color de pixel
TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 20);

// Derecha 7 Segmento A

// X
```

```
int8H = ( JuegoDerecha7SegAposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegAposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 21);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 22);

// Y
int8H = ( JuegoDerecha7SegAposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegAposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 23);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 24);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 25);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 26);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 27);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 28);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 29);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 30);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 31);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 32);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 33);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 34);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 35);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 36);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000001) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 37);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 37);
endif


// derecha 7 Segmento B

// X
int8H = ( JuegoDerecha7SegBposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegBposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 38);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 39);

// Y
int8H = ( JuegoDerecha7SegBposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegBposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 40);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 41);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 42);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 43);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 44);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 45);

// codificacion
```

```
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 46);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 47);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 48);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 49);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 50);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 51);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 52);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 53);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000010) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 54);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 54);
endif


// derecha 7 Segmento C

// X
int8H = ( JuegoDerecha7SegCposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegCposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 55);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 56);

// Y
int8H = ( JuegoDerecha7SegCposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegCposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 57);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 58);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 59);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 60);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 61);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 62);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 63);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 64);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 65);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 66);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 67);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 68);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 69);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 70);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00000100) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 71);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 71);
endif


// Derecha 7 Segmento D

// X
int8H = ( JuegoDerecha7SegDposX ) >> 8;
```

```
int8L = 0x000000FF bitand ( JuegoDerecha7SegDposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 72);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 73);

// Y
int8H = ( JuegoDerecha7SegDposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegDposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 74);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 75);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 76);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 77);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 78);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 79);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 80);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 81);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 82);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 83);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 84);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 85);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 86);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 87);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00001000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 88);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 88);
endif



// Derecha 7 Segmento E

// X
int8H = ( JuegoDerecha7SegEposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegEposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 89);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 90);

// Y
int8H = ( JuegoDerecha7SegEposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegEposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 91);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 92);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 93);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 94);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 95);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 96);
```

```
// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 97);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 98);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 99);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 100);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 101);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 102);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 103);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 104);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00010000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 105);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 105);
endif



// Derecha 7 Segmento F

// X
int8H = ( JuegoDerecha7SegFposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegFposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 106);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 107);

// Y
int8H = ( JuegoDerecha7SegFposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegFposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 108);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 109);

// ancho
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 110);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 111);

// alto
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 112);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 113);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 114);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 115);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 116);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 117);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 118);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 119);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 120);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 121);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB00100000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 122);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 122);
endif
```

```
// Derecho 7 Segmento G

// X
int8H = ( JuegoDerecha7SegGposX ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegGposX );
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 123);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 124);

// Y
int8H = ( JuegoDerecha7SegGposY ) >> 8;
int8L = 0x000000FF bitand ( JuegoDerecha7SegGposY);
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 125);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 126);

// ancho
int8H = Juego7SegAlto >> 8;
int8L = 0x000000FF bitand Juego7SegAlto;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 127);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 128);

// alto
int8H = Juego7SegAncho >> 8;
int8L = 0x000000FF bitand Juego7SegAncho;
TCPestado = SetNthCharacter(int8H, TCPcadenaescritura, 129);
TCPestado = SetNthCharacter(int8L, TCPcadenaescritura, 130);

// codificacion
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 131);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 132);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 133);
TCPestado = SetNthCharacter(2, TCPcadenaescritura, 134);


// # subrectangulos
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 135);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 136);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 137);
TCPestado = SetNthCharacter(0, TCPcadenaescritura, 138);

// datos de color de pixel
if(JuegoDerecha7SegMascara bitand 0xB01000000) then
  TCPestado = SetNthCharacter(Juego7SegColor, TCPcadenaescritura, 139);
else
  TCPestado = SetNthCharacter(CampoColor, TCPcadenaescritura, 139);
endif




