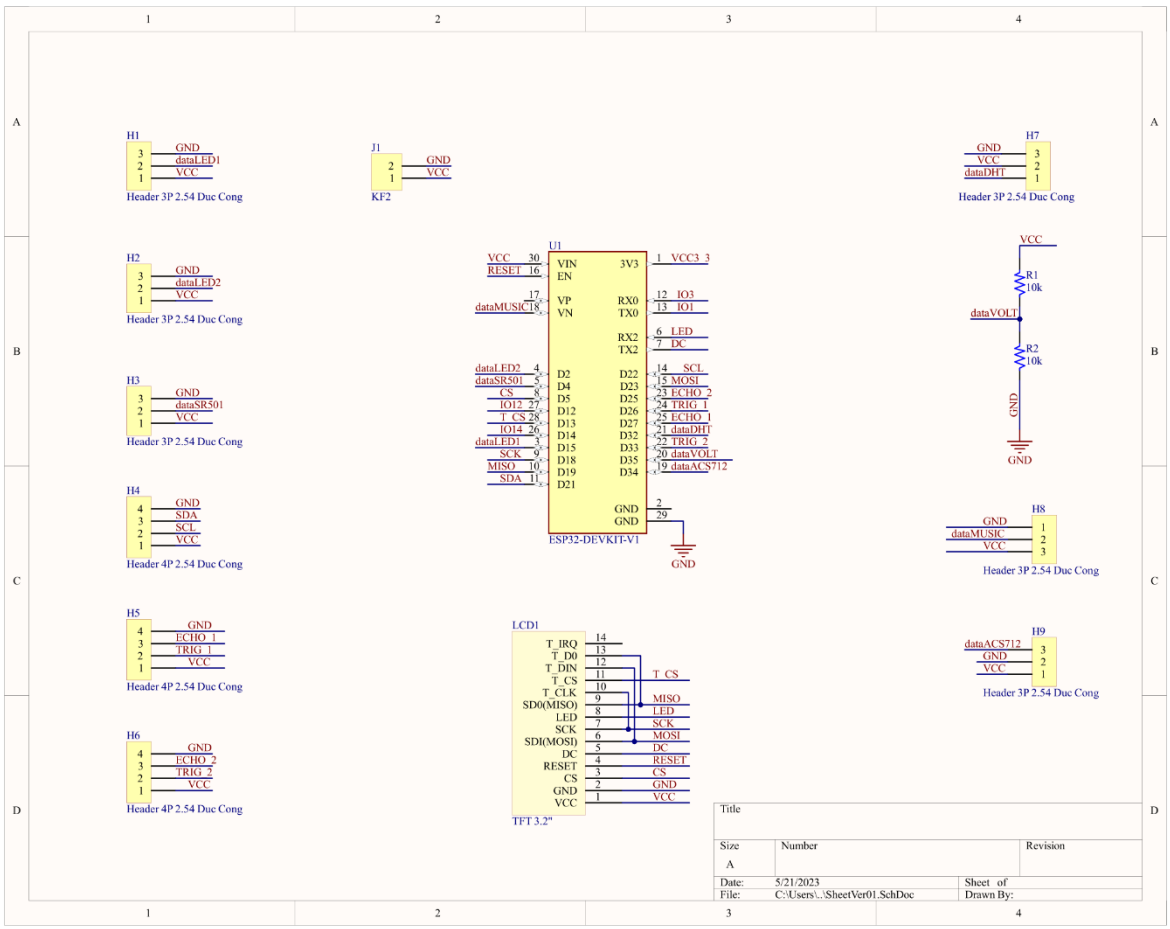
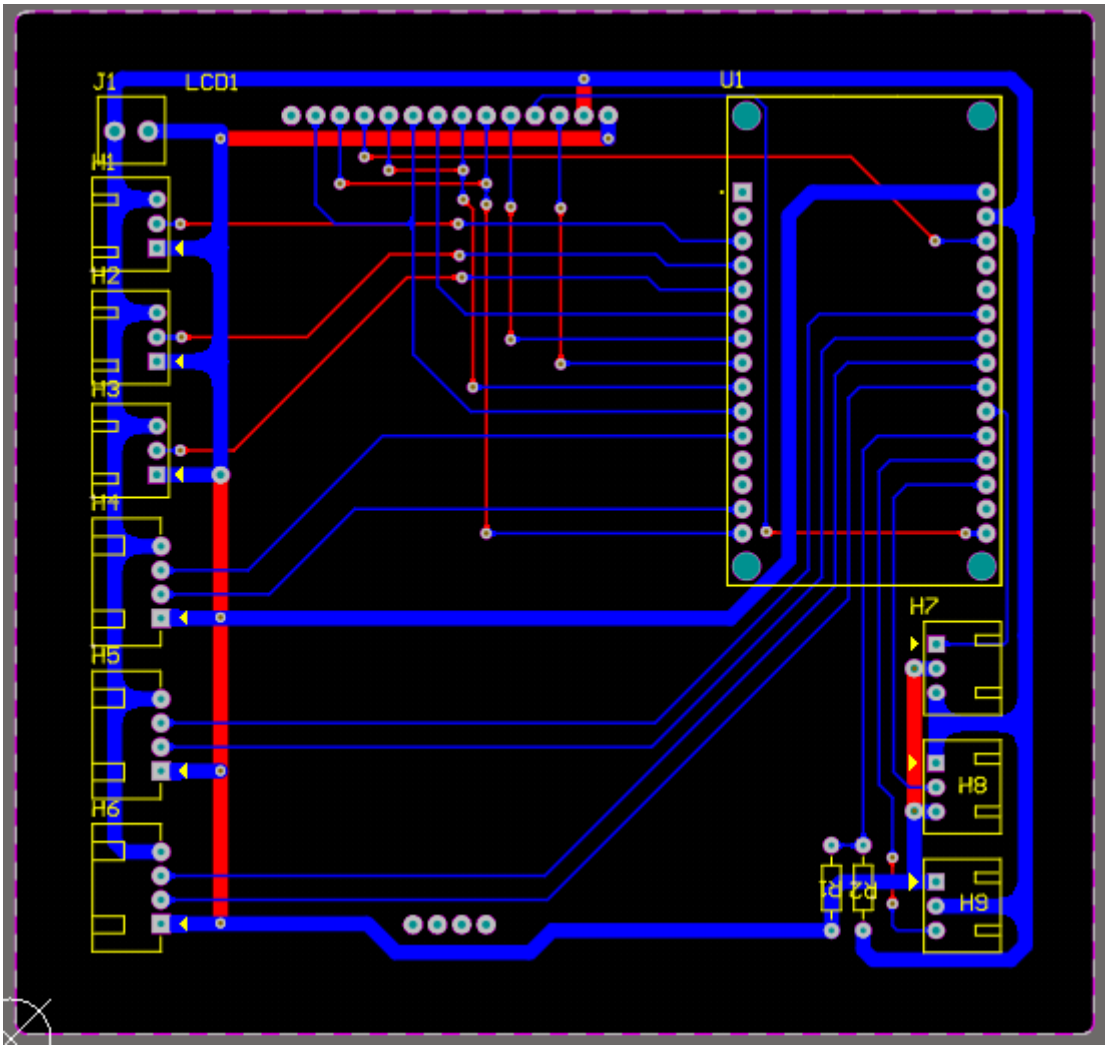

H1
GND
dataLED1
VCC
Header 3P 2.54 Duc Cong
J1
GND
VCC
KF2
H7
GND
VCC
dataDHT
Header 3P 2.54 Duc Cong
H2
GND
dataLED2
VCC
Header 3P 2.54 Duc Cong
U1
VCC 30
RESET 16
17
dataMUSIC 18
dataLED2 4
dataSR501 5
CS 8
IO12 27
T_CS 28
IO14 26
dataLED1 3
SCK 9
MISO 10
SDA 11
VIN
EN
VP
VN
D2
D4
D5
D12
D13
D14
D15
D18
D19
D21
3V3
RX0
TX0
RX2
TX2
D22
D23
D25
D26
D27
D32
D33
D35
D34
GND
GND
1 VCC3 3
12 IO3
13 IO1
6 LED
7 DC
14 SCL
15 MOSI
23 ECHO_2
24 TRIG_1
25 ECHO_1
21 dataDHT
22 TRIG_2
20 dataVOLT
19 dataACS712
2
29
ESP32-DEVKIT-V1
GND
VCC
R1
10k
dataVOLT
R2
10k
GND
GND
H3
GND
dataSR501
VCC
Header 3P 2.54 Duc Cong
H4
GND
SDA
SCL
VCC
Header 4P 2.54 Duc Cong
H8
GND
dataMUSIC
VCC
Header 3P 2.54 Duc Cong
H5
GND
ECHO_1
TRIG_1
VCC
Header 4P 2.54 Duc Cong
LCD1
T_IRQ 14
T_D0 13
T_DIN 12
T_CS 11
T_CLK 10
SD0(MISO) 9
LED 8
SCK 7
SDI(MOSI) 6
DC 5
RESET 4
CS 3
GND 2
VCC 1
T_CS
MISO
LED
SCK
MOSI
DC
RESET
CS
GND
VCC
TFT 3.2"
H9
dataACS712
GND
VCC
Header 3P 2.54 Duc Cong
H6
GND
ECHO_2
TRIG_2
VCC
Header 4P 2.54 Duc Cong
Title
Size
A
Number
Revision
Date: 5/21/2023
Sheet of
File: C:\Users\..\SheetVer01.SchDoc
Drawn By:


J1
LCD1
U1
H1
H2
H3
H4
H5
H6
H7
H8
H9
R1
R2

SOURCE CODE:

```
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>
//----------------------------------------Link Blynk
#define BLYNK_TEMPLATE_ID "TMPL65cD1TjN_"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "GtNTzrn9yhOlqLVk8Nnpz0AIAsky5VkD"
//----------------------------------------Include Library
#include "SPI.h"
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <string.h>
#include "DHT.h"
#include <BH1750.h>
#include <Wire.h>
#include <FastLED.h>
#include "bitmap.h"
//----------------------------------------DHT define
#define DHTPIN 32
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
//----------------------------------------Include pin
#define CS_PIN  13
XPT2046_Touchscreen ts(CS_PIN);
//----------------------------------------Wifi and Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Minh Triet .";
char pass[] = "0898907304";
//----------------------------------------LED define
#define LED_PIN     15
#define LED_PIN_1     2
#define NUM_LEDS    72
CRGB leds_1[NUM_LEDS];
CRGB leds[NUM_LEDS];
//----------------------------------------BH1750 define
BH1750 lightMeter;
//----------------------------------------Pin configuration and initialization
for LCD TFT
#define TFT_DC 17
#define TFT_CS  5
#define led_lcd 16
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
//----------------------------------------Cam bien sieu am define
```

```
#define trig1  26
#define echo1  27
#define trig2  33
#define echo2  25
//---------------------------------------SR501 define
#define dataSR501  4
//---------------------------------------Cam bien sieu am value
int numHuman;
String queue = "";
int timeoutcounter=0;
int distanse1;
int distanse2;
//---------------------------------------read acs712 (ampe)
#define analogAmpe  34
float amps;
//---------------------------------------read (volt)
#define analogVolt 35
float volt;
//---------------------------------------Defines colors
// Assign human-readable names to some common 16-bit color values:
#define BLACK       0x0000
#define BLUE        0x001F
#define RED         0xF800
#define GREEN       0x07E0
#define CYAN        0x07FF
#define MAGENTA     0xF81F
#define YELLOW      0xFFE0
#define WHITE       0xFFFF
#define ORANGE      0xFD20
#define DARKORANGE  0xFB60
#define MAROON      0x7800
#define BLACKM      0x18E3
////////////////////////////////////
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000
//---------------------------------------Variable for detecting touch screen
when touched
#define MINPRESSURE 10
#define MAXPRESSURE 1000
//---------------------------------------Value sample
// int lux = 250;
int mode = 0;                    //Start sample value different Normal
int modeBtnOld = 0;              //Start sample value different Normal
int modeBtn = 1;                 //Start sample value different Normal
//Value for dht
int Humidity;
```

```
float Temperature;
float Fahreheit;
float hif;
float hic;
//Value for mode
int brightnessRoom = 250;
int brightnessRoomOld = 0;
int brightnessLed=30;
//---------------------------------------The x and y points for the
Temperature bar
int x_bar_t = 20;
int y_bar_t = 60;
//---------------------------------------The variable to hold the conversion
value from the temperature value to the value for the temperature bar
int T_to_Bar;
//---------------------------------------Menu = 0 to display the Main Menu
Display, Menu = 1 to control the LED and Menu = 2 to display DHT11 sensor data
int Menu = 0;
//---------------------------------------Variable for the x, y and z points
on the touch screen
int TSPointZ;
int x_set_rotatoon_135;
int y_set_rotatoon_135;
//---------------------------------------Millis variable to update the
temperature and humidity values
unsigned long previousMillis = 0; //--> will store last time updated
unsigned long previousMillis_Power = 0;
unsigned long previousMillis_WriteDHT = 0;
// constants won't change:
const long interval = 2000; //--> interval
//=======================================Connect to wifi
bool initWiFi()
{
  tft.fillScreen(WHITE);
  tft.drawRGBBitmap(60, 100, loading, 190, 29);
  int col[8];
  col[0] = tft.color565(35, 35, 35);
  col[1] = tft.color565(53, 53, 53);
  col[2] = tft.color565(71, 71, 71);
  col[3] = tft.color565(89, 89, 89);
  col[4] = tft.color565(107, 107, 107);
  col[5] = tft.color565(124, 124, 124);
  col[6] = tft.color565(142, 142, 142);
  col[7] = tft.color565(160, 160, 160);
  WiFi.begin(ssid, pass);
  Serial.println("Connecting to WiFi...");
  int dem = 0;
  while (WiFi.status() != WL_CONNECTED)
```

```
  {
    //create begin screen
    for (int i = 8; i > 0; i--)
    {
      tft.fillCircle(150 + 15 * (cos(-(i + 0) * PI / 4)), 160 + 15 * (sin(-(i
+ 0) * PI / 4)), 3, col[0]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 1) * PI / 4)), 160 + 15 * (sin(-(i
+ 1) * PI / 4)), 3, col[1]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 2) * PI / 4)), 160 + 15 * (sin(-(i
+ 2) * PI / 4)), 3, col[2]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 3) * PI / 4)), 160 + 15 * (sin(-(i
+ 3) * PI / 4)), 3, col[3]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 4) * PI / 4)), 160 + 15 * (sin(-(i
+ 4) * PI / 4)), 3, col[4]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 5) * PI / 4)), 160 + 15 * (sin(-(i
+ 5) * PI / 4)), 3, col[5]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 6) * PI / 4)), 160 + 15 * (sin(-(i
+ 6) * PI / 4)), 3, col[6]);
      delay(10);
      tft.fillCircle(150 + 15 * (cos(-(i + 7) * PI / 4)), 160 + 15 * (sin(-(i
+ 7) * PI / 4)), 3, col[7]);
      delay(10);
    }
    if (dem > 50)
      break;
    Serial.print(".");
    dem = dem + 1;
  }
  Blynk.begin(auth, ssid, pass);
  tft.fillScreen(BLACK);
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  return true;
}
//=======================================VOID SETUP()
void setup() {
  Serial.begin(115200);
```

```
  pinMode(led_lcd, OUTPUT);
  pinMode(trig1, OUTPUT);
  pinMode(echo1, INPUT);
  pinMode(trig2, OUTPUT);
  pinMode(echo2, INPUT);
  pinMode(dataSR501, INPUT_PULLUP);
  digitalWrite(led_lcd, HIGH);

  dht.begin();
  Wire.begin();
  lightMeter.begin();
  FastLED.addLeds<WS2812, LED_PIN, RGB>(leds, NUM_LEDS);
  FastLED.addLeds<WS2812, LED_PIN_1, RGB>(leds_1, NUM_LEDS);

  tft.begin();
  tft.setRotation(3);
  ts.begin();
  ts.setRotation(0);


  initWiFi();

  Menu_display();
}
//================================================================================READ
BLYNK()
//button brightness room
BLYNK_WRITE(V4)
{
  brightnessRoom = param.asInt();
}
//button mode normal
BLYNK_WRITE(V5)
{
  modeBtn = 1;
}
//button mode read
BLYNK_WRITE(V6)
{
  modeBtn = 2;
}
//button mode sleep
BLYNK_WRITE(V7)
{
  modeBtn = 3;
}
//================================================================================VOID
LOOP()
```

```
void loop() {
  // put your main code here, to run repeatedly:
  Blynk.run();

  readNumHuman();
  WriteDHT11Data();
  //---------------------------------------Main Menu Display
  if (Menu == 0) {
    DrawTempAmountOfPeople(numHuman);
    ShowPower();
    GetTSPoint();
    //---------------------------------------Conditions for detecting touch screen when touched
    if (TSPointZ > MINPRESSURE && TSPointZ < MAXPRESSURE) {

      //---------------------------------------Conditions for detecting when the Button for controlling the LED is touched and its command (Enter the LED controlling menu)
      if (x_set_rotatoon_135 > 17 && x_set_rotatoon_135 < (17+280) && y_set_rotatoon_135 > 90 && y_set_rotatoon_135 < (90+40))
      {
        Menu = 1;
        DrawButtonControlLEDPress();
        delay(100);
        DrawButtonControlLED();
        delay(100);
        tft.fillScreen(BLACK);
        delay(10);
        DrawTempLux(brightnessRoom);
        DrawButtonTempTru();
        DrawButtonTempCong();
        DrawButtonTempNormal();
        DrawButtonTempRead();
        DrawButtonTempSleep();
        DrawMode();
        DrawButtonBack(10, 200);
      }
      //---------------------------------------

