DWIN
ideal partner for you
Professional, Creditable, Successful
DWIN's High Quality HMI, The Best Solution for Industrial Field Applications!

DWIN DGUS Display Development Guide

Version 4.0

Revision on Jan,2014

Beijing DWIN Technology Co., Ltd.

INDEX

1. General Introduction

1.1 System Structure of DGUS

DGUS（**DWIN Graphic Utilized Software**） is a new cost effective GUI software platform developed by DWIN Technology. Based on the K600+ Kernel hardware platform, GUI design, combined with a simple command interface, can be achieved quickly, eliminating the need for complicated programming and expensive development environments.

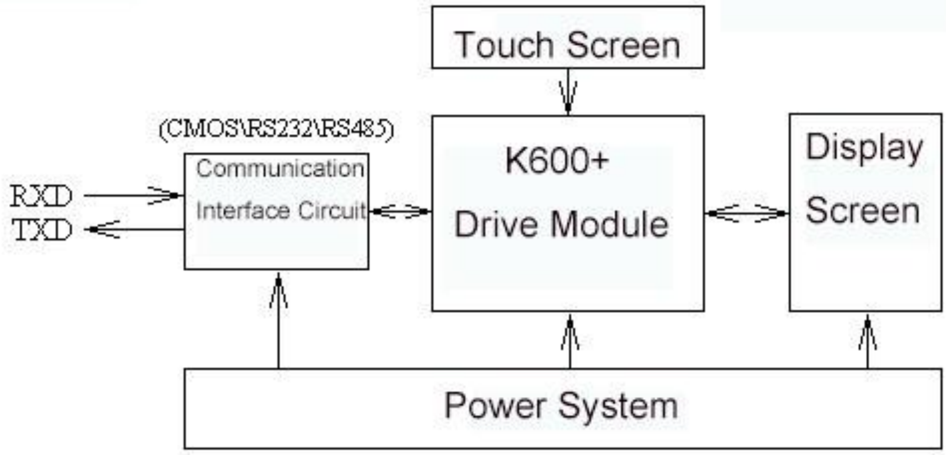


Figure 1 Hardware Description of DGUS

As shown in Figure 1, there are five parts as basic hardware structures for DGUS LCMs.

1) K600+ kernel: critical infrastructure;
2) Display: unit for displaying
3) Touch screen: in option.
4) DC/DC system: power supply for entire system
5) Communication interface: UART with 3 kinds of current level in option (CMOS, RS232 and RS485.

Scheme of DGUS Software is showing as below, Figure 2.

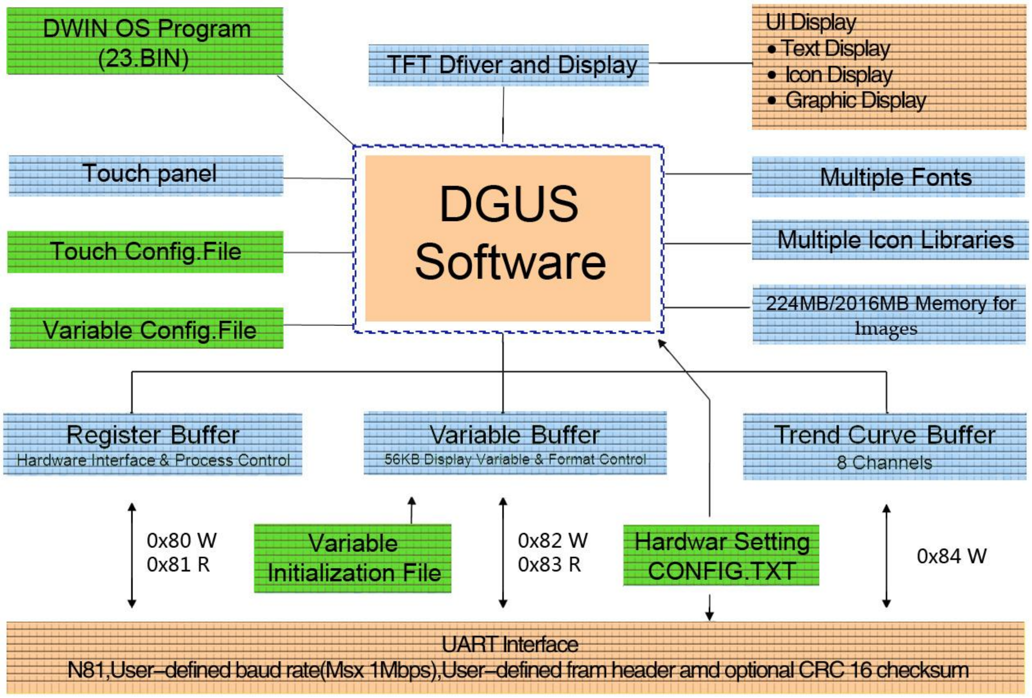
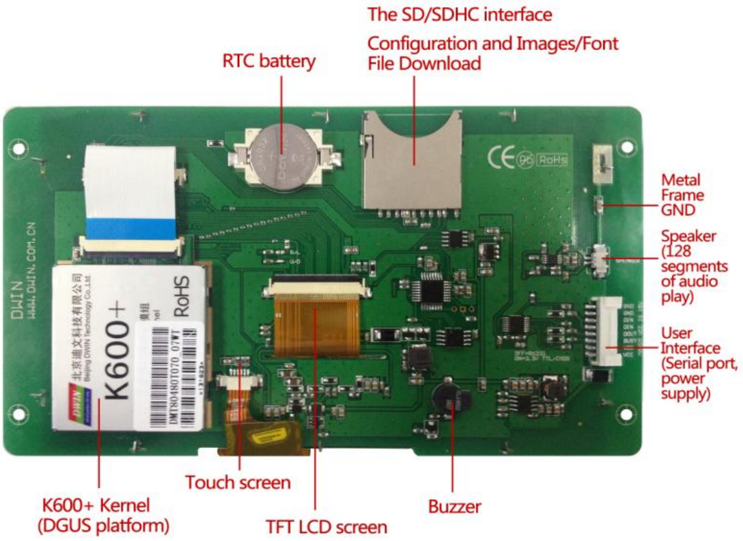


Figure2 Scheme of DGUS software

The display before delivery preinstalled by DGUS software was called DGUS display as default settings. Typically, a piece of DGUS display was composed as following (take DMT80480T070_07WT as example)


The SD/SDHC interface
Configuration and Images/Font File Download
RTC battery
Metal Frame GND
Speaker (128 segments of audio play)
User Interface (Serial port, power supply)
K600+ Kernel (DGUS platform)
Touch screen
TFT LCD screen
Buzzer

1.2 Features

There are key features of DGUS as below:

- GUI was broke up into widgets, configuring under pages. Each widget is displayed directly and controlled via variables.

  Users are required to read-and-write variables via serial port only to make consequential changes for widget.

  Example: Display a temperature value with two decimal places. Two steps requested:

  Step1 Configuration： Via PC software offered by DWIN Technology, users may add up a data variable on the page with format set up(font size, color, unit, scale, DSN, data category).Later on transform and download generated bin file through SD card after immediate-preview on DGUS software

  Step2 Running： User only need to refresh data to corresponding address of DataSource at scheduled times. When pages switched to right one, displaying works with preset format.

- Keyboard /Touch input is controlled via 13.bin file according to definition of each page. User's machines are only required to read variables at scheduled times, or serial breakoff activated if parameters varied.

  Example: Set up temperature two-digit decimal value by touching. Two steps requested:

  Step1 Configuration： Via PC software offered by DWIN Technology, users may add up a data variable on the page with format set up(font size, color, unit, scale, DSN, data category).Later on transform and download generated bin file via SD card after immediate-preview on DGUS software

Step2 Running: Touching to activate each buttons makes auto inputting. User’ s machine is free to check record whenever it needed.

- 56KB variable space, 8 channels for curve drawings. Extreme fast response speed (80mS as maximum);
- 256-byte to configure register and read-and-write of serial port for hardware operation
- 256MB(1GB/2GB extended) Flash memory for quantities of icons, images and font save.
- 128pcs of displaying widgets and open-ended touching widgets can be installed as maximum on each page. Overlay of displaying widgets are supported.
- SD/SDHC, FAT32, SD card for configuration operations especially applicable for production
- RTC, backlight adjustment, buzzer functions are integrated
- Support audio play, P-cap touch screen and images memory are allowed to set up high-reliable database.
- DWIN OS embedded that allowed part of codes to run on the DGUS display directly which makes development easier. It makes display possible to being acted as master control.

  DWIN OS integrated the arithmetical operation including MAC and CRC, storage, serial port communications, basic protocol (Modbus, DL/T645), peripheral driving, as well as DGUS process control.
- Not only reliable hardware platform offered (DWIN HMI architecture based ASIC experienced went through more than 10 years on industrial application), but proprietary-owned software assistance (assembly code design, sizing 50KB approximately in total), makes DWIN display better performed in industry.
- TUV CE and RoHs passed.

1.3 DGUS Data Frame

To make it easier for calculation of MCUs, the data in DGUS module is in integer, unsigned integer, long integer and double long integer format.

Integer: -32768 (0x8000) to +32767 (0x7FFF).

Unsigned integer: 0 (0x0000) to 65535 (0xFFFF).

Long integer: -2147483648 (0x80000000) to +2147483647 (0x7FFFFFFF).

Double long integer: -9223372036854775808 to 9223372036854775807.

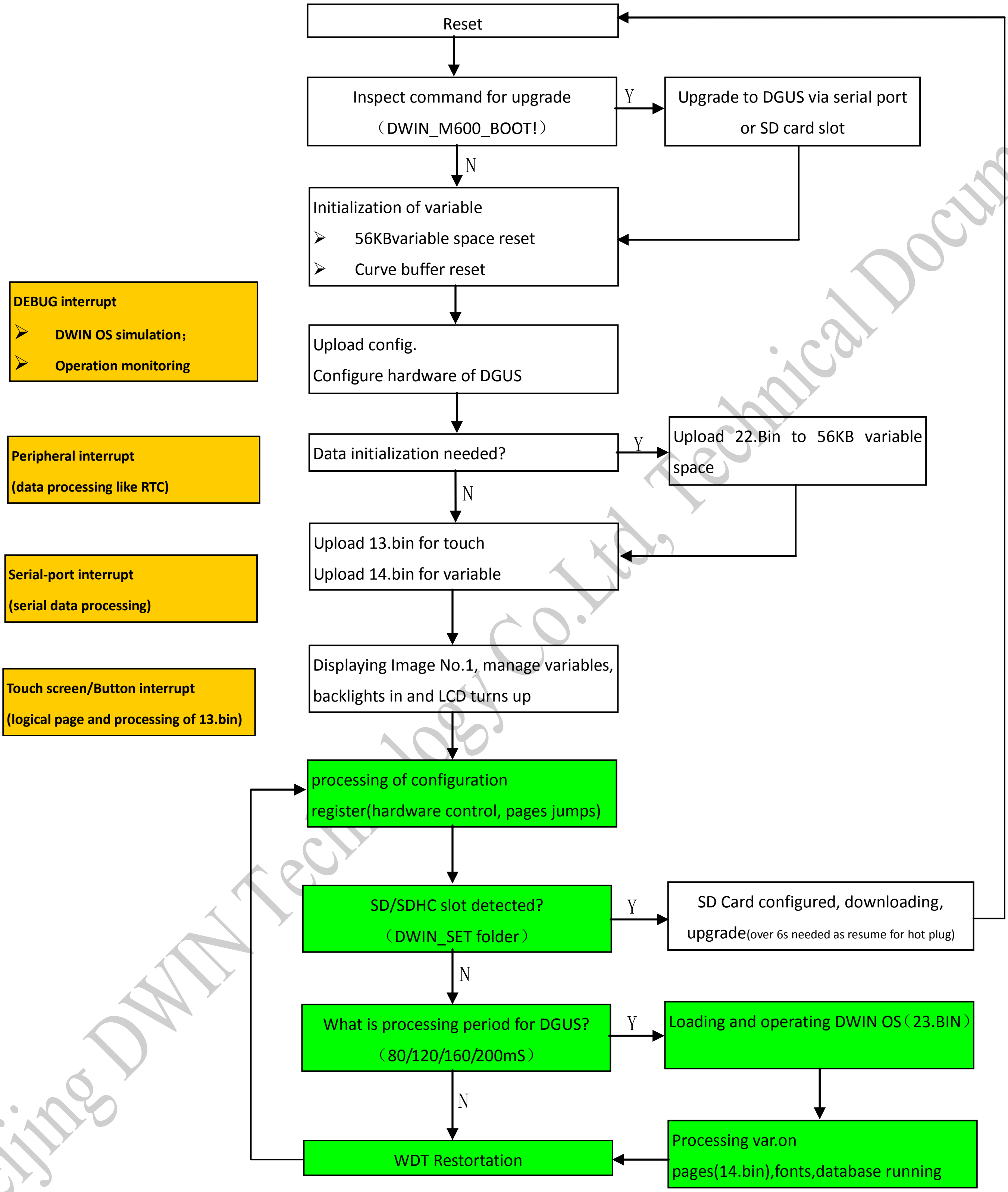
Decimal numbers are represented by fix-point decimals.

Example: 0x4D2(1234) indicates 12.34, if there are two decimal digits.

The DGUS module uses the 16 bit color system. Refer to the chart below to view color palette definition.

65K-color Definition																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Define	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
	Red 0xF800					Green 0x07E0						Blue 0x001F				

1.4 DGUS Processing Flow Chart


Reset
Inspect command for upgrade
（DWIN_M600_BOOT!）
Y
Upgrade to DGUS via serial port or SD card slot
N
Initialization of variable
➢ 56KBvariable space reset
➢ Curve buffer reset
DEBUG interrupt
➢ DWIN OS simulation;
➢ Operation monitoring
Upload config.
Configure hardware of DGUS
Peripheral interrupt
(data processing like RTC)
Data initialization needed?
Y
Upload 22.Bin to 56KB variable space
N
Serial-port interrupt
(serial data processing)
Upload 13.bin for touch
Upload 14.bin for variable
Touch screen/Button interrupt
(logical page and processing of 13.bin)
Displaying Image No.1, manage variables, backlights in and LCD turns up
processing of configuration register(hardware control, pages jumps)
SD/SDHC slot detected?
（DWIN_SET folder）
Y
SD Card configured, downloading, upgrade(over 6s needed as resume for hot plug)
N
What is processing period for DGUS?
（80/120/160/200mS）
Y
Loading and operating DWIN OS（23.BIN）
N
WDT Restortation
Processing var.on pages(14.bin),fonts,database running

Instruction:

DWIN OS programming once completely in each DGUS period (80/120/160/200mS), that is why circulation or delays caused by loop command is prohibited in DWIN OS.

2. DGUS Formula

2.1 Storage of DGUS

There are 7 spaces for display working as shown below. Each further explanation can be founded in relevant chapters. Part DGUS Hardware Configuration will be explained in a solo **Chapter 2.2**. O.S is not a standard factor, but in option for custom use to secondary development on automation solution.

Category	Description
DGUS Register	System and Space for Register
Images Memory	Image saves
RAM	User-defined variable storage
FLASH	Both system and users used part respectively. User may use targeting data save including font and icon.
Curve Buffer	For writing curve data to display temporarily.
OS Register	256 registers were used for operation of OS command.
DGUS Hardware Configuration	Parameter Configuration related with display including baud rate, rotation angle as well as backlight etc.

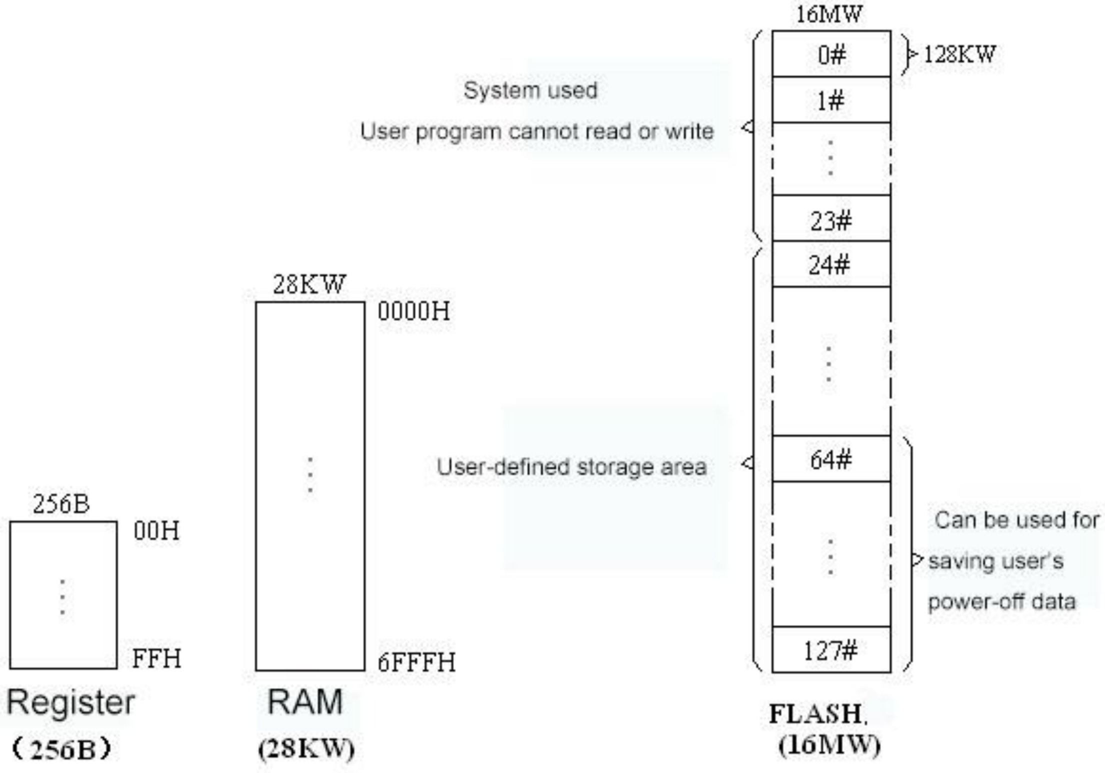

16MW
0#
1#
23#
24#
64#
127#
128KW
System used
User program cannot read or write
User-defined storage area
Can be used for saving user's power-off data
28KW
0000H
6FFFH
256B
00H
FFH
Register
(256B)
RAM
(28KW)
FLASH
(16MW)

Figure 3 DGUS Register Space

2.1.1 DGUS Register

256 bytes in total. See Register in Figure 3 as above.

Use can access command to make backlight control, buzzer control, images switching, RTC, read-and-write to FLASH, timer control, display reset etc. For Further information about register, please refers to **Chapter 4**.

2.1.2 Images Memory

24-bit BMP pictures with same resolution as module are required. Besides, suffix number for image naming is mandatory for reorganization in downloading process.

For example, if one pcs of image will be saved in Position 20, image could be named as “20_TEST.BMP” or “20.BMP” or “020TEST.BMP”, while “TEST20.BMP” is not allowed. Maximum quantity of image store in varied resolutions shown as below

Category	Max. of Image Space	Max. of Database	Quantity						
			320*240	480*272	640*480	800*480	800*600	1024*600	1024*768
Standard	210MB	89MB	836	836	278	278	209	167	139
1GB Extended	932MB	450MB	3728	3728	1242	1242	932	745	621
2GB Extended	1896MB	960MB	7584	7584	2528	2528	1896	1516	1264

2.1.3 RAM

There is 28K Word RAM internally as shown in Figure 3. Each address is double byte with high byte in front and low byte in end, ranging from 000H~6FFFH, which is used for addressing variable on each page. Besides, users may take undistributed RAM address as normal storage in common.

2.1.4 FLASH

There is 16M Word FLASH internally as shown in Figure 3. Each address is double byte with high byte in front and low byte in end. A 32MB flash memory, divided into 128 addresses, is designed for the font library. Each address occupies 256KB, corresponding with an address from 0 to 127.

In sections, No.0-No.23 were for none-accessed where is ready for default system. No.24-No.127 is used to store custom font file or icon resources. While, No.64-No.127 section could be used as users’ FLASH for power-down save of history data. Please refer to **Chapter 2.3.6** for definition of 13_Touch.Bin and 14_Var.Bin.

Font ID	Size(KB)	Description	Example
0	3072	#0 ASCII font.	0_DWIN_ASC.HZK
13	256	13 touch configuration	13_Touch.BIN
14	2048	14 variable configurations (up to1024 pages with max. 64 variables per page).	14_VAR.BIN
22	256	Variable initializing file for the initial value of 56KB access variable.	22_variable initializing.BIN
23	256	User program based on DWIN OS.	23_Software.BIN
24-127	26	Font, icon library (64-127 space can be use as database).	User defined
No.0-23 Font (6MB) retained for future use. User are advised to start from No.24.			

Tips: Export the data from Font ID 32-127 via SD card interface

Create a file naming after Font ID in <DWIN_SET> folder with the extension “.DAT” (e.g.: 32_test.DAT), the minimum size should be 256KB. The corresponding font data will be written into the first 256KB space of the file. For further information, please refer to Chapter 2.3.4.

2.1.5 Curve Buffer

In DGUS there left a curve buffer like FIFO for 8 pcs of curve data running. This space is only allowed to write.

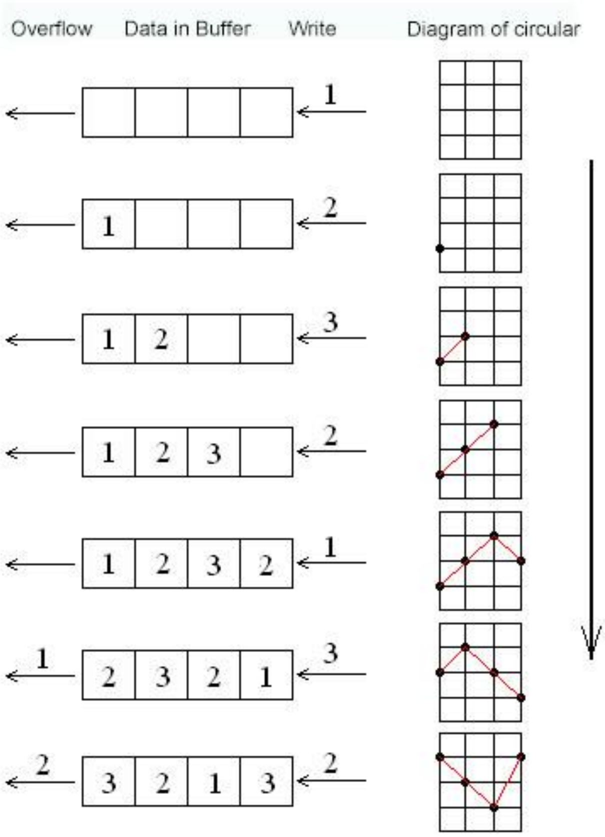
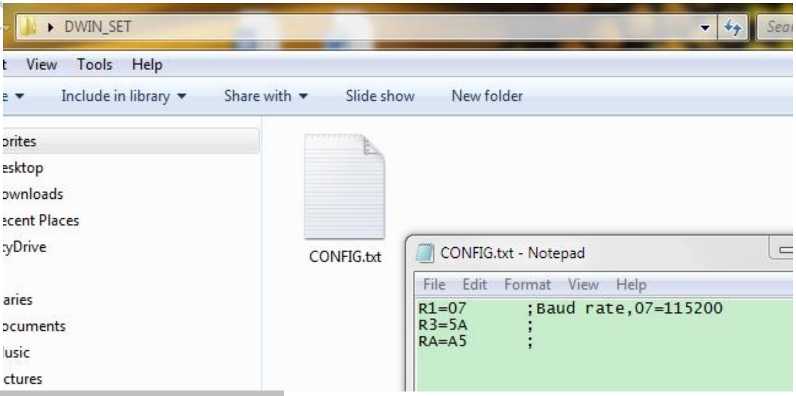


Figure 4 Curve Buffer

2.2 DGUS Parameter Configuration

System settings of DGUS would be configured in CONFIG.TXT. All parameters of register such like scripting language were written and download by SD card. Each parameter is described by line while un-used parameter can be skipped.

Format: R?=HH. **? is serial number of register. HH is hex configured value of register, Capital writing is mandatory set. No need to write N/A and skipped allowed if unneeded.**



```
R1=07      ;Baud rate,07=115200
R3=5A      ;
RA=A5      ;
```

Example: RA=A5: Configure Reg. RA to 0xA5

Do not write ra=5a or RA=5a.

PLEASE refers Chapter 2.2.6 for completed example.

2.2.1 Resolution (R0)

Display Resolution was configured by R0 as below.

R0	Resolution（H*V）	Part Number	Description
00	640*480	DMT64480T056_03W	
01	640*480	DMT64480T057_01W	
02	800*480	DMT80480T070_07W	
03	800*600	DMT80600T080_07W	
04	1024*768	Customization	
05	1024*768	DMT10768T057_01W	
06	800*600	Customization	
07	800*600	Customization	
08	800*600	MVGA01、MDVI01	
09	1024*768	DMT10768T150_02W	
0A	1280*800	NC	
0B	1024*600	DMT10600T102_02W	
0C	1366*768	NC	
0D	240*320	Customization	
0E	320*240	Customization	T035_02W belongs this mode in earlier
0F	480*272	DMT48270T043_03W	
10	480*272	Customization	
11	800*480	Customization	
12	320*240	DMT32240T035_02W	

DO NOT configure register R0, which defines the module drive mode in case of any incorrect manipulation

2.2.2 Bit Clock Phase Selection (R4)

Due to TCON difference on display, there are two kinds of initial displaying data and R4 is used to setting the timing relation of the input data and clock.

R4=00 The input data is locked at the rising edge of clock;

R4=FF The input data is locked at the falling edge of clock;

DO NOT configure register R0 and R4, which probably makes distortion of images or deckle edge

2.2.3 Baud rate（R1,R5,R9）

Baud rate were set on R1,R5 and R9

When R1 equals 00-10, R5 and R9 invalid. The corresponding baud rate as options showing below *(Unit:Kbps)*

R1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
BAUD	1.2	2.4	4.8	9.6	19.2	38.4	57.6	115.2	28.8	76.8	62.5	125	250	230.4	345.6	691.2	921.6

- Whe R1 equals 11, Baud rate up to R5 and R9.

  R5:R9=6250000/Baud rate R5:R9 is a double byte. R5 is high byte and R9 is low.

  E.g.: Set up 10000bps, R5:R9=6250000/10000＝625=0x0271 R5=02 R9=71

  DEFAULT baud rate: R1=7, 1152000bps.

2.2.4 Frame Header (R3,RA)

Block	1	2	3	4	5
Definition	Header	Length	Command	Data	Command and CRC checksum
Length	2	1	1	N	2
Instruction	R3:RA Definition	Length including command, data and checksum	0x80-0x84		R2.4 make calls if activated or not

Structurally, five data blocks composed of serial data frame in DGUS as above and two purposes for setting up data frame header:

1) For recognition of data frame and synchronize operation;

2) Do header as device address for differentiating in simultaneous working of multiple DGUS.

