GT-SP user guide 3 (Host Control Edition)

Ver.e00

1 Introduction

◆This document is **an introductory guide for first-time developers** of our **GT-SP** (LCD Touch Screen Modules GUI control type).

◆The GT-SP can be connected to most major microcontrollers through a serial I/F. This section describes the basics of controlling the standard 7-inch GT-SP model (GTWV070S3A00P) with the **Arduino Nano Every** as the host controller.

◆As a prerequisite, this document is intended for those who understand the basic usage of GT Design Studio and can connect circuits and do programming using Arduino.

Contents

1.1 What to prepare

- GT-SP (This document assumes that the 7-inch size standard product **GTWV070S3A00P** is used)
- Arduino Nano Every
- USB cable
- Soldering set
- PC (OS: Windows 10 / 11 recommended)
- Connection board (e.g., a breadboard)
- DC power supply (5V), power cables

*Please **download the latest versions of the following software from our website** (free of charge).

- GT Design Studio (design tool provided by us).
- GT Design Studio sample projects, Arduino sample programs.

  [North America / EU] https://www.noritake-elec.com/support/

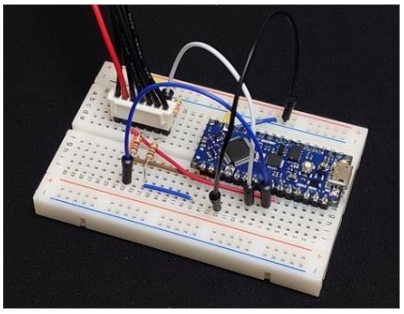
  [Asia / Other] https://www.noritake-itron.jp/eng/cs/soft/42

2 Connecting to a host controller

◆Select the I/F and baud rate by switching the jumpers on the back of the board. The **SBUSY signal** reflects the status of the receive buffer, while the **TRDY signal** indicates whether data is currently held in the transmit buffer. For more detailed information, please refer to the GT-SP hardware specifications.

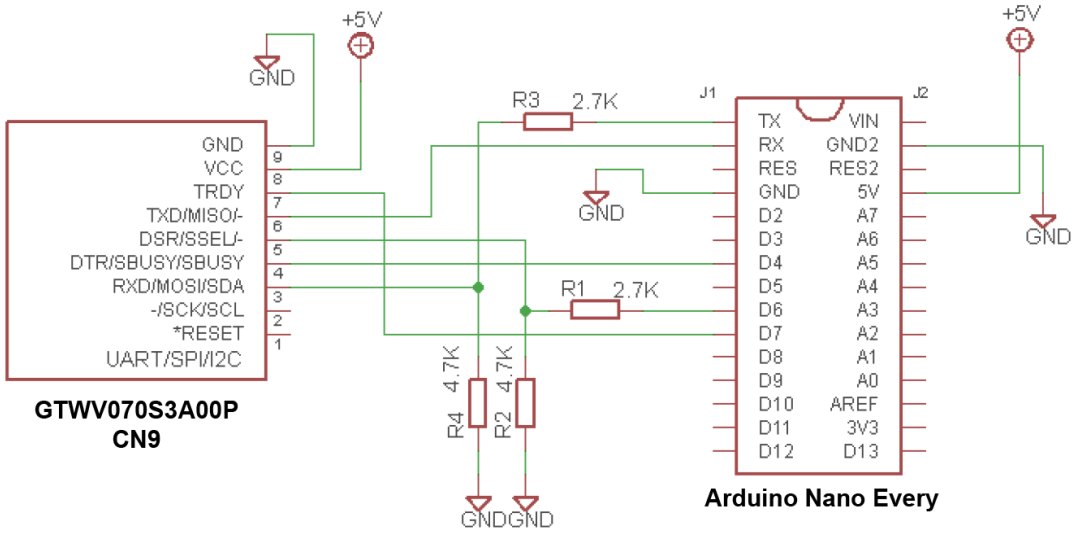
◆This section presents an example of connecting to the GT-SP using a UART connection (38,400 bps, no reset). Connect the host controller to the GT-SP using a breadboard or similar device, as in the following circuit diagram.