      //---------------------------------------Condition to detect when the button to display DHT11 sensor data is touched and the command (Enter the menu displays DHT11 sensor data)
      if (x_set_rotatoon_135 > 17 && x_set_rotatoon_135 < (17+280) && y_set_rotatoon_135 > 160 && y_set_rotatoon_135 < (160+40))
      {
        Menu = 2;
        DrawButtonTempHumPress();
        delay(100);
```

```
        DrawButtonTempHum();
        delay(100);
        tft.fillScreen(BLACK);
        delay(10);
        tft.drawLine(15, 40, 300, 40, MAROON);
        tft.drawLine(15, 39, 300, 39, MAROON);

        tft.setTextSize(2);
        tft.setTextColor(BLUE);
        tft.setCursor(40, 13);
        tft.print("Temperature & Humidity");

        draw_bar(x_bar_t, y_bar_t);

        tft.drawLine(190, 60, 190, 227, MAROON);
        tft.drawLine(190, 127, 300, 127, MAROON);

        tft.fillRect(202, 60, 100, 27, CYAN);
        tft.setTextSize(2);
        tft.setTextColor(BLACK);
        tft.setCursor(205, 65);
        tft.print("Humidity");

        tft.fillRect(202, 140, 100, 43, GREEN);
        tft.setTextSize(2);
        tft.setTextColor(BLACK);
        tft.setCursor(227, 145);
        tft.print("Heat");
        tft.setCursor(220, 165);
        tft.print("Index");

        DrawButtonBack(8, 6);

        GetDHT11Data();
        delay(100);
      }
    }
  }

  //---------------------------------------Menu or Mode to control the LED
  if (Menu == 1) {
    ControlTheLED();
  }
  //---------------------------------------

  //---------------------------------------Menu or Mode to display DHT11
sensor data
  if (Menu == 2) {
```

```
    ShowDHT11Data();
  }
  //----------------------------------------
  if(brightnessRoomOld != brightnessRoom)
  {
    Blynk.virtualWrite(V4, brightnessRoom);
    if(Menu == 1) DrawTempLux(brightnessRoom);
    brightnessRoomOld = brightnessRoom;
  }
  Serial.println(modeBtn);
  if(numHuman == 0)
  {
    FastLED.setBrightness(0);
    FastLED.show();
  }
  else
  {
    brightnessMode(modeBtn);
  }
  // brightnessMode(modeBtn);
  //SR501(modeBtn, brightnessRoom);
  ControlButtonMode(modeBtn);
}
//========================================================================

//========================================================================GetT
SPoint()
void GetTSPoint()
{
  TS_Point p = ts.getPoint();
  p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.height());
  p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.width());
  y_set_rotatoon_135 = map(p.x, 0, 240, 0, tft.height());
  x_set_rotatoon_135 = map(tft.width() - p.y, 0, 320, 0, tft.width());
  TSPointZ = p.z;
}
//========================================================================Draw
ButtonTempHum()
void DrawButtonTempHum()
{
  tft.fillRoundRect(17, 160, 280, 40, 10, WHITE);
  tft.fillRoundRect(19, 162, 276, 36, 10, BLUE);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(25, 173);
  tft.print("Temperature & Humidity");
}
//========================================================================
```

```
//=========================================================================DrawButtonTempHumPress()
void DrawButtonTempHumPress()
{
  tft.fillRoundRect(17, 160, 280, 40, 10, BLACKM);
}
//=========================================================================

//=========================================================================DrawButtonBack(x, y)
void DrawButtonBack(int x_btn_back, int y_btn_back)
{
  tft.fillRoundRect(x_btn_back, y_btn_back, 30, 30, 5, MAROON);
  tft.fillRoundRect(x_btn_back+2, y_btn_back+2, 26, 26, 5, YELLOW);
  tft.setTextSize(2);
  tft.setTextColor(BLACKM);
  tft.setCursor(x_btn_back+7, y_btn_back+7);
  tft.print("<");
}
//=========================================================================

//=========================================================================DrawButtonBackPress(x, y)
void DrawButtonBackPress(int x_btn_back, int y_btn_back)
{
  tft.fillRoundRect(x_btn_back, y_btn_back, 30, 30, 5, BLACKM);
}
//=========================================================================

//=========================================================================GetDHT11Data()
void GetDHT11Data()
{
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  Humidity = dht.readHumidity();;
  // Read temperature as Celsius (the default)
  Temperature = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  Fahreheit = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(Humidity) || isnan(Temperature) || isnan(Fahreheit)) {

    return;
  }
```

```
  // Compute heat index in Fahrenheit (the default)
  hif = dht.computeHeatIndex(Fahreheit, Humidity);
  // Compute heat index in Celsius (isFahreheit = false)
  hic = dht.computeHeatIndex(Temperature, Humidity, false);
  Blynk.virtualWrite(V0, Temperature);
  Blynk.virtualWrite(V1, Humidity);
  Serial.print("\n");
  Serial.print("Humidity: " + String(Humidity) + "%");
  Serial.print("\t");
  Serial.print("Temperature:" + String(Temperature) + " C");
}
//=========================================================================

//=========================================================================draw
_bar (Temperature Bar)
void draw_bar(int x_bar, int y_bar)
{
  tft.fillRoundRect(x_bar, y_bar, 35, 120, 5, DARKORANGE);
  tft.fillCircle(x_bar+17, y_bar+140, 30, DARKORANGE);
  tft.fillRoundRect(x_bar+4, y_bar+4, 27, 120, 2, BLACKM);
  tft.fillCircle(x_bar+17, y_bar+140, 25, BLACKM);
  tft.fillRect(x_bar+8, y_bar+8, 19, 120, DARKORANGE);
  tft.fillCircle(x_bar+17, y_bar+140, 21, DARKORANGE);

  //tft.fillRect(41, 58, 19, 108, RED);

  tft.drawLine(x_bar+37, y_bar+8, x_bar+42, y_bar+8, RED);
  tft.setTextSize(1);
  tft.setTextColor(RED);
  tft.setCursor(x_bar+47, y_bar+4);
  tft.println("50");

  tft.drawLine(x_bar+37, y_bar+115, x_bar+42, y_bar+115, RED);
  tft.setCursor(x_bar+47, y_bar+111);
  tft.println("0");
}
//=========================================================================

//=========================================================================Menu
_display()
void Menu_display()
{
  tft.fillScreen(BLACK);
  tft.setTextSize(3);
  DrawButtonControlLED();
  DrawButtonTempHum();
}
```

```
//=========================================================================

//=========================================================================ControlTheLED()
void ControlTheLED()
{
  GetTSPoint();

  if (TSPointZ > MINPRESSURE && TSPointZ < MAXPRESSURE)
  {
    if (x_set_rotatoon_135 > 0 && x_set_rotatoon_135 < (0+50) && y_set_rotatoon_135 > 26 && y_set_rotatoon_135 < (26+60))
    { brightnessRoom -= 10;
      DrawButtonTempTruPress();
      delay(100);
      DrawButtonTempTru();
    }
    //---------------------------------------Condition to detect when the Cong Button is touched and the command
    if (x_set_rotatoon_135 > 260 && x_set_rotatoon_135 < (260+60) && y_set_rotatoon_135 > 26 && y_set_rotatoon_135 < (26+60))
    { brightnessRoom += 10;
      DrawButtonTempCongPress();
      delay(100);
      DrawButtonTempCong();
    }
    //---------------------------------------Condition to detect when the Normal Button is touched and the command
    if (x_set_rotatoon_135 > 35 && x_set_rotatoon_135 < (35+60) && y_set_rotatoon_135 > 120 && y_set_rotatoon_135 < (120+60))
    {
      modeBtn = 1;
      DrawButtonTempNormalPress();
      delay(100);
      DrawButtonTempNormal();
    }
    //---------------------------------------Condition to detect when the Read Button is touched and the command
    if (x_set_rotatoon_135 > 130 && x_set_rotatoon_135 < (130+60) && y_set_rotatoon_135 > 120 && y_set_rotatoon_135 < (120+60))
    {
      modeBtn = 2;
      DrawButtonTempReadPress();
      delay(100);
      DrawButtonTempRead();
    }
    //---------------------------------------Condition to detect when the Sleep Button is touched and the command
```

```
    if (x_set_rotatoon_135 > 225 && x_set_rotatoon_135 < (225+60) &&
y_set_rotatoon_135 > 120 && y_set_rotatoon_135 < (120+60))
    {
      modeBtn = 3;
      DrawButtonTempSleepPress();
      delay(100);
      DrawButtonTempSleep();
    }
    //--------------------------------------
  }

  if(mode != modeBtn)
  {
    tft.fillRect(165, 210, 80, 30, BLACK);
    mode = modeBtn;
  }
  if(brightnessRoomOld != brightnessRoom)
  {
    Blynk.virtualWrite(V4, brightnessRoom);
    DrawTempLux(brightnessRoom);
    brightnessRoomOld = brightnessRoom;
  }
  DrawTempMode(modeBtn);
  GetTSPoint();

  if (TSPointZ > MINPRESSURE && TSPointZ < MAXPRESSURE)
  {
    if (x_set_rotatoon_135 > 10 && x_set_rotatoon_135 < (5+45) &&
y_set_rotatoon_135 > 195 && y_set_rotatoon_135 < (195+45)) {
      Menu = 0;
      DrawButtonBackPress(10, 200);
      delay(100);
      DrawButtonBack(10, 200);
      //delay(100);
      Menu_display();
    }
  }
}
//=========================================================================

//=========================================================================ShowDHT11Data()
void ShowDHT11Data() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
```

```
  GetDHT11Data();
}

T_to_Bar = map(Temperature, 0.0, 50.0, 108, 0);

tft.fillRect(x_bar_t+8, (y_bar_t+8)+T_to_Bar, 19, 108-T_to_Bar, ORANGE);
tft.fillRect(x_bar_t+8, y_bar_t+8, 19, T_to_Bar, BLACK);

tft.setTextSize(2);
tft.setTextColor(ORANGE, BLACK);