Presume that R3=AA RA=BB, Serial Command have to be started from 0xAA 0xBB (for example read command in register AA BB03 81 00 10) in mandatory. Only in this way DGUS can be answered.

DEFAULT: R3=5A RA=A5   Header: 0x5A A5

2.2.5 Register Configuration of Operating Mode (R2, RC)

Reg. R2, RC is defined by ‘bit’ to configure operating mode. Shadow marking is default set.

- **R2 (SYS_CFG configuration Byte)**

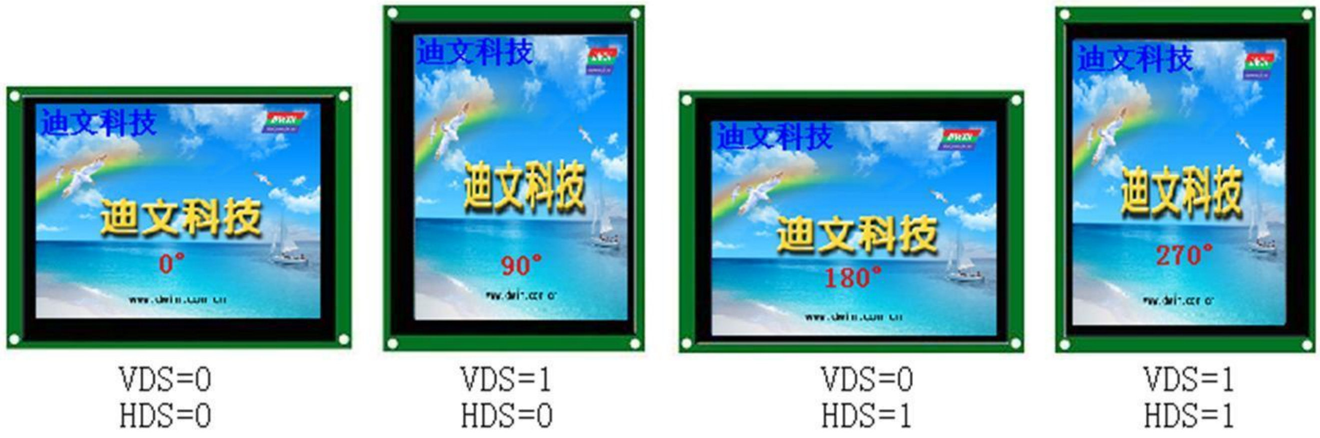
<table>
<tr><th>Bit</th><th>Ratio</th><th>Definition</th><th>Description</th></tr>
<tr><td>.7</td><td>0x80</td><td>VDS</td><td>0=Normal display.
1=90 °Rotation.</td></tr>
<tr><td>.6</td><td>0x40</td><td>HDS</td><td>0=Normal Display.
1=180 °Rotation (upside down).</td></tr>
<tr><td>.5</td><td>0x20</td><td>TP_LED</td><td>0=Brightness is not up to status of touch screen
1=Brightness depends on status of touch screen. Further set up in R6, R7,R8</td></tr>
<tr><td>.4</td><td>0x10</td><td>FCRC</td><td>0=Disable CRC16 checksum in the serial communication.
1= Enable CRC16 checksum in the serial communication</td></tr>
<tr><td>.3</td><td>0x08</td><td>TPSAUTO</td><td>0=Disable auto-upload of key code(user query)
1=Enable auto-upload of key code or data.</td></tr>
<tr><td>.2</td><td>0x04</td><td>L22_Init_En</td><td>0=Initialize 56KB access variable data to 0x00.
1=Initialize 56KB access variable data from 22*.bin.</td></tr>
<tr><td>.1</td><td>0x02</td><td>FRS1</td><td rowspan="2">The shorter period will shorten response time for variable display, but reduce the efficiency of data processing.
<table>
<tr><td>Cycle</td><td>80mS</td><td>120mS</td><td>160mS</td><td>200mS</td></tr>
<tr><td>FRS1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>FRS0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
</table>
For the resolution 1024*768, upon 120mS cycle are recommended
The cycle period determins the speed of Animation Icon display.</td></tr>
<tr><td>.0</td><td>0x01</td><td>FRS0</td></tr>
</table>

- **RC(AUX_CFG Config. Byte) Instruction**

Bit	Ratio	Definition	Description
.7	0x80	Reserved	Write 0
.6	0x40	RUN_OS_EN	0= Disable DWIN OS, equally “STOP_DWIN_OS” in config.txt 1= Enable DWIN OS, equally”RUN_DWIN_OS” in config.txt.
.5	0x20	TP_BUZZ_EN	0=Buzzer works with clicking valid area. 1=No Buzzer, but parameters writings in Register 0x02 is allowed to control the buzzer.
.4	0x10	PAGE128_EN	0=64 variables as maximum quantities of variable on one page.Options of 64 variables in software must be clicked. 1=128 variables as maximum quantities of variable on one page. Options of 128 variables in software must be clicked.
.3	0x08	Undefined	0=manual set for checksum result answer after CRC checksum enabled. 1=auto answer checksum result after CRC checksum enabled.
.2	0x04	Undefined	0= Triple spots calibration 1=Five spots calibration
.1	0x02	Undefined	Write 0
.0	0x01	Undefined	Write 0

2.2.6 Display Direction （R2.7,R2.6）

Four kinds of displaying direction in option that set via R2.7(VDS)、R2.6(HDS)



Example of Config. File

```
R1=07        ; Baud rate, 0x07: 115200bps.
R2=20        ; SYS_CFG, Brightness can be changed via screen clicking, the parameters set up in R6, R7,R8
R6=40        ; Brightness of backlight, 0x40: 100% brightness.
R7=10        ; Brightness of backlight of sleep mode, 0x10: 25% brightness.
R8=14        ; Light-up time，units: 1.0 seconds，0x14=20 seconds.
R3=A5         ; High-byte of frame header: 0xA5.
RA=5A         ; Low-byte of frame header: 0x5A.
```

2.2.7 Backlight Control via Touch Screen（R2.5,R6,R7,R8）

When R2.5=1, backlight determined by touch screen status. If backlight standby, first touch will not activate.

R#	Range	Description
R6	0x00-0x40	Backlight in once touch pressed under backlight control via touch screen activated.
R7	0x00-0x40	Backlight out if no touch for a while after under mode of backlight control via touch screen activated.
R8	0x01-0xFF	Time of backlight in under mode of backlight control via touch screen activated.

E.g.: R2.5=1 R6=40 R7=10 R8=1E

If no touch within 30s (0x1E), brightness will go down by 0x10(25% down), appositively it will get back to 0x40(100%) if touch it again.

Reminder: user can modify register R0 – RA by SD card, also can use command 0xFE07 to modify the parameters on touch screen.

2.2.8 Calibration

Method 1: Click touch screen 20 times in 4 seconds on none-button area to activate calibration mode.

1) Quickly tap the touch screen more than 20 times in 4 seconds. Do not click button area.
2) Click until a long beep emits out. For the models without a buzzer, user can time for 4 seconds or judge whether or not the variables are refreshed.
3) Enter calibration mode, follow crossing point to touch for calibration
4) Calibration done and return to the starting page.

Method 2 (for V4.5 and higher version):

Write "TP_CORRECT" in CONFIG.TXT in root directory of SD card to activate calibration made once.

Caution-For V4.3 and higher versions, touch screen calibration will be disabled when SD card is disabled.

DGUS Display functioned automatically to see if calibration valid. If invalid calibrated like mis-operation, DGUS display will not be set incorrectly. Unless of unlock of SD card slot, lock of slot makes calibration does not happens in operation

2.3 SD Card Operation

All files downloaded in DGUS Modules by SD or SDHC is FAT 32 in mandatory. Please start from a quick format to SD card ensuring integrity of data downloading because sometimes SD card in consuming level is not application and failed in use. **Please see Chapter 2.3.5**

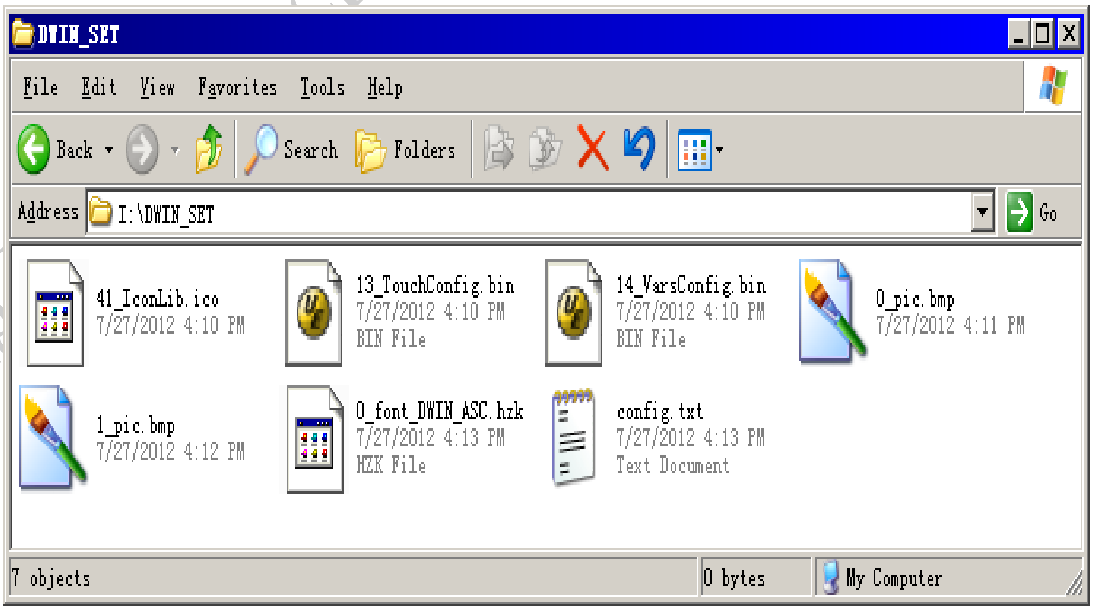
1) Plug SD card into the slot on the module to download files.
2) Downloading process starts automatically after the initialization with blue screen.
3) When downloading finished, config and image in page 1 will be displayed.

2.3.1 How to use SD card to operate.

1) Create a <DWIN_SET> folder in the root directory of the SD card.
2) Copy the pictures, fonts and config.files into <DWIN_SET> folder, as shown below.
3) Images: *.bmp Touch Configuration: 13*.bin Variable Configuration: 14*.bin
Font: *.HZK / *.DZK ICO File: *.ICO Other Binary: *.bin
System Configuration: CONFIG.txt

Reminder 1: Besides of CONFIG.txt, other files named with a numbering prefix which is the locating number for section of this file in Flash.

Reminder 2: There must have 6 seconds hold between Plug-in and plug-out, or downloading will not started.



I:\DWIN_SET

41_IconLib.ico 7/27/2012 4:10 PM
13_TouchConfig.bin 7/27/2012 4:10 PM BIN File
14_VarsConfig.bin 7/27/2012 4:10 PM BIN File
0_pic.bmp 7/27/2012 4:11 PM
1_pic.bmp 7/27/2012 4:12 PM
0_font_DWIN_ASC.hzk 7/27/2012 4:13 PM HZK File
config.txt 7/27/2012 4:13 PM Text Document

7 objects 0 bytes

SD Card File Format			
File Type	**Naming Rule**	**Example**	**Description**
Pictures	Picture ID+ (optional) file name.BMP	00_starting page.BMP	24-bit BMP pictures with same resolution of DWIN module are required
Fonts	Font ID+ (optional) file name.BIN/DZK/HZK	32_ASCII. DZK	Generated by the Font Generator
Icon Library	Icon file ID+ (optional) file name.ICO	41_iconlibrary. ICO	Generated by DWIN Toolbox "DWICON"
Default ASCII	0*.HZK	0_DWIN_ASC.HZK	Generated by DWIN Toolbox "No.0 font library".
Touch configuration	13*.BIN	13_touch configuration file.BIN	Generated by DGUS_SDK.
Variable configuration	14*.BIN	14_variables configuration file. BIN	Generated by DGUS_SDK.
Variables Initialization	22*.BIN	22_Initialization.BIN	
User Code	23*.BIN	23_Water_Treatment.BIN	Base on DWIN OS.
Hardware settings	CONFIG.TXT	CONFIG.TXT	

2.3.2 Audio File downloads.

Part of DGUS display support 128 segments of audio play(further information to specification) that downloaded into display in advance. Same as font file downloading, audio file also named with suffix number on it, from 0-127, for example 12testing.wav,

Format: WAV, 32KHz, 16bit, Mono. 32KB/S for downloading. Flash of DGUS is not related with audio.

2.3.3 Firmware Upgrade/Downgrade

DWIN will release latest firmware from time to time. User are free to enquiry to sales and drop latest one (DGUS_V*.BIN) into DWIN_SET and download directly.

2.3.4 Export of Database via SD Card

Database is a space from image memory for data saves. Space size, position are determined by users that varies based on different kernel size. In course of use, encryption and Forward Error Correction (FEC) processed in data procedure to ensure the reliability.

Category	**Max. of Image Space**	**Max. of Database**	**Quantity**						
			320*240	**480*272**	**640*480**	**800*480**	**800*600**	**1024*600**	**1024*768**
Standard	210MB	89MB	836	836	278	278	209	167	139
1GB Extended	932MB	450MB	3728	3728	1242	1242	932	745	621
2GB Extended	1896MB	960MB	7584	7584	2528	2528	1896	1516	1264

Step of database export:

1) Work out location of starting page

Assumed that initial address is ADR, so the exporting page ID is highest double bytes + 256

2) Build up a DAT file, naming with initial page ID and same size as exporting database (ALT to 128KB)

Initial Page ID+(optional)File name.DAT

Drop this DAT file into DWIN_SET and auto uploading to existed DAT file once SD card inserted

E.g.: Exporting database 0x00 10 00 00 to 0x00 17 FF FF, in total 1MB(512KW)

Accordingly, SD Card exporting ID is 0x00 10+256=272

Please set down a folder sizing 1MB in DWIN_SET: 272databass.DAT or something like this. Then insert SD card to export data.

Data Speed: 180KB/S. For large size data exporting, users are recommended to divide it to multiple files.

For further information of Database read-and-write, please go to **Chapter 4.2**

2.3.5 SD/SDHC Interface Lock/Unlock

1) SD/SDHC Lock and Unlock:

Specific codes in CONFIG.TXT can be used to disable the SD card slot on the DGUS module with a password to avoid accidental operation.

	Code to Disable SD Card	Description
Part 1	SD_LOCK	Fixed.
Part 2	1000	Password address in variable SRAM, 0x0000 – 0x6FF8.
Part 3	ABCD1234	Password to re-enable SD card, 8 bytes.

Code in CONFIG.TXT to re-Enable SD card: **SD_UNLOCK**.

E.g.: presume password is 12345678, saved in 0x6000 address in variable SRAM.

Steps to disable SD card:

1. Write “SD_LOCK_6000_12345678” to CONFIG.TXT.
2. Copy CONFIG.TXT into DWIN_SET folder in SD card.
3. Plug SD card into slot on DGUS module to disable it.

Steps to re-enable SD card:

Method 1: Send password to module via serial to activate SD card once.

We take 0xA55A as frame header, send command: A5 5A 0B 82 60 00 31 32 33 34 35 36 37 38.

Method 2: Using <Text Input> to type password can activate SD card once.

Method 3: write re-able SD card command in CONFIG.TXT in root directory of SD card and plug SD card into slot on DGUS module to re-able SD card.

WARNING- FAILURE TO INPUT CORRECT PASSWORD WILL RESULT IN SD CARD INTERFACE PERMANENT LOCKOUT! SAFEKEEP YOUR PASSWORD!

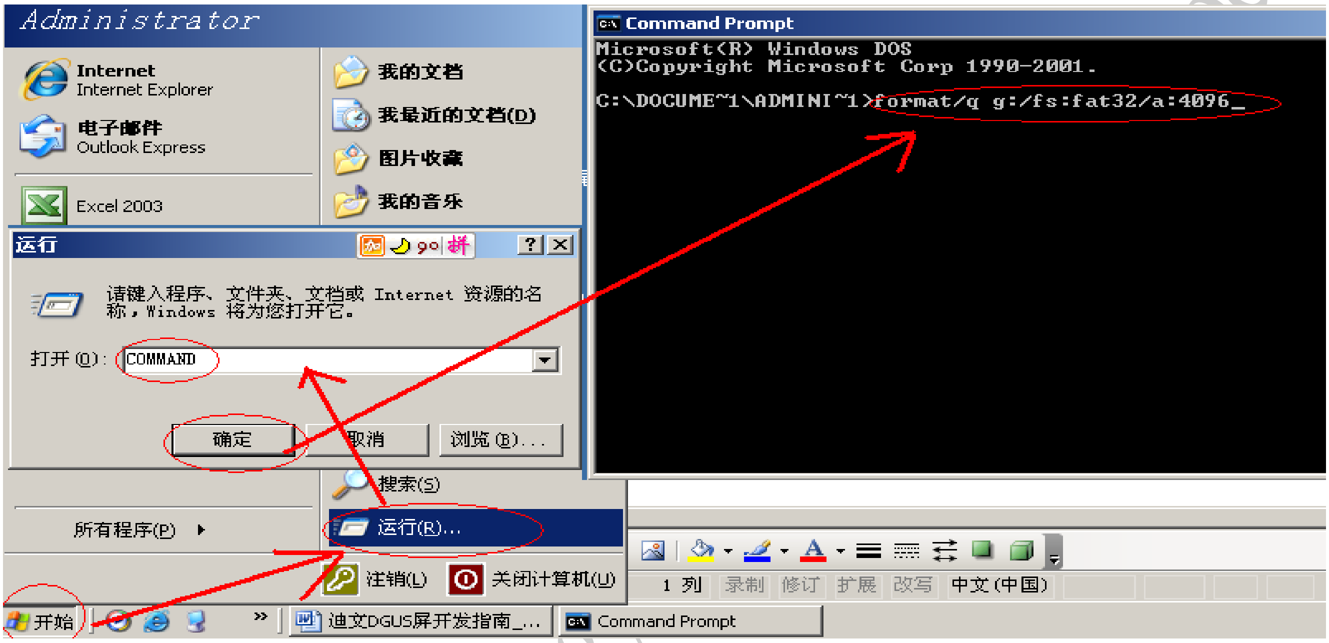
2) Format SD/SDHC

Format your SD card: if part of your data in SD card is not downloaded into module, please format your SD card as the instruction below.

Step 1: open RUN function in Windows and run DOS using “command”.

Step 2: type command to format: “format/q g:/fs:fat32/a:4096”, and click <enter> to finish formatting.

The letter in red is the disk number of SD card.



请键入程序、文件夹、文档或 Internet 资源的名称，Windows 将为您打开它。

打开(O)：COMMAND

确定 取消 浏览(B)...

Command Prompt

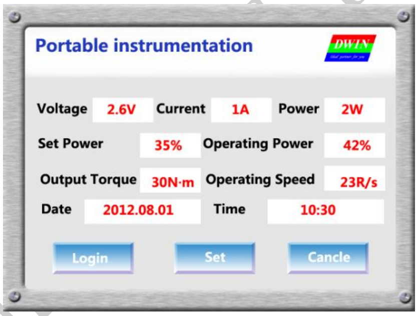
```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-2001.

C:\DOCUME~1\ADMINI~1>format/q g:/fs:fat32/a:4096_
```

2.3.6 Basic Operational Process from a Project.

Step 1: Planning of Variables

✧ VP should be arranged by continuous addresses for read/write convenience.

✧ Avoid overlap of VP and SP addresses.



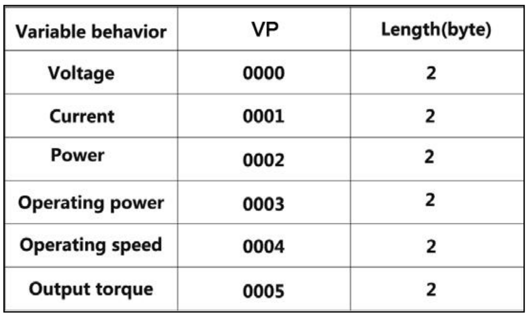
Portable instrumentation

Voltage 2.6V Current 1A Power 2W

Set Power 35% Operating Power 42%

Output Torque 30N·m Operating Speed 23R/s

Date 2012.08.01 Time 10:30



Variable behavior	VP	Length(byte)
Voltage	0000	2
Current	0001	2
Power	0002	2
Operating power	0003	2
Operating speed	0004	2
Output torque	0005	2

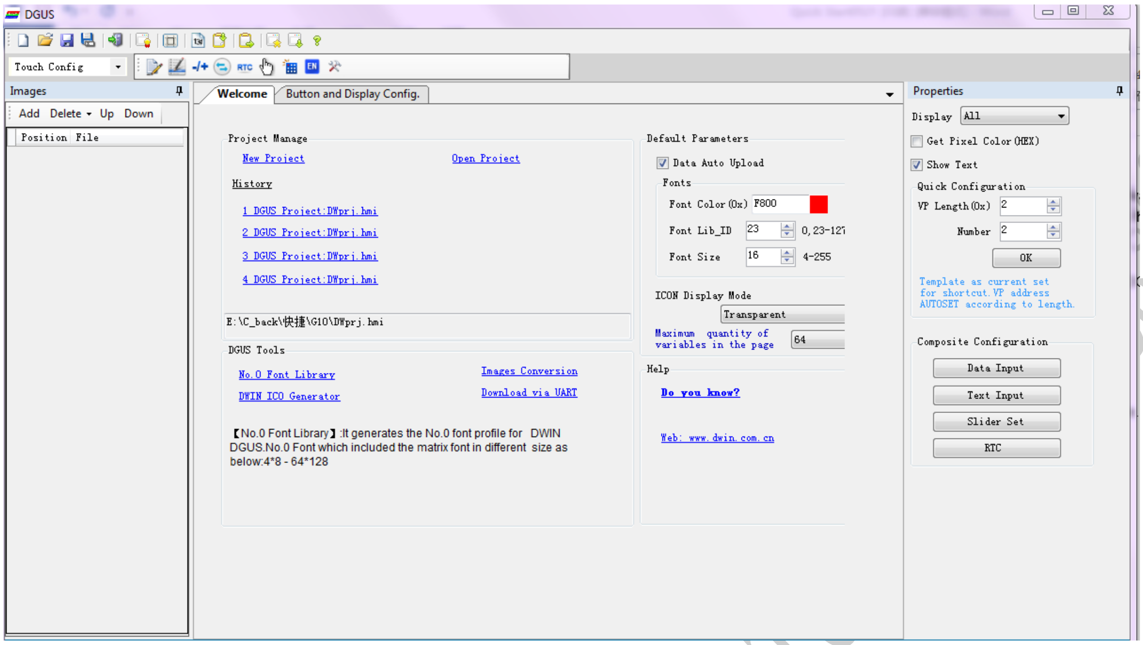
Step 2: Interface Design

✧ Outsourcing or employ designers to create Pictures, icons and fonts are generated by the image processing software.

✧ Run DGUS_SDK and set down a new project loading all materials with well set resolution and corresponded Touch Variable(13_Touch Configuration.BIN) and Display Variable(14_Variables Configuration. BIN), programming it.

Step 3: Configuration of User Interface

✧ Config. file for communication and operation parameters of button are generated by DGUS_SDK



1 DGUS Project:DWprj.hmi
2 DGUS Project:DWprj.hmi
3 DGUS Project:DWprj.hmi
4 DGUS Project:DWprj.hmi

E:\C_back\快捷\G10\DWprj.hmi

【No.0 Font Library】:It generates the No.0 font profile for DWIN DGUS.No.0 Font which included the matrix font in different size as below:4*8 - 64*128

☑ Data Auto Upload

Font Color (0x) F800 🟥

Font Lib_ID 23 0,23-127

Font Size 16 4-255

ICON Display Mode: Transparent

Maximum quantity of variables in the page 64

Web: www.dwin.com.cn

Display All

□ Get Pixel Color (HEX)

☑ Show Text

VP Length (0x) 2

Number 2

Template as current set for shortcut.VP address AUTOSET according to length.

Step 4: Debugging & Modification

✧ Testing and revising the interface by viewing effects on DGUS module. (Step 2 - 3)

✧ Connect serial port of DGUS module and user's MCU, debugging.

Step5: Data transfer to LCMs

✧ Config. files, fonts, icon files, pictures and other files must be stored in SD cards as DWIN_SET folder that taking for a while to auto download after inserting to slot.

Definition:

✧ **Touch Configuration (13.Bin)**

Touch Function includes all display actions with touches which means different behaviors on operation makes varied outcomes. For example, 'Basic Touch' is the operation for press effects and interface switches, and 'Return Value' is not only limited to Basic Touch, but change the value of sort variables. There are 9 widgets in software for touch configuration: Variable Data Input, Popup Window, Incremental Adjustment, Slider Adjustment, the RTC settings, Basic Touch Control, Return Key Code, Text Input, Firmware Parameter Settings.

✧ **Display Variable (14.Bin)**

Display Variable is unit on pages for displaying and changing status, which is, controlled via command or others approaches, including icon, data displaying, timer, text etc. All variables have an attribute in common, address of variables. This address is space in RAM that is distributed in settings of variable attributes. User is allowed to manage variables by modifying the data in distributed RAM space. It is something like a kind of container in terms of communications management of program, operation is only required to work with data inside which can be displayed directly on LCD, such form like data, icon, pointer, graphics, animation as well as curves.

Variable Icon, Animation Icon, Slider, Wordart. Image Animation, Icon Rotation, Data Variable etc. as widgets embedded on DGUS_SDK for customer's visual developments.

3. Serial Port

Serial mode of DGUS module is asynchronous, full duplex serial port (UART). Each byte occupies 10 bits: 1 start bit, 8 data bits, and 1 stop bit.

The SD card can define baud rate. All data transfer is in hexadecimal format with MSB priority. E.g.: transferring 0x1234, 0x12 will be transferred first, then 0x34 after.

Busy pin is invalid for DGUS module; keep it unconnected.

Volume of serial FIFO buffer is 4KB(around 230400-691200bps continuous send), minimum capacity of data transfer in DGUS circle (80/120/160/200ms). Maximum capacity depends on the complexity of GUI. Therefore, DWIN recommends sending no more than 4KB data to the DGUS module in a DGUS cycle.

3.1 Data Frame

Data frame is made up by 4 parts, shown as below.

Data	**1**	**2**	**3**	**4**	**5**
Definition	Frame Header	Data length	Command	Data	CRC checksum of the command and data
Data Length	2	1	1	n	2
Description	Defined by R3 & RA in CONFIG.TXT	Data length, including command, data and checksum	0x80-0x84		Defined by R2 in CONFIG.TXT

The maximum length of a data packet is 254 bytes (without CRC checksum) or 252 bytes (with CRC checksum).

CRC checksum is only available for command and data, instead of data length and frame header, with ANSI CRC-16(X16+X15+X2+1) format.