2.1 Connection schematic example for a UART connection (without reset).



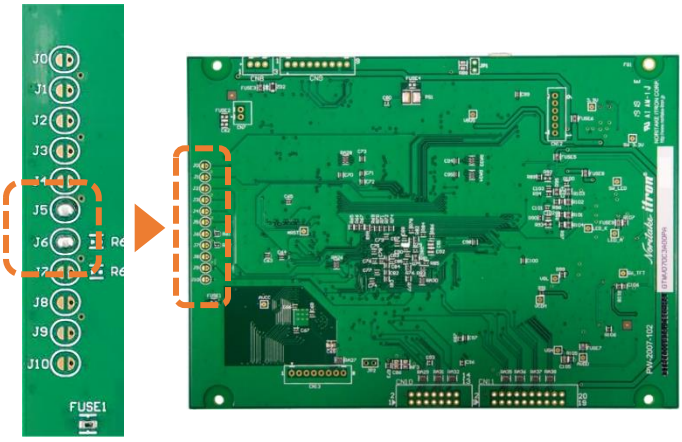
◆When connecting to a host controller that operates at 5V, such as the Arduino Nano Every, it is necessary to convert the signal level to match the 3.3V input on the display unit.

◆In this case, the level conversion is done by dividing the voltage with a resistor.



2.2 I/F settings on the GT-SP side (jumper settings)

◆ Set the jumpers on the GT-SP to **switch the interface settings**. Please refer to the hardware specifications for the part number being used and set the jumper.



◆For GTWV070S3A00P, only solder and shortcircuit J5 and J6 to switch to UART (38,400 bps).

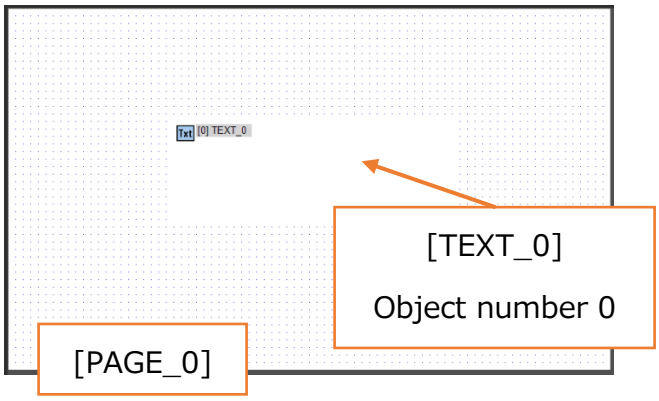
3 Sending control commands

◆The GT-SP can be controlled by **sending commands from the host**.

◆In this section, "**Hello World**" is displayed in a text object. The display brightness repeatedly switches back and forth from less luminant to full luminance.

3.1 Project preparation

◆In GT Design Studio, create the following project (GT Design Studio sample project: **sample1.gud**) and register it to the GT-SP.



3.2 Program operation

◆Register and run the following program (Arduino sample program: **arduino_sample1.ino**) in Arduino.

```
#include <string.h>
//ピン接続 | Pin assign
#define GT_DTR      4       //DTR
#define GT_DSR      6       //DSR
#define GT_TRDY     7       //TRDY <--未使用 | Disused

void setup()
{
    //ピン初期設定 | Pin Initialization
    pinMode(GT_DTR, INPUT);
    pinMode(GT_DSR, OUTPUT);
    pinMode(GT_TRDY, INPUT);
    digitalWrite(GT_DSR, LOW);

    //Serial setting
    Serial1.begin(38400);    //38400bps Baud rate
    Serial1.setTimeout(100); //シリアルタイムアウト | Serial timeout
}

void loop()
{
  //オブジェクト制御コマンド-プロパティ設定 | Object Control Command - Property Settings
  Serial1.print("CMD");   //コマンドヘッダ | Command header
  Serial1.write(0xD3);    //オブジェクト-プロパティ設定 | Object-Property Setting
  Serial1.write(0x00);    //オブジェクトNo. 下位バイト | Object No. Lower byte
  Serial1.write(0x00);    //オブジェクトNo. 上位バイト| Object No. Upper byte
  Serial1.write(0x40);    //プロパティNo. 下位バイト | Property No. Lower byte
  Serial1.write(0x00);    //プロパティNo. 上位バイト| Property No. Upper byte
  Serial1.write(0x0B);    //データ長 最下位バイト | Data length Least significant byte
  Serial1.write(0x00);    //データ長 下位バイト | Data length second byte
  Serial1.write(0x00);    //データ長 上位バイト | Data length third byte
  Serial1.write(0x00);    //データ長 最上位バイト| Data length Most significant byte
  Serial1.print("Hello World"); //データ | Data

  //全体制御コマンド-輝度設定 (50%) | Control Command - Brightness Setting
  Serial1.print("CMD");   //コマンドヘッダ | Command header
  Serial1.write(0x58);    //輝度設定 | Brightness setting
  Serial1.write(0x80);    //輝度50% | Brightness 50%
  delay(500);

  //全体制御コマンド-輝度設定 (100%) | Control Command - Brightness Setting
  Serial1.print("CMD");  //コマンドヘッダ | Command header
  Serial1.write(0x58);   //輝度設定 | Brightness setting
  Serial1.write(0xFF);   //輝度100% | Brightness 100%
  delay(500);
}
```

3.3 Commands and program description

◆Users can perform numerous orders by sending commands to the GT-SP in the below format.