// Transmitir por socket RFB
TCPestado = TransmitString(TCPcadenaescritura, COMM_TCP);
```

CONDITIONS

Condition Block: Com Abierta? (Id: 11)
Operator Type: AND
TRUE Exit to: Enviar version protocolo (Id: 16)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Com Abierta? (Id: 18)
Operator Type: AND
TRUE Exit to: caracteres en espera (Id: 39)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Com Abierta? (Id: 23)
Operator Type: AND
TRUE Exit to: Enviar modo autenticacion (Id: 26)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Com Abierta? (Id: 28)
Operator Type: AND
TRUE Exit to: caracteres en espera (Id: 48)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Com Abierta? (Id: 33)
Operator Type: AND
TRUE Exit to: Enviar mensaje inicializacion (Id: 36)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Caracteres>0 (Id: 38)
Operator Type: AND
TRUE Exit to: Recibir version protocolo cliente (Id: 21)
FALSE Exit to: delay (Id: 44)

Is caractersenespera
Greater?
Than 0

Condition Block: Caracteres>0 (Id: 50)
Operator Type: AND
TRUE Exit to: Recibir mensaje inicializacion cliente (Id: 31)
FALSE Exit to: delay (Id: 52)

Is caractersenespera
Greater?
Than 0

Condition Block: Com Abierta? (Id: 56)
Operator Type: AND
TRUE Exit to: Caracteres en espera (Id: 58)
FALSE Exit to: Delay 2000 (Id: 13)

Communication HandleCOMM_TCP
Communication Open?

Condition Block: Caracteres>0 (Id: 61)
Operator Type: AND
TRUE Exit to: Recibir mensaje de protocolo cliente (Id: 66)

FALSE Exit to: delay (Id: 63)

Is Greater? Than | caractersenespera | 0

Condition Block: Com Abierta? (Id: 69)
Operator Type: AND
TRUE Exit to: Enviar mapa de colores (Id: 101)
FALSE Exit to: Delay 2000 (Id: 13)

Communication Open? | HandleCOMM_TCP

Condition Block: Com Abierta? (Id: 73)
Operator Type: AND
TRUE Exit to: Redibujar puntajes izquierda (Id: 197)
FALSE Exit to: Delay 2000 (Id: 13)

Communication Open? | HandleCOMM_TCP

Condition Block: Caracteres>0 (Id: 79)
Operator Type: AND
TRUE Exit to: Recibir peticiones cliente (Id: 78)
FALSE Exit to: delay 1 ms (Id: 111)

Is Greater? Than | caractersenespera | 0

Condition Block: Com Abierta? (Id: 85)
Operator Type: AND
TRUE Exit to: delay 1 ms (Id: 111)
FALSE Exit to: Delay 2000 (Id: 13)

Communication Open? | HandleCOMM_TCP

Condition Block: Com Abierta? (Id: 138)
Operator Type: AND
TRUE Exit to: Caracteres>0 (Id: 79)
FALSE Exit to: Delay 2000 (Id: 13)

Communication Open? | HandleCOMM_TCP

Condition Block: Pulsacion Juego (Id: 156)
Operator Type: AND
TRUE Exit to: Redibujar Bola & Paletas (Id: 155)
FALSE Exit to: Caracteres en espera (Id: 75)

Is Equal? To | JuegoPulsacion | 1

Condition Block: Puntajes (Id: 173)

Operator Type: AND
TRUE Exit to: Redibujar puntajes izquierda (Id: 213)
FALSE Exit to: Caracteres en espera (Id: 75)

Is Equal?	JuegoRedibujaPuntajes
To	1

Condition Block: sonido (Id: 185)
Operator Type: AND