3.2 Command

Working under variable-oriented pattern, All modes and GUI status of DGUS are entirely controlled by variables. Therefore, user only need to process up variables via serial. There are only 5 pcs of command for operation easily in 3 categories, one for access to DGUS register(0x81), one for access to RAM(0x82,x083) and one for curve display(0x84).

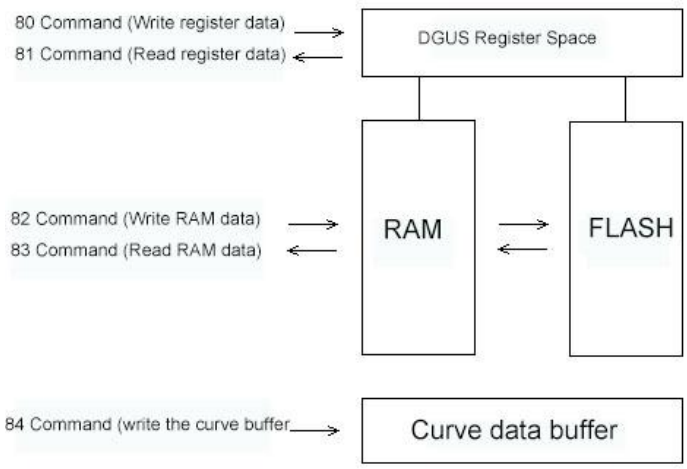

80 Command (Write register data)
81 Command (Read register data)
DGUS Register Space
82 Command (Write RAM data)
83 Command (Read RAM data)
RAM
FLASH
84 Command (write the curve buffer
Curve data buffer

Figure 5 Principle of Command in DGUS

Function	CMD	Data	Description
Access Register	**0x80**	ADR(0x00-0xFF)+Data_Pack	Write data in designated addresses in register.
	0x81	ADR(0x00-0xFF)+RD_LEN(0x00-0xFF)	Read data in designated addresses in register.
		ADR(0x00-0xFF)+RD_LEN+Data_Pack	Response of DGUS module.
	256Byte register for interface with hardware control, following addressing operation based on **Byte**.		
Access Variable SRAM	**0x82**	ADR_H:L(0x0000-0x6FFF)+DATA0…DATAn	Write data in designated addresses in variable SRAM.
	0x83	ADR_H:L(0x0000-0x6FFF)+RD_LEN(0x00-0x7F)	Read data in designated addresses in variable SRAM.
		ADR_H:L+RD_LEN+DATA0……DATAn	Response of DGUS module.
	28K(56K Byte) variable register for GUI variable store,following addressing operation based on **Word**.		
Curve Buffer	**0x84**	CH_Mode（Byte）+DATA0（Word）+…+DATAn	Write data in curve buffer. CH_Mode defines channels for trend curve channel of follow-up data order: ➢ Each bit of CH_Mode corresponds to one channel; e.g.: CH_Mode .0 corresponds to channel 0, .7 corresponds to channel 7 1 in particular bit indicates the presence of the corresponding channel. ➢ Data of lower channel is prior ranged. e.g.: CH_Mode = 0x83 (10000011B), indicates a follow-up data format : (channel 0 + channel 1+ channel 7) +...+ (channel 0 + channel 1+ channel 7).
	8K Word in DGUS for 8pcs of curve drawing. All data in this buffer is 16-bit unsigned number.		

3.3 Access Register

Command 0x80：Execute order of write to DGUS register

Command 0x81：Execute order of read to DGUS register

If frame header is 0x5AA5, no CRC checksum, format as follows.

➢ Command of Write:

Header		Length	Command	Initial Address of Reg.	Data Pack
0x5A	0xA5	F_Len	0x80	W_ADR	W_Data
2 Bytes		1Byte	1Byte	1Byte	N Byte

W_ADR：Initial Address of data in pack and write to following address of register in sequence.

W_Data：Data that will be written in DGUS.

➢ Command of Read:

Header		Length	Command	Initial Address of Reg.	Loading Length(bytes)
0x5A	0xA5	F_Len	0x81	R_ADR	R_Num
2 Bytes		1Byte	1Byte	1Byte	1Byte

R_ADR：User will read and load data where started from this address.

R_Num：User will read and load bytes in data starting from R_ADR as initial address.

3.3.1 Read-and-Write RTC

0x1F	RTC_COM_ADJ	W	1	0x5A: modifications for RTC data via serial port, then, reset.
0x20	RTC_NOW	R/W	16	YY:MM:DD:WW:HH:MM:SS

➢ Read RTC by serial port

Save current RTC on Reg. 0x20 and use Command 0x81 to read.

Calendar Loading（YY:MM:DD:WW:HH:MM:SS）: 5A A5 03 81 20 07

TIME UPLOADING（HH:MM:SS）: 5A A5 03 81 24 03

- Write RTC by serial

Command 0x80 to rewrite Reg.0x1F as 0x5A and write preset time from Reg.0x20

E.g.:

Setup time as 2013-11-08 18:56:00. Send as follows:

5A A5 0A 80 1F 5A 13 11 08 00 18 56 00

Reminder：Y/M/D/H/M/S need to be set ONLY，week and lunar calendar in process automatically by DGUS.

Position on Week can be whatever set such like 00 in above example.

3.3.2 Read-and-Write Font

0x40	En_Lib_OP	R/W	1	0x5A: modifications for Font data via serial port, then, reset.
0x41	Lib_OP_Mode	W	1	0xA0：read signified font file to variable register.
0x42	Lib_ID	W	1	Custom font space, 0x40-0x7F, 128KW for each Maximum Flash is 8MW(16MB)
0x43	Lib_Address	W	3	Initial Add.(word) of signified font space: 0x00:00:00-0x01:FF:FF
0x46	VP	W	2	Initial Add.(word) of signified variable register: 0x00:00-0x6F:FF
0x48	OP_Length	W	2	Data length(word): 0x00:01-0x6F:FF。

FONT No. 64-127 (64pcs font, 16MB) can be operated via serial-port and load to variable register. (If system required, users are free to use Command 0x82 load from variable register)

E.g.: Load 4KW(0x10 00)data to variable register starting 0x10 00 from Add.0x00 00 00 in Font No.80.

Send: 5A A5 0C 80 40 5A A0 50 00 00 00 10 10 10 00

Reminder: Do not over size the font space,namely Lib_Address+OP_Length<= 0x02 00 00

3.3.3 128 Segments of Audio Play

0x50	Play_Music _Set	W	3	0x5A:Play_Strat:Play_Num Play_Start : position of start，Play_Num: numbers for successive play（0x00 stop）
0x53	Volume_Adj	W	2	Write: 0x5A:VOL adjustment, Volume=VOL/64,default 0x40。

Some of DGUS display have function of audio play(1.024s/seg). Command 0x80 could be used for control of audio play and adjustment of volum

E.g.: An audio with 3.5s in length, saving from Position 6 to Position9, which was divided into 4 segments. Please send following command with 100% volume.:

Play:5A A5 07 80 50 5A 06 04 5A 40

Stop: 5A A5 05 80 50 5A 06 00

Volume up to 150%(64*1.5=96 0x60): 5A A5 04 80 53 5A 60

N/A area will be skipped over in audio processing.

3.4 Access RAM

Command 0x82：Write operation to RAM

Command 0x83：Read operation to RAM

If frame header is 0x5AA5, no CRC checksum, format as follows.

- Command of Write:

Header		Length	Command	Initial Address of RAM	Data Pack
0x5A	0xA5	F_Len	0x82	W_ADR	W_Data
2 Bytes		1Byte	1Byte	2 Byte	N Byte

W_ADR：Initial Address of data in pack and write to following address of RAM in sequence.

W_Data：Data that will be written in RAM.

E.g.: Use Command 0x82 to write data 0x0064, 0x0032 to Add. 0x1000,0x1001 in RAM

Send: 5A A5 07 82 10 00 00 64 00 32

Response: N/A

- Command of Read:

Header		Length	Command	Initial Address of ARM	Loading Length(bytes)
0x5A	0xA5	F_Len	0x83	R_ADR	R_Num
2 Bytes		1Byte	1Byte	1Byte	1Byte

R_ADR：User will read and load data where started from this address.

R_Num：User will read and load bytes in data starting from R_ADR as initial address.

E.g.: Use Command 0x83 to read data from Add. 0x1000, 0x1001 in RAM

Send: 5A 5A A5 04 83 10 00 02

Response: 5A A5 08 83 10 00 02 xx xx xx xx

3.5 Access FLASH

As shown in Figure 5, in DGUS register, Reg. 0x40-Reg.0x49 manages the data communications between RAM and FLASH. The point for users to access FLASH is control of DGUS register.

FLASH reading operation: Copy data from FLASH to RAM via Command 0x80, then read it out via Command 0x83.

FLASH writing operation: prewriting data to RAM via Command 0x82, then write data in RAM into FLASH via Command 0x80

Prior to operation get finished, users could use Command 0x81 to check Reg. 40H in loop until it gets zero.

3.6 Access Curve Buffer

If frame header is 0x5AA5, no CRC checksum, format as follows.

Header		Length	Command	Mode of Channel Distribution	Data Pack
0x5A	0xA5	F_Len	0x83	R_ADR	R_Num
2 Bytes		1Byte	1Byte	1Byte	N Byte

CH_M：This controls the position that data to be written. There are 8 bits sequencing from bit0, bit1,……bit7, signifying Channel 0, Channel 1, Channle X with valued 1in valid.

Line_Data：Data that send to buffer, Word as format.

E.g. Write DATA 0x0001, 0x0002 to Channel 0, and write DATA 0x0003, 0x0004 to Channel 5.

Send: 5A　A5　0A　84　21　（0001 0003）（0002 0004）

Response: N/A

3.7 Serial Port CRC Checksum Program Reference

ANSI CRC-16(X16+X15+X2+1）format was adopted in DGUS and program C as follows:

```
unsigned char CRCTABH[256]={0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41
                    0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0
                    0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1
                    0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1
                    0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40
                    0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1
                    0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40
                    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0
                    0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41
                    0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41
                    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40
                    0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1
                    0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41
                    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0
                    0x80,0x41,0x00,0xC1,0x81,0x40};
unsigned char CRCTABL[256]={0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06
                    0x07,0xC7,0x05,0xC5,0xC4,0x04,0xCC,0x0C,0x0D,0xCD
                    0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09
                    0x08,0xC8,0xD8,0x18,0x19,0xD9,0x1B,0xDB,0xDA,0x1A
                    0x1E,0xDE,0xDF,0x1F,0xDD,0x1D,0x1C,0xDC,0x14,0xD4
                    0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3
                    0x11,0xD1,0xD0,0x10,0xF0,0x30,0x31,0xF1,0x33,0xF3
                    0xF2,0x32,0x36,0xF6,0xF7,0x37,0xF5,0x35,0x34,0xF4
                    0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A
                    0x3B,0xFB,0x39,0xF9,0xF8,0x38,0x28,0xE8,0xE9,0x29
                    0xEB,0x2B,0x2A,0xEA,0xEE,0x2E,0x2F,0xEF,0x2D,0xED
                    0xEC,0x2C,0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26
                    0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,0xA0,0x60
                    0x61,0xA1,0x63,0xA3,0xA2,0x62,0x66,0xA6,0xA7,0x67
                    0xA5,0x65,0x64,0xA4,0x6C,0xAC,0xAD,0x6D,0xAF,0x6F
                    0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68
                    0x78,0xB8,0xB9,0x79,0xBB,0x7B,0x7A,0xBA,0xBE,0x7E
                    0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,0xB4,0x74,0x75,0xB5
                    0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71
                    0x70,0xB0,0x50,0x90,0x91,0x51,0x93,0x53,0x52,0x92
                    0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,0x9C,0x5C
                    0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B
                    0x99,0x59,0x58,0x98,0x88,0x48,0x49,0x89,0x4B,0x8B
                    0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C
                    0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42
                    0x43,0x83,0x41,0x81,0x80,0x40};

unsigned char index,crch.crcl;
                      crch=0xff;
                      crcl=0xff;
                      for(i=0;i<j;i++)
                 {    index=crch^txdat[i];      //txdat[i] is data sending
                      crch=crcl^CRCTABH[index];
                      crcl=CRCTABL[index];
```

4 DGUS Register

4.1 DGUS Register (0x80/0x81 to access via UART)

Register Address	Definition	Length (Byte)	Description	
0x00	Version	1	DGUS version number, BCD format, 0x10 indicates V1.0.	
0x01	LED_NOW	1	LED brightness, 0x00-0x40.	
0x02	BZ_TIME	1	Buzzer beeping time, by every 10ms.	
0x03	PIC_ID	2	Read: read current picture ID. Write: jump to appointed picture.	
0x05	TP_Flag	1	0x5A: there is update of touching coordinates. Others: no updating. Touch panel data is no longer updated if user did not clear the flag after data reading.	
0x06	TP_Status	1	0x01: first click. 0x03: pressing down. 0x02: uplift pressing. Others: null.	
0x07	TP_Position	4	Coordinate of touching position: X_H:L, Y_H:L.	
0x0B	TPC_Enable	1	0x00: disable the touch panel. Others: enable the touch panel. Default setting: 0xFF.	
0x0C-0x0F	RUN_TIME	4	Running time after power on, BCD format, hour occupies 2 bytes, the max is 9999:59:59.	
0x10-0x1A	R0-RA	11	Mapping of SD card config. register, read only.	
0x1F	RTC_COM_ADJ	1	0x5A: RTC data is rewritten through serial port, clear after RTC auto updating.	
0x20	RTC_NOW	16	YY:MM:DD:WW:HH:MM:SS	
Send serial command to modify current time, e.g.: A5 5A 0A 80 1F 5A 12 10 25 0412 00 01. (BCD Format) "04" means Thursday, it can be written as any day you choose.				
0x30-0x3F	Reserve	16	Undefined.	
0x40	En_Lib_OP	1	0x5A: applying writing in font flash memory, clear after operation.	
0x41	Lib_OP_Mode	1	0x50: Transfer data from variable flash to font flash memory. 0xA0: Transfer data from font flash memory to variable SRAM.	
0x42	Lib_ID	1	Designate font address for data exchange. (0x40-0x7F) Every font space is 128KW, the maximum Flash space is 8MW (16MB).	
0x43	Lib_Address	3	Designate address in font library for data exchange. Specify the first (word) address for data operation in font storage, 0x00:00:00-0x01:FF:FF.	
0x46	VP	2	Designate variable SRAM addresses for data exchange. 0x00:00-0x6F:FF.	
0x48	OP_Length	2	Length of exchanged data, by word. 0x00:01-0x6F:FF.	
Save 1KW variable data string starting from 0x1000 address into #64 font ID with starting 0x0000 address, send serial command: A5 5A 0C 80 40 5A 50 40 00 00 00 10 00 02 00.				
0x4A	Timer0	2	16-bit software timer, in term of 4ms, auto-decrement to 0.	Maximum error is +/-4ms.
0x4C	Timer1	1	8-bit software timer, in term of 4ms, auto-decrement to 0.	
0x4D	Timer2	1	8-bit software timer, in term of 4ms, auto-decrement to 0.	
0x4E	Timer3	1	8-bit software timer, in term of 4ms, auto-decrement to 0.	
0x4F	Key_code	1	Address of key code for 13 touch control config. file, 0x00: null. Clear after operation executed.	
0x50-0xEA	Reserve	158	Undefined.	
0xEB	Trendline _Clear	1	Specially defined data write in order to clear of corresponding curve buffer 0x55: Clear of all curve buffers； 0x56-0x5D：Clear of channel CH0-CH7; Notice: register will be returned to 0 after clear up of curve buffer.	
0xEC-0xED	Reserve	2	Undefined.	
0xEE-0xEF	Reset_Triger	2	Write 0x5AA5 to reset DGUS once.	
0xF0-0xFF	Reserve	16	Undefined.	

4.2 Read-and-Write of Database

0x56	En_DBL_OP	R/W	1	0x5A: user applies for working with register of database and clear after operation. Once time of read-and-write operation in each period of DGUS running.
0x57	OP_Mode	W	1	0x50：Write data from variable storage to database. 0xA0：Loading data from database to variable storage.
0x58	DBL_Address	W	4	Word Address of Database. 0x00:00:00:00-1D:FF:FF:FF, 480MW as maximum（960MB，depends on Flash） Savings starts from No. 64MB of memory space, overlay with images space. Each 1Byte database occupies 2Bytes in memory. When export data from SD card, the size of font is 64KW(128KB), numbering from 256. Font ID for 960MB database: 256-7935. Read-and-write times: 100,000 times.
0x5C	VP	W	2	Initial address of designated variable storage in database: 0x00:00-0x6F:FF
0x5E	OP_Length	W	2	Data length: 0x00:01-0x6F:FF。

Category	**Max. of Image Space**	**Max. of Database**	**Quantity**						
			320*240	**480*272**	**640*480**	**800*480**	**800*600**	**1024*600**	**1024*768**
Standard	210MB	89MB	836	836	278	278	209	167	139
1GB Extended	932MB	450MB	3728	3728	1242	1242	932	745	621
2GB Extended	1896MB	960MB	7584	7584	2528	2528	1896	1516	1264

In course of use, encryption and Forward Error Correction (FEC) processed in data procedure to ensure the reliability. Psychically, multiple pages sizing 64KW(128KB) composed of database with 100,000 times for read-and-write (Order of writing makes consuming one time), moreover, address in operation is successive, and paging issue managed by DGUS automatically.

- Initial Add of Database(0x00 00 00 00, corresponding to No.64MB physical storage) related with Image ID and savings coefficient K1

Resolution	**320*240**	**480*272**	**640*480**	**800*480**	**800*600**	**1024*600**	**1024*768**
K1	1	1	3	3	4	5	6
PIC_ID	128	128	42-43	42-43	32	25-26	21-22
"128" signifies No.128 in not in use for image save if database starts from Add.0x00; “42-43” signifies both invalid							

- Calculation for data from space to database.
  Presume that if we have N pcs of images to store, the initial address of database in use is :
  （（N*K1）-128）*64*1024 round numbers to 64KW(128KB)。
  E.g.: 200pcs images under Res.480x272, the Adr_Min:[(200*1)-128]*64*1024=0x00 48 00 00
- Exporting : Please see the step of exporting from SD card on **Chapter 2.3.4**

4.3 Button Pressing Activated.

0x4F	Key_code	W	1	Keyboard Code to activate 0x13 touch bin file: 0x01-0xFF，0x00 signifies invalid. DGUS will clear register of keyboard code after processing.

In application of DGUS without keyboard interface offered while requested in customer’s end, Reg. 0x4F provided a interface to keyboard-control for GUI. User only needs to drop code in Reg. 0x4F to satisfy this feature.

E.g.: in 13.bin with Page No.10, key code 0xF1 defined to enter Page where asked a data input. So, Command sending under Page No. 10 5A A5 03 80 4F F1 will activate a keyboard control and page jumps accordingly. Keyboard control could be mix up with touch screen operation in use.

5. DGUS Development Steps

5.1 Development in General

Unlike the previous LCMs, which adopted commands-oriented or timing sequence to manage GUI, DGUS module performed based on real-time variables with programmable file configured, transmitting via UART or SD card. Software flow chart of different development methods for temperature controller is shown as above.

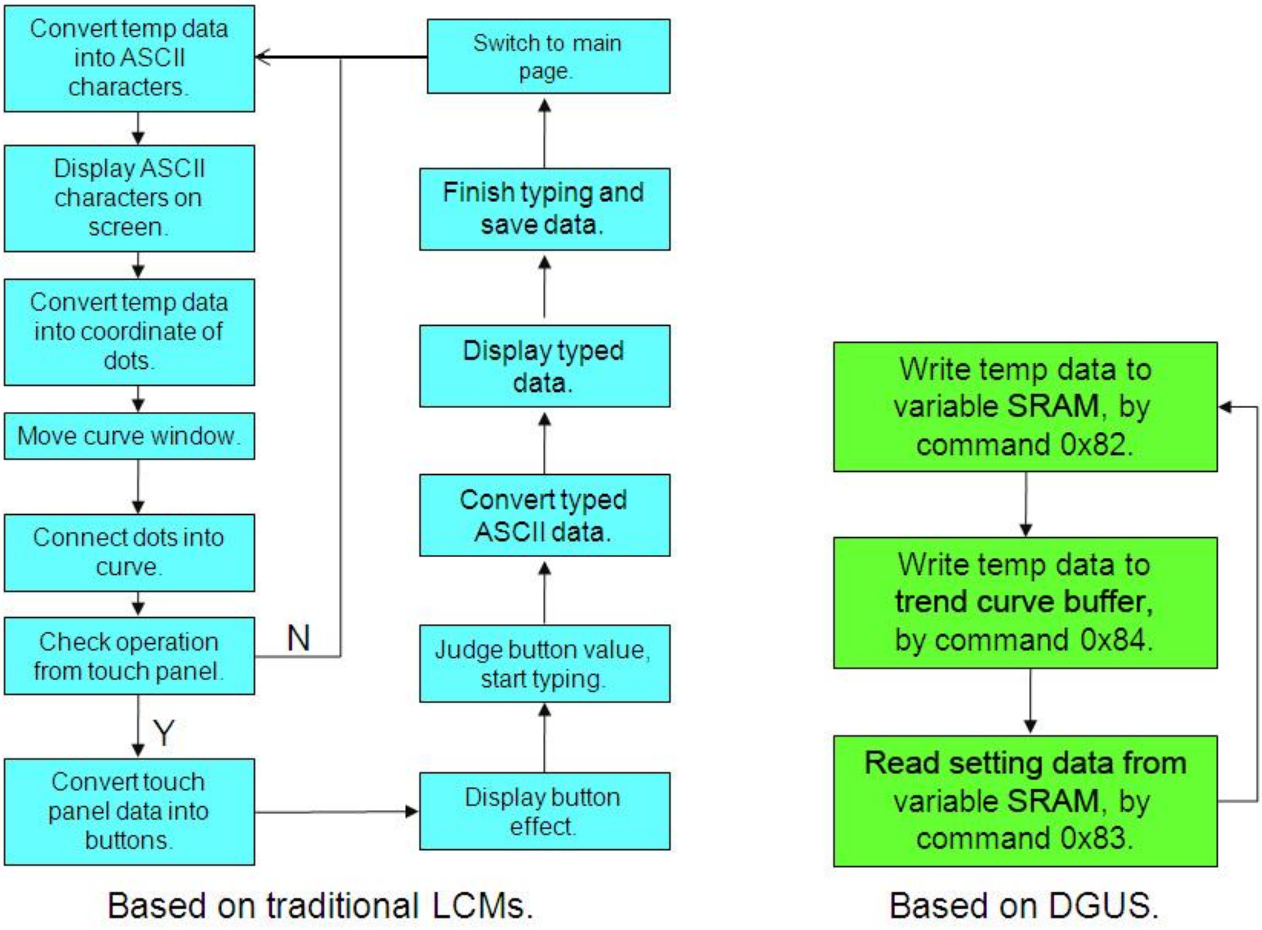

Convert temp data into ASCII characters.
Display ASCII characters on screen.
Convert temp data into coordinate of dots.
Move curve window.
Connect dots into curve.
Check operation from touch panel.
N
Y
Convert touch panel data into buttons.
Display button effect.
Judge button value, start typing.
Convert typed ASCII data.
Display typed data.
Finish typing and save data.
Switch to main page.
Based on traditional LCMs.
Write temp data to variable SRAM, by command 0x82.
Write temp data to trend curve buffer, by command 0x84.
Read setting data from variable SRAM, by command 0x83.
Based on DGUS.

Figure 6 Differences between DGUS and Module based on Terminal Assistant

5.2 Distribution of Variable Address

As shown in Figure 7, users are free to create and add new variable and assign a RAM address for this variable that depends on varied space. However, the key point for integer variables is that RAM space equals its initial address in set. See Figure 7.

。

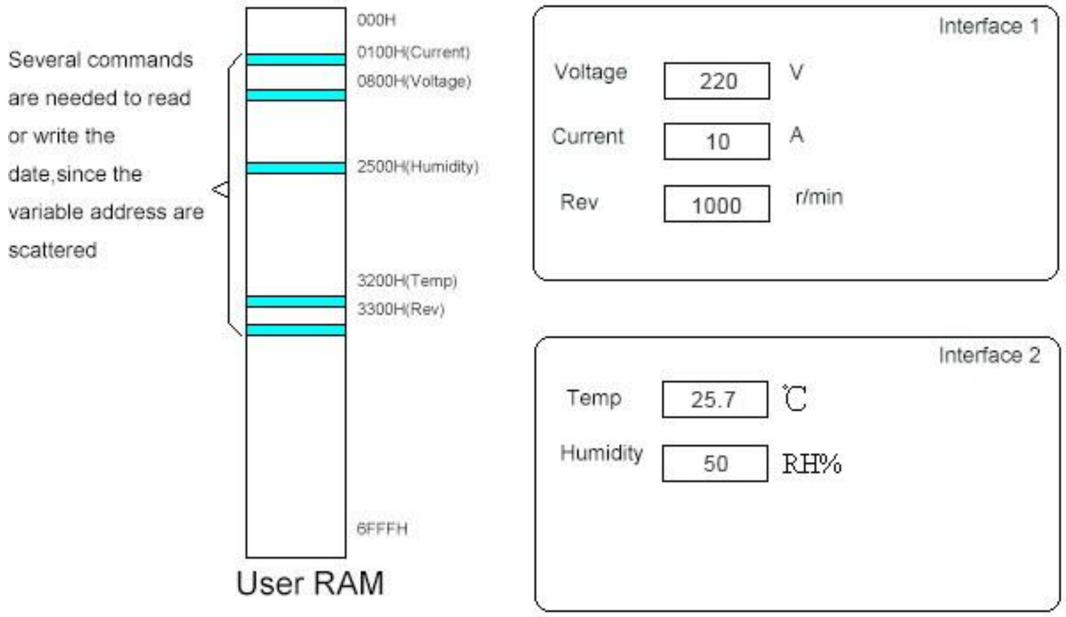


Figure 7 Principle of distribution of variables address

It is conclude from Chapter 3 that communications in DGUS is successive access. If distance between variable addresses is too large, inconveniences brings up because multiple commands needed for read-and-write operation. Therefore, the principle in Figure 8 is highly recommended for distribution of variables.