tft.setCursor(75, 100);
tft.print(Temperature);
if (Temperature < 10) tft.print(" ");
tft.setCursor(160, 100);
tft.print((char)247);
tft.println("C");


tft.setCursor(75, 135);
tft.print(Fahreheit);
if (Fahreheit < 100) tft.print(" ");
tft.setCursor(160, 135);
tft.print((char)247);
tft.println("F");

tft.setTextSize(3);
tft.setTextColor(CYAN, BLACK);
tft.setCursor(205, 95);
tft.print(Humidity);
tft.print(" %");

tft.setTextSize(1);
tft.setTextColor(GREEN, BLACK);
tft.setCursor(205, 200);
tft.print(hic);
tft.print(" ");
tft.print((char)247);
tft.print("C");
if (hic < 10) tft.print(" ");

tft.setTextSize(1);
tft.setTextColor(GREEN, BLACK);
tft.setCursor(205, 220);
tft.print(hif);
tft.print(" ");
tft.print((char)247);
tft.print("F");
```

```
  if (hif < 100) tft.print(" ");

  GetTSPoint();

  if (TSPointZ > MINPRESSURE && TSPointZ < MAXPRESSURE)
  {
    if (x_set_rotatoon_135 > 3 && x_set_rotatoon_135 < (3+50) &&
y_set_rotatoon_135 > 1 && y_set_rotatoon_135 < (1+50)) {
      Menu = 0;
      DrawButtonBackPress(8, 6);
      delay(100);
      DrawButtonBack(8, 6);
      //delay(100);
      Menu_display();
    }
  }
}
void WriteDHT11Data()
{
  unsigned long currentMillis_WriteDHT = millis();
  if (currentMillis_WriteDHT - previousMillis_WriteDHT >= interval)
  {
    // save the last time you blinked the LED
    previousMillis_WriteDHT = currentMillis_WriteDHT;
    GetDHT11Data();
  }
}
//================================================================================
void DrawTempLux(int lux)
{
  int px_x = map(lux, 0, 1000, 0, 216);
  tft.fillRoundRect(50, 36, 220, 40, 10, WHITE);
  tft.fillRoundRect(52, 38, px_x, 36, 10, BLUE);
  tft.setTextSize(2);
  tft.setTextColor(RED);
  tft.setCursor(120, 49);
  tft.print(lux);
  tft.setCursor(170, 49);
  tft.print("lux");
}
void DrawButtonTempTru()
{
  tft.fillRoundRect(10, 41, 30, 30, 5, YELLOW);
  //tft.fillRoundRect(52, 38, 216, 36, 10, BLUE);
  tft.setTextSize(2);
  tft.setTextColor(BLACKM);
  tft.setCursor(20, 50);
  tft.print("-");
```

```
}
void DrawButtonTempCong()
{
  tft.fillRoundRect(280, 41, 30, 30, 5, YELLOW);
  //tft.fillRoundRect(52, 38, 216, 36, 10, BLUE);
  tft.setTextSize(3);
  tft.setTextColor(BLACKM);
  tft.setCursor(288, 46);
  tft.print("+");
}
void DrawButtonTempNormal()
{
  tft.fillRoundRect(35, 120, 60, 60, 10, WHITE);
  tft.fillRoundRect(37, 122, 56, 56, 10, RED);
  tft.setTextSize(1);
  tft.setTextColor(BLACKM);
  tft.setCursor(47, 146);
  tft.print("Normal");
}
void DrawButtonTempRead()
{
  tft.fillRoundRect(130, 120, 60, 60, 10, WHITE);
  tft.fillRoundRect(132, 122, 56, 56, 10, YELLOW);
  tft.setTextSize(1);
  tft.setTextColor(BLACKM);
  tft.setCursor(145, 146);
  tft.print("Read");
}
void DrawButtonTempSleep()
{
  tft.fillRoundRect(225, 120, 60, 60, 10, WHITE);
  tft.fillRoundRect(227, 122, 56, 56, 10, BLUE);
  tft.setTextSize(1);
  tft.setTextColor(BLACKM);
  tft.setCursor(240, 146);
  tft.print("Sleep");
}
void DrawMode()
{
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(95, 210);
  tft.print("Mode:");
}
void DrawTempMode(int mode)
{
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
```

```
  if(mode == 1)
  {
    tft.setCursor(165, 210);
    tft.print("Normal");
    Blynk.virtualWrite(V5, 1);
  }
  if(mode == 2)
  {
    tft.setCursor(165, 210);
    tft.print("Read");
    brightnessMode(2);
    Blynk.virtualWrite(V6, 1);
  }
  if(mode == 3)
  {
    tft.setCursor(165, 210);
    tft.print("Sleep");
    brightnessMode(3);
    Blynk.virtualWrite(V7, 1);
  }

}
void DrawTempAmountOfPeople(int numHuman)
{
  tft.setTextSize(2);
  tft.setTextColor(WHITE,BLACK);
  tft.setCursor(45, 20);
  tft.print("Amount of people: ");
  if(numHuman < 10) tft.print(" ");
  tft.print(numHuman);
}
void DrawButtonControlLED()
{
  tft.fillRoundRect(17, 90, 280, 40, 10, WHITE);
  tft.fillRoundRect(19, 92, 276, 36, 10, GREEN);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(65, 103);
  tft.print("Control the LED");
}
//============================================================================

//============================================================================DrawButtonControlLEDPress()
void DrawButtonControlLEDPress()
{
  tft.fillRoundRect(17, 90, 280, 40, 10, BLACKM);
```

```
}
void DrawButtonTempTruPress()
{
  tft.fillRoundRect(10, 41, 30, 30, 5, BLACKM);
}
void DrawButtonTempCongPress()
{
  tft.fillRoundRect(280, 41, 30, 30, 5, BLACKM);
}
void DrawButtonTempNormalPress()
{
  tft.fillRoundRect(35, 120, 60, 60, 10, BLACKM);
}
void DrawButtonTempReadPress()
{
  tft.fillRoundRect(130, 120, 60, 60, 10, BLACKM);
}
void DrawButtonTempSleepPress()
{
  tft.fillRoundRect(225, 120, 60, 60, 10, BLACKM);
}
void readNumHuman()
{
  distanse1 = 0;
  distanse2 = 0;
  measureDistanse1();
  measureDistanse2();
  if(distanse1<10 && queue.charAt(0)!='1')
  {
    queue+="1";
  }
  else if(distanse2<10 && queue.charAt(0)!='2')
  {
    queue+="2";
  }
  if(queue.equals("12"))
  {
    numHuman++;
    Serial.print("queue: ");
    Serial.println(queue);
    queue="";
  }
 else if(queue.equals("21") && numHuman>0)
 {
    numHuman--;
    Serial.print("Hang doi: ");
    Serial.println(queue);
    queue="";
```

```
 }
  if(queue.length()>2 || queue.equals("11") || queue.equals("22")
||timeoutcounter>100)
  {
    queue="";
  }
  if(queue.length()==1)
  {timeoutcounter++;}
  else {timeoutcounter=0;}
  Blynk.virtualWrite(V3, numHuman);
}
void measureDistanse1()
{
  digitalWrite(trig1, LOW);
  delayMicroseconds(2);
  digitalWrite(trig1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig1, LOW);
  unsigned long time1 = pulseIn(echo1, HIGH);
  distanse1 = int(time1 / 2 / 29.412);
}
void measureDistanse2()
{
  digitalWrite(trig2, LOW);
  delayMicroseconds(2);
  digitalWrite(trig2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig2, LOW);
  unsigned long time2 = pulseIn(echo2, HIGH);
  distanse2 = int(time2 / 2 / 29.412);
}
void ShowPower()
{
  unsigned long currentMillis_Power = millis();
  if (currentMillis_Power - previousMillis_Power >= interval)
  {
    previousMillis_Power = currentMillis_Power;
    readPower();
  }
  if(Menu == 0)
  {
    DrawTempPower(amps,volt);
  }
  Blynk.virtualWrite(V2, amps*volt);
}
void DrawTempPower(float amps, float volt)
{
  tft.setTextSize(2);
```

```
  tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);
  tft.setCursor(15, 57);
  tft.print("Power: ");
  tft.print(volt);
  tft.print("V");
  tft.print(" ");
  tft.print(amps);
  tft.print("A");
  tft.print(" ");
  tft.print(amps*volt);
  tft.print("W");
}
void readPower()
{
  int rawValue = analogRead(analogAmpe);
  float voltage = (rawValue / 4096.0) * 3300;
  amps = ((2500 - voltage) / 66);
  int aVolt = analogRead(analogVolt);
  volt = (((aVolt-1024) * 3.3)*2) / 4096.0;
  if(amps < 0) {amps=0;}
  if(volt < 0) {volt=0;}
}
void ledColor(int modeColor, int Brightness)
{
  //modeColor:
  //1: White (Normal)
  //2: SaddleBrown (Read)
  //3: Goldenrod1 (Sleep)
  if(modeColor == 1)
  {
    FastLED.setBrightness(Brightness);
    fill_solid(leds,NUM_LEDS, CRGB::White);
    fill_solid(leds_1,NUM_LEDS, CRGB::White);
    FastLED.show();
  }
  if(modeColor == 2)
  {
    FastLED.setBrightness(Brightness);
    fill_solid(leds,NUM_LEDS, CRGB::SaddleBrown);
    fill_solid(leds_1,NUM_LEDS, CRGB::SaddleBrown);
    FastLED.show();
  }
  if(modeColor == 3)
  {
    FastLED.setBrightness(Brightness);
    fill_solid(leds,NUM_LEDS, CRGB::Goldenrod1);
    fill_solid(leds_1,NUM_LEDS, CRGB::Goldenrod1);
    FastLED.show();
```

```
  }
}
void brightnessMode(int numColor)
{
  float lux = lightMeter.readLightLevel();
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
  if(lux < (brightnessRoom - 10) || lux > (brightnessRoom + 10))
  {

    if(brightnessLed == 0) brightnessLed = 1;
    if(brightnessLed == 255) brightnessLed = 254;
    if(lux < brightnessRoom)
    {
      brightnessLed++;
    }
    if(lux > brightnessRoom)
    {
      brightnessLed--;
    }
    Serial.println(brightnessLed);
  }
  ledColor(numColor,brightnessLed);
  Serial.println(brightnessRoom);
}
void ControlButtonMode(int modeBtn)
{
  if(modeBtnOld != modeBtn)
  {
    if(modeBtn == 1)
    {
      Blynk.virtualWrite(V5, 1);
      Blynk.virtualWrite(V6, 0);
      Blynk.virtualWrite(V7, 0);
    }
     if(modeBtn == 2)
    {
      Blynk.virtualWrite(V5, 0);
      Blynk.virtualWrite(V7, 0);
    }
     if(modeBtn == 3)
    {
      Blynk.virtualWrite(V5, 0);
      Blynk.virtualWrite(V6, 0);
    }
    modeBtnOld = modeBtn;
  }
```

```
}
void SR501(int modeBtn, int brightnessRoom)
{
  int readDataSR501 = digitalRead(dataSR501);
  if(readDataSR501 == HIGH)
  {
    // FastLED.setBrightness(brightnessRoom);
    // fill_solid(leds_1,NUM_LEDS, CRGB::White);
    // FastLED.show();
    Serial.println("ON");
  }
  else
  {
    // FastLED.setBrightness(brightnessRoom);
    // fill_solid(leds_1,NUM_LEDS, CRGB::Black);
    // FastLED.show();
    Serial.println("OFF");
  }
}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<
```

FILE BITMAP.H:

```
const unsigned short loading[5510] PROGMEM={