◆This section describes only the commands used in this sample program. Please refer to the GUI mode compatible commands in the software specifications for other commands.

Header	Command	Parameter
“CMD”	0x00 – 0xFF	0x00 – 0xFF
3 byte	1 byte	n byte

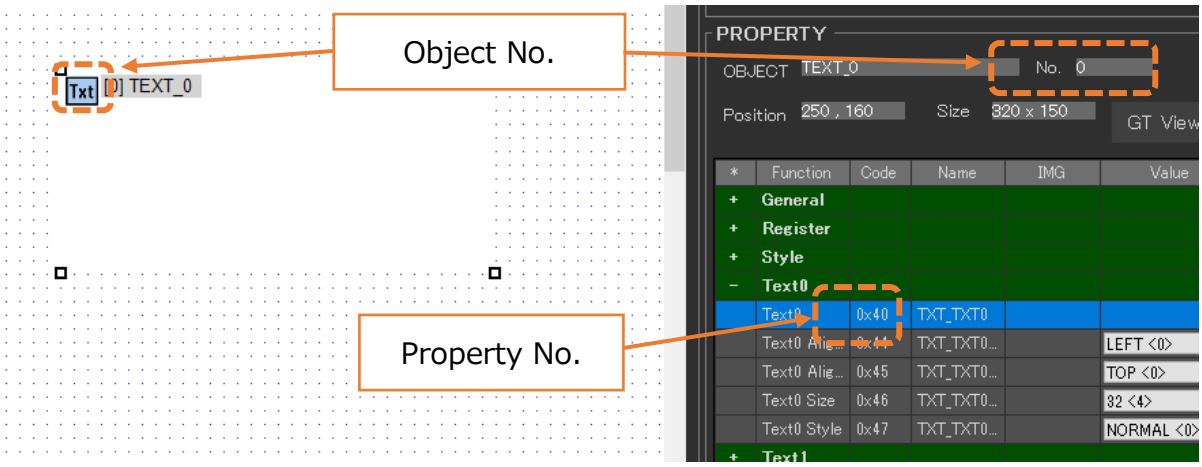
3.3.1 Object control commands - property settings

◆**This command changes the properties of objects** on the GT-SP. Users can specify the object and property numbers, and after sending the "data length (bytes)" and "data," the contents of the specified property will be changed. Users can also control objects' display contents, colors, and values.

◆In this example, a command to display " **Hello World** " (11-byte character string) is sent to the Text0 property (property No. 0x40) of the text object (object No. 0). Various numbers and data lengths must be sent in order from lower to upper bytes.

Header	Command	Parameter			
		Object No. (2byte) 0 = 0x00 00	**Property No. (2byte) 0x40 = 0x00 40**	**Data (4 bytes) 11 = 0x00 00 00 0B**	**Data**
“CMD”	0xD3	0x00, 0x00	0x40, 0x00	0x0B, 0x00, 0x00, 0x00	“Hello World”

◆The object and property numbers can also be found on the GT Design Studio screen in the following places:



3.3.2 Overall control command - display brightness setting

◆**This command sets the display brightness** of GT-SP. Sending a value from 0x00 (0% brightness) to 0xFF (100% brightness) will change the brightness of the backlight LED.

◆This example sends the command to blink — 50% (0x80) and 100% (0xFF) brightness every 500 ms.

Header	**Command**	**Parameter**
		Data (1byte) **0% = 0x00, 50% = 0x80, 100% = 0xFF**
“CMD”	0x58	0x80 or 0xFF