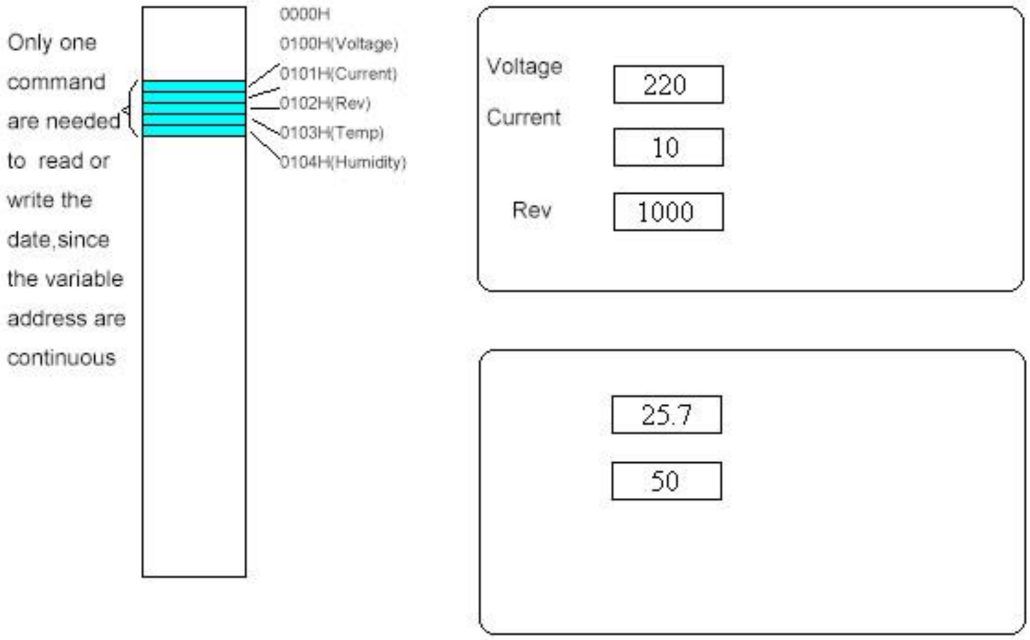


Figure 8 Distributions of Variables

When user in process of setting up a new project, Figure 9 as principle of variable space distribution is recommended. Please make sure to arrange a suitable size for RAM to each page in order to operation conveniences. Please see Figure 9.

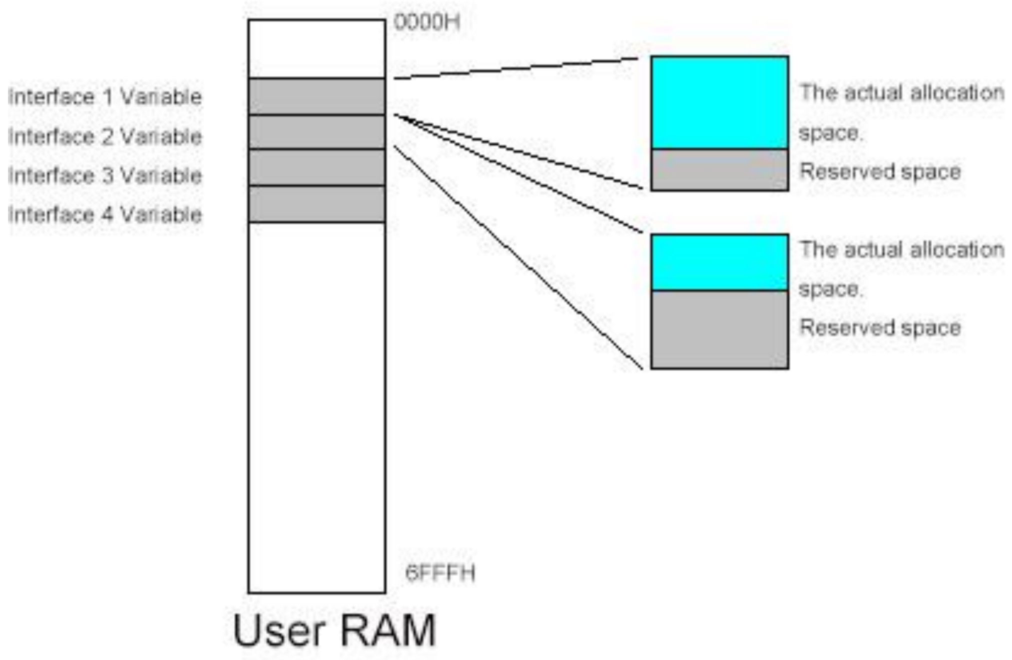


Figure 9 Principle of Variable Space Distribution

5.3 DGUS Running Cycle/Period

DGUS have 4 options as running time: 200ms, 180ms, 120ms, and 80ms.

As shown in Graph n, there is a timer in the DGUS. The timer generates an interrupt according to the "running period" configured by users. The DGUS program will refresh the variables on current user interface whenever an interrupt is generated.

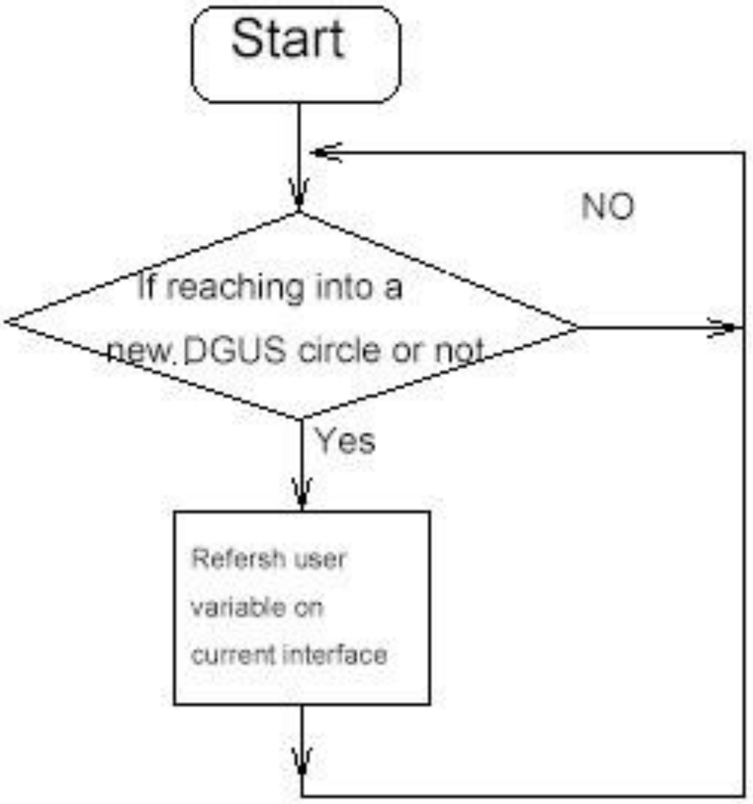

Start
NO
If reaching into a new DGUS circle or not
Yes
Refersh user variable on current interface

Figure 9 DGUS Running Cycle

5.4 Variable Refresh

As shown in Figure 10, DGUS keeps refreshing the data from associated RAM addresses, following the running cycle designated by user. At the same time slot, data was written into associated address of RAM as long as touch screen input activated. That is, DGUS would fulfill data communications between display interface and RAM periodically.

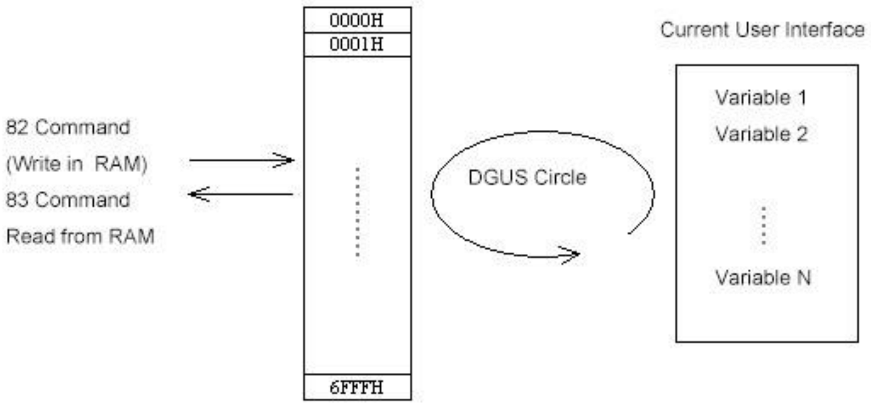


Figure 10 Variable Refresh

We will naturally come to the conclusion from the essence of DGUS Cycle elaborated above: user's read and write of the RAM area via serial port is random and barely has any time constraints, while it is periodical for when the variables are updated to the interface through RAM area. Under this premise, assuming the current DGUS cycle 200ms, if the user send more than two commands which update the same variable during one cycle, the last command will be the one which actually works. It follows that when the user is writing his own variable refresh program, the cycle of refreshing command sending is suppose to no less than the DGUS running cycle, because sending more command essentially is nonsense(excluding Curve Refresh).

5.5 Availability Ratio of Running Cycle

The CPU time consumed on refreshing all variables of one page is related to the amount and type of user-defined variables. But, generally, actual time costing on variable refresh is less than DGUS running period. Please refer to the figure below.

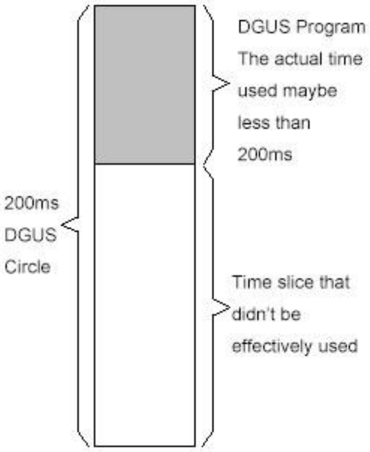


Figure 10 Availability Ratio of Running Cycle

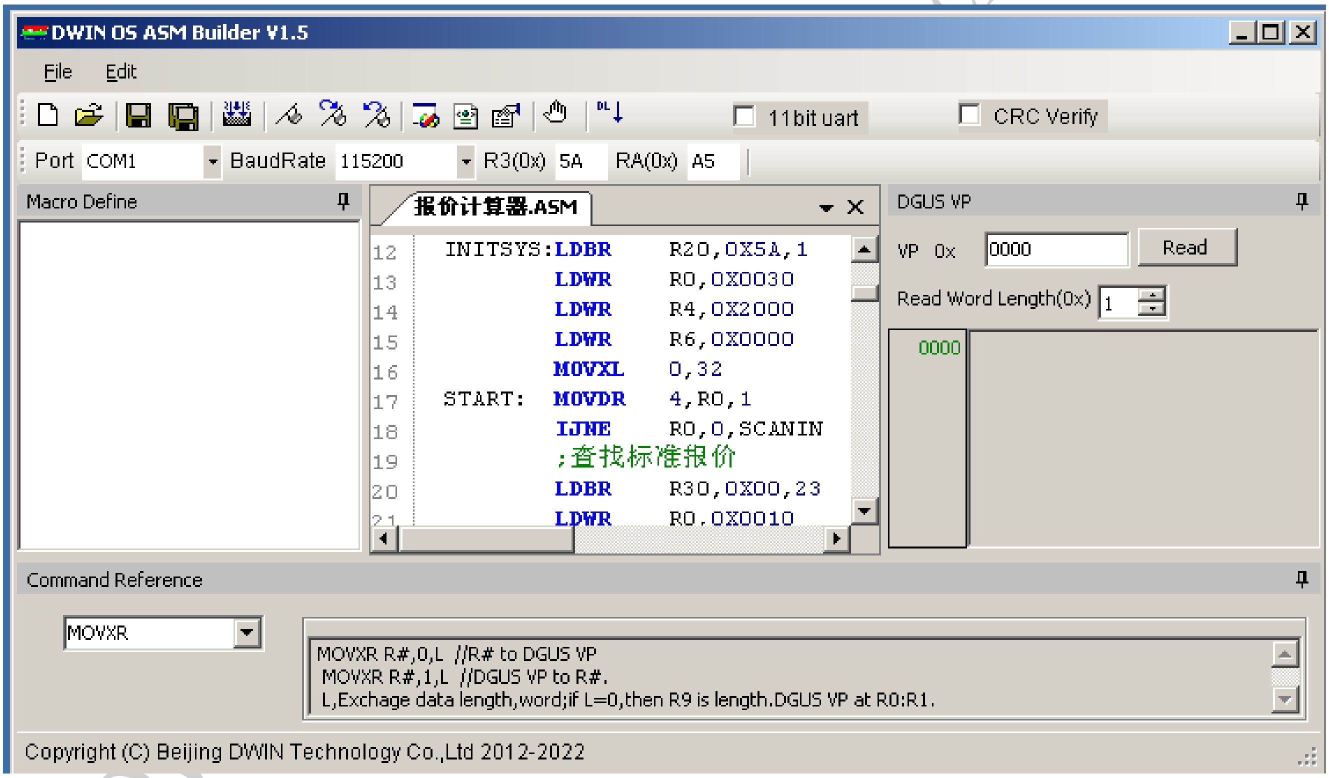
6. Advanced Development with DGUS--DWIN OS Programming

As stated before that DGUS is only for basic development with overall solution on purpose of passive displaying via serial port communications, let alone further complicated calculations in applications required.

In some of small and medium automation application, user may consider DGUS display as master unit if whole system is composed of relatively independent units with all functions equipped such devices like Modbus supported. In this network 485, DGUS display applied with machine to consist of a system based on DWIN OS running directly in targeting of replacement of users' original CPU.

DWIN OS embedding in DGUS is integrated with a lot of automated processing concerned software such as serial communication, CRC, solving linear equation, database operation, user algorithm etc. It adopts formula like scripting language projected, to make a fast secondary-development on user's end.

The interface of OS software shows as follows:



Port COM1 BaudRate 115200 R3(0x) 5A RA(0x) A5

VP 0x 0000

Read Word Length(0x) 1

0000

```
   INITSYS:LDBR     R20,0X5A,1
           LDWR     R0,0X0030
           LDWR     R4,0X2000
           LDWR     R6,0X0000
           MOVXL    0,32
   START:  MOVDR    4,R0,1
           IJNE     R0,0,SCANIN
           ;查找标准报价
           LDBR     R30,0X00,23
           LDWR     R0,0X0010
```

MOVXR

```
MOVXR R#,0,L  //R# to DGUS VP
 MOVXR R#,1,L  //DGUS VP to R#.
 L,Exchage data length,word;if L=0,then R9 is length.DGUS VP at R0:R1.
```

Copyright (C) Beijing DWIN Technology Co.,Ltd 2012-2022

Based on DGUS configured, DWIN OS have 256KB(32764 lines of code) space for coding while endless loop is not allowed to occurred because it runs through in each DGUS cycle(80/120/160/200mS)

Typically, the application of DWIN OS is working with Modbus protocol in order to substitute standard HMI, functioning as master unit. In this way not only gets down the cost but safety and reliability of system have been improved tremulously because standard HMI mostly is working under IPC or PC framework, combining with general operating system like Windows CE.

In addition to basic functions mentioned before, use either do programming with that in same as MCU.

6.1 Introduction to Programming

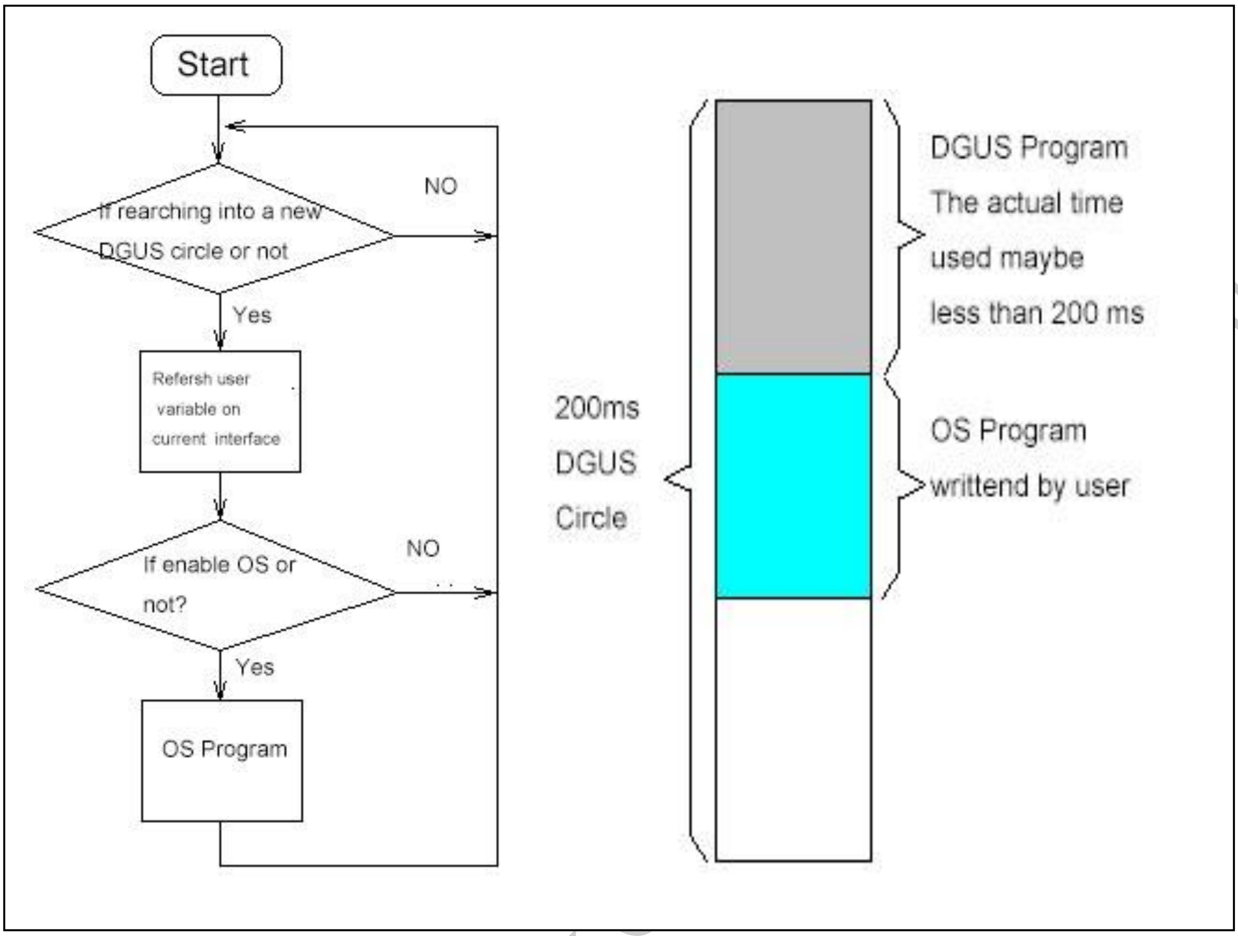

Start
If rearching into a new DGUS circle or not
NO
Yes
Refersh user variable on current interface
If enable OS or not?
NO
Yes
OS Program
200ms DGUS Circle
DGUS Program The actual time used maybe less than 200 ms
OS Program writtend by user

Figure 11 Running Cycle with DWIN OS involved

As shown on Figure 11, DGUS running cycle are in best use effectively with OS involved.

6.2 OS Register

OS has 256 pcs of registers named from R0 to R255, among which R0-R9 in specified use while others could be free to access.

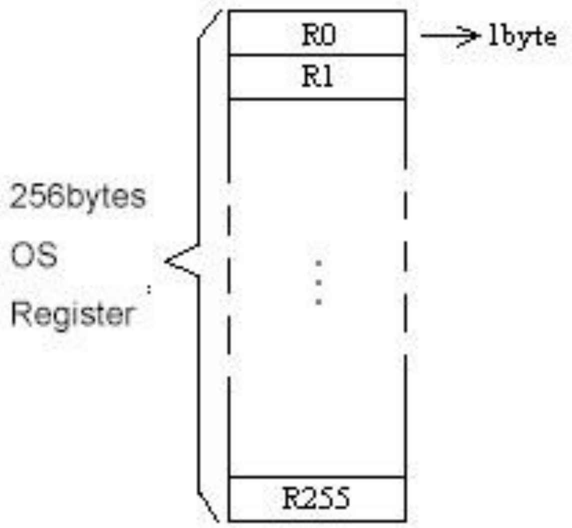


Figure 12 OS Register

6.3 Addressing Storage in OS Command OS

As shown in Figure 12, OS Command Set is capable of addressing storage space, RAM, FLASH and curve buffer

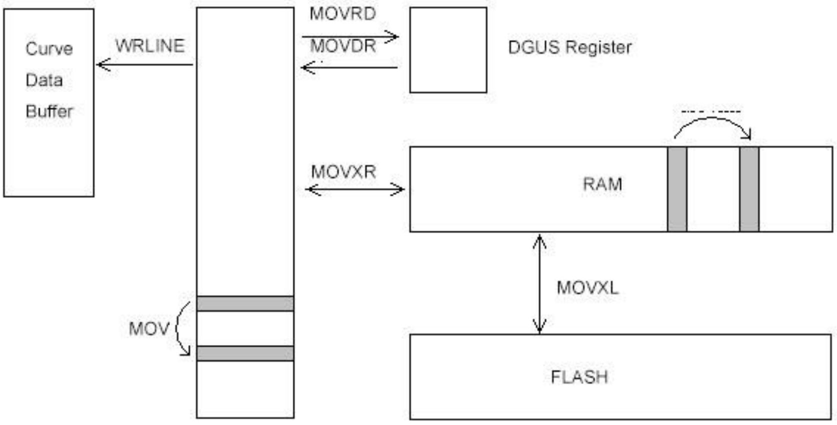


Figure 13 Register that allows OS to addressing

6.4 OS Program Structure

For any running period of DGUS, OS always starts from first line of edited program till command "END" happens, then exits and replay from the new entrance of DGUS running cycle. This process is analogous to PLC.

What need to be noticed that program structure of OS have to be compiled in sequence from top to bottom without endless loop in(such like while in Language C) and time-lapse segment, otherwise lags in responses or blue screen(reset) always. In order to makes complicated algorithm process, OS provided command like 'Compare Jump' and 'Function Call' to use for procedures operated.

6.5 Basic Protocol

DWIN OS Register Variable： R0-R255，256 Byte；

DGUS Register： Corresponding DGUS 0x80/0x81to access Register (0x00-0xFF)

DGUS Variable： Corresponding DGUS 0x82/0x83 to access variable storage(0x0000-0x6FFF)

Font Space： Corresponding (0x20-0x7F) ,24MB

- Pseudo Assembly Code:

EQU: substitute in compiling

Example:

```
PICID   EQU 3
WORD    EQU 2
MOVDR PICID, R10, WORD          //   Equal to "MOVDR 3, R10, 2"
```

Tips： EQU definition can be found from Macro Define in OS software

DB: Define a BYTE or a WORD

Example:

```
LDADR  TAB1    //   Save the data in TAB1 (24bits) to R5, R6, R7
TAB1:   DB  1,2,3,4
        DB  1000, 2000, 3000, 4000, -100
        DB  "A Sample String"
```

"Symbol ;": For comments only.

6.6 DWIN OS Command Set

R# stands for the single/multiple Registers with certain index. R0-R255.

<> present Instant Numbers. In assembly, 100, 0x64, 64H, 064H are as the same as 100in decimal.

Functions	**Commands** CMDs	**Parameters**	**Remark**
None	NOP		A Non-Operation. NOP
Data exchange between DGUS Variables & DWIN_OS Registers	MOVXR	R#, <MOD>, <NUM>	R#: DWIN_OS Register(s). <MOD>: 0: Register to Variable; 1: Variable to Register <NUM>: Data length in Words for exchange: 0x00-0x80 When <NUM>=0x00; Length is determined by R9. DGUS Variable Pointer defined by R0, R1. *e.g.:* ***MOVXR R20, 0, 2***
Load several 8 bit numbers to DWIN_OS Registers	LDBR	R#, <DATA>, <NUM>	R#: DWIN_OS Register(s). <DATA>: Data to load <NUM>: The indexes of Registers to hold the data, 0x00 present 256 Registers. *e.g.:* ***LDBR R8, 0x82, 3***
Load a 16 bit number to DWIN_OS Register	LDWR	R#, <DATA>	R#: DWIN_OS Registers. <Data>: Loaded data. *e.g.:* ***LDWR R8, 1000*** ***LDWR R8,-300***
Look up in Program Space (Program Space to DWIN_OS Registers)	MOVC	R#, <NUM>	R#: DWIN_OS Register(s). <NUM>: Data length of result for look up Table address pointer is defined by R5, R6, R7 *e.g.:* ***MOVC R20, 10***
Data Exchange between Registers in DWIN_OS	MOV	R#S, R#T, <NUM>	R#S: Source Register(s) OR registers R#T: Target Register(s) OR registers <NUM>: Data length for Exchanging. 0x00 indicates the length determined by R9. *e.g.:* ***MOV R8, R20, 3***
Data transfer from DWIN_OS Register to DGUS Register	MOVRD	R#, D#, <NUM>	R#: DWIN_OS Register(s). D#: DGUS Registers(s). <NUM>: Data length for Exchanging. *e.g.:* ***MOVRD R10, 3, 2***
Data transfer from DGUS Register to DWIN_OS Register	MOVDR	D#, R#, <NUM>	R#: DWIN_OS Register(s). D#: DGUS Registers(s). <NUM>: Data length for Exchanging. *e.g.:* ***MOVDR 3, R10, 2***
Data Exchange between DGUS Variables and Font Library	MOVXL	<MOD>, <NUM>	<MOD>:0= Transfer from Font Lib to DGUS Variables 1= DGUS Variable to Font Library <NUM>: Data length (Word) Address of DGUS Variable is defined by R0:R1 **Operation Mode of Font File: (MOD=0,1)** Font index is defined by R4 (0x20 – 0x7F), R5:R6:R7 is the operation starting address in Font Lib. Cancelled if out of boundary.