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0010 (16)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0020 (32)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0030 (48)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0040 (64)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0050 (80)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0060 (96)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0070 (112)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0080 (128)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0090 (144)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00A0 (160)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00B0 (176)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00C0 (192)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00D0 (208)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00E0 (224)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00F0 (240)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0100 (256)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0110 (272)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0120 (288)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0130 (304)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0140 (320)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0150 (336)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0160 (352)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0170 (368)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF,   // 0x0180 (384)
pixels
0xFFFF, 0xDEFB, 0xFFDF, 0xCE79, 0xDEFB, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0190 (400)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xF79E, 0xFFDF, 0xEF7D, 0xF79E, 0xC618, 0x8410, 0xFFDF,   // 0x01A0 (416)
pixels
0xE73C, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x01B0 (432)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xB596, 0xDEFB, 0xEF5D, 0xFFFF,
0xD6BA, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x01C0 (448)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0xFFFF,
0xDEDB, 0xFFDF, 0xC618, 0xF7BE, 0xBDD7, 0xFFFF, 0xF7BE,   // 0x01D0 (464)
pixels
0xFFFF, 0xCE79, 0xFFFF, 0xDEDB, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x01E0 (480)
pixels
0xFFFF, 0xFFFF, 0xEF7D, 0xFFFF, 0xC618, 0xFFDF, 0xFFDF, 0xF7BE, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xDEDB,   // 0x01F0 (496)
pixels
0xFFFF, 0xDEDB, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0xF7BE, 0xEF7D, 0xAD55,   // 0x0200 (512)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFDF, 0xFFDF, 0xE71C, 0xDEFB, 0xF79E, 0xB596,   // 0x0210 (528)
pixels
0xF7BE, 0xD6BA, 0xF79E, 0x94B2, 0xFFFF, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0220 (544)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0230 (560)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xBDD7, 0x31A6, 0x4208, 0x3186,   // 0x0240 (576)
pixels
0x528A, 0x0861, 0x10A2, 0xB596, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0250 (592)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0x9CD3, 0x6B4D, 0x39C7,
0x2124, 0x18E3, 0x0861, 0x0000, 0x31A6, 0x0841, 0x630C,   // 0x0260 (608)
pixels
0xAD55, 0xBDF7, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0270 (624)
pixels
```

```
0xEF5D, 0x73AE, 0x5ACB, 0x0861, 0x2965, 0x2945, 0x52AA, 0x0861, 0x630C,
0x6B6D, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0280 (640)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x39C7, 0x52AA, 0x1082, 0x4228,
0x10A2, 0x4208, 0x0020, 0x630C, 0x2104, 0x5ACB, 0x0841,   // 0x0290 (656)
pixels
0x738E, 0x1082, 0x8C71, 0xC638, 0xCE79, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF5D, 0x6B4D,   // 0x02A0 (672)
pixels
0x2104, 0x5AEB, 0x0000, 0x4A49, 0x4228, 0x8430, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xB596, 0x4A49, 0x0861, 0x6B4D, 0x1082,   // 0x02B0 (688)
pixels
0x39C7, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDF7,
0x8C71, 0x1082, 0x31A6, 0x39C7, 0x0020, 0xA534, 0xFFFF,   // 0x02C0 (704)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x9492,
0x4208, 0x4A69, 0x18E3, 0x2104, 0x0000, 0x18E3, 0x0841,   // 0x02D0 (720)
pixels
0x2124, 0x0000, 0x6B6D, 0x18E3, 0xDEDB, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x02E0 (736)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x02F0 (752)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xCE59, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0300 (768)
pixels
0x0000, 0x18C3, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0310 (784)
pixels
0xFFFF, 0xF79E, 0x5ACB, 0x632C, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3,   // 0x0320 (800)
pixels
0x8C71, 0xBDD7, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0x0861, 0x0000,   // 0x0330 (816)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0841, 0x0841, 0x4A49,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0340 (832)
pixels
0xFFFF, 0xFFFF, 0x9CD3, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x6B4D, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0350 (848)
pixels
0x0020, 0x0000, 0x39C7, 0x8C71, 0xE73C, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xF79E, 0x18C3, 0x5AEB, 0x0020, 0x0000,   // 0x0360 (864)
pixels
0x0000, 0x6B4D, 0x0000, 0x10A2, 0xCE79, 0xFFFF, 0xFFFF, 0xFFFF, 0xD69A,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3,   // 0x0370 (880)
pixels
```

```
0x73AE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x528A, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0xC638, 0xFFFF, 0xFFFF,   // 0x0380 (896)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xD6BA, 0x8410, 0x2104, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0390 (912)
pixels
0x0000, 0x0000, 0x2945, 0x3186, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x03A0 (928)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x03B0 (944)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF7D, 0x10A2,
0x10A2, 0x39E7, 0x0000, 0x0000, 0x0000, 0x0000, 0x2945,   // 0x03C0 (960)
pixels
0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618,   // 0x03D0 (976)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x2124, 0x10A2, 0x0000, 0x0000, 0x0000, 0x39C7,   // 0x03E0 (992)
pixels
0xCE59, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFDF, 0x2124, 0x0000, 0x0000, 0x0000,   // 0x03F0 (1008)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0x18E3, 0x4208, 0xEF7D, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0400 (1024)
pixels
0x52AA, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4228, 0xFFFF,
0xA534, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0410 (1040)
pixels
0x0000, 0x0000, 0x6B4D, 0x9CD3, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0x2945, 0x0000, 0x0000, 0x0000, 0x0000, 0x2124,   // 0x0420 (1056)
pixels
0x0000, 0x18C3, 0xE71C, 0xFFFF, 0xFFFF, 0xFFDF, 0x2124, 0x0000, 0x0000,
0x0000, 0x0020, 0xB596, 0x8C51, 0x0000, 0x31A6, 0xFFFF,   // 0x0430 (1072)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xF79E, 0x18E3, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0xCE59, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0440 (1088)
pixels
0xFFFF, 0xF7BE, 0x4208, 0x0000, 0x0020, 0x52AA, 0x0841, 0x0000, 0x0000,
0x10A2, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0450 (1104)
pixels
0x0000, 0x0000, 0x1082, 0xB5B6, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0460 (1120)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0470 (1136)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x632C, 0x0000, 0x0841, 0x31A6,
0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0xCE59, 0xFFFF,   // 0x0480 (1152)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x8C71, 0x10A2, 0x0000, 0x0000,   // 0x0490 (1168)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4A49, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1082, 0xF7BE,   // 0x04A0 (1184)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0x31A6, 0x0000, 0x0000, 0x0000, 0x0000, 0x39C7, 0x2104,   // 0x04B0 (1200)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x6B6D, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xB5B6, 0x0000,   // 0x04C0 (1216)
pixels
0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x2124, 0xB596, 0x5ACB, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x04D0 (1232)
pixels
0x0000, 0x4228, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x8410, 0x0000,
0x0000, 0x0000, 0x0000, 0x5AEB, 0xFFDF, 0x7BCF, 0x0000,   // 0x04E0 (1248)
pixels
0xBDF7, 0xFFFF, 0xFFFF, 0xFFFF, 0x8C71, 0x0000, 0x0000, 0x0000, 0x0841,
0xCE79, 0x7BCF, 0x0000, 0x18C3, 0xB5B6, 0xFFFF, 0xFFFF,   // 0x04F0 (1264)
pixels
0xFFFF, 0xFFFF, 0x3186, 0x18E3, 0xAD55, 0x632C, 0x0000, 0x0000, 0x0000,
0x5ACB, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x7BEF, 0x0020,   // 0x0500 (1280)
pixels
0x0000, 0x3186, 0xE71C, 0xFFFF, 0x73AE, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18E3,   // 0x0510 (1296)
pixels
0x31A6, 0x0000, 0x8430, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0520 (1312)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0530 (1328)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xE71C, 0x10A2, 0x0000, 0x0000, 0x0000, 0x0000,
0x18C3, 0x0000, 0x31A6, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0540 (1344)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0x2945, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0550 (1360)
pixels
0x0000, 0x8430, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x2104, 0x1082,
0x0000, 0x0000, 0x0000, 0x0000, 0x31A6, 0xD69A, 0xFFFF,   // 0x0560 (1376)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x6B6D, 0x0000,