TRUE Exit to: enviar campana (Id: 182)
FALSE Exit to: Puntajes (Id: 173)

Is Equal?	JuegoSonido
To	1

CONTINUE BLOCKS

There are no continue blocks in this flowchart.

VARIABLES

TITLE: Strategy Database
STRATEGY: RFB
DATE: 03/29/16 TIME: 07:08:58

NUMERIC VARIABLES

NAME	TYPE	INIT. VALUE	REF. COUNT
BolaAlto	INT	RUN	0 15
BolaAncho	INT	RUN	0 17
BolaColor	INT	RUN	0 2
BolaPosX	INT	RUN	0 19
BolaPosY	INT	RUN	0 51
BolaVelocidadX	INT	RUN	0 64
BolaVelocidadY	INT	RUN	0 52
CampoAbajo	INT	RUN	0 17
CampoArriba	INT	RUN	0 34
CampoColor	INT	RUN	0 36
CampoDerecha	INT	RUN	0 8
CampoIzquierda	INT	RUN	0 13
caractersenespera	INT	RUN	0 12
estadocartas	INT	RUN	0 2
int8H	INT	RUN	0 304
int8L	INT	RUN	0 304
Juego7SegAlto	INT	RUN	0 57
Juego7SegAncho	INT	RUN	0 57
Juego7SegColor	INT	RUN	0 29
JuegoDerecha7SegAposX	INT	RUN	0 5
JuegoDerecha7SegAposY	INT	RUN	0 5
JuegoDerecha7SegBposX	INT	RUN	0 5
JuegoDerecha7SegBposY	INT	RUN	0 5
JuegoDerecha7SegCposX	INT	RUN	0 5
JuegoDerecha7SegCposY	INT	RUN	0 5
JuegoDerecha7SegDposX	INT	RUN	0 5
JuegoDerecha7SegDposY	INT	RUN	0 5
JuegoDerecha7SegEposX	INT	RUN	0 5
JuegoDerecha7SegEposY	INT	RUN	0 5
JuegoDerecha7SegFposX	INT	RUN	0 5
JuegoDerecha7SegFposY	INT	RUN	0 5
JuegoDerecha7SegGposX	INT	RUN	0 5
JuegoDerecha7SegGposY	INT	RUN	0 5
JuegoDerecha7SegMascara	INT	RUN	0 34
JuegoIzquierda7SegAposX	INT	RUN	0 6
JuegoIzquierda7SegAposY	INT	RUN	0 6
JuegoIzquierda7SegBposX	INT	RUN	0 6
JuegoIzquierda7SegBposY	INT	RUN	0 6
JuegoIzquierda7SegCposX	INT	RUN	0 6
JuegoIzquierda7SegCposY	INT	RUN	0 6
JuegoIzquierda7SegDposX	INT	RUN	0 6
JuegoIzquierda7SegDposY	INT	RUN	0 6
JuegoIzquierda7SegEposX	INT	RUN	0 6
JuegoIzquierda7SegEposY	INT	RUN	0 6
JuegoIzquierda7SegFposX	INT	RUN	0 6
JuegoIzquierda7SegFposY	INT	RUN	0 6
JuegoIzquierda7SegGposX	INT	RUN	0 6
JuegoIzquierda7SegGposY	INT	RUN	0 6
JuegoIzquierda7SegMascara	INT	RUN	0 34
JuegoPulsacion	INT	RUN	0 7
JuegoPulsacionDelay	INT	RUN	0 2
JuegoRedibujaPuntajes	INT	RUN	0 7
JuegoReset	INT	RUN	0 3
JuegoSonido	INT	RUN	0 20
PaletaDerachaAncho	INT	RUN	0 10
PaletaDerechaAlto	INT	RUN	0 25
PaletaDerechaColor	INT	RUN	0 2

PaletaDerechaPosX	INT	RUN	0	11
PaletaDerechaPosY	INT	RUN	0	29
PaletaDerechaPuntaje	INT	RUN	0	25
PaletaDerechaTecla	INT	RUN	0	6
PaletaDerechaVelocidadY	INT	RUN	0	5
PaletaIzquierdaAlto	INT	RUN	0	25
PaletaIzquierdaAncho	INT	RUN	0	8
PaletaIzquierdaColor	INT	RUN	0	2
PaletaIzquierdaPosX	INT	RUN	0	9
PaletaIzquierdaPosY	INT	RUN	0	27
PaletaIzquierdaPuntaje	INT	RUN	0	25
PaletaIzquierdaTecla	INT	RUN	0	6
PaletaIzquierdaVelocidadY	INT	RUN	0	5
TCPestado	INT	RUN	0	801
tmp0	INT	RUN	0	14
tmp1	INT	RUN	0	3
tmp2	INT	RUN	0	2
tmp3	INT	RUN	0	4
tmp4	INT	RUN	0	5
zFBtmp1	INT	RUN	0	5
zFBtmp2	INT	RUN	0	5
zFBtmp3	INT	RUN	0	3
zFBtmp4	INT	RUN	0	3
zFBtmp5	INT	RUN	0	3
zFBtmp6	INT	RUN	0	3

STRING VARIABLES

NAME	INIT.	WIDTH	REF. COUNT	VALUE
TCPcadenaescritura	RUN	512	827	
TCPcadenalectura	RUN	512	7	

COMMUNICATION HANDLES

NAME	REF. COUNT	INIT.	VALUE
COMM_TCP	32	RUN	