			Operation Mode of Database(MOD=2,3) Initial address configure by Reg. R4:R5:R6:R7 *e.g.:* *MOVXL 0, 300*
Data exchange between DGUS Variables	MOVXX	<NUM>	<NUM>: Data length (Word) If <NUM> is 0. it means the length to be defined by R8:R9. Address for DGUS Source Variable is R0:R1; Address for DGUS Target Variable is R2:R3 *e.g.:* *MOVXX 100*
Registers Indexed Addressing	MOVA		R2 defines the address for source Register(s) R3 defines the address for target Register(s) R9 defines the length for exchange, in BYTES. *e.g.:* *MOVA*
32bit integers addition	ADD	R#A, R#B, R#C	C=A+B, A,B are 32bit integers, C is 64bit integer. *e.g.:* *ADD R10, R20, R30*
32bit integers subtraction	SUB	R#A, R#B, R#C	C=A-B, A,B are 32bit integers, C is 64bit integer. *e.g.:* *SUB R10, R20, R30*
64bit MAC for long integers	MAC	R#A, R#B, R#C	C=(A*B+C), A, B are 32bit integers, C is 64bit integer. *e.g.:* *MAC R10, R20, R30*
64bit integers division	DIV	R#A, R#B, <MOD>	A/B, A is quotient, B is reminder. A and B are 64bit register. <MOD>: 0: The quotient will not be rounded. 1: The quotient WILL BE ROUNDED. *e.g.:* *DIV R10, R20, 1*
Expand Variable to 32bit	EXP	R#S, R#T, <MOD>	Expand the data in R#S to 32bit and save to R#T R#S: Source register(s) R#T: Target register <MOD>: Data type of R#S. 0=8Bit unsigned；1=8bit signed 2=16bit unsigned；3=16bit integer *e.g.:* *EXP R10, R20, 2*
32bit unsigned MAC	SMAC	R#A, R#B, R#C	C=A*B+C A and B are 16bit unsigned integer, C is 32bit unsigned integer. *e.g.:* *SMAC R10, R20, R30*
Register self-increase	INC	R#, <MOD>, <NUM>	R#=R#+NUM, unsigned self-increasing calculation <MOD>: Data type of R#; 0=8bit;1=16bit *e.g.:* *INC R10, 1,5*
Register self-decrease	DEC	R#, <MOD>, <NUM>	R#=R#-NUM, unsigned self-decreasing calculation <MOD>: Data type of R#; 0=8bit;1=16bit *e.g.:* *DEC R10, 0,1*
Load Address	LDADR	<ADRH>, <ADRM>, <ADRL>	Load <ADRH:ADRM:ADRL> to R5:R6:R7 *e.g.:* *LDADR TAB* *LDADR 0x123456*
Logical Calculation: AND	AND	R#A, R#B, <NUM>	A=A AND B, Logical “AND” calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES

			e.g.: AND R10, R20,1
Logical Calculation: OR	OR	R#A, R#B, <NUM>	A=A OR B, Logical "OR" calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES *e.g.: OR R10, R20,1*
Logical Calculation: XOR	XOR	R#A, R#B, <NUM>	A=A XOR B, Logical "XOR" calculation for series of Registers. <NUM>: Data length of R#A, R#B in BYTES *e.g.: XOR R10, R20,1*
Integer Linear Equation	ROOTLE	00, 00, 00	Calculate the Y value according to the given X value, which is a point on the line defined by (X_0, Y_0) and (X_1, Y_1) in 16bit integer. Input: X=R10, X_0=R14, Y_0=R16, X_1=R18, Y_1=R20 Output: Y=R12 *e.g.: ROOTLE*
ANSI CRC-16	CRCA	R#S, R#T, R#N	Perform ANSI CRC-16 calculation on series of Registers. ANSI CRC-16(X16+X15+X2+1) R#S: Registers for Input R#T: Registers to hold the result, 16bit, LSB mode. R#N: Save the length for CRC byte data, 8bit *e.g.: CRCA R10, R80, R9*
CCITT CRC-16	CRCC	R#S, R#T, R#N	Perform CCITT CRC-16 calculation on series of Registers. CCITT CRC-16(X16+X12+X5+1) R#S: Registers for Input R#T: Registers to hold the result, 16bit, MSB mode. R#N: Save the length for CRC byte data, 8bit *e.g.: CRCC R10, R80, R9*
Read MODBUS data frame from COM_0_Rx_FIFO	RMODBUS	R#A, R#T, R#C	Check the FIFO in COM_0 received valid MODBUS data frame, if yes, will move the data to register and clear the Receiving FIFO. R#A: Specified Registers will store the first 3 bytes of MODBUS data pack (Address, CMDs, and Data Length). ***If length is 0x00, it present no length matching, data after it (the 4th byte) indicate the length exclude address, instruction and CheckSum.*** R#C: Register for return value/status. It will hold the data returned; 0x00 indicates no valid MODBUS data frame is received; 0xFF stands for valid MODBUS data frame is received and stored in R#T registers. R#T: Target register to store the MODBUS data after validation is successful. *e.g.: RMODBUS R10, R20, R13*
Bit decomposition	BITS	R#, <VP>	Decompose the 8 bits in R# to 8 DGUS Variable (Byte) specified by VP. Bit 1 becomes 0x0001, bit 0 becomes 0x0000. R#: The register need to decompose.

			<VP>: DGUS VP address. *e.g.: BITS R10, 0x2000*
Bit integration	BITI	R#, <VP>	Integrate 8 DGUS Variable (Byte) specified by VP into 1 byte Bit Variable (MSB). 0x0000 becomes bit 0, other value become bit 1. R#: The register need to decompose. <VP>: DGUS VP address. *e.g.: BITI R10, 0x2000*
HEX to ASC	HEXASC	R#S, R#T, <MOD>	R#S: 32bit Integer needs to be converted to ASCII R#T: Target registers for ASCII after conversion. <MOD>: Convert Mode. High 4 bits indicate the length of integers; Lower 4 bits indicates the length of decimals. The ASCII string after conversion is signed, right aligned; empty slots will be filled by 0x20. For data 0x12345678: <MOD>=0x62, result is +054198.96; <MOD>=0xF2, result is +3054198.96 *e.g.: HEXASC R20, R30, 0x62*
Sequence comparison	TESTS	R#A, R#B, <NUM>	Compare the values in R#A and R#B by sequence. If not match, return the current address of R#A to R0 register; If match, return 0x00 to R0 register. R#A: Starting register for register series A; R#B: Starting register for register series B; <NUM>: max length for data comparison. *e.g.: TESTS R10, R20, 16*

Configuration for COM$_1$	COMSET	<MODE>, <BSH>, <BSL>	Set the configuration for Serial port COM$_1$:
			<MODE>:
			0x00=N81 mode;
			0x01=E81 mode;
			0x02=O81 mode;
			0x03=N82 mode;
			<BSH:L>: Factors for Baud Rate. Value is 6250000/(Desired Baud Rate). The receive FIFO will be cleared when you set the baud rate.
			e.g.: COMSET 0, 54

Conditional bitjump	JB	R#, <BIT>, <NUM>	Evaluate the <bit> in R# register. If 1, jump to <NUM>; if 0, proceed to next instruction. R#: The register contains data to be evaluated. <Bit>: the index of the bit to be evaluated. 0x00-0x0F (MSB). <NUM>: Jump position control. Num.7 control the direction: 1 = forward; 0=backward; result for NUM&0x7F indicates the number of instructions to jump. *e.g.: JB R10, 15,TEST1* *NOP*

			TEST1: ADD R8,R12, R16
Variable conditional jump (Not Equal)	CJNE	R#A, R#B, <NUM>	Compare the value of 2 8bit registers (R#A and R#B). If equal, proceed to next instruction; if not equal, jump to <NUM>. *e.g.: TEST1: NOP* *INC R10, 0, 1* *CJNE R10,R11, TEST1*
Integer conditional jump (Less than)	JS	R#A,R#B, <NUM>	Compare the value for 2 bit integer in R#A and R#B. If A>=B, proceed to next instruction; If A<B, jump to <NUM> *e.g.: JS R10, R12, TEST1* *NOP* *TEST1: NOP*
Value conditional jump (Number and Variable)	IJNE	R#, <INST>, <NUM>	Compare the value in 8 bit Register and a instant Number <INST>. If equal, process to next instruction; if not equal, jump to <NUM>. *e.g.: IJNE R10, 100, TEST1* *NOP* *TEST1: NOP*
Compulsorily terminate current input thread	EXIT	R#A, R#B, 00	Compulsorily terminate current input thread. R#A: decide to change page. 0x00= Don't change; 0x01= change page. R#B: The Picture ID to return back (16bit) *e.g.: EXIT R10, R11*
Return	RET	00, 00, 00	Return to main program by calling this function in sub-program. *e.g.: RET*
Call sub-function	CALL	<PCH>, <PCL>, 00	Call sub-program in position of program counter <PCH:L> (0x0000-0x7FFB). Maximum support 32 levels of Program Nesting. *e.g.: CALL TEST*
Direct Jump	GOTO	<MOD>, <PCH>, <PCL>	<MOD>=0x00: Jump to <PCH:L> <MOD>=0x01: Relatively Jump to (PC+1+<PCH:L>) <MOD>=0x02: Relatively Jump to (PC+1-<PCH:L>) <PCH:L>=0xFFFF indicates the value is in R0:R1 *e.g.: GOTO TEST1* *NOP* *TEST1: NOP*
Data send by Serial port	COMTXD	<COM>, R#S, R#N	Send data to the specified serial port. <COM>: Serial port select. 0=COM1(DGUS User port); 1=COM2(System reserved) R#S: the registers hold the data to send R#N: The registers contained the byte length info to send. If 0x00 indicates sending 256 bytes of data. *e.g.: COMTXD 0, R10, R9*
Print via serial port	CPRTS	<COM>, <VPH>,	Check the content for print is exist at the DGUS

		<VPL>	Variables which <VP> pointed to, if yes, print it via serial port. <VP> correspond to the VP value defined in 0xFE07 in DGUS LCMs. Printing Status will be cleared after printing is done. <COM>: Serial port select. 0=COM1(DGUS User port); 1=COM2(System reserved) *e.g.:* ***CPRTS 0, 0x2000***
Check COM$_0$_Rx_FIFO	RDXLEN	00, R#, 00	Return the length in byte (0-253) for received data in FIFO buffer are for COM$_1$ and save it in R# register. 0x00 indicates empty. *e.g.:* ***RDXLEN 0, R10***
Read from COM$_0$_RX_FIFO	RDXDAT	00, R#A, R#B	Read <R#B> (1-253) bytes from FIFO buffer of COM1 and move it to R#A registers. FIFO’s length will adjust automatically. *e.g.:* ***RDXDAT 0, R11, R10***
Direct send data via Serial-port	COMTXI	00, R#, <NUM>	Send the data inside <NUM> series of Registers indexed with R#. *e.g.:* ***COMTXI 0, R20, 16***
Read the content for current Input Method	SCAN	R#, <NUM>, 00	Load <Num> of character that inputted under current input method to register at (R#+1), and R# will hold the length info. Characters are counted backward from the current cursor. *e.g.:* ***SCAN R20, 6***
Write Curve buffer for specified channel	WRLINE	R#S, R#1, <CH>	Calculate the result of numbers of 16bit unsigned integers by R#S added an offset value V_BIAS then write it to the buffer for curve defined by <CH> (0x00-0x07). R#I indicate the register hold 3 bytes, N, V_BIAS. *e.g.:* ***WRLINE R80, R10, 2***
Erase specified Font Lib	ERASE	<L_ID>, 5A, A5	<L_ID>: The Font Library ID to be erased. From 0x20-0x7f *e.g.:* ***ERASE 40***
Sum of addition (Error detection)	SUMADD	R#S, R#T, R#N	Calculate the sum of data in 1 byte for error detection. R#S: Registers to calculate (Input) R#T: Result in 1 byte, 8 bit. R#N: Register for length of series. 8 bit. *e.g.:* ***SUMADD R10, R80, R9***
Sum of addition with Carry (Error detection)	SUMADDC	R#S, R#T, R#N	Calculate the sum of data in 1 byte with carry for error detection. R#S: Registers to calculate (Input) R#T: Result in 1 byte, 8 bit. R#N: Register for length of series. 8 bit. *e.g.:* ***SUMADDC R10, R80, R9***
Sum of XOR calculation (Error detection)	SUMXOR	R#S, R#T, R#N	Calculate the result for XOR operation for data in 1 byte for error detection.

			R#S: Registers to calculate (Input) R#T: Result in 1 byte, 8 bit. R#N: Register for length of series. 8 bit. *e.g.:* ***SUMXOR R10, R80, R9***
Convert HEX to Compressed BCD code	HEXBCD	R#S, R#T, <MOD>	Convert data in HEX to compressed BCD code. 0x1000 will be converted to 0x10, 0x00. R#S: Initial address for registers stored data in HEX R#T: Starting address for registers stored data for result in BCD code. <MOD>: High 4 bits indicate the numbers of byte for HEX data. (0x01-0x08); Low 4 bits indicate the numbers of byte for BCDoutput. (0x01-0x0A). *e.g.:* ***HEXBCD R10, R80, 0x23***
Convert Compressed BCD to HEX code	BCDHEX	R#S, R#T, <MOD>	Convert data in compressed BCD to HEX. 0x1000 will be converted to 0x3E8 (1000). R#S: Initial address for registers stored data in compressed BCD code. R#T: Starting address for registers stored data for result in HEX. <MOD>: High 4 bits indicate the numbers of byte for compressed BCD data. (0x01-0x0A); Low 4 bits indicate the numbers of byte for HEX output. (0x01-0x08). *e.g.:* ***BCDHEX R10, R80, 0x32***
Convert ASCII string to HEX characters	ASCHEX	R#S, R#T, <LEN>	Convert ASCII String to signed 64 bit HEX data. R#S: Starting address for registers stored ASCII Strings R#T: A 64bits register to hold the output 64bit Hex data. <LEN>: The length for ASCII string, include sign bit and decimal point. 0x01-0x15. *e.g.:* ***ASCHEX R10, R80, 0x05***
Read DL/T645 data frame from COM$_0$_Rx_FIFO	RD645	R#A, R#T, R#C	Check the FIFO in COM$_0$ received valid DL/T645 data frame, if yes, will move the data to register and clear the Receiving FIFO. R#A: Address Register stores 6 bytes of data. R#C: Register for return value/status. It will hold the data returned; 0x00 indicates no valid DL/T645 data frame is received; 0xFF stands for valid DL/T645 data frame is received and stored in R#T registers. R#T: Target registers to store the DL/T645 data after validation is successful in format: *CMDCode + DataLength + data* *e.g.:* ***RMODBUS R10, R20, R13***
Time sequence funcion	TIME	R#A,R#B,<MOD>	R#A and R#B：register for saving 6bytes time variables which format is BCD; MOD=0, calculating A=A-B，to count relative value between two time data. A MUST BE greater than B, when A<B, 0xEF which is the first word of R#A was returned automatically

			without calculation
			MOD=1，compute A=B-RTC；
			MOD=2，compute A=RTC-B。
			e.g.: TIME R0,R10,0

Add display variables	ADDL14	R#A,R#B,<MOD>	R#A: register for saving one display variable(32Bytes); R#B: site position that added for variables, 0x00-0x1F, maximum 32 pcs of variables can be added. <MOD>： 0x5A= Add to designated position Other=Delete designated position and null for R#A at this point. e.g.: ADDL14 R80,R81,0x5A
Square root Computing	SQRT	R#A,R#B	Compute a square root of 64-bit unsigned number R#A and save to R#B R#A: Saved a 8 byte unsigned number; R#B: Saved a 4 byte unsigned number e.g.: SQRT R80,R90
End of the program	END	FF, FF, FF	*e.g.: END*

7. Touch Config. File（13.BIN）

The Touch Config. File, which contains several touch commands, can be generated by DGUS_SDK. Each command occupies 16, 32 or 48 bytes and includes 6 parts.

Part	Definition	Data Length	Description
1	Pic_ID	2	Picture ID
2	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye). Xs=FFFF: the function of the button will be activated by key code in register 0xF4, set Ys_H as key code then disable press-down effect.
3	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
4	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
5	TP_Code	2	Touch key code: 0xFF**: Invalid key code. 0xFE**: Function buttons, e.g.: 0xFE00 indicates it’s a Variable Data Input button. 0x00**: Touch key code in ASCII format, e.g.: 0x0031 means “1”.
6	TP_FUN	16/32	When TP_Code = 0xFE**, parameters of functional buttons.

7.1 Touch Control/Keyboard Control Overview

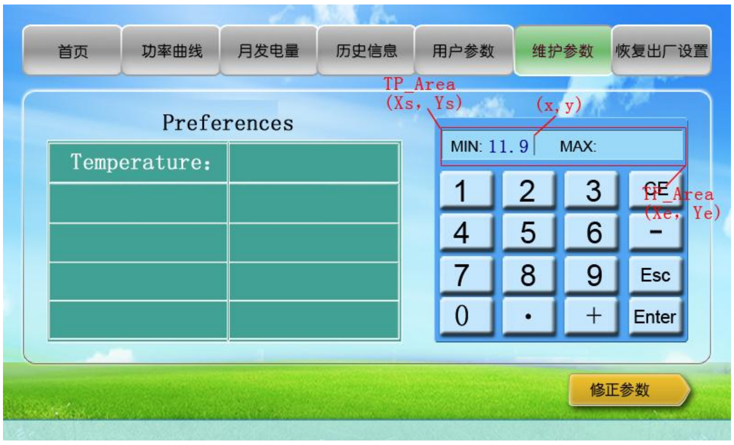
Num.	Key Code	Function	Description
01	00	Variable Data Input	Integer and fixed-point decimals to designated variable space
02	01	Popup Window	Touch to active a popup window and return to the top of menu
03	02	Incremental Adjustment	Button for +/- adjustment, both steps and up/down limits are allowed. Circulation set in Range 0-1 for check box in options.
04	03	Slider	Slider operation for data input and steps set available
05	04	RTC	Touch Keyboard to Set RTC for entry calendar(Y/M/D/H/M/S)
06	05	Return Value	Send pressed value to variables upon button touched, bit-variable included.
07	06	Text	Character input and cursor editing supported.

			ASCII supported including 8bit coding Unicode and multiple languages mixed allowed with OS
08	07_00	Write from register to variable space	Offer approaches to revise the register via touch screen in order to control hardware indirectly. E.g.: read out backlight to variable, then send back it for adjustment of brightness.
09	07_01	Write from variable space to register	
10	07_02	Images transfers to bitmap in vertical	Transfer designated colored bitmap to single bitmap, then save it in VP. Mainly used for printing out of current page.
11	07_05	Images transfer s to bitmap in horizontal	
12	07_03	Data send to COM1	Data in designated VP was sent to COM1 if touch the screen.
13	07_04	Date send to COM2	Data in designated VP was sent to COM2 if touch the screen. COM2 is used for extended functions of display only.
14	07_06	Coordinates of touch screen send to COM2	Coordinates in touch area was sent to COM2 if touch the screen. COM2 is used for extended functions of display only.

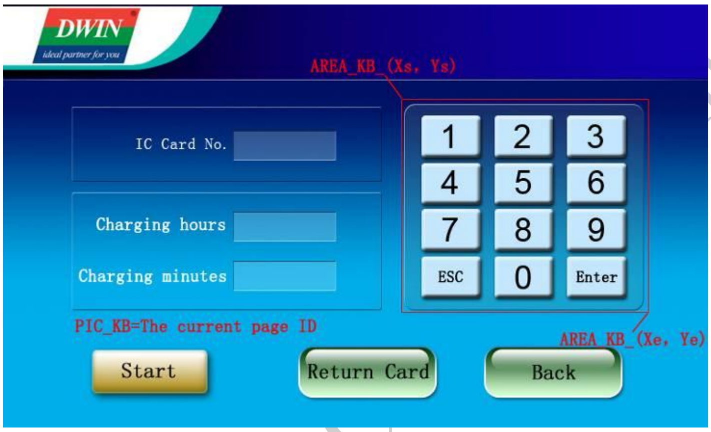
7.2 Variable Data Input（0x00）

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE00
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	V_Type	1	Inputted variables format. 0x00: integer (word). 0x01: long integer (double word). 0x02: unsigned byte (high byte of VP address). 0x03: unsigned byte (low byte of VP address). 0x04: double long integer, -9223372036854775808 to 9223372036854775807.
0x14	N_Int	1	Integer digits, e.g.: input 1234.56, so N_Int = 0x04.
0x15	N_Dot	1	Decimal digits, e.g.: input 1234.56, so N_Dot = 0x02.
0x16	(x,y)	4	Position of cursor, right alignment.
0x1A	Color	2	Font color.
0x1C	Lib_ID	1	Address of ASCII Font file, 0x00: default #0 ASCII font.
0x1D	Font_Hor	1	Font size, by pixel numbers in X-direction.
0x1E	Cursor_Color	1	Cursor color. 0x00: black, others: white.
0x1F	Hide_En	1	0x00: encrypted display, others: unencrypted display.
0x20	0xFE	1	0xFE
0x21	KB_Source	1	0x00: call keypad from current page. Others: call keypad from designated page.
0x22	PIC_KB	2	Picture ID of keypad. Null if KB_Source = 0x00.
0x24	AREA_KB	8	Cut area for keypad (Xs, Ys) (Xe, Ye). Null if KB_Source = 0x00.
0x2C	AREA_KB_Position	4	Paste position of keypad on current page. Null if KB_Source = 0x00.
0x30	0xFE	1	0xFE
0x31	Limits_En	1	0xFF: enable range limit of inputting value, null if over range. Others: disable range limit.
0x32	V_min	4	Floor of range (long integer, 4 bytes).
0x36	V_max	4	Ceiling of range (long integer, 4 bytes).
0x3A	Reserve	6	0x00 fixed.

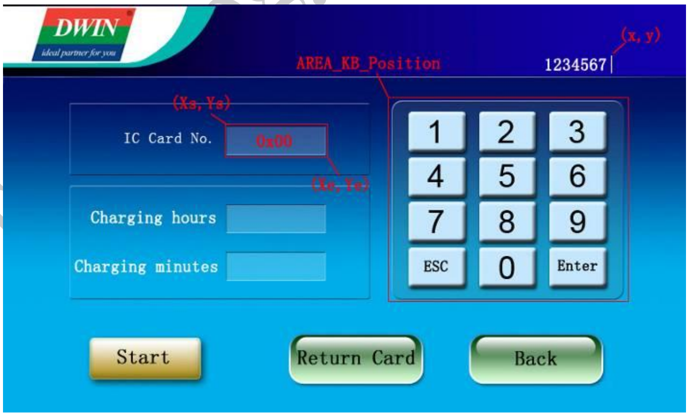
Valid key codes: 0x0030 – 0x0039 (Number 0 - 9), 0x002E (.), 0x002D (+/-), 0x00F0 (cancel), 0x00F1 (confirm),



Call keypad from current page (KB_Source = 0x00).



Call keypad from designated page (KB_Source = 0x01): keypad is activated after click.

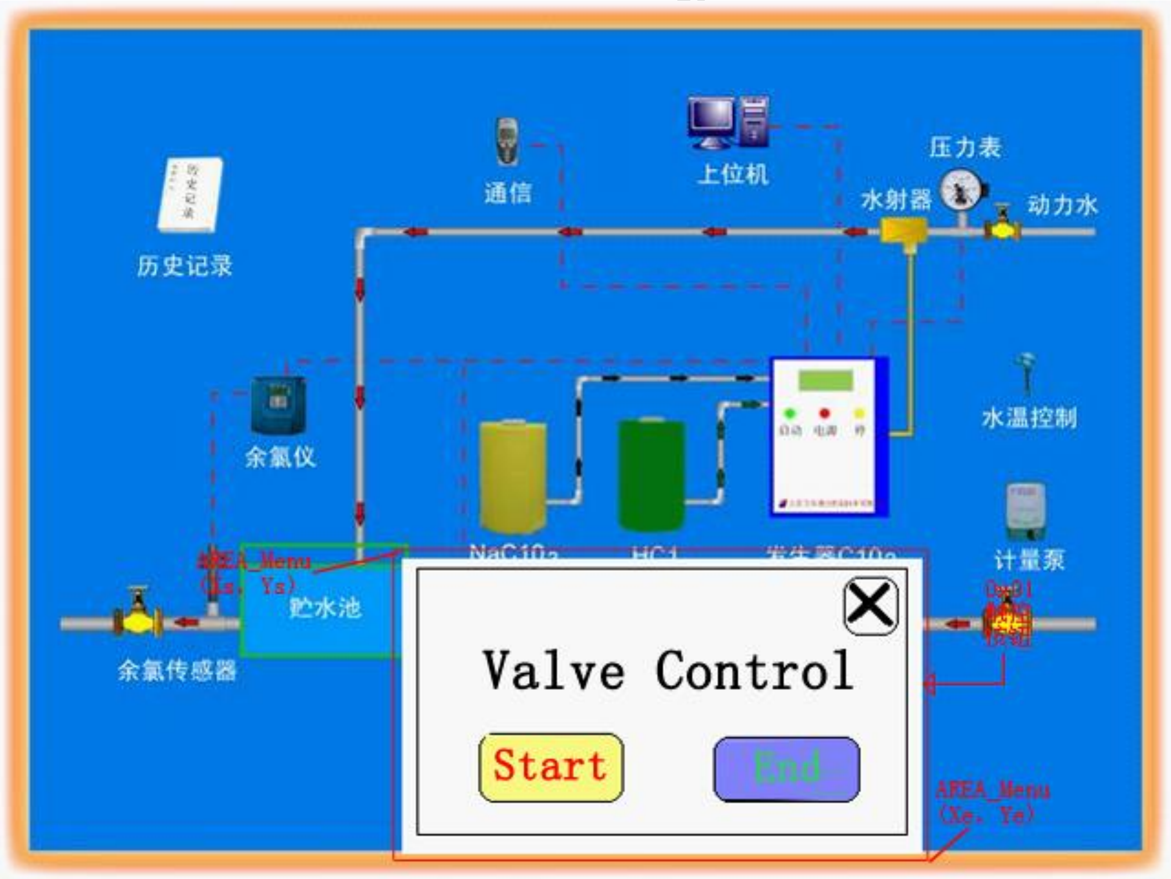


Call keypad from designated page (KB_Source = 0x01): page with keypad.

7.3 Popup Window (0x01)

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE01
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	VP_Mode	1	Key code format. 0x00: write key code in VP address (word). 0x01: write low byte of key code in high byte of VP. 0x02: write low byte of key code in low byte of VP. 0x10-0x1F: write data from last bit of key code into designated bit of VP address. (0x10 corresponds to VP.0, 0x1F corresponds to VP.F)
0x14	Pic_Menu	2	Picture ID of popup window.
0x16	AREA_Menu	8	Cut area for popup window: (Xs, Ys) (Xe, Ye).
0x1E	Menu_Position_X	2	Paste position of popup window: X coordinate.
0x20	0xFE	1	0xFE
0x21	Menu_Position_Y	2	Paste position of popup window: Y coordinate.
0x23	NULL	13	0x00 fixed.