0x0000, 0x0000, 0x0000, 0xC618, 0x9CD3, 0x0000, 0x0000,   // 0x0570 (1392)
pixels
```

```
0x0000, 0x0020, 0x0861, 0x8430, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x7BCF, 0x0000, 0x18E3, 0xCE79,   // 0x0580 (1408)
pixels
0xF79E, 0x6B6D, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0861, 0x738E, 0xB596, 0x2945, 0x0000, 0x0000, 0x0000,   // 0x0590 (1424)
pixels
0x52AA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0x0861, 0x0841, 0x0000,
0x0000, 0x2104, 0xB5B6, 0x4228, 0x0000, 0xA534, 0xFFFF,   // 0x05A0 (1440)
pixels
0xFFFF, 0xFFFF, 0x4228, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000,
0x0000, 0x0000, 0x0841, 0xE73C, 0xFFFF, 0xFFFF, 0xDEDB,   // 0x05B0 (1456)
pixels
0x0841, 0x7BCF, 0xFFFF, 0xC618, 0x0000, 0x0000, 0x0000, 0xB596, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x73AE, 0x0000, 0x0000, 0x3186,   // 0x05C0 (1472)
pixels
0xFFFF, 0xFFFF, 0xDEDB, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x10A2, 0x18E3, 0x0000,   // 0x05D0 (1488)
pixels
0x18C3, 0x9CF3, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x05E0 (1504)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x05F0 (1520)
pixels
0xFFFF, 0xAD55, 0x0000, 0x0000, 0x52AA, 0x39E7, 0x0841, 0x8430, 0x0000,
0x0000, 0xB596, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0600 (1536)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x5AEB, 0x0000,
0x0000, 0x18C3, 0x9492, 0x630C, 0x0000, 0x0000, 0x0000,   // 0x0610 (1552)
pixels
0x0841, 0x0020, 0x1082, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0xD69A, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0620 (1568)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xAD75, 0x0861, 0x0000, 0x0000, 0x0000,
0x0000, 0x1082, 0x10A2, 0x0000, 0x0841, 0x0000, 0x2965,   // 0x0630 (1584)
pixels
0x528A, 0x0020, 0x9492, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xAD55, 0x0000, 0x5AEB, 0xFFFF, 0xFFFF, 0xEF7D,   // 0x0640 (1600)
pixels
0x2104, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x31A6, 0xFFFF,
0xFFFF, 0x94B2, 0x0000, 0x0000, 0x0000, 0x0841, 0x8410,   // 0x0650 (1616)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xB5B6, 0x0020, 0x9CD3, 0xCE59, 0x1082, 0x0000,
0x0000, 0x0000, 0x18E3, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0660 (1632)
pixels
0x9492, 0x0000, 0x0000, 0x0000, 0x0000, 0x10A2, 0x0000, 0x0000, 0x0000,
0x0020, 0xC618, 0xFFFF, 0xFFFF, 0xBDF7, 0x0000, 0x0020,   // 0x0670 (1648)
pixels
```

```
0x4A69, 0x10A2, 0x0000, 0x0000, 0x0000, 0xA534, 0xFFFF, 0xFFFF, 0xFFFF,
0xA514, 0x0861, 0x0000, 0x0000, 0x0841, 0xCE79, 0xFFFF,   // 0x0680 (1664)
pixels
0x7BCF, 0x0000, 0x0000, 0x1082, 0x0000, 0x1082, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xCE79,   // 0x0690 (1680)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x06A0 (1696)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF79E,   // 0x06B0 (1712)
pixels
0x18C3, 0x0000, 0x632C, 0x4228, 0x0000, 0x0000, 0x0000, 0x0841, 0xCE79,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x06C0 (1728)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF5D, 0x39E7, 0x0000, 0x0000, 0x4A49,
0xFFFF, 0xB596, 0x0000, 0x0000, 0x3186, 0xA534, 0xAD75,   // 0x06D0 (1744)
pixels
0xCE59, 0x8C51, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x31A6, 0xDEDB, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x06E0 (1760)
pixels
0xFFFF, 0xFFFF, 0x94B2, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0841, 0xCE79, 0x2124, 0x0000, 0x0000, 0x0000,   // 0x06F0 (1776)
pixels
0xA534, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x7BEF,
0x0000, 0x4208, 0xDEDB, 0xFFDF, 0x9CD3, 0x10A2, 0x0000,   // 0x0700 (1792)
pixels
0x6B4D, 0xCE59, 0x9CD3, 0xAD55, 0x4208, 0x0000, 0xA534, 0xC638, 0x4A69,
0x0000, 0x0000, 0x0000, 0x0000, 0x9492, 0xFFFF, 0xFFFF,   // 0x0710 (1808)
pixels
0xFFFF, 0xD69A, 0x0861, 0xC638, 0xFFFF, 0x3186, 0x0000, 0x0000, 0x0000,
0x0000, 0xB5B6, 0xFFFF, 0xFFFF, 0xEF7D, 0x18E3, 0x0000,   // 0x0720 (1824)
pixels
0x0000, 0x0000, 0x2104, 0xDEFB, 0x0000, 0x0000, 0x0000, 0x0000, 0x0841,
0xB596, 0xFFFF, 0xFFFF, 0x31A6, 0x0000, 0x0000, 0x0000,   // 0x0730 (1840)
pixels
0x0841, 0x4228, 0x0020, 0x738E, 0xFFFF, 0xFFFF, 0xFFFF, 0x94B2, 0x18E3,
0x4A49, 0x0000, 0x0000, 0x0020, 0x4208, 0x0861, 0x0841,   // 0x0740 (1856)
pixels
0x4A49, 0xDEFB, 0xAD55, 0xA534, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0xB596, 0xFFFF, 0xFFFF,   // 0x0750 (1872)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0760 (1888)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x9CF3, 0x2104, 0x1082,   // 0x0770 (1904)
pixels
```

```
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2965, 0xFFDF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0780 (1920)
pixels
0xFFFF, 0xFFFF, 0x528A, 0x0000, 0x0000, 0x0000, 0x0000, 0x31A6, 0x18E3,
0x0000, 0x0841, 0xA534, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0790 (1936)
pixels
0x52AA, 0x18E3, 0x8410, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x4A49,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618,   // 0x07A0 (1952)
pixels
0x39E7, 0x0000, 0x0000, 0x0000, 0x0000, 0x0841, 0x0000, 0x0861, 0x0000,
0x630C, 0x1082, 0x0000, 0x0000, 0x0000, 0x18E3, 0xE73C,   // 0x07B0 (1968)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0x6B6D, 0x0000, 0x0000,
0x0020, 0x2104, 0x0000, 0x0000, 0x18E3, 0x94B2, 0xFFFF,   // 0x07C0 (1984)
pixels
0xFFFF, 0xFFFF, 0xD69A, 0x73AE, 0x0000, 0x0000, 0x0000, 0x0000, 0x4228,
0x0000, 0x0000, 0x0841, 0xCE59, 0xFFFF, 0xFFFF, 0x8C71,   // 0x07D0 (2000)
pixels
0x0000, 0x4208, 0x4228, 0x0000, 0x0020, 0x8430, 0x8C71, 0x10A2, 0xEF7D,
0xFFFF, 0xFFFF, 0xFFFF, 0x8C71, 0x0000, 0x0000, 0x0000,   // 0x07E0 (2016)
pixels
0x0000, 0x4228, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x7BCF, 0xFFFF,
0xD69A, 0x0000, 0x0000, 0x0000, 0x0000, 0x6B4D, 0xFFFF,   // 0x07F0 (2032)
pixels
0x2965, 0x94B2, 0xFFFF, 0xFFFF, 0xFFFF, 0x738E, 0x3186, 0xAD75, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0841, 0xDEFB, 0xFFFF,   // 0x0800 (2048)
pixels
0xFFFF, 0xFFFF, 0xE71C, 0xBDD7, 0x630C, 0x8C71, 0x4A49, 0x738E, 0x4228,
0x73AE, 0x7BCF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0810 (2064)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0820 (2080)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xEF7D, 0x39C7, 0x1082, 0x0000, 0x0000,   // 0x0830 (2096)
pixels
0x0000, 0x0000, 0x0000, 0x0020, 0xCE79, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0840 (2112)
pixels
0xD69A, 0x0020, 0xA514, 0xB5B6, 0x0841, 0x0000, 0x0000, 0x0000, 0x10A2,
0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xA514, 0x4A49,   // 0x0850 (2128)
pixels
0xB5B6, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0xE73C, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0x0020, 0x0000,   // 0x0860 (2144)
pixels
0x31A6, 0x0000, 0x0000, 0x7BCF, 0x528A, 0xDEDB, 0x10A2, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x2124, 0xCE59, 0xFFFF, 0xFFFF,   // 0x0870 (2160)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x7BEF, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x9CF3, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0880 (2176)
pixels
0xFFFF, 0xEF5D, 0x0861, 0x630C, 0x2945, 0x0000, 0x0000, 0x0000, 0x0000,
0x0861, 0xEF5D, 0xFFFF, 0xFFFF, 0xEF7D, 0x18C3, 0x0000,   // 0x0890 (2192)
pixels
0x0000, 0x0000, 0x0841, 0xDEFB, 0xFFFF, 0x1082, 0xDEDB, 0xFFFF, 0xFFFF,
0xF79E, 0x10A2, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020,   // 0x08A0 (2208)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x8430, 0xF7BE, 0x2104,
0x0000, 0x0000, 0x0000, 0x0861, 0x528A, 0x0020, 0x52AA,   // 0x08B0 (2224)
pixels
0xEF7D, 0xFFFF, 0xF79E, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x2104, 0xD69A, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x08C0 (2240)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x08D0 (2256)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x08E0 (2272)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xB5B6, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x08F0 (2288)
pixels
0x0000, 0x18C3, 0xBDD7, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7BE, 0x3186, 0x0000,   // 0x0900 (2304)
pixels
0xB5B6, 0xFFDF, 0x2124, 0x0000, 0x0000, 0x0000, 0x18C3, 0xEF7D, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xCE59, 0x0020, 0x0000, 0x0000,   // 0x0910 (2320)
pixels
0x0000, 0x18C3, 0x73AE, 0x0020, 0x2124, 0xE73C, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFDF, 0x528A, 0x0000, 0x18E3, 0xE71C, 0x1082,   // 0x0920 (2336)
pixels
0x0000, 0x0000, 0x8C51, 0xFFFF, 0x8410, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x2965, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0930 (2352)
pixels
0xFFFF, 0xFFFF, 0x8C51, 0x0000, 0x0000, 0x0000, 0x0861, 0x2104, 0x0000,
0x0000, 0x94B2, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7BE,   // 0x0940 (2368)
pixels
0x3186, 0x630C, 0x3186, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x9CD3,
0xFFFF, 0xFFFF, 0x73AE, 0x0000, 0x0000, 0x0000, 0x1082,   // 0x0950 (2384)
pixels
0x0000, 0x2945, 0x3186, 0x0000, 0x6B6D, 0xFFFF, 0xFFFF, 0xFFFF, 0x94B2,
0x0000, 0x0000, 0x0000, 0x0000, 0x31A6, 0x0000, 0x0000,   // 0x0960 (2400)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x3186, 0x2965, 0x0000, 0x0000, 0x0000,
0x0000, 0x0020, 0x0000, 0x0000, 0x73AE, 0xFFFF, 0xFFFF,   // 0x0970 (2416)
pixels
```

```