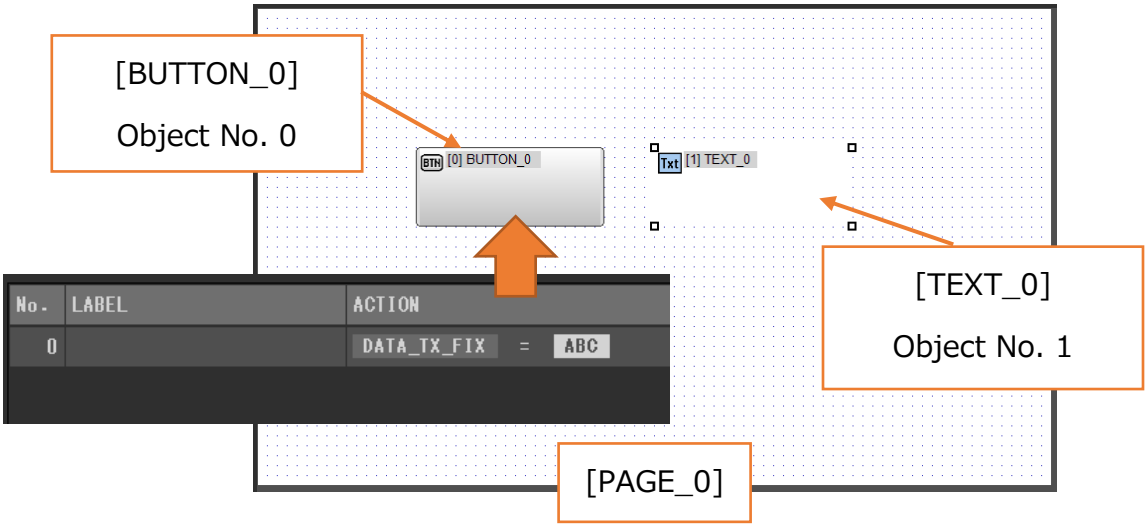
4 Receiving response data

◆The GT-SP sends data to the host **following touch events** or by **commands from the host**.

◆This section describes a program where the host receives the character string "ABC" sent from the GT-SP after a button touch event and then displays that character string in a text object on the GT-SP.

4.1 Project preparation

◆Create the following project (GT Design Studio sample project: **sample2.gud**) in GT Design Studio. Make the project output "ABC" when the button is pressed, and register it in the GT-SP.



4.2 Running the program

◆Register and execute the following program (Arduino sample program: **arduino_sample2.ino**) in Arduino. The declaration section is omitted.

```
//変数 | Variable
String gtsp_res_data;
bool screen_reflesh = false; //画面再描画フラグ | Display refresh flag

////////////////////////////////////////////////
// loop
////////////////////////////////////////////////
void loop()
{

  //GT-SP データ受信 | GT-SP data received
  if ( Serial1.available() ) {
    gtsp_res_data = gtsp_signal_read();
    screen_reflesh = true;
  }

  //画面描画 | Screen update
  if (screen_reflesh == true){
    gtsp_ObjPrpSet_string(1, 0x40, gtsp_res_data);
  }
  screen_reflesh = false;
}

/********************** 関数 | Function **********************/
////////////////////////////////////////////////
//  オブジェクト制御コマンド-プロパティ設定（文字列用） | Object Control Command -Property
Setting (String)
////////////////////////////////////////////////
void gtsp_ObjPrpSet_string(int obj, int prp, String val )
{
  gt_print("CMD");      //コマンドヘッダ | Command header
  gt_put(0xd3);         //オブジェクト-プロパティ設定コマンド | Object-Property Setting
  gt_put(obj >> 0);     //オブジェクトNo. 下位バイト | Object No. Lower byte
  gt_put(obj >> 8);     //オブジェクトNo. 上位バイト| Object No. Upper byte
  gt_put(prp >> 0);     //プロパティNo. 下位バイト | Property No. Lower byte
  gt_put(prp >> 8);     //プロパティNo. 上位バイト| Property No. Upper byte
  gt_put(val.length() >> 0);    //データ長 最下位バイト | Data length Least significant
byte
  gt_put(val.length() >> 8);    //データ長 下位バイト | Data length second byte
  gt_put(val.length() >> 16);   //データ長 上位バイト | Data length third byte
  gt_put(val.length() >> 24);   //データ長 最上位バイト| Data length Most significant
byte
  gt_print(val);        //文字列送信 | Send String to GT-SP
}

////////////////////////////////////////////////
//  1byte 送信 | Send byte to GT-SP
```

```
////////////////////////////////////////////////////
void gt_put(unsigned char onebyte)
{
  while ( digitalRead(GT_DTR) == HIGH ) {}  //busycheck
  Serial1.write(onebyte);
}

////////////////////////////////////////////////////
//  文字列送信 | Send String to GT-SP
////////////////////////////////////////////////////
void gt_print(String val)
{
  int val_i;
  //文字列を 1 文字ずつ送信 | Send string character by character
  for (val_i = 0; val_i < val.length(); val_i++)
  {
    while ( digitalRead(GT_DTR) == HIGH ) {}  //busycheck
    Serial1.print(val.substring(val_i, val_i+1));
  }
}

////////////////////////////////////////////////////
//  データ受信（文字列用） | Receive data from GT-SP (for String)
////////////////////////////////////////////////////
String gtsp_signal_read(){

  byte res_dl[4] = "";
  unsigned long dl;
  char res_data_char[255]="";

  if (Serial1.find("RESb", 4)){
    Serial1.readBytes(res_dl, 4); //データ長抽出 | data length extraction
    dl = (unsigned long)(res_dl[0] + (res_dl[1]<<8) + (res_dl[2]<<16) +
(res_dl[3]<<24)); //データ長変換 | data length conversion
    Serial1.readBytes(res_data_char, dl); //データ抽出 | data extraction
    return String(res_data_char); //String 型変換、リターン | String type conversion,
return
  }
}
```