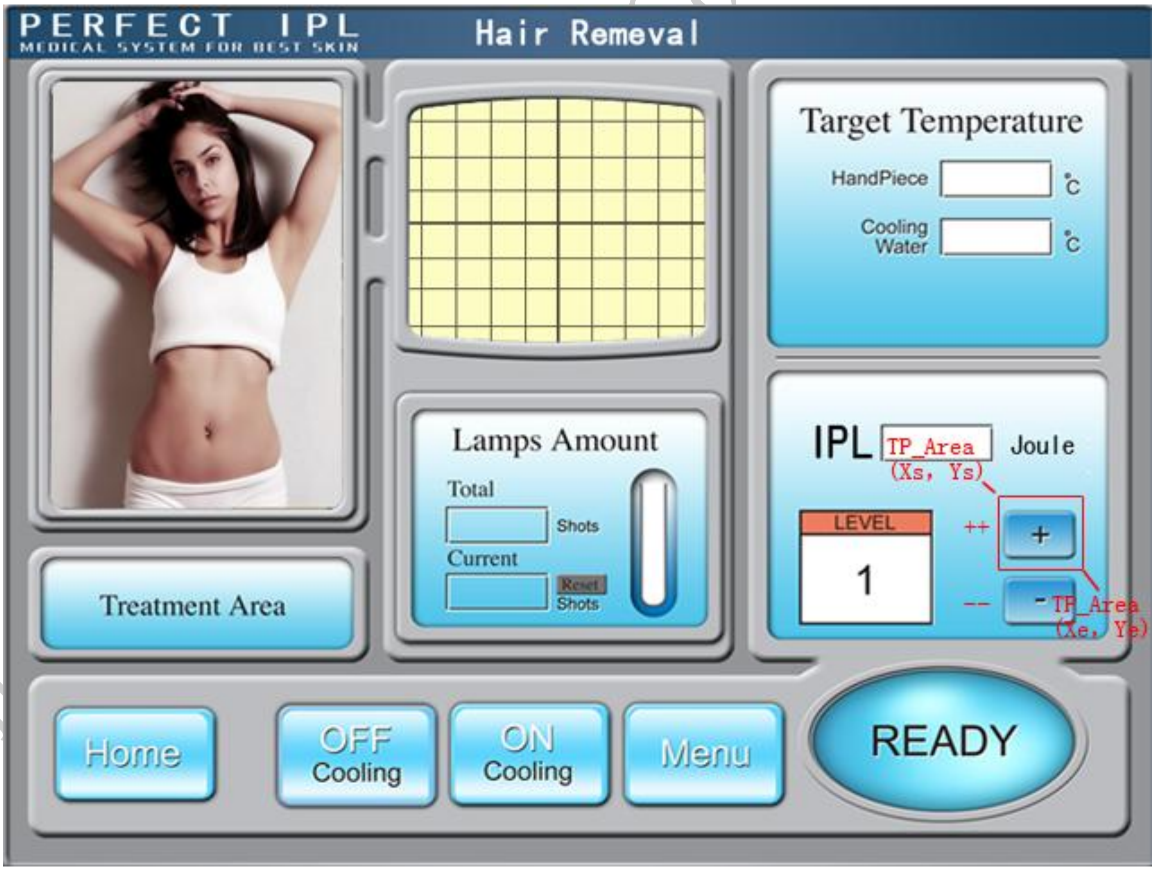
Valid key code: 0x0000 – 0x00FF, 0xFF: cancel.



Key code (0x0000 – 0x00FE) of “Start” and “End” button will be written in VP address. Designate 0x00FF key code for “Esc” button. By the way, drop-down menu also could be designed by this command.

7.4 Incremental Adjustment (0x02)

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	0xFF**.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE02
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	VP_Mode	1	Adjust value mode. 0x00: adjust value in VP address (integer). 0x01: adjust value in high byte of VP address (unsigned byte). 0x02: adjust value in low byte of VP address (un signed byte). 0x10-0x1F: adjust value in designated bit of VP address. (0x10 corresponds to VP.0, 0x1F corresponds to VP.F) Step size must be 0 or 1.
0x14	Adj_Mode	1	Adjust mode. 0x00: --, others: ++.
0x15	Return_Mode	1	Loop. 0x00: disable loop, others: enable loop.
0x16	Adj_Step	2	Step size: 0x0000-0x7FFF.
0x18	V_Min	2	Floor of range (integer), low byte is valid when VP_Mode is 0x01 or 0x02.
0x1A	V_Max	2	Ceiling of range (integer), low byte is valid when VP_Mode is 0x01 or 0x02.
0x1C	Key_Mode	1	0x00: continuous press to adjust successively 0x01: one-step adjust as pressing
0x1D	NULL	3	0x00 fixed.



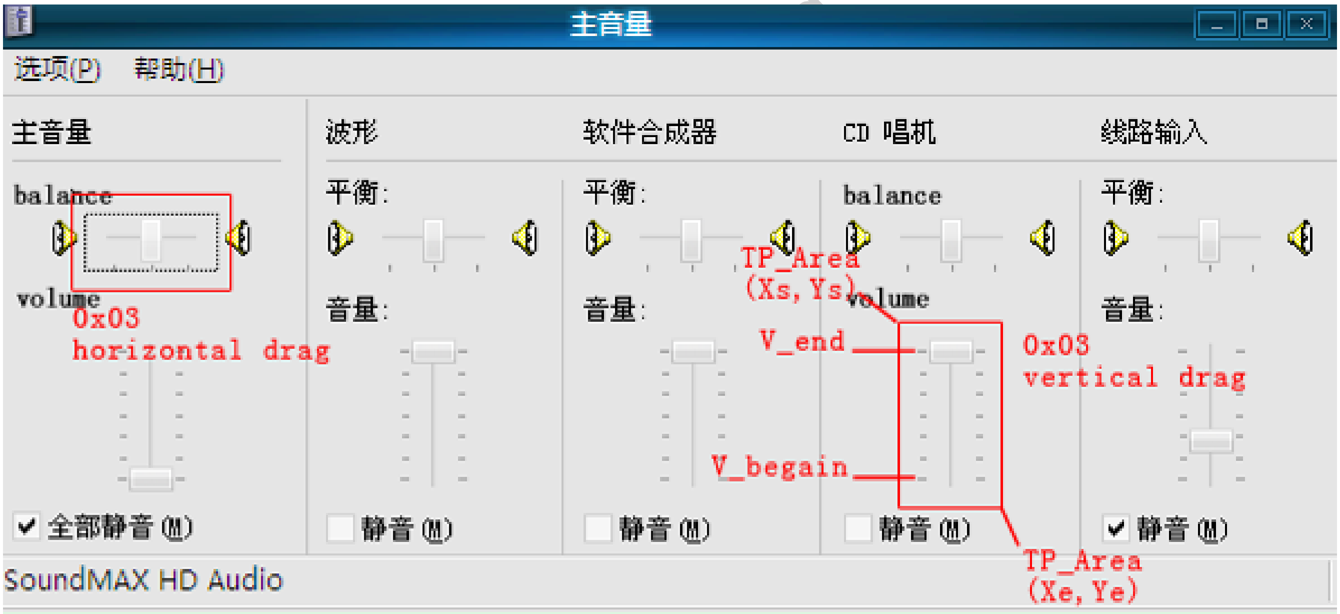
Set two buttons for "+" (Adj_Mode=0x01) and "–" (Adj_Mode=0x00).

Set range as 0 – 1, and match up with function Variable Icon, check function will be achieved. (Press once to pick up and twice to cancel.)

7.5 Slider Adjustment (0x03)

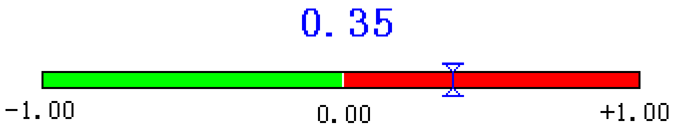
Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	0xFF**
0x0C	Pic_On	2	0xFF**
0x0E	TP_Code	2	0xFE03
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	Adj_Mode	1	➢ First 4 bits define data format. 0x0*: adjust value in VP address (integer). 0x1*: adjust value in high byte of VP (unsigned byte). 0x2*: adjust value in low byte of VP (unsigned byte). ➢ Last 4 bits define sliding mode. 0x*0: horizontal. 0x*1: vertical.
0x14	Area_Adj	8	Effective sliding area (Xs, Ys) (Xe, Ye), must be equal to value of TP_Area.
0x1C	V_begin	2	Start return value (integer).
0x1E	V_end	2	End return value (integer).

Slider is activated after holding for 0.5 second to avoid maloperation.



Slider function is applied to indicate current volume (**refer to Chapter 8.2.3**).

Values can also be indicated by <Data Variable> function to have current value (**refer to Chapter 8.3.1**).

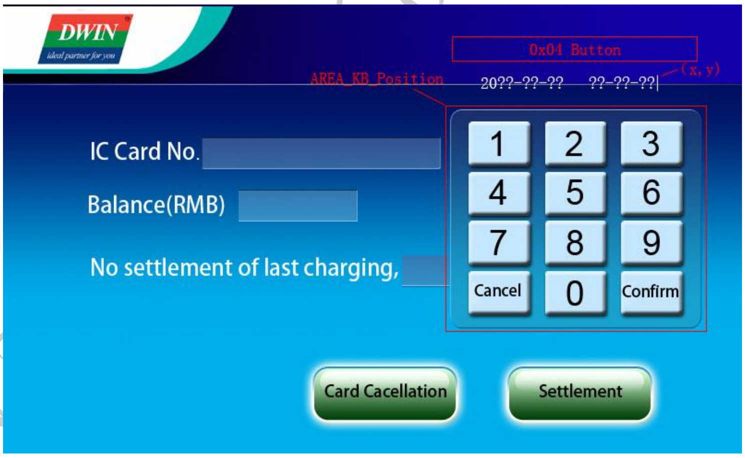


Slider Adjustment does not support machine buttons (key code in register 0X4F).

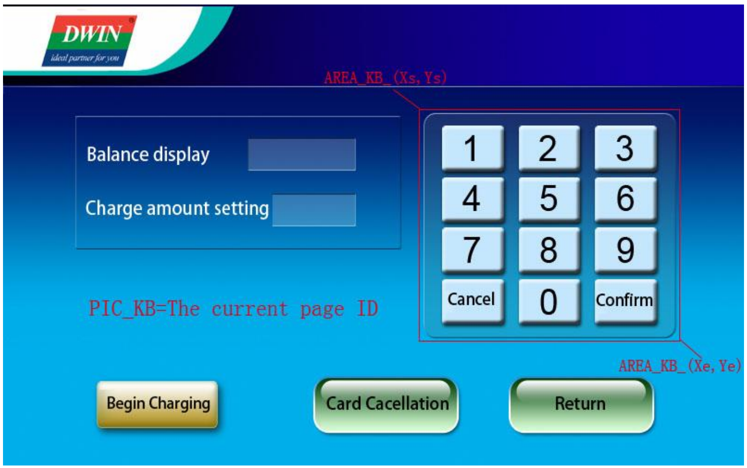
7.6 The RTC Settings (0x04)

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE04
0x10	0xFE	1	0xFE
0x11	0x00 00 00	3	0x00 00 00 fixed.
0x14	(x, y)	4	Position of cursor, right alignment.
0x18	Color	2	Font color.
0x1A	Lib_ID	1	Address of font file.
0x1B	Font_Hor	1	Font size, by pixel numbers in X-direction.
0x1C	Cursor_Color	1	Cursor color. 0x00: black, others: white.
0x1D	KB_Source	1	0x00: call keypad from current page. Others: call keypad from designated page.
0x1E	PIC_KB	2	Picture ID of keypad. Null if KB_Source = 0x00.
0x20	0xFE	1	0xFE
0x21	AREA_KB	8	Cut area for keypad (Xs, Ys) (Xe, Ye). Null if KB_Source = 0x00.
0x29	AREA_KB_Position	4	Paste position of keypad on current page. Null if KB_Source = 0x00.
0x2D	NULL	3	0x00 fixed.

Parameters are the same with function <Variable Input>.



Keyboard is not on the current page(KB_Source=0x01): Keyboard Page



Keyboard is not on the current page(KB_Source=0x01): Keyboard activated

7.7 Return Key Code (0x05)

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE05
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	VP_Mode	1	Adjust value mode. 0x00: adjust value in VP address (integer). 0x01: adjust value in high byte of VP address (integer). 0x02: adjust value in low byte of VP address (integer). 0x10-0x1F: write data from last bit of key code into designated bit of VP address. (0x10 corresponds to VP.0, 0x1F corresponds to VP.F)
0x14	Key_Code	2	Return key code.
0x16	NULL	10	0x00 fixed.

7.8 Text Input (0x06)

➢ Key code table for text input

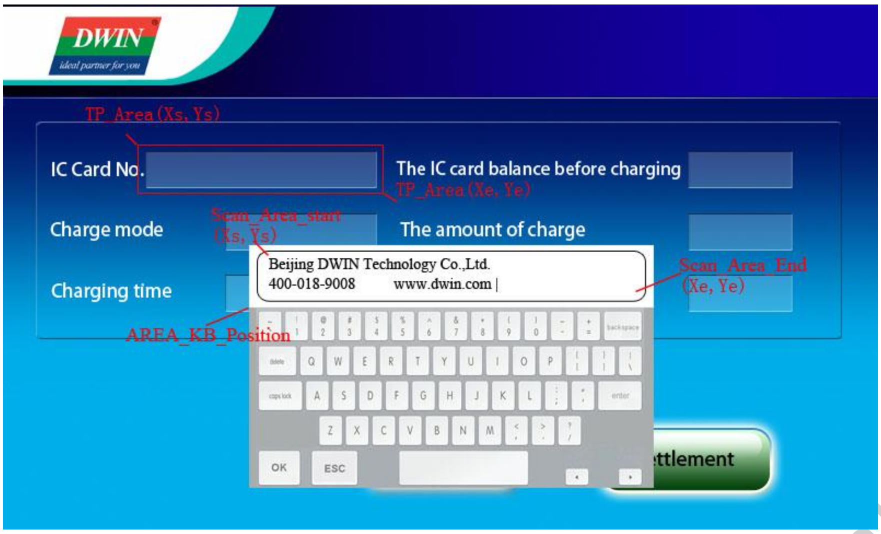
Key code consists of 2 bytes. Low byte indicates lower-case letters, while high byte indicates capital letters. Refer to the table below to see key code table. All key codes follow ASCII table.

Key	Ordinary	Capital	Key	Ordinary	Capital	Key	Ordinary	Capital	Key	Ordinary	Capital
0x7E60	`	~	0x5171	q	Q	0x4161	a	A	0x5A7A	z	Z
0x2131	1	!	0x5777	w	W	0x5373	s	S	0x5878	x	X
0x4032	2	@	0x4565	e	E	0x4464	d	D	0x4363	c	C
0x2333	3	#	0x5272	r	R	0x4666	f	F	0x5676	v	V
0x2434	4	$	0x5474	t	T	0x4767	g	G	0x4262	b	B
0x2535	5	%	0x5979	y	Y	0x4868	h	H	0x4E6E	n	N
0x5E36	6	^	0x5575	u	U	0x4A6A	j	J	0x4D6D	m	M
0x2637	7	&	0x4969	i	I	0x4B6B	k	K	0x3C2C	,	<
0x2A38	8	*	0x4F6F	o	O	0x4C6C	l	L	0x3E2E	.	>
0x2839	9	(	0x5070	p	P	0x3A3B	;	:	0x3F2F	/	?
0x2930	0	)	0x7B5B	[	{	0x2227	'	"	0x2020	SP	SP
0x5F2D	-	_	0x7D5D	]	}	0x0D0D	Enter	Enter			
0x2B3D	=	+	0x7C5C	\	|						

Note: The key code of text input should be less than 0x80 (ASCII code). Key code "0x0D" will be automatically transferred into 0x0D 0x0A. Key code 0x00 and 0xFF: null.

➢ Function keys

Key	Definition	Description
0x00F0	Cancel	Cancel the operation, no affect to variable data.
0x00F1	Return	Save the input text to the designated address and return.
0x00F2	Backspace	Backspace, delete one character.
0x00F3	Delete	Delete.
0x00F4	CapsLock	Caps lock. Must assign the button effect to enable it.
0x00F7	Left	Cursor forwards for one character.
0x00F8	Right	Cursor backwards for one character.



Note: when users prefer to keyboard(key value in 0x4F) for text input, if CapsLock needed, please set animation area of button on the area where CapsLock input to be reminded. Only in this way, CapsLock reminder will show up on the area.

- **ASCII Input**

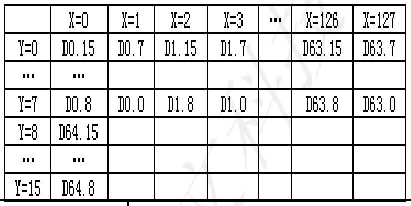
Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE06
0x10	0xFE	1	0xFE
0x11	*VP	2	Variable pointer.
0x13	VP_Len_Max	1	Max length of text, by word (0x01-0x7B). 0xFFFF as end mark will be added at the end of text. Max address number of text should be VP_Len_Max + 1
0x14	Scan_Mode	1	Input mode. 0x00: re-input, 0x01: modify existing text.
0x15	Lib_ID	1	Address of font file.
0x16	Font_Hor	1	Font size, by pixel numbers in X-direction.
0x17	Font_Ver	1	Font size, by pixel numbers in Y-direction. Should be 2 times of pixels in X-direction if Lib_ID = 0x00.
0x18	Cursor_Color	1	Cursor color. 0x00: black, others: white.
0x19	Color	2	Text color.
0x1B	Scan_Area_Start	4	Top-left coordinates of text (Xs, Ys).
0x1F	Scan_Return_Mode	1	0x55: save input terminator and valid data length at (VP-1) position. High byte in (VP-1) for input terminator: 0x5A indicates input is finished, other value shows input is in-process. Low byte in (VP-1) data length for valid input, counted in bytes. 0x00: disable input status return.
0x20	0xFE	1	0xFE
0x21	Scan_Area_End	4	Bottom-right coordinates of text (Xe, Ye).
0x25	KB_Source	1	0x00: call keypad from current page. Others: call keypad from designated page.
0x26	PIC_KB	2	Picture ID of keypad. Null if KB_Source = 0x00.
0x28	AREA_KB	8	Cut area for keypad (Xs, Ys) (Xe, Ye). Null if KB_Source = 0x00.
0x30	0xFE	1	0xFE

0x31	AREA_KB_Position	4	Paste position of keypad on current page. Null if KB_Source = 0x00.
0x35	DISPLAY_EN	1	0x00: unencrypted display, 0x01: encrypted display.
0x36	NULL	10	0x00 fixed

7.9 Firmware Parameter Settings (0x07)

Address	Definition	Data Length	Description
0x00	Pic_ID	2	Picture ID.
0x02	TP_Area	8	Touch button area: (Xs, Ys) (Xe, Ye).
0x0A	Pic_Next	2	Picture jump to. 0xFF**: disable picture switch.
0x0C	Pic_On	2	Press-down effect. 0xFF**: disable press-down effect.
0x0E	TP_Code	2	0xFE07
0x10	0xFE	1	0xFE
0x11	Mode	1	Setup mode selection, see following mode.
0x12	DATA_PACK	14	Data pack of setup.

➢ Setup Mode

<table>
<tr><th>Mode</th><th>Data Pack</th><th>Notes for Data Pack</th><th>Function</th></tr>
<tr><td>0x00</td><td>No</td><td>No</td><td>Transmit data from register to variable SRAM 0x6F00-0x6FFF (low bytes).</td></tr>
<tr><td>0x01</td><td>No</td><td>No</td><td>Transmit data from variable SRAM (low bytes) to register and reset module parameters of R1-R3, R5-RA.</td></tr>
<tr><td rowspan="3">0x02</td><td>Tran_Area</td><td>Coordinates of top-left and bottom-right of area.</td><td rowspan="3">Convert designated area to monochrome bitmap (vertical mode) and save the data to designated VP address.
A. Width should be even.
B. Height should be multiple of 8.
C. VP data format shown as below:
VP: status indicator, refreshed to 0x5555 after operation.
VP+1: horizontal length, by word.
VP+2: numbers of data segment.
VP+3: bitmap data, with MSB priority.
If the key code automatically upload is enabled (R2.3=1), module will upload message (value in VP address upload to 0x5555) to serial port.
The command is mainly for printing of current screen.</td></tr>
<tr><td>*VP</td><td>VP address for restoring bitmap data.</td></tr>
<tr><td colspan="2">
<table>
<tr><th></th><th>X=0</th><th>X=1</th><th>X=2</th><th>X=3</th><th>…</th><th>X=126</th><th>X=127</th></tr>
<tr><td>Y=0</td><td>D0.15</td><td>D0.7</td><td>D1.15</td><td>D1.7</td><td></td><td>D63.15</td><td>D63.7</td></tr>
<tr><td>…</td><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Y=7</td><td>D0.8</td><td>D0.0</td><td>D1.8</td><td>D1.0</td><td></td><td>D63.8</td><td>D63.0</td></tr>
<tr><td>Y=8</td><td>D64.15</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Y=15</td><td>D64.8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td rowspan="2">0x03</td><td>*VP</td><td>Variable pointer.</td><td rowspan="2">Upload data in designated VP address to serial port.
Range of Tx_LEN: 0x0001-0xFFFF.</td></tr>
<tr><td>Tx_LEN</td><td>Length of data to be sent.</td></tr>
<tr><td>0x04</td><td colspan="3">Save function with 0x03, uploading data to COM2 (reserved port).</td></tr>
<tr><td rowspan="2">0x05</td><td>Tran_Area</td><td>Coordinates of top-left and bottom-right of area.</td><td rowspan="2">Convert designated area to monochrome bitmap (horizontal mode) and save the data to designated VP address.
➢ Width should be multiple of 16.
➢ VP data format as shown below:
VP: status indicator, refreshed to 0x5555 after operation.
VP+1: horizontal length, by word.
VP+2: numbers of data segment.
VP+3: bitmap data, with MSB priority.
If the key code automatically upload is enabled (R2.3=1), module will upload message (value in VP address upload to 0x5555) to serial port.
The command is mainly for printing of current screen.</td></tr>
<tr><td>*VP</td><td>VP address for restoring bitmap data.</td></tr>
<tr><td rowspan="2">0x06</td><td>Frame_Head</td><td>Frame header (2 bytes)</td><td rowspan="2">Send the current touched position to COM2 (serial port for reserving the system), the format is:
Frame_Head + X + Y + Check (The cumulative Sum for 1 byte of X, Y) +
Frame_end.</td></tr>
<tr><td>Frame_End</td><td>Frame end (2 bytes)</td></tr>
</table>

8 Variable Config. File (14.BIN)

The Variable Config. file, contained multiple variable commands that can be generated by DGUS_SDK. Since each command occupies 32 bytes and each page contains 64 variable commands, space for each page is 2KB (0x0800). Max page number is 1024, and max volume of variable Config. file is 2MB. Priority of display is last in-first out (LIFO). Variable command contains 6 sections.

No.	Definition	Data Length	Description
1	0x5A	1	Fixed
2	Type	1	Variable type.
3	*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
4	Len_Dsc	2	The whole process length (in terms of words).
5	*VP	2	Variable pointer, 0x0000-0x6FFF. Write 0x0000 for the variables that do not need address assigning. The command will be disabled when the high byte is 0xFF.
6	Description	N	Parameters of variable.

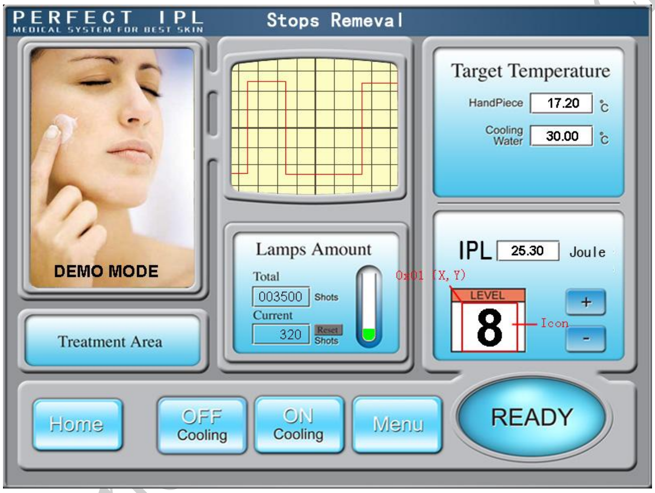
8.1 Display Variables Overview

Num.	Key Code	Function	Description
01	00	Variable Icon	Icon display related with a data variable. If variable changes, icon will be switches accordingly which is widely used for dashboard, progress bar application.
02	01	Animation Icon	3 kinds of icon status corresponding to a data variable: no display, display fixed Animation icon display which is widely used for alerting
03	02	Slider	Slider related with a data variable for value changes. Normally it was used for liquid level, dial board, progress bar.
04	03	WordArt	Use created icon material in wordart to display data.
05	04	Image Animation	Auto play of images in a certain speed. Normally used in welcome page or screensaver.
06	05	Icon Rotation	Use a pointer as icon file to display data changes on a dashboard.
07	06	Bit Variable Icon	Connect status 0 or 1 on each bit of a variable to display 2 status, 8 pcs of status in option totally, then use icon to display the status. Normal use for display of on-off state
08	10	Data Variable	Display a variable in designated format, including decimals, font type, and alphabet.
09	11	Text	Display charter strings in text area.
10	12_00	Digital RTC	Display RTC in form of text with custom format
11	12_01	Analog Clock	Use ICON to display RCT in form of watch-face
12	13	HEX Variable	Use ASCII to display variables in byte HEX interval. E.g.: display 1234 to 12:34 on timer.
13	20	Dynamic Trend Curve	Combined with Command 0x84 to configure curve in real-time. Display area, coordinate, scales(zoom in/out) can be managed.
14	21_01	Dot	Dot set（x,y,color）
15	21_02	Line	Dot Connection (color,(x0,y0),…(xn,yn)）
16	21_03	Rectangle	Rectangle displayed. Color/position/size can be managed.
17	21_04	Rectangle Area Fill	Fill designated rectangle area, color/position/size can be managed.
18	21_05	Circle	Display entire arc, color/position/size can be managed.
19	21_06	Picture Cut/Paste	Cut an area from designated image to current page.
20	21_07	Icon Display	ICON display, icon library in option.
21	21_08	Area Fill	Closed area fill.
22	21_09	Spectrum	Spectrum display according to variable data. Color/position can be managed.
23	21_0A	Segment	
24	21_0B	Arc Display	
25	21_0C	Character	Display Character according to variable data.
26	21_0D	Rectangle XOR	Mark the data in designated color..
27	21_0E	Bicolorable Graph	Bicolorable graph regarded to variable data, corresponding to 0/1.
28	21_0F	Bitmap	65K bitmap data. Normal used for images downloading reminder.
29	21_10	Zoom in and Paste	Zoom in and past the area to the place. It is mainly combined with Command 0F for image displaying in real time
30	22	Table Display	Display the data in subfield table which is defined via two-dimensional array.

8.2 Variable Icon

8.2.1 Variable Icon (0x00)

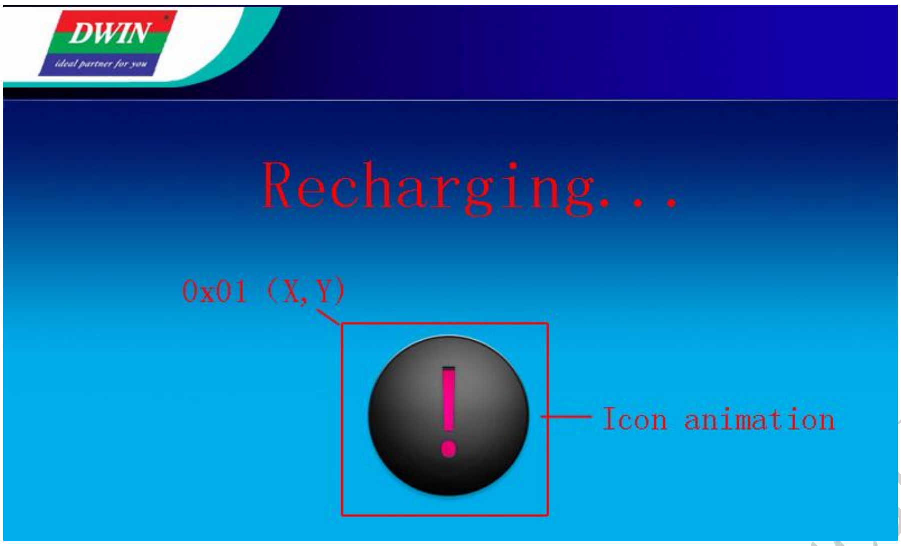
Address		Definition	Data Length	Description
0x00		0x5A00	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x0008	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	(x, y)	4	Display position, top-left coordinate of icon.
0x0C	0x03	V_Min	2	Floor of range, null if over range.
0x0E	0x04	V_Max	2	Ceiling of range, null if over range.
0x10	0x05	Icon_Min	2	Icon address in icon file corresponding to min value.
0x12	0x06	Icon_Max	2	Icon address in icon file corresponding to max value.
0x14	0x07:H	Icon_Lib	1	Address of icon file.
0x15	0x07:L	Mode	1	Icon display mode. 0x00: transparent. Others: opaque.