0xFFFF, 0x52AA, 0x0000, 0x0020, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000,
0xB596, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0xD6BA,   // 0x0980 (2432)
pixels
0xC638, 0xDEFB, 0x8430, 0xEF7D, 0xDEDB, 0xCE59, 0xFFDF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0990 (2448)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x09A0 (2464)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF7D, 0x2124,
0xC638, 0x10A2, 0x0000, 0x0861, 0x0020, 0x0000, 0x2104,   // 0x09B0 (2480)
pixels
0xD6BA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0x0000, 0x0861, 0x4A69,   // 0x09C0 (2496)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x3186, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xBDF7, 0x0000, 0x0000, 0x0000, 0x0000, 0x4208,   // 0x09D0 (2512)
pixels
0xC638, 0x0861, 0x0000, 0x7BEF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xE71C,
0x39E7, 0x0000, 0x0000, 0x0861, 0x0000, 0x0000, 0x0841,   // 0x09E0 (2528)
pixels
0xB596, 0xFFFF, 0xC638, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x632C, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x09F0 (2544)
pixels
0xA534, 0x0000, 0x0000, 0x0000, 0x0841, 0x2104, 0x0000, 0x0000, 0x9CF3,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF79E, 0x2104, 0x0000,   // 0x0A00 (2560)
pixels
0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0020, 0xC618, 0xFFFF, 0xFFFF,
0xE71C, 0x1082, 0x0000, 0x0000, 0x0861, 0x0000, 0x0000,   // 0x0A10 (2576)
pixels
0x0000, 0x0861, 0xC638, 0xFFFF, 0xFFFF, 0xF7BE, 0x2124, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0A20 (2592)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x528A, 0xEF5D,
0x2965, 0x0000, 0x18E3, 0xF79E, 0xFFFF, 0xD69A, 0x0841,   // 0x0A30 (2608)
pixels
0x0000, 0x5ACB, 0xDEDB, 0x0000, 0x0000, 0x0000, 0x0000, 0x39E7, 0xFFFF,
0xFFFF, 0xFFFF, 0xA514, 0x0841, 0x0841, 0x0000, 0x0841,   // 0x0A40 (2624)
pixels
0x0000, 0x0861, 0x0020, 0x0000, 0x2965, 0xC638, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0A50 (2640)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0A60 (2656)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x5AEB, 0x0000, 0x2945, 0x0000,
0x0000, 0x4A49, 0x10A2, 0x0000, 0x0841, 0xD6BA, 0xFFFF,   // 0x0A70 (2672)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0x528A, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0A80 (2688)
pixels
0x0000, 0x0000, 0x2945, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDF7,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0A90 (2704)
pixels
0x10A2, 0xC638, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xD6BA, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0xF7BE, 0xFFFF,   // 0x0AA0 (2720)
pixels
0xEF5D, 0x18C3, 0x0000, 0x0000, 0x0020, 0x528A, 0x528A, 0x0020, 0x8C71,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xE71C, 0x2945, 0x0000,   // 0x0AB0 (2736)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xA514, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x4A49, 0x0000, 0x0000, 0x52AA,   // 0x0AC0 (2752)
pixels
0x0841, 0x0000, 0x0000, 0x0000, 0x8430, 0xFFFF, 0xFFDF, 0x5ACB, 0x0000,
0x0000, 0x1082, 0x2965, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0AD0 (2768)
pixels
0x630C, 0xFFDF, 0xFFFF, 0xFFFF, 0x8430, 0x2124, 0x31A6, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0AE0 (2784)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4A69, 0xB5B6, 0x10A2, 0x0000,
0xA534, 0xFFFF, 0xFFFF, 0xFFFF, 0x4A49, 0x0000, 0x0841,   // 0x0AF0 (2800)
pixels
0x39E7, 0x0000, 0x0000, 0x0000, 0x0000, 0x9492, 0xFFFF, 0xFFFF, 0xFFDF,
0x18E3, 0x0000, 0x0000, 0x0841, 0xB596, 0x0861, 0x0000,   // 0x0B00 (2816)
pixels
0x39E7, 0x1082, 0x0000, 0x630C, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0B10 (2832)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0B20 (2848)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xAD55, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0841, 0xDEDB, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0B30 (2864)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x7BEF,
0x0000, 0x0000, 0x0000, 0x0000, 0x18E3, 0x1082, 0x0000,   // 0x0B40 (2880)
pixels
0x0861, 0xD6BA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0x0020, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x10A2, 0xAD75,   // 0x0B50 (2896)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xD69A, 0x5ACB, 0x0020, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x7BCF, 0xFFFF, 0xFFFF, 0xFFFF, 0x5ACB,   // 0x0B60 (2912)
pixels
0x0000, 0x0000, 0x8430, 0xFFFF, 0xE71C, 0x0861, 0x94B2, 0xFFDF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x94B2, 0x0000, 0x0000, 0x0000,   // 0x0B70 (2928)
pixels
```

```
0x0000, 0x0000, 0x0000, 0x0000, 0xAD55, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0x5AEB, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0B80 (2944)
pixels
0x0000, 0x0020, 0xDEDB, 0xFFFF, 0xFFFF, 0xBDF7, 0x0841, 0x630C, 0x2965,
0x1082, 0x0000, 0x0000, 0x0000, 0x0861, 0xD69A, 0xFFFF,   // 0x0B90 (2960)
pixels
0xFFFF, 0xE71C, 0x10A2, 0x0861, 0x1082, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x2965, 0x2965, 0x0000, 0x0000,   // 0x0BA0 (2976)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0xDEFB,
0xFFFF, 0xE73C, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0BB0 (2992)
pixels
0x0000, 0x0000, 0x0000, 0x8C71, 0xFFFF, 0xFFFF, 0xFFFF, 0x528A, 0x0000,
0x0000, 0x0000, 0x39E7, 0x0000, 0x0000, 0x632C, 0x2124,   // 0x0BC0 (3008)
pixels
0x0000, 0x4228, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0BD0 (3024)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0BE0 (3040)
pixels
0xFFFF, 0xBDD7, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x2945, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0BF0 (3056)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0x0020, 0x0000,
0x0000, 0x0000, 0x4208, 0x5AEB, 0x0000, 0x2104, 0xF79E,   // 0x0C00 (3072)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC638, 0x0020, 0x0000, 0x0000, 0x4A49,
0xAD75, 0x73AE, 0x0000, 0x0000, 0x8C71, 0xFFFF, 0xFFFF,   // 0x0C10 (3088)
pixels
0xFFFF, 0xB596, 0x0000, 0x3186, 0x0861, 0x0000, 0x0000, 0x0000, 0x0020,
0xC638, 0xFFFF, 0xFFFF, 0xFFFF, 0xAD75, 0x0000, 0x0000,   // 0x0C20 (3104)
pixels
0x73AE, 0xFFFF, 0xDEDB, 0x18C3, 0x0020, 0xDEDB, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0x4A49, 0x0000, 0x0000, 0x18E3, 0x3186, 0x0000,   // 0x0C30 (3120)
pixels
0x0000, 0x0000, 0x94B2, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x52AA,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0C40 (3136)
pixels
0x5ACB, 0xFFFF, 0xFFFF, 0xA534, 0x0861, 0x0000, 0x0000, 0x0020, 0x9CF3,
0x94B2, 0x0000, 0x0000, 0x52AA, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0C50 (3152)
pixels
0x9492, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x8C71, 0xA514, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0C60 (3168)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x9492, 0xFFFF, 0xFFFF, 0xFFFF,
0x4A69, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0C70 (3184)
pixels
```

```
0x0000, 0x73AE, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDF7, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0C80 (3200)
pixels
0x9CD3, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0C90 (3216)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x9CF3,   // 0x0CA0 (3232)
pixels
0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0xDEDB,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0CB0 (3248)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x8C51, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x2965, 0xFFDF, 0xFFFF, 0xFFFF,   // 0x0CC0 (3264)
pixels
0xFFFF, 0xFFFF, 0xB5B6, 0x0000, 0x0000, 0x0000, 0xC638, 0xFFFF, 0xF79E,
0x2104, 0x2965, 0xE73C, 0xFFFF, 0xFFFF, 0xFFDF, 0xAD55,   // 0x0CD0 (3280)
pixels
0x1082, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4228, 0x7BEF,
0x52AA, 0x632C, 0x4228, 0x0000, 0x0000, 0x0020, 0x39C7,   // 0x0CE0 (3296)
pixels
0x1082, 0x0000, 0x18E3, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDD7,
0x0020, 0xBDD7, 0xFFFF, 0xE73C, 0x10A2, 0x0000, 0x0000,   // 0x0CF0 (3312)
pixels
0x7BCF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0x18C3, 0x528A, 0xE73C,
0x2965, 0x0000, 0x0000, 0x0000, 0x1082, 0xE71C, 0xFFFF,   // 0x0D00 (3328)
pixels
0xFFFF, 0xC638, 0x0020, 0x0000, 0x0000, 0x18C3, 0xFFDF, 0xFFDF, 0x2104,
0x1082, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0x52AA, 0x0000,   // 0x0D10 (3344)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0x0020, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x1082, 0x738E, 0x738E, 0x2945,   // 0x0D20 (3360)
pixels
0x0000, 0x0000, 0x0000, 0x73AE, 0xFFFF, 0xFFFF, 0xD69A, 0x0841, 0x0000,
0x0000, 0x0020, 0xAD75, 0x0020, 0x0000, 0x0000, 0x94B2,   // 0x0D30 (3376)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xF7BE, 0xAD75, 0x18C3, 0x0000, 0x0000, 0x0000,
0x39E7, 0xAD55, 0x0841, 0x0000, 0x31A6, 0xF7BE, 0xFFFF,   // 0x0D40 (3392)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0D50 (3408)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xD6BA, 0x1082, 0x0000,   // 0x0D60 (3424)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x3186, 0x0841, 0xD69A, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0D70 (3440)
pixels
```

```
0xFFFF, 0xFFFF, 0xF79E, 0x0861, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000,