◆In the following example, functions are used to send and receive data. A GT-SP busy check is performed when sending data from the host to prevent the GT-SP receiver buffer from overflowing.

4.3 Explanation of response data

◆The response data is sent from the GT-SP to the host in the following format. Here, we will explain **the "binary type" response** sent by this sample program. For other responses, please refer to the software specification.

Header	Type	Data
“RES”	0x00 – 0xFF	0x00 – 0xFF
3 byte	1 byte	n byte

◆When the GT-SP transmits binary data, such as the event actions "**DATA_TX_FIX**" or "**DATA_TX**," or by reading data by command, numerical or string data is sent from the GT-SP in the following "binary type" format.

Header	Type	Data	
		Data length (4byte)	Data
“RES”	“b”	0x04, 0x00, 0x00, 0x00	“ABC” & 0x00

◆In this sample project, pressing the button generates an event and executes the action "**DATA_TX_FIX = ABC**." At that time, 0x00 is added to the end of the data (character string "ABC"). The transmission data length is "4", and 4 bytes are transmitted sequentially from the lower byte. The result is sent to the host in the following format.

[User tip] The transmitted data can also be checked using **the Terminal function** in GT Design Studio.

4.4 Program description

◆In the sample program, **gtsp_signal_read** is called when the host receives serial data from GT-SP. Then, the GT-SP determines whether the data starts with "RESb" and obtains a data length of 4 bytes. It then gets the data for the data length and returns it as a string.

◆By sending object No., property No., and string data to **gtsp_ObjPrpSet_string**, an arbitrary string is displayed in the text object.

◆Data is sent to the GT-SP by performing a GT-SP busy check to prevent the GT-SP receive buffer overflow.

5 Header files

◆When a project is registered to the GT-SP in GT Design Studio, **a header file is automatically generated** in the folder where the project is saved.

5.1 IDX.h (page/object reference number definition)

◆All page and object numbers use the format "**XXXX_IDX.h**" set by GT Design Studio. Rather than referring to page numbers and object numbers by values in the host-side program, this format ensures the maintainability of project changes. //***** GUI Define / PAGE Infomation **********

```
//----- GUI_DEF_PG_0    "PAGE_0" --------------------
#define    GUI_DEF_PG_PAGE_0_NO    0

//***** GUI Define / OBJ Infomation **********

//----- GUI_DEF_OBJ_0    "BAR_0" --------------------
#define    GUI_DEF_OBJ_BAR_0_NO    0

//----- GUI_DEF_OBJ_1    "BAR_1" --------------------
#define    GUI_DEF_OBJ_BAR_1_NO    1
....
```

Page number

Object number

5.2 DEF.h (parameter/property value definitions)

◆The primary parameter values for all pages and objects are formatted as "**XXXX_DEF.h**." It should also be used to restore the primary host values.

5.3 CONFIG.h (parameter/property reference No. definition)

◆In "**XXXX_CONFIG.h**," the parameter No. for referencing parameters is designated as "**XXXX_CONFIG.h**." It should be used to specify the number when reading or writing parameters to and from the host.

6 Disclaimers and limitations

◆The information provided in this document does not guarantee it will work perfectly in all environments.

◆The sample code provided may be used in part or in its entirety only for use in our products. Additionally, it is the customer's responsibility to verify the operation of the final product.

◆The contents of this document may not be up-to-date due to ongoing development. Please check the website for the latest information.

◆Tools provided in the form of installers may contain licensed programs. Do not reverse engineer for analysis, decompilation, or disassembly.

◆All trademarks used in this guide are the property of their respective owners.

7 Contact information

Please contact us via the inquiry form if you have any questions or requests.

◆ [North America / EU] https://www.noritake-elec.com/support/contact-us

◆ [Asia / Other] https://www.noritake-itron.jp/eng/contact/