8.2.2 Animation Icon (0x01)

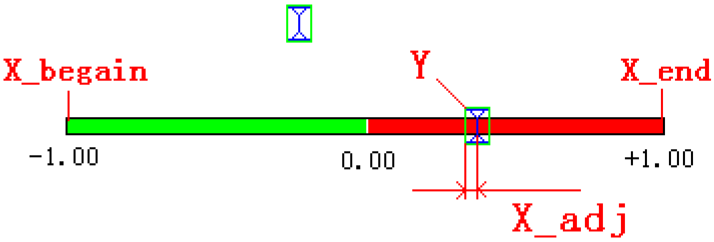
Address		Definition	Data Length	Description
0x00		0x5A01	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000A	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer of initial icon. High word: unsigned integer. Low word: reserved, status of animation. (0x0000-0x0FFFF)
0x08	0x01	(x, y)	4	Display position, top-left coordinate of icon.
0x0C	0x03	0x0000	2	0x0000 fixed.
0x0E	0x04	V_Stop	2	Value corresponding to stop animation.
0x10	0x05	V_Start	2	Value corresponding to start animation.
0x12	0x06	Icon_Stop	2	Icon at V_Stop value.
0x14	0x07	Icon_Start	2	Start/end icons for animation at V_Start value.
0x16	0x08	Icon_End	2	
0x18	0x09:H	Icon_Lib	1	Address of icon file.
0x19	0x09:L	Mode	1	Icon display mode. 0x00: transparent. Others: opaque.

If the value in VP address is equal to neither V_Stop nor V_Start, icons are not displayed on screen



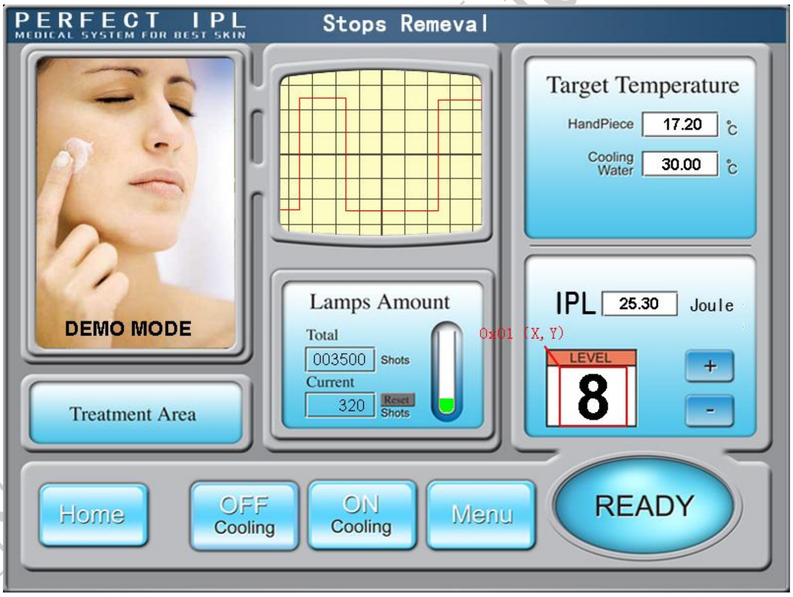
8.2.3 Slider (0x02)

Address		Definition	Data Length	Description
0x00		0x5A02	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x0009	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	V_begin	2	Variable corresponding to start point.
0x0A	0x02	V_end	2	Variable corresponding to end point.
0x0C	0x03	X_begin	2	Starting position of slider. X coordinates for horizontal sliders. (Y coordinates for vertical sliders.)
0x0E	0x04	X_end	2	Ending position of slider. X coordinates for horizontal sliders. (Y coordinates for vertical sliders.)
0x10	0x05	Icon_ID	2	Icon address in icon file.
0x12	0x06	Y	2	Position of slider. Y coordinates for vertical sliders. (X coordinates for horizontal sliders.)
0x14	0x07:H	X_adj	1	X/Y axis offset to the left/top.
0x15	0x07:L	Mode	1	Slider mode. 0x00: horizontal, others: vertical.
0x16	0x08:H	Icon_Lib	1	Address of icon file.
0x17	0x08:L	Icon_mode	1	Icon display mode. 0x00: transparent, others: opaque.
0x18	0x09:H	VP_DATA_Mode	1	0x00: integer (whole VP address). 0x01: high byte in VP address. 0x02: low byte in VP address.



8.2.4 WordArt (0x03)

Address		Definition	Data Length	Description
0x00		0x5A03	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x0007	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	X, Y	4	Top-left coordinate of words, left aligned or top-right coordinate of words, right aligned.
0x0C	0x03	Icon0	2	Icon corresponding to number 0, by sequence of “01234567890-.”.
0x0E	0x04:H	Icon_Lib	1	Address of icon file.
0x0F	0x04:L	Icon_Mode	1	Icon display mode. 0x00: transparent, others: opaque.
0x10	0x05:H	Int_Num	1	Length of integer digits.
0x11	0x05:L	Dec_Num	1	Length of decimal digits.
0x12	0x06:H	VP_Data_Mode	1	0x00: integer (2 bytes), from -23768 to 32767 0x01: long integer (4 bytes), from -2147483648 to 2147483647 0x02: *VP high byte, no unsigned, from 0 to 255 0x03: *VP low byte, no unsigned, from 0 to 255 0x04: ultra-long integer(8 bytes), from -9223372036854775808 to 9223372036854775807 0x05: unsigned integer(2 bytes), from 0 to 65535 0x06: unsigned long integer(4 bytes), from 0 to 4294967295
0x13	0x06:L	ALI	1	0x00: left-aligned, 0x01: right-aligned.



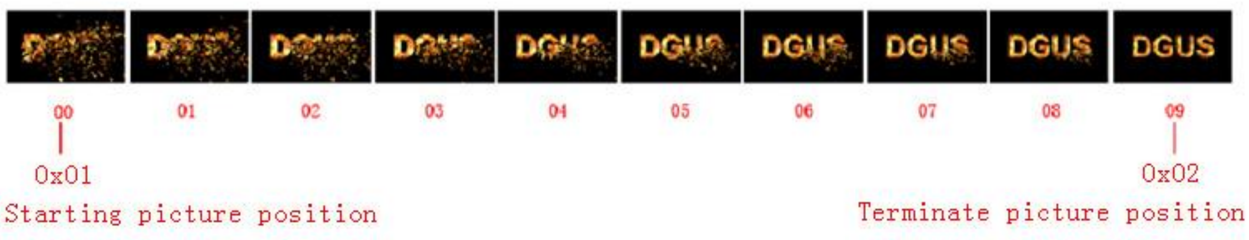
8.2.5 Image Animation (0x04)

Address		Definition	Data Length	Description
0x00		0x5A04	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x0004	2	The whole process length (in terms of words).
0x06	0x00	0x0000	2	0x0000 fixed.
0x08	0x01	Pic_Begin	2	Starting picture of animation.
0x0A	0x02	Pic_End	2	Ending picture of animation.
0x0C	0x03:H	Frame_Time	1	Switching speed of animation, by every 8ms.

Start image ID should be smaller than end image ID.

Set a <Image Animation> on end image to loop.

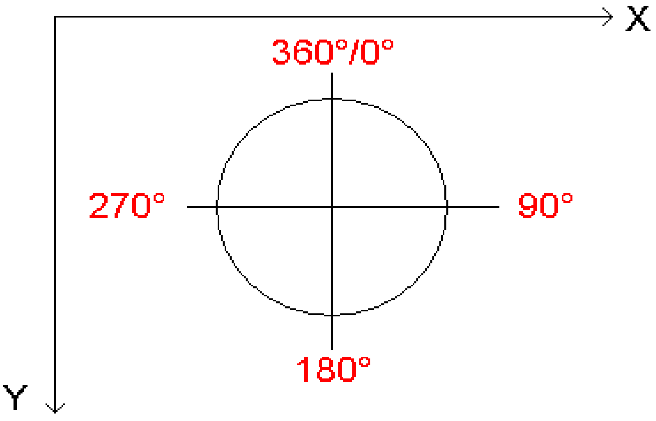
Send commands or set <Touch Control> button to interrupt animation.



8.2.6 Icon Rotation (0x05)

Address		Definition	Data Length	Description
0x00		0x5A05	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000C	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	Icon_ID	2	Icon address in icon file.
0x0A	0x02	Icon_Xc	2	Rotation center of icon: X coordinate.
0x0C	0x03	Icon_Yc	2	Rotation center of icon: Y coordinate.
0x0E	0x04	Xc	2	Rotation center on current screen: X coordinate.
0x10	0x05	Yc	2	Rotation center on current screen: Y coordinate.
0x12	0x06	V_Begin	2	Value corresponding to starting angle, null if over range.
0x14	0x07	V_End	2	Value corresponding to ending angle, null if over range.
0x16	0x08	AL_Begin	2	Starting angle, range from 0 to 720 (0x000 - 0x2D0), by every 0.5 °.
0x18	0x09	AL_End	2	Ending angle, range from 0 to 720 (0x000 - 0x2D0), by every 0.5 °.
0x1A	0x0A: H	VP_Mode	1	VP mode. 0x00: integer (whole VP address). 0x01: high byte in VP address. 0x02: low byte in VP address.
0x1B	0x0A:L	Lib_ID	1	Address of icon file.
0x1C	0x0B	Mode	1	Icon display mode. 0x00: transparent, others: opaque.

This function is mainly used for dash board. Rotation is always clockwise, AL_Begin should be larger than AL_End, (or a 360 will be added to AL_End by system).



8.2.7 Bit Variable Icon (0x06)

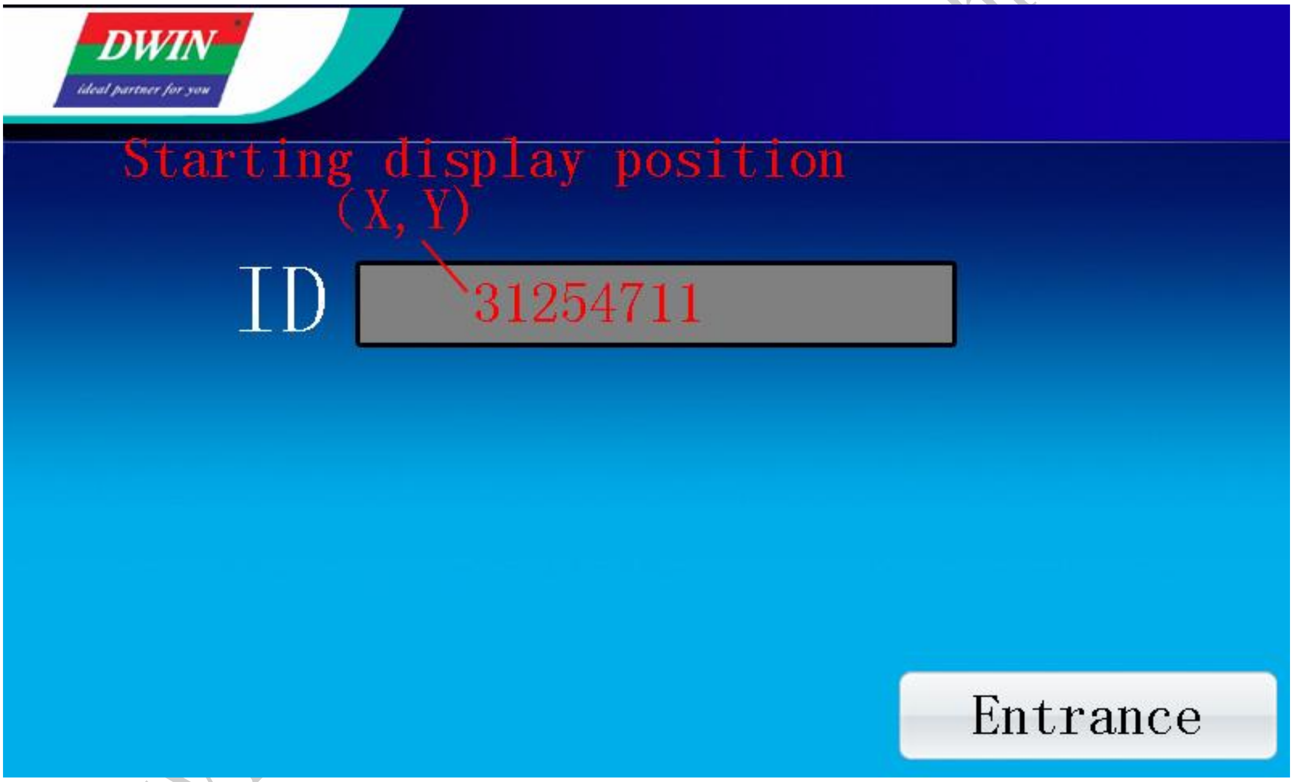
<table>
<tr><th colspan="2">Address</th><th>Definition</th><th>Data Length</th><th>Description</th></tr>
<tr><td>0x00</td><td></td><td>0x5A06</td><td>2</td><td></td></tr>
<tr><td>0x02</td><td></td><td>*SP</td><td>2</td><td>Stack pointer, default setting is 0xFFFF (set by Config. file).</td></tr>
<tr><td>0x04</td><td></td><td>0x000C</td><td>2</td><td>The whole process length (in terms of words).</td></tr>
<tr><td>0x06</td><td>0x00</td><td>*VP</td><td>2</td><td>Variable pointer, by word.</td></tr>
<tr><td>0x08</td><td>0x01</td><td>*VP_AUX</td><td>2</td><td>Substitutive variable pointer, reserved 2 words. User software unable to access.</td></tr>
<tr><td>0x0A</td><td>0x02</td><td>Act_Bit_Set</td><td>2</td><td>Display is on when bit value of VP is 1.</td></tr>
<tr><td>0x0C</td><td>0x03:H</td><td>Display_Mode</td><td>1</td><td>
<table>
<tr><th rowspan="2">Display_Mode</th><th colspan="2">Bit Value</th></tr>
<tr><th>0</th><th>1</th></tr>
<tr><td>0x00</td><td>ICON0S</td><td>ICON1S</td></tr>
<tr><td>0x01</td><td>ICON0S</td><td>Null.</td></tr>
<tr><td>0x02</td><td>ICON0S</td><td>Animation:
ICON1S-ICON1E.</td></tr>
<tr><td>0x03</td><td>Null.</td><td>ICON1S</td></tr>
<tr><td>0x04</td><td>Null.</td><td>Animation:
ICON1S-ICON1E.</td></tr>
<tr><td>0x05</td><td>Animation:
ICON0S-ICON0E.</td><td>ICON1S</td></tr>
<tr><td>0x06</td><td>Animation:
ICON0S-ICON0E.</td><td>Null.</td></tr>
<tr><td>0x07</td><td>Animation:
ICON0S-ICON0E.</td><td>Animation:
ICON1S-ICON1E.</td></tr>
</table>
</td></tr>
<tr><td>0x0D</td><td>0x03:L</td><td>Move_Mode</td><td>1</td><td>Bit icons arranged mode.
0x00: X++, space unreserved for undesignated bits.
0x01: Y++, space unreserved for undesignated bits.
0x02: X++, space reserved for undesignated bits.
0x03: Y++, space reserved for undesignated bits.</td></tr>
<tr><td>0x0E</td><td>0X04:
H</td><td>Icon_Mode</td><td>1</td><td>Icon display mode.
0x00: transparent, 0x01: opaque.</td></tr>
<tr><td>0x0F</td><td>0x04:L</td><td>Icon_Lib</td><td>1</td><td>Address of icon file.</td></tr>
<tr><td>0x10</td><td>0x05</td><td>ICON0S</td><td>2</td><td>Icon ID for bit0 in non-animation mode, or starting icon ID for bit0 in animation mode.</td></tr>
<tr><td>0x12</td><td>0x06</td><td>ICON0E</td><td>2</td><td>Ending icon ID for bit0 in animation mode.</td></tr>
<tr><td>0x14</td><td>0x07</td><td>ICON1S</td><td>2</td><td>Icon ID for bit1 in non-animation mode, or starting icon ID for bit1 in animation mode.</td></tr>
<tr><td>0x16</td><td>0x08</td><td>ICON1E</td><td>2</td><td>Ending icon ID for bit1 in animation mode.</td></tr>
<tr><td>0x18</td><td>0x09</td><td>X, Y</td><td>4</td><td>Top-left coordinates of starting icons.</td></tr>
<tr><td>0x1C</td><td>0x0B</td><td>DIS_MOV</td><td>2</td><td>Spacing between icons.</td></tr>
<tr><td>0x1E</td><td></td><td></td><td></td><td>0x00 fixed</td></tr>
</table>

8.3 Text Variable

8.3.1 Data Variable (0x10)

Address		Definition	Data Length	Description
0x00		0x5A10	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000D	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	X, Y	4	Top-left coordinate of text string.
0x0C	0x03	COLOR	2	Text color.

0x0E	0x04:H	Lib_ID	1	Address of font file.	
0x0F	0x04:L	Font_X_Dots	1	Horizontal pixel numbers.	
0x10	0x05:H	ALI	1	0x00: right-aligned, 0x01: left-aligned, 0x02: centered.	
0x11	0x05:L	Int_Num	1	Length of integer digits.	The sum should be less than 20.
0x12	0x06:H	Dec_Num	1	Length of decimal digits.	
0x13	0x06:L	VP_Data_Mode	1	VP mode. 0x00: integer (2 bytes). -32768 – 32767 0x01: long integer (4 bytes). -2147483648 – 2147483647 0x02: high byte in VP address. 0 – 255 0x03: low byte in VP address. 0 – 255 0x04: double long integer (8 bytes). -9223372036854775808 – 9223372036854775807 0x05: unsigned integer (2 bytes). 0 – 65535 0x06: unsigned long integer (4 bytes). 0 – 4294967295	
0x14	0x07:H	Len_unit	1	Length of unit. 0x00: without unit.	
0x15	0x07:L	String_Unit	Max11	Unit data, by ASCII code.	



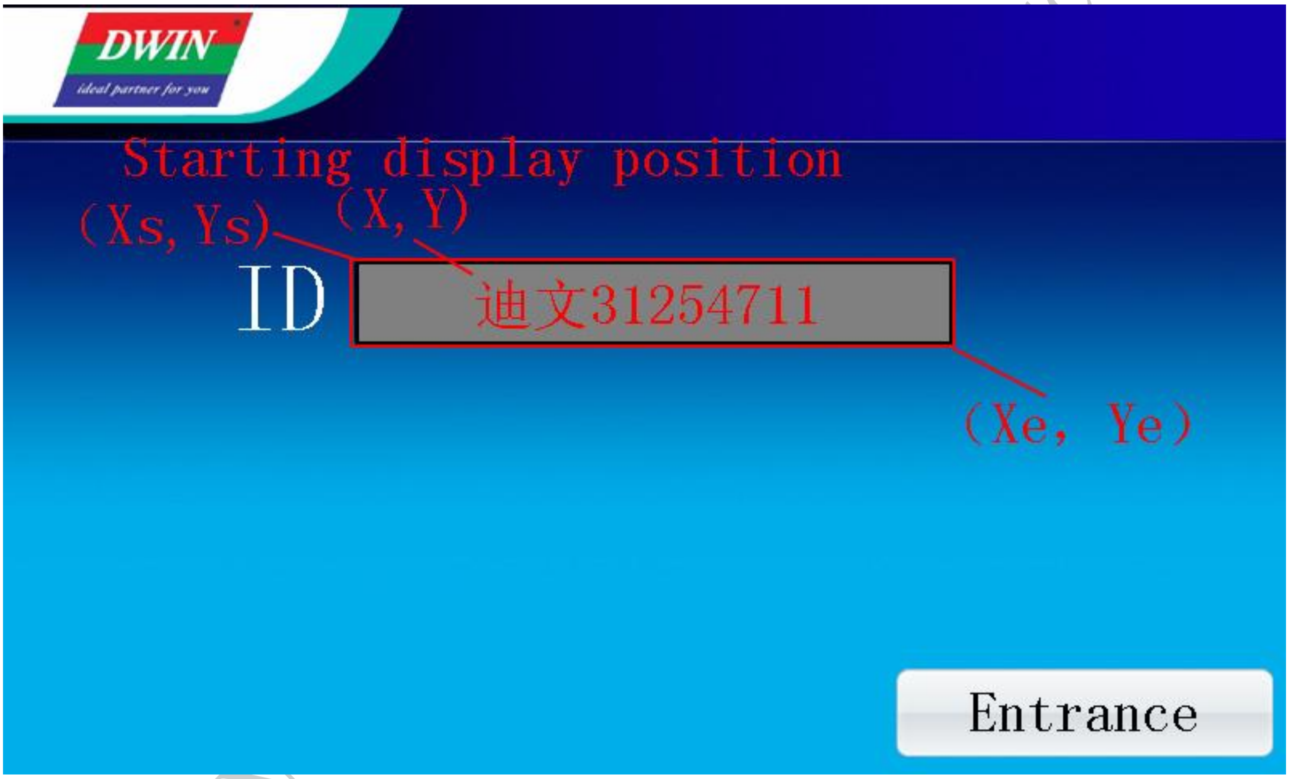
8.3.2 Text (0x11)

Address		Definition	Data Length	Description
0x00		0x5A11	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000D	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	X, Y	4	Top-left coordinate of text string.
0x0C	0x03	Color	2	Text color.
0x0E	0x04	Xs Ys Xe Ye	8	Scope of text box, top-left and bottom-right coordinates.
0x16	0x08	Text_length	2	Text length, by byte. Data will not display if it is changed into 0xFFFF or over range.

0x18	0x09:H	Font0_ID	1	Address of font file for encoding mode 0x01 - 0x04.
0x19	0x09:L	Font1_ID	1	Address of font file for encoding mode 0x00 and 0x05, also other non-ASCII font for encoding mode 0x01 - 0x04.
0x1A	0x0A:H	Font_X_Dots	1	Font size in X-direction. X should be Y/2 for encoding mode 0x01-0x04.
0x1B	0x0A:L	Font_Y_Dots	1	Font size in Y-direction. Must be even.
0x1C	0x0B:H	Encode_Mode	1	Spacing between letters is defined by .7 bit. .7 = 0: adapted spacing automatically. .7 = 1: fixed spacing. Encoding mode is defined by .6 to .0 bit. 0: 8 bit coding, 1: GB2312, 2: GBK, 3:BIG5, 4: SJIS, 5: UNICODE.
0x1D	0x0B:L	HOR_Dis	1	Character spacing.
0x1E	0x0C:H	VER_Dis	1	Line spacing.
0x1F	0x0C:L			0x00 fixed

Dots number in Y-direction must be even.

All ASCII characters from 4*8 pixels to 64*128 pixels are included in 0_DWIN_ASCII.hzk.



8.3.3 RTC (0x12)

- **Digital RTC**

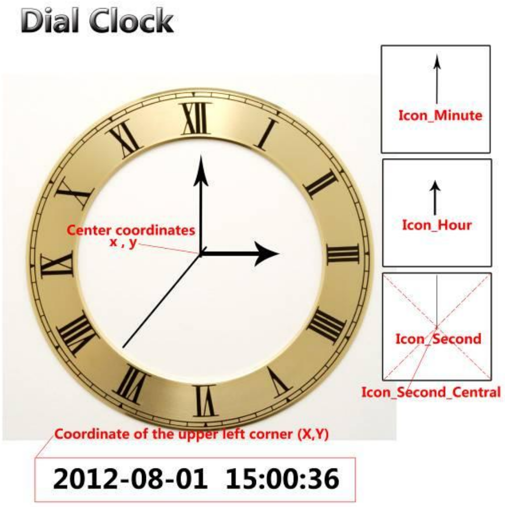
Address		Definition	Data Length	Description
0x00		0x5A12	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000D	2	The whole process length (in terms of words).
0x06	0x00	0x0000	2	0x0000 fixed.
0x08	0x01	X, Y	4	Top-left coordinates of text.
0x0C	0x03	Color	2	Text color.
0x0E	0x04:H	Lib_ID	1	Address of ASCII font file.
0x0F	0x04:L	Font_X_Dots	1	Font size in X-direction.
0x10	0x05	String_Code	MAX16	Character string, by the RTC code table and ASCII code. E.g.: current time is 2012-05-02 12:00:00 Wednesday, ● Y-M-D H: Q: S 0x00, will be displayed as “2012-05-02 12:00:00”. ● M-D W H: Q 0x00, will be displayed as “05-02 WED 12:00”.

- **RTC Code table**

Description	Encoding	Format
Year	Y	2000-2099
Month	M	01-12
Day	D	01-31
Hour	H	00-23
Minute	Q	00-59
Second	S	00-59
Date	W	SUN MON TUE WED THU FRI SAT
Coding end	0x00	

➢ **Analog Clock**

Address		Definition	Data Length	Description
0x00		0x5A12	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000D	2	The whole process length (in terms of words).
0x06	0x00	0x0001	2	0x0001
0x08	0x01	X, Y	4	Rotation center of analog clock on current screen.
0x0C	0x03	Icon_Hour	2	Hour hand icon address in icon file, 0xFFFF: null.
0x0E	0x04	Icon_Hour_Central	4	Rotation center of hour hand icon.
0x12	0x06	Icon_Minute	2	Minute hand icon address in icon file, 0xFFFF:null.
0x14	0x07	Icon_Minute_Central	4	Rotation center of minute hand icon.
0x18	0x09	Icon_Second	2	Second hand icon address in icon file, 0xFFFF: null.
0x1A	0x0A	Icon_Second_Central	4	Rotation center of second hand icon.
0x1E	0x0C:H	ICON_Lib	1	Address of icon file.
0x1F			1	0x00.



8.3.4 HEX Variable (0x13)

Address		Definition	Data Length	Description
0x00		0x5A13	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000D	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Starting variable pointer of data string, data is encoded with BCD format. The data will be displayed in HEX format when half-byte data is greater than 0x9, e.g.: 0x32: display 32, 0xBF: display BF.
0x08	0x01	X, Y	4	Top-left coordinate of text.
0x0C	0x03	Color	2	Text color.

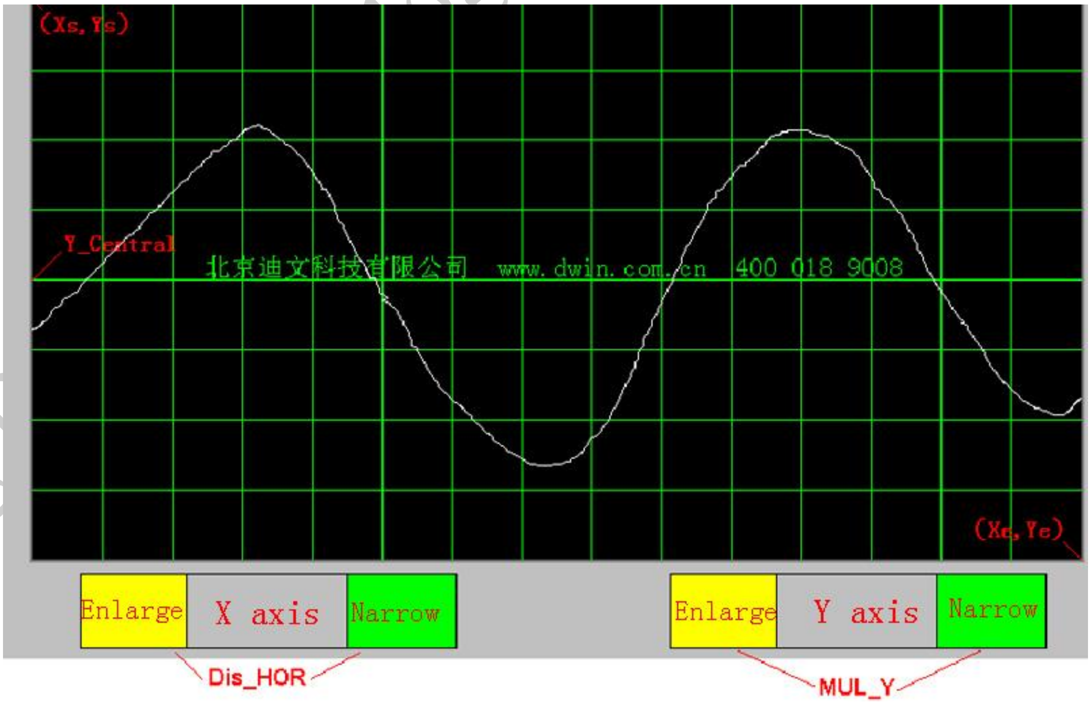
0x0E	0x04:H	Byte_Num	1	Byte numbers to be displayed, 0x01 - 0x0F.
0x0F	0x04:L	Lib_ID	1	Address of font file. The format of font must be 8bit encoding, half-width, if Lib_ID is not 0x00.
0x10	0x05:H	Font_X	1	Font size in X-direction.
0x11	0x05:L	String_Code	MAX15	Encoded separators string, used to define the format of Timer. Every time a Timer data (BCD code) is read, one ASCII char will be added after as separator. Some special chars: 0x00: none, Timer data will be concatenated; 0x0D: new line.

8.4 Graphic Variables

8.4.1 Dynamic Trend Curve (0x20)

Address		Definition	Data Length	Description
0x00		0x5A20	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000A	2	The whole process length (in terms of words).
0x06	0x00	0x0000	2	0x0000
0x08	0x01	Xs Ys Xe Ye	8	Scope of trend curve window, null if over range.
0x10	0x05	Y_Central	2	Center line coordinates of trend curve in Y-direction.
0x12	0x06	VD_Central	2	Trend curve value at center line, normally average of max & min value.
0x14	0x07	Color	2	Trend curve color.
0x16	0x08	MUL_Y	2	Magnification in Y-direction, by every 1/256, 0x0000 - 0x7FFF.
0x18	0x09:H	CHANEL	1	Chanel for trend curve, 0x00 – 0x07.
0x19	0x09:L	Dis_HOR	1	Transverse spacing between sample point, 0x01 – 0xFF.

Use command 0x84 to send trend curve data, please refer to **Chapter 3.2 Command Set** for detailed command format.



Scale and position of curve can be modified by buttons on screen if the variable description is saved in SP address.

- To scale the trend curve automatically with Incremental Adjustment (0xFE02), without user’s program.

- To move the trend curve up and down using Slider adjustment (0xFE03) to revise the value of Y_Central, without user's program.
- If thicker lines requested, user may drop more than one curve variables that allowed to move in Coordinate Y sourcing from same data channel.

MUL_Y calculation of full-scale trend curve:

MUL_Y= (Ye-Ys)*256/ (Vmax-Vmin).

Ye Ys are Y coordinates of trend curve window, Vmax Vmin are Max and Min value of trend curve.

E.g.: a 12-bit A/D data acquisition, Vmax= 4095, Vmin= 0, to display trend curve fully-scale between Ys = 50 and Ye = 430, MUL_Y= (430-50)*256/ (4095-0)= 23.7, rounded down to 23.

8.4.2 Basic Graphics (0x21)

Address		Definition	Data Length	Description
0x00		0x5A21	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x0008	2	
0x06	0x00	*VP	2	Variable pointer.
0x08	0x01	Area	8	Set displaying area: upper-left, down-right coordinat, null if over rang Only valid to Command 0x0001-0x0005、0x0009、0x000A、0x000B
0x10	0x05:H	Dashed_Line_En	1	0x5A:command for drawings with line segment(Command 0x02、0x03、0x09、0x0A). Display in dash dot line or imaginary line. Others: Full line display
0x11	0x05:L	Dash_Set	4	Imaginary line Format in 4 segments: Seg 1: dot matrix of full line; Seg 2: dot matrix of imaginary line Seg 3:second section of dot matrix of full line; Seg 4: second section of dot matrix of imaginary line. e.g.: set 0x10 0x04 0x10 0x04 as imaginary line, will display dot line.
0x15			13	Retained, write 0x00

Basic drawings defines a function of 'drawing-board'firstly in 14.bin, while plot drawing operations are determined by variable storage which points from *VP

User can change the variable storage to make different drawings comes true.

- **Instruction of variable data format in storage space**

ADDRESS	DEFINITION	DESCRIPTION
VP	CMD	Command for drawings
VP+1	Data_Pack_Num_Max	Maximum quantity of data pack: Drawings(0x0002) Amount of lines: vertex-1
VP+2	DATA_Pack	Data

- **Data Pack for Basic Graphic**

CMD	Function	Description of Data Pack, by word			
		Relative Address	Data Length	Definition	Description
0x0001	Dot	0x00	2	(x, y)	Dot coordinates, high byte of X coordinate is judgment condition
		0x02	1	Color	Dot color.
0x0002	Line	0x00	1	Color	Line color.

		0x01	2	(x, y)0	Vertex 0 coordinates, high byte of X coordinate is judgment condition
		0x03	2	(x, y)1	Vertex 1 coordinates, high byte of X coordinate is judgment condition
		0x01+2*n	2	(x, y)n	Vertex n coordinates, high byte of X coordinate is judgment condition
0x0003	Rectangle	0x00	2	(x, y)s	Top-left coordinates, high byte of X coordinate is judgment condition
		0x02	2	(x, y)e	Bottom-right coordinates.
		0x04	1	Color	Rectangle's color.
0x0004	Rectangle Area Fill	0x00	2	(x, y)s	Top-left coordinates, high byte of X coordinate is judgment condition
		0x02	2	(x, y)e	Bottom-right coordinates.
		0x04	1	Color	Filled color.
0x0005	Circle	0x00	2	(x, y)	Circle center coordinates, high byte of X coordinate is judgment condition
		0x02	1	Rad	Radius of circle.
		0x03	1	Color	Circle color.
0x0006	Picture Cut/Paste	0x00	1	Pic_ID	Image ID of cutting area, high byte of X coordinate is judgment condition
		0x01	2	(x, y)s	Top-left coordinates of the cutting area.
		0x03	2	(x, y)e	Bottom-right coordinates of the cutting area.
		0x05	2	(x, y)	Paste position on current screen, upper left coordinate
0x**07	Icon Display	0x00	2	(x, y)	Top-left coordinates of icon, high byte of X coordinate is judgment condition
		0x02	1	ICON_ID	Icon ID in icon file, high byte of command specifies address of icon file, display mode is transparent.
0x0008	Area Fill	0x00	2	(x, y)	Sampling dot coordinates, high byte of X coordinate is judgment condition
		0x02	1	Color	Filled color.
0x0009	Spectrum	0x00	1	Color0	Connect (X0, Y0s), (X0, Y0e) with color0, high byte of X coordinate is judgment condition
		0x01	1	X0	
		0x02	1	Y0s	
		0x03	1	Y0e	
0x000A	Segment	0x00	1	Color	Connect (Xs, Ys), (Xe, Ye) with Color, high-byte of Xs is judging condition.
		0x01	1	Xs	
		0x02	1	Ys	
		0x03	1	Xe	
		0x04	1	Ye	
0x000B	Arc Display	0x00	1	Color0	Arc color
		0x01	2	(X,Y)0	Central point value, high byte of X-value is criteria
		0x03	1	RAD0	radius
		0x04	1	DEG_S0	Initial angle, unit 0.5 °, 0-720
		0x05	1	DEG_E0	Terminated angle, unite 0.5 °,0-720
0x000C	Character	0x00	1	Color0	Charter color
		0x01	2	(X,Y)0	Position and upper-left point coordinate. X-value is criteria
		0x03H	0.5	Lib_ID	Font position
		0x03L	0.5	En_Mode	character encoding scheme: 0=8bit 1=GB2312 2=GBK 3=BIG5 4=SJIS 5=UNICODE
		0x04H	0.5	X_Dots	Lattice in X direction
		0x04L	0.5	Y_Dots	Lattice in Y direction
		0x05	1	Text0	Character data, only valid on high byte of 8-bit encode. If encoding is 01-04 and ASCII data, default No.0 font will be used for display.
0x000D	Rectangle XOR	0x00	2	(x,y)s	Upper left coordinate of rectangle area. High byte of X coordinate is judgment condition
		0x02	2	(x,y)e	Lower right corner coordinate of rectangle area

		0x04	1	Color	XOR color and 0xFFFF for opposite color operation	
0x000E	Bicolorable Graph	0x00	2	(x,y)s	Upper left coordinate of bitmap, high byte of X coordinate is judgment condition	
		0x02	1	X_Dots	Lattice in X direction	
		0x03	1	Y_Dots	Lattice in Y direction	
		0x04	1	Color1	The color that corresponded to "1"bit	
		0x05	1	Color0	The color that corresponded to "1"bit, if set Color0 same as Color1 which means "0" bit is no need to display, just skip it directly	
		0x06	N	Data_Pack	Data display with MSB. Considering the conveniences of data write and read, each line have to align to one word, namely next line should always start from a new data word.	
0x000F	Bitmap	0x00	2	(x,y)s	Upper left coordinate of bitmap, high byte of X coordinate is judgment condition。	
		0x02	1	X_Dots	Lattice in X direction	Maximum:196*146(4:3) or 226*126 （16:9）。
		0x03	1	Y_Dots	Lattice in Y direction	
		0x04	N	Data_Pack	Data display, each word occupies one dot(MSB,5R6G5B data format)	
x0010	Paste display after zoom-in	0x00	2	（x,y）	Paste on upper left corner after zoom in, high byte of X is judgment condition.	When the area to be zoomed in lies within the after-zoom-in image area, it should be aligned to the bottom-right. More magnification can be obtained via nested zoom-in.
		0x02	2	（x,y） s	Upper-left coordinate that to be zoom-out	
		0x04	2	（x,y） e	Bottom-right coordinate that to be zoom-in	

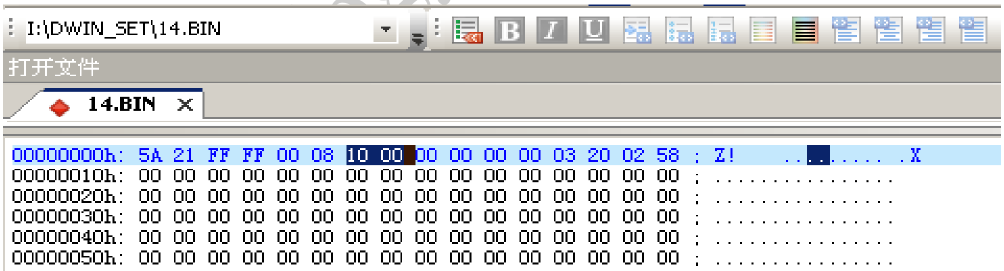
Condition:

0xFF: current drawing operation finished.

0xFE: the operation will be ignored.

- **E.g.: Basic drawings (take 0x0006 cut/paste as example)**

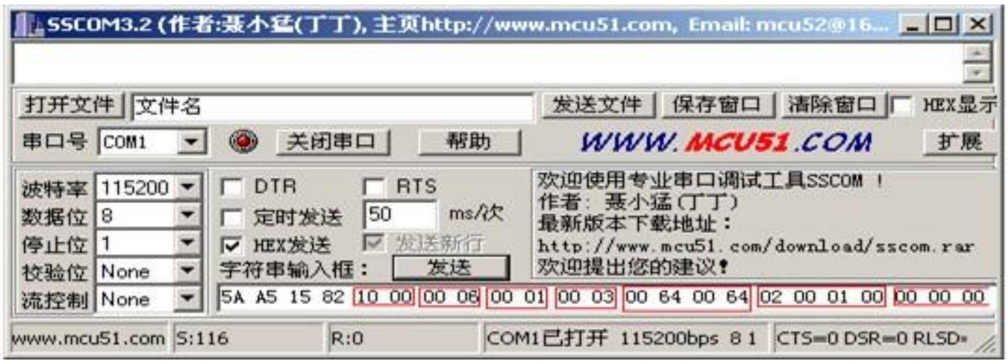
Step 1: create and define a 'drawing-board' variable in 14.BIN, pointing to Add.0x1000 in VP, showing as follows.



```
00000000h: 5A 21 FF FF 00 08 10 00 00 00 00 00 03 20 02 58 ; Z!     ........ .X
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

Step2 Download 14.BIN into DGUS display via SD card.

Step3 Write 0x0006 to Add. 0x1000(*VP),namely, cut (100,100)(512,256) to paste position(0,0) on current page. DGUS display will implement the command on page that works with Command Drawing-board if content in VP keeps going.



串口号 COM1

波特率 115200

数据位 8

停止位 1

校验位 None

流控制 None

□ DTR □ RTS

□ 定时发送 50 ms/次

☑ HEX发送 ☑ 发送新行

字符串输入框：

5A A5 15 82 10 00 00 06 00 01 00 03 00 64 00 64 02 00 01 00 00 00 00

欢迎使用专业串口调试工具SSCOM ！
作者： 聂小猛(丁丁)
最新版本下载地址：
http://www.mcu51.com/download/sscom.rar
欢迎提出您的建议！

www.mcu51.com S:116 R:0 COM1已打开 115200bps 8 1 CTS=0 DSR=0 RLSD=

8.4.3 Table Display（0x22）

Address		Definition	Data Length	Description
0x00		0x5A22	2	
0x02		*SP	2	Stack pointer, default setting is 0xFFFF (set by Config. file).
0x04		0x000C	2	The whole process length (in terms of words).
0x06	0x00	*VP	2	Starting VP address of data in table.
0x08	0x01:H	TAB_X_Num	1	Column number, 0x01 - 0xFF.
0x09	0x01:L	TAB_Y_Num	1	Row number, 0x01 - 0xFF.
0x0A	0x02:H	TAB_X_Start	1	Starting column to be displayed, 0x00 - 0xFF.
0x0B	0x02:L	TAB_Y_Start	1	Starting row to be displayed, 0x00 - 0xFF.
0x0C	0x03:H	Unit_Data_Num	1	➢ 0x01 - 0x7F: data length for one cell. ➢ 0x00: data in VP address defines the length of each column. When Unit_Data_Num is 0x00 the starting address of data will be (row number/2, round up to integer) backward from VP address. When Unit_Data_Num=0x00, saving position of data in table put off later and take whole word address. E.g.:*VP=0x1000，TAB_X_Num=0x07, therefore： 0x1000-0x1003 saves Row No.0-No.6, low-byte of 1003 invalid and start save from Address 0x1004
0x0D	0x03:L	Encode_Mode	1	.7 Automatically adjustment of spacing in text display ➢ .7=0 adjust it automatically； ➢ .7=1 manual adjustment and character width set as fixed number of dots .6 Sheet content format ➢ .6=0 text display； ➢ .6=1 first two words indicates the format as reference bellowing NOTICE[1] .5 Boarder line display ➢ .5=0 display boarder line ➢ .5=1 do not display .4 Undefined, write 0. .3-.0 Text code 0=8bit 1=GB2312 2=GBK 3=BIG5 4=SJIS 5=UNICODE
0x0E	0x04	Xs Ys Xe Ye	8	Table area, top-left and bottom-right coordinates.
0x16	0x08	Color_line	2	Boarder color.
0x18	0x09	Color_text	2	Text color.
0x1A	0x0A:H	Font0_ID	1	Address of font for encoding mode 0x01 - 0x04.
0x1B	0x0A:L	Font1_ID	1	Address of font for encoding mode 0x00 and 0x05.
0x1C	0x0B:H	Font_X_Dots	1	Font size in X-direction.
0x1D	0x0B:L	Font_Y_Dots	1	Font size in Y-direction.
0x1E	0x0C:H	TAB_X_Adj_Mod	1	Displaying or not the column header when TAB_X_Start is NOT 0. 0x00: valid display, 0x01: invalid display.
0x1F	0x0C:L	TAB_Y_Adj_Mod	1	Displaying or not the row header when TAB_Y_Start is NOT 0. 0x00: valid display, 0x01: invalid display.

NOTICE[1]:When Encode_mode.6=1, the first two words of data in each unite define format of table:

✧ High byte of the first word:

0x00=integer (2 bytes)	range from -32768 to 32767
0x01=long integer (4 bytes)	range from -2147483648 to 2147483647
0x02=*VP high byte, unsigned number	range from 0 to 255
0x03=*VP low byte, unsigned number	range from 0 to 255
0x04= overlength integer (8 bytes)	range from -9223372036854775808 to 9223372036854775807
0x05=unsigned integer (2 bytes)	range from 0 to 65535
0x06= unsigned long integer (4 bytes)	range from 0 to 4294967295
0x10= time format One，12:34:56	BCD
0x11= time format Two，12-34-56	BCD

0x12= time format Three，YYYY-MM-DD HH:MM:SS BCD

0xFF=Text format

- ✧ First word of low byte：

  Mode=0x00-0x06

  fixed-point format of the variable data,he high 4bit shows integer digits and the low 4bit signified decimal digits.

  Mode=0x10-0X11 : Byte length of BCD

  Mode=Others : Undefined

- ✧ Second word: text color

  If the actual content is shorter than the prescript length of the Unit Data_Num, 0xFFFF has been used as the terminator of cell text the particularly large tables have been modified by value of TAB_X_Start、TAB_Y_Start via touch screen in order to drag and move

8 DGUS FAQ

1 What is DGUS?

DGUS is the abbreviation of DWIN Graphical User Software.

It is mainly designed for the MCU users to develop interface with full graphical and touch screen rapidly and reliably.

2 How to use DGUS?

Configuration design was completed by PC software to develop HMI via DGUS that makes both interactive mode and procedure of control to be independent in consequence that decrease amount of code while code that reading-and-writing of variable memory via serial port required only.

3 How simple that DGUS is?

For example, what is the most complicated in application of display is oscilloscope. Yet, the only work MCU have to be burden is data transmitting to DGUS LCMs via serial port, which is collected from A/D on the premise that DGUS play. Regarding with other such function as zoom-in/out of curve display, horizontal translation, all things can be featured by DGUS performing independently without code of MCU involved.

4 It is admitted that human-machine interface compounded in configuration mode with superiority of speedy working however deprived of distinctive design.

The essential difference between configuration of DGUS and traditional HMI is that DWIN LCMs has 256MB flash memory (maximum can be extend to 2GB),and the graph database is defined by user itself,which means as long as it can be designed by Photoshop, DWIN LCMs COULD BE support in order to idea displayed in 360 degrees.

5 Compared with traditional HMI, what are the typical characteristics of DGUS?

The biggest difference is the software platform. Traditional HMI is designed by general-purpose operating system like WinCE,Linux,Android,etc, whereas DGUS is specialized software, which is the proprietary of DWIN and curing in hardware with following the typical characteristics:

(a)High reliability, stability,and strong anti-interference ability.

(b)Free of royalty fees leads to cheaper cost.

(c)Protect user intellectual property effectively, and avoid destructive competition from copycat of counterparts.

6 Compared with traditional crystal display or UART LCM, what's the typical characteristic of DGUS?

The essence of DGUS is GUI platform on hardware, comparing with traditional crystal display or UART LCM. The typical characteristic is easy to use for second development, high quality, and easy for production. Besides, it is simple to form serialization of products on general hardware platform.

7 How fast is DGUS? Is it necessary for MCU to detect BUSY?

For DGUS, the minimum feasible delay of variable display is 80ms, namely the variable display will be updated for 12 times/1s,which is able to satisfy demand of real-time.

DGUS is designed with whole new thinking. The UART buffer space will never overflow, thereby no need to judge BUSY.

8 128 set of variable in maximum can be displayed in one page, is it enough?

The variable of DGUS is highly abstracted (e.g.one trend curve display is a variable),and also considering the graphic variable contains variety of information in consequence that 10 variables displaying on one page is working enough for general application. Take a temperature controller as example, only 4 the real variables involved: current temperature, setting temperature, the upper limit and lower limit of warnings.

While, most users may regards the keyboard buttons as variables. Actually the press buttons are described by touch configure file, and there's no limit of the quantity, thus it doesn't occupy variable resource.

9 How can DGUS print the current content that the screen displayed to printer?

Standard printer drive is embedded in DWIN OS Builder, which can directly drive port printer to print the display content in specific area.

10 If MODBUS device or PLC can directly connect DGUS screen or not?

Yes, it can. But this has to use DWIN OS which is embedded in DGUS product to do a simple interface program. Users can download the relevant application case from DWIN website.

11 DGUS variable storage area, what to do if want it not to be 0x0000 when power on?

In the CONFIG file, set the secondary (0x04,L22_En)in the R2 register (R2=04) .

At the same time, you are requested to create a variable initialization file and name format as 22+name.bin

Download the 22**.bin and the CONFIG file into DWIN LCM by SD card. When power on the screen, DGUS will auto load 56KB access variable data from 22**.bin as initialization data.

12 What caused my DGUS get stuck when operated (response slow, Icon and animation incoherence)?

This is because there are too much data for DGUS to handle, and it isn't able to handle effectively. The reasons are the below three:

Use transparent ICON display when design the ICON, but there are too much spare area left, leading to some display that mainly used ICON such as Word art display, analog clock display, slider display slowly.

The way to improve it:

When design Icon,try to decrease the spare area that isn't needed.

When use pop-up keyboard, the pop-up area is selected too large (For example: pop up a keyboard with are 800*600 in the 1024*768 screen)leading to too much real time information to handle with, which can effect the display speed.

Customer put many variables in one page, leading to increasing processing task and slowing down processing .This case is very rare.

If it is that case, user can improve DGUS processing speed by set R2 register correctly. Please refer the list below:

DGUS Processing Ability	R2.1 (FreshTime_Se1)	R2.0 (RunMode_Se1)	Explanations
100%	0	0	Standard Mode
125%	0	1	High-speed Mode1
125%	1	0	High-speed Mode2
160%	1	1	High-speed Mode3

13 I want to complete an alarm menu popping up, in the menu there are close button, the operator can touch the button to close the menu. How can it be completed by DGUS?

It can be completed in three ways as below:

Set the alarm menu as an Icon; use the Icon display (0x5A00).Setone "alarm variable" to control the Icon display or not.

Design touch control on each page of every variable Icon .Use the return key value (0xFE05) to pass back the button value, and save it in the "Key variable".

When user's software change the value of "Alarm value", it can complete to control the alarm menu pop up or not.

If the operator press the "close button" area, the user can check the value of "key variable". According to the value of" alarm variable", if the button is effective or not can be known, then decide if close the display or not.

14 To prevent injuries caused by the improper operate,you must press the screen for 0.5 seconds.How to do if I want operate as soon as I press the screen?

Solution 1:Adjust user code.

```
READREG(0x06,0x01,Reg)
if(Reg==0x01||Reg==0x03)
TPOK=1;
if(Reg==0x02)
TPOK=0;
if(TPOK)    P++;
```

Solution 2:Develop the upper code by DWIN OS

15 How to solve the situation that DC/DC,bigger energy consumption and Large current noise interfere the A / D acquisition?

(a) When design the feeder must supply power separate alignment and concentrate at the power.

(b) If there are some circuits cannot be powered separate alignment,power the large interference device (like DGUS LCM) first and power the circuit require high signal-to-noise ratio at end.

(c)Series DGUS LCM with an over 2.2mH 1A(depend on the power of DGUS) power inductors smoothing the current and reducing the current noise.

16 Why when power on the DGUS,it will keeping flicker or work well after flicker for some times? How to solve this?

This is because the power supply is not enough,such as large resistance (including the line resistance),low Output current limit.

You can parallel a large capacity electrolytic capacitor,capacitance is calculated as below:

C=1250/(voltage of the DGUS power supply-DUGS nominal minimum operating voltages) uF

The select of capacitors under the different power supply as the following form shows (the voltage of DUGS power supply is measured from the VCC port) .

DUGS power Supply (v)	nominal minimum operating voltages (v)	capacitance (uF)	Recommend
6	3.6	521	10V 680uF
5	3.6	893	10V 1000uF
4.5	3.6	1389	10V 1500uF
9	7	625	16V 680uF
12	7	250	25V 330uF
24	7	74	50V 100uF

17 I bought an DGUS LCM which nominal operating voltage is 7-24V and current is 300mA@12V,now charging by 18V 0.5A Intrinsically safe power supply via 500m cable(10R).If the DGUS LCM will work well?

First,calculate the power:

Power=18*0.5=9W   minimum load power=（12*0.3）*2=7.2W 7.2<9 the power meets requirements

Second, calculate if voltage for the minimum load power point is in the normal operating range of the DGUS LCM.

The voltage for the minimum load power point is Vcc/2=9V which is between 7V-42V.

The current for the minimum load power point is Vcc/(2*R)=900mA which is under 3.6/9=400Ma.

Conclusion: The DGUS LCM will work well.

The current for the point is calculated as below:

I=（V-Sqrt（V*V-4*R*P））/(2*R)   V is the voltage for the power,R is the resistance for the cable,P is the loadpower

V=18 R=10 P=3.6   I=0.23A   current voltage=18-0.23*10=15.7V

18 If we can order special DGUS software for our own project?

Special DUGS software needs more research investment with high human cost, but we are willing to make a win-win cooperation if the profit is enough to the investment.(Bulk purchase or make an additional payment for the research)

In fact,the function of DGUS is mature now and many special requirement can be achieved by our engineer with DWIN OS.

Record of Revision

Date	Content	DGUS Version
2012-10-26	First edition.	V3.7
2012-11-05	1. Revise