0x0000, 0x10A2, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0D80 (3456)
pixels
0xBDF7, 0x0000, 0x0000, 0x0000, 0x7BCF, 0xAD55, 0x52AA, 0x0020, 0x2124,
0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0x5ACB, 0x0000, 0x0000,   // 0x0D90 (3472)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x0861,   // 0x0DA0 (3488)
pixels
0x0000, 0x73AE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0x31A6, 0x0000, 0x8C71,
0xFFFF, 0xFFFF, 0x7BCF, 0x0000, 0x0000, 0xA514, 0xFFFF,   // 0x0DB0 (3504)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0x18C3, 0x31A6, 0xF79E, 0x4208, 0x0000,
0x0000, 0x0000, 0x0000, 0x9CD3, 0xFFFF, 0xFFFF, 0xC638,   // 0x0DC0 (3520)
pixels
0x0020, 0x0000, 0x0000, 0x0841, 0x8C71, 0xAD55, 0x0000, 0x0000, 0x8C71,
0xFFFF, 0xFFFF, 0xFFFF, 0xAD55, 0x0000, 0x2945, 0x2104,   // 0x0DD0 (3536)
pixels
0x0000, 0x0000, 0x0000, 0x39C7, 0xD69A, 0x632C, 0x0000, 0x0000, 0x0000,
0x0000, 0x2945, 0xFFFF, 0xFFFF, 0xEF7D, 0x39E7, 0x0000,   // 0x0DE0 (3552)
pixels
0x0000, 0x6B6D, 0xFFFF, 0xFFFF, 0xFFFF, 0x4A69, 0x31A6, 0x0000, 0x0000,
0x0020, 0x0000, 0x0000, 0x0000, 0x8C51, 0xFFFF, 0xFFFF,   // 0x0DF0 (3568)
pixels
0xFFFF, 0xFFFF, 0xEF7D, 0x3186, 0x0000, 0x0000, 0x2965, 0xFFFF, 0xFFFF,
0x6B6D, 0x0000, 0x10A2, 0xDEFB, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0E00 (3584)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFDF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0E10 (3600)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xAD75, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0E20 (3616)
pixels
0x0000, 0x18E3, 0xD69A, 0x2104, 0x73AE, 0xD6BA, 0xEF7D, 0xC618, 0xAD75,
0xDEFB, 0xEF5D, 0xFFFF, 0xE71C, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0E30 (3632)
pixels
0xBDF7, 0x0000, 0x5ACB, 0xE71C, 0x4A49, 0x0000, 0x0000, 0x0000, 0x0000,
0x8430, 0xF79E, 0xFFFF, 0xFFFF, 0xEF5D, 0x5ACB, 0x0000,   // 0x0E40 (3648)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2965, 0xF7BE, 0xFFFF,
0xFFFF, 0xCE59, 0x2945, 0x0000, 0x0000, 0x0000, 0x4228,   // 0x0E50 (3664)
pixels
0xD69A, 0x1082, 0x0000, 0x0000, 0x0000, 0x0000, 0x1082, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0000, 0x52AA,   // 0x0E60 (3680)
pixels
0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0x9492, 0x0000, 0x18E3, 0xDEDB, 0xFFFF,
0x5ACB, 0x0000, 0x0000, 0xA534, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0E70 (3696)
pixels
```

```
0xD6BA, 0x4A69, 0x0000, 0x0000, 0x18E3, 0x0000, 0x0000, 0x0000, 0x0000,
0x2124, 0xCE79, 0xFFFF, 0xFFFF, 0xEF5D, 0x18C3, 0x8430,   // 0x0E80 (3712)
pixels
0x0020, 0x0000, 0x0000, 0x0020, 0x0000, 0x10A2, 0xEF5D, 0xFFFF, 0xFFFF,
0xFFFF, 0x52AA, 0x0000, 0x0000, 0x1082, 0x9CF3, 0x6B6D,   // 0x0E90 (3728)
pixels
0x0000, 0x18E3, 0xF7BE, 0xD69A, 0x4228, 0x0000, 0x0000, 0x0000, 0x2965,
0xFFFF, 0xFFFF, 0xE71C, 0x5ACB, 0x0000, 0x0000, 0x0841,   // 0x0EA0 (3744)
pixels
0xD69A, 0xFFFF, 0xC618, 0x0861, 0x10A2, 0x0000, 0x3186, 0x0841, 0x528A,
0x0000, 0x0020, 0x0020, 0x9CF3, 0xFFFF, 0xFFFF, 0xEF7D,   // 0x0EB0 (3760)
pixels
0xD6BA, 0x0861, 0x0000, 0x0000, 0x2104, 0xFFFF, 0xFFFF, 0x52AA, 0x0000,
0x1082, 0xB5B6, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7BE, 0x738E,   // 0x0EC0 (3776)
pixels
0x9CD3, 0xF7BE, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x73AE, 0x73AE,
0xB596, 0x2965, 0xD6BA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0ED0 (3792)
pixels
0xE73C, 0x31A6, 0x9CD3, 0xCE79, 0x6B6D, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF,
0xEF7D, 0x18C3, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0EE0 (3808)
pixels
0x0020, 0x0000, 0x0020, 0x0841, 0x39E7, 0x2104, 0x0861, 0x2965, 0x18E3,
0x528A, 0x0841, 0x7BCF, 0xF7BE, 0xFFFF, 0xC638, 0x1082,   // 0x0EF0 (3824)
pixels
0x4A69, 0xFFDF, 0x8430, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x630C,
0x6B6D, 0xDEFB, 0x632C, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0F00 (3840)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x9CD3, 0xFFFF, 0xFFFF, 0xFFFF, 0x9CF3,
0x0000, 0x0000, 0x0000, 0x0000, 0x31A6, 0xBDF7, 0x18E3,   // 0x0F10 (3856)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0xAD55, 0x52AA, 0x0000, 0x0020, 0x8C51,
0xBDD7, 0x18E3, 0x0000, 0x0000, 0x0000, 0xB596, 0xFFFF,   // 0x0F20 (3872)
pixels
0xFFFF, 0xFFFF, 0x8410, 0x0000, 0x0000, 0x0841, 0x738E, 0x18E3, 0x0000,
0x0000, 0x31A6, 0x9492, 0x8410, 0x8410, 0x31A6, 0x0000,   // 0x0F30 (3888)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x2965, 0x9CD3, 0x0020, 0x94B2, 0xFFFF,
0xFFFF, 0xFFFF, 0x7BEF, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0F40 (3904)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0xB5B6, 0xFFFF, 0xFFFF, 0xFFFF, 0xAD55,
0x0000, 0x0000, 0x73AE, 0xFFFF, 0xA514, 0x0000, 0x39C7,   // 0x0F50 (3920)
pixels
0xFFFF, 0xFFFF, 0xAD75, 0x0000, 0x0000, 0x0000, 0x0020, 0x2104, 0x4A49,
0x1082, 0x0000, 0x0000, 0x0000, 0xA514, 0xFFFF, 0xFFFF,   // 0x0F60 (3936)
pixels
0xFFFF, 0xB5B6, 0x0000, 0x0000, 0x0861, 0x0000, 0x18E3, 0x0000, 0x18E3,
0x0000, 0x4A49, 0x4A69, 0xA534, 0x52AA, 0x0861, 0x0841,   // 0x0F70 (3952)
pixels
```

```
0x0000, 0x0000, 0x0020, 0x39C7, 0x528A, 0x0000, 0x0000, 0xA514, 0xFFFF,
0xFFFF, 0xFFFF, 0x4228, 0x10A2, 0x0000, 0x0000, 0x10A2,   // 0x0F80 (3968)
pixels
0x1082, 0xBDD7, 0xFFFF, 0xFFFF, 0xA514, 0x0020, 0x0000, 0x2104, 0x2104,
0x1082, 0x632C, 0xFFFF, 0xFFFF, 0xEF5D, 0x3186, 0x0000,   // 0x0F90 (3984)
pixels
0x10A2, 0x0841, 0x0000, 0x18C3, 0xA514, 0xFFFF, 0xFFFF, 0x73AE, 0x0000,
0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000,   // 0x0FA0 (4000)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0xAD55, 0xFFFF, 0xFFFF, 0x8C51, 0x0000, 0x39E7,   // 0x0FB0 (4016)
pixels
0x18C3, 0x0000, 0x0000, 0x18C3, 0x0841, 0x0000, 0x0000, 0x0000, 0x0841,
0x0000, 0x0000, 0x5ACB, 0x8430, 0x0000, 0x0000, 0x0000,   // 0x0FC0 (4032)
pixels
0x0000, 0x0020, 0x5ACB, 0xFFFF, 0xFFFF, 0xFFFF, 0x8C71, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0FD0 (4048)
pixels
0x0000, 0x0000, 0x3186, 0x1082, 0x0000, 0x632C, 0xFFFF, 0xFFFF, 0x9CF3,
0x0000, 0x0000, 0x0000, 0xA514, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0FE0 (4064)
pixels
0xAD75, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2965,   // 0x0FF0 (4080)
pixels
0x0861, 0x0000, 0x9CF3, 0xCE79, 0x0861, 0xB596, 0xFFFF, 0xFFFF, 0xFFFF,
0xE73C, 0x10A2, 0x0000, 0x0000, 0x10A2, 0x52AA, 0x0020,   // 0x1000 (4096)
pixels
0x0000, 0x0000, 0xA514, 0xFFFF, 0xFFFF, 0xFFFF, 0x630C, 0x0000, 0x0000,
0x10A2, 0x4208, 0x0841, 0x0000, 0x2104, 0xDEFB, 0xFFFF,   // 0x1010 (4112)
pixels
0xF7BE, 0x7BEF, 0x0000, 0x0841, 0x1082, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x6B6D, 0xFFFF, 0xFFFF, 0xFFFF, 0x9492,   // 0x1020 (4128)
pixels
0x0000, 0x8C71, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x1030 (4144)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x2104, 0xC618, 0xFFFF, 0xFFFF, 0xEF7D,
0x10A2, 0x0000, 0x18E3, 0x8C51, 0x5ACB, 0x0000, 0x3186,   // 0x1040 (4160)
pixels
0xFFFF, 0xFFFF, 0x4A49, 0x0000, 0x39C7, 0xE71C, 0xEF7D, 0x630C, 0x0020,
0xEF5D, 0xFFFF, 0xC618, 0x0020, 0x0000, 0x6B6D, 0x0000,   // 0x1050 (4176)
pixels
0x0000, 0x0000, 0x4A69, 0xFFFF, 0xFFFF, 0xDEDB, 0x1082, 0x0000, 0x0000,
0x0000, 0x5AEB, 0x0020, 0x0000, 0x0000, 0x0000, 0x1082,   // 0x1060 (4192)
pixels
0x10A2, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4A69, 0x1082, 0x73AE,
0xFFDF, 0xFFFF, 0x8C71, 0x2965, 0x0000, 0x0000, 0x0000,   // 0x1070 (4208)
pixels
```

```
0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x528A, 0x31A6, 0x0000, 0x0000,
0x0861, 0x2104, 0x0000, 0x39C7, 0x9CD3, 0x0000, 0x4228,   // 0x1080 (4224)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xD69A, 0x0000, 0x0000, 0x3186, 0xA534, 0x0000,
0x3186, 0x0861, 0x0000, 0x18E3, 0x39E7, 0x0000, 0x0000,   // 0x1090 (4240)
pixels
0x0000, 0x0000, 0x0000, 0x52AA, 0xC618, 0xCE59, 0x2945, 0x0000, 0x0000,
0x0000, 0x0000, 0xB5B6, 0xFFFF, 0xFFFF, 0x6B6D, 0x0000,   // 0x10A0 (4256)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x39E7, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x10B0 (4272)
pixels
0x0861, 0x0020, 0x8C71, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF5D, 0x0861,
0x0000, 0x0000, 0x7BCF, 0xFFFF, 0x632C, 0x1082, 0x2104,   // 0x10C0 (4288)
pixels
0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0x6B6D, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x2945, 0xE73C, 0xFFFF, 0xFFFF, 0xFFDF,   // 0x10D0 (4304)
pixels
0x1082, 0x2124, 0x31A6, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000,
0x2965, 0xF7BE, 0xFFFF, 0xFFFF, 0xE71C, 0x6B6D, 0x2124,   // 0x10E0 (4320)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x2945, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x10F0 (4336)
pixels
0x0000, 0x0000, 0x528A, 0xFFFF, 0xFFFF, 0xFFFF, 0x9492, 0x0000, 0x0020,
0xD6BA, 0xFFFF, 0xEF7D, 0x2945, 0x10A2, 0xF79E, 0xFFFF,   // 0x1100 (4352)
pixels
0x5AEB, 0x39C7, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0x73AE, 0x738E, 0xFFFF,
0xCE59, 0x0841, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x1110 (4368)
pixels
0x0000, 0xAD55, 0xFFFF, 0xA514, 0x0020, 0x0000, 0x0000, 0x0000, 0x3186,
0x4228, 0x0000, 0x0000, 0x4A69, 0xFFFF, 0xD69A, 0x0000,   // 0x1120 (4384)
pixels
0x0000, 0x0000, 0x0000, 0x0841, 0xC618, 0x1082, 0x7BCF, 0xFFFF, 0xFFFF,
0xFFFF, 0x630C, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x1130 (4400)
pixels
0x0000, 0x0000, 0x0000, 0x18C3, 0x18C3, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0020, 0x0861, 0x3186, 0xD6BA, 0xFFFF, 0xFFFF,   // 0x1140 (4416)
pixels
0xFFFF, 0xB596, 0x0020, 0x0000, 0x0020, 0x2945, 0x0000, 0x0020, 0x0000,
0x0000, 0x632C, 0x6B4D, 0x7BEF, 0x8C71, 0x2945, 0x8430,   // 0x1150 (4432)
pixels
0x0861, 0x1082, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18E3, 0x0020,
0xDEFB, 0xFFFF, 0xFFFF, 0xBDF7, 0x0000, 0x0000, 0x0000,   // 0x1160 (4448)
pixels
0xA514, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x31A6,   // 0x1170 (4464)
pixels
```

```
0x8430, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDF7, 0x0000, 0x0000, 0x0000,
0x18E3, 0xB5B6, 0x4208, 0x0020, 0x0020, 0x94B2, 0xFFFF,   // 0x1180 (4480)
pixels
0xFFFF, 0xFFFF, 0x8410, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x2104, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDF7, 0x0020,   // 0x1190 (4496)
pixels
0x4228, 0xA534, 0x0000, 0x3186, 0xF7BE, 0x0861, 0x0000, 0xBDD7, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x18C3, 0x0000, 0x0000,   // 0x11A0 (4512)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0xEF7D,
0x7BEF, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2965,   // 0x11B0 (4528)
pixels
0x8C71, 0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0x0841, 0x0000, 0x4A49, 0xFFFF,
0xF7BE, 0x2104, 0x52AA, 0xFFFF, 0xFFFF, 0x4228, 0x0841,   // 0x11C0 (4544)
pixels
0xCE59, 0xFFFF, 0xFFFF, 0xDEFB, 0x1082, 0xA534, 0xFFFF, 0xBDD7, 0x0020,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2965, 0xF7BE,   // 0x11D0 (4560)
pixels
0xFFFF, 0xD69A, 0x2104, 0x0000, 0x0000, 0x0000, 0x4A69, 0x73AE, 0x0000,
0x0000, 0x73AE, 0xFFFF, 0xDEFB, 0x0020, 0x0000, 0x0000,   // 0x11E0 (4576)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x52AA, 0xE73C, 0xFFFF, 0xFFFF, 0xCE79,
0xAD75, 0x2965, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x11F0 (4592)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x4A49, 0x8C51, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x738E,   // 0x1200 (4608)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x630C, 0xCE59,
0xFFFF, 0xFFFF, 0xFFFF, 0xF79E, 0xFFFF, 0xCE79, 0x31A6,   // 0x1210 (4624)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0xF7BE, 0x8410, 0x4A69, 0xEF5D,
0xFFFF, 0x52AA, 0x0000, 0x0000, 0x0000, 0x10A2, 0x0000,   // 0x1220 (4640)
pixels
0x0000, 0x0000, 0x0000, 0x6B6D, 0x4208, 0x1082, 0x1082, 0x0000, 0x0000,
0x0000, 0x0000, 0x0841, 0x5AEB, 0xA534, 0xFFFF, 0xFFFF,   // 0x1230 (4656)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x39C7, 0x0000, 0x1082, 0x5ACB, 0x0020,
0x2104, 0x0000, 0x2104, 0xF7BE, 0xFFFF, 0xFFFF, 0xF7BE,   // 0x1240 (4672)
pixels
0x738E, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0xEF7D,
0xFFFF, 0xFFFF, 0xFFFF, 0xE73C, 0x0861, 0x1082, 0x2124,   // 0x1250 (4688)
pixels
0x0000, 0x0861, 0x73AE, 0x0000, 0x0000, 0x8C51, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0x5ACB, 0x4228, 0x0020, 0x0000, 0x0000,   // 0x1260 (4704)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x0020, 0x0000,
0x0000, 0x0000, 0x0000, 0x4A69, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1270 (4720)
pixels
```

```
0xFFFF, 0xFFFF, 0x7BEF, 0x0841, 0x0000, 0x0000, 0x2965, 0x528A, 0x0000,
0x1082, 0xEF5D, 0xF79E, 0x2945, 0x0000, 0x2104, 0xB596,   // 0x1280 (4736)
pixels
0xAD75, 0x2965, 0x0000, 0x8C71, 0xFFFF, 0xDEFB, 0x0841, 0x0841, 0x0000,
0x0000, 0x632C, 0x7BCF, 0x1082, 0xE73C, 0xFFFF, 0xFFFF,   // 0x1290 (4752)
pixels
0x9CD3, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020,
0x4228, 0x18E3, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x12A0 (4768)
pixels
0x0000, 0x4208, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618,
0x738E, 0x18E3, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x12B0 (4784)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x39C7, 0x4228, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xE73C, 0x0020, 0x0000,   // 0x12C0 (4800)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x18C3, 0x9CD3, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x5AEB, 0x0000,   // 0x12D0 (4816)
pixels
0x0000, 0x0000, 0x0020, 0x630C, 0x18E3, 0x2124, 0xFFDF, 0xFFFF, 0xBDD7,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x12E0 (4832)
pixels
0x0000, 0x632C, 0x18E3, 0x0000, 0x0000, 0x0000, 0x0000, 0x2945, 0x2124,
0x738E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x12F0 (4848)
pixels
0xFFFF, 0xFFFF, 0xB5B6, 0x0000, 0x0000, 0x18E3, 0x0000, 0x0861, 0x0000,
0x2124, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x2945,   // 0x1300 (4864)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x8C51, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xEF7D, 0x2965, 0x0000, 0x0000, 0x0000,   // 0x1310 (4880)
pixels
0x0000, 0x0000, 0x0841, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0x4228, 0x0841, 0x4228, 0x0000, 0x0000,   // 0x1320 (4896)
pixels
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x630C,
0x73AE, 0xBDF7, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1330 (4912)
pixels
0xFFFF, 0x8C51, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x2104, 0xF79E,
0xFFFF, 0xA514, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x1340 (4928)
pixels
0x0000, 0xAD75, 0xFFFF, 0xDEDB, 0x0841, 0x31A6, 0x0020, 0x0000, 0x9492,
0x9492, 0x2945, 0xF79E, 0xFFFF, 0xFFFF, 0xEF7D, 0x4A69,   // 0x1350 (4944)
pixels
0x4228, 0x7BEF, 0x3186, 0x31A6, 0x9492, 0x10A2, 0x8410, 0x52AA, 0x39C7,
0x9492, 0x18E3, 0x73AE, 0x4A49, 0x2965, 0xBDF7, 0xF79E,   // 0x1360 (4960)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xBDD7,
0x39E7, 0x7BEF, 0x0841, 0x0861, 0x4208, 0x0020, 0x39E7,   // 0x1370 (4976)
pixels
```

```
0x0841, 0x6B6D, 0x8410, 0xE73C, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0x4208, 0x632C, 0x528A, 0x39C7,   // 0x1380 (4992)
pixels
0x7BEF, 0x2965, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0xB596, 0x2104, 0x6B6D,   // 0x1390 (5008)
pixels
0x52AA, 0x39C7, 0x8C71, 0x8410, 0xFFFF, 0xFFFF, 0xFFFF, 0x630C, 0x39C7,
0x528A, 0x8410, 0x31A6, 0x8410, 0x18E3, 0x8C51, 0x31A6,   // 0x13A0 (5024)
pixels
0x7BCF, 0x0841, 0x8C71, 0x5AEB, 0x3186, 0xEF7D, 0xFFFF, 0xFFDF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x13B0 (5040)
pixels
0xFFFF, 0x31A6, 0x5AEB, 0x4208, 0x6B4D, 0x5ACB, 0x5AEB, 0xEF7D, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x8C71, 0x5AEB, 0x4A49,   // 0x13C0 (5056)
pixels
0x31A6, 0x8C71, 0x94B2, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xCE59, 0xA534, 0x4A69, 0x9492, 0x39C7, 0xA534,   // 0x13D0 (5072)
pixels
0xBDD7, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xDEFB, 0xE73C, 0x7BCF, 0x6B6D, 0x1082, 0x4228, 0x39C7,   // 0x13E0 (5088)
pixels
0x0020, 0x0000, 0x39E7, 0x10A2, 0xA534, 0xA534, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7BE,   // 0x13F0 (5104)
pixels
0x31A6, 0x10A2, 0x1082, 0x39E7, 0x10A2, 0xB5B6, 0xFFFF, 0xFFFF, 0xFFFF,
0x8410, 0x0000, 0x2945, 0x0020, 0x4A69, 0x8C51, 0xFFFF,   // 0x1400 (5120)
pixels
0xFFFF, 0xFFFF, 0x94B2, 0x0000, 0x2104, 0x0841, 0x2124, 0x2945, 0xA534,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xF7BE, 0xFFFF,   // 0x1410 (5136)
pixels
0xF7BE, 0xFFDF, 0xFFFF, 0xEF5D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF7D,
0xFFFF, 0xFFFF, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1420 (5152)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xAD75, 0xD69A, 0xFFFF, 0xD69A, 0xFFDF, 0xD69A, 0xF7BE,   // 0x1430 (5168)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xE71C, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xF7BE,   // 0x1440 (5184)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1450 (5200)
pixels
0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xEF5D, 0xDEDB, 0xFFFF, 0xFFFF,
0xFFDF, 0xFFFF, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB,   // 0x1460 (5216)
pixels
0xFFFF, 0xFFFF, 0xF7BE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xD6BA,   // 0x1470 (5232)
pixels
```

```
0xFFDF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xEF5D, 0xFFDF, 0xFFFF,   // 0x1480 (5248)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1490 (5264)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xF79E, 0xFFFF, 0xD69A, 0xFFDF, 0xFFFF, 0xD6BA, 0x9CF3,   // 0x14A0 (5280)
pixels
0xFFDF, 0xCE79, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xCE79, 0xDEDB,   // 0x14B0 (5296)
pixels
0xD69A, 0xFFFF, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0x8C71,
0xEF7D, 0xAD75, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x14C0 (5312)
pixels
0xFFFF, 0x8410, 0xE71C, 0xCE59, 0xEF5D, 0xF7BE, 0xFFDF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x14D0 (5328)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x14E0 (5344)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x14F0 (5360)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1500 (5376)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1510 (5392)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1520 (5408)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1530 (5424)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1540 (5440)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1550 (5456)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1560 (5472)
pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1570 (5488)
pixels
```

```
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x1580 (5504)
pixels
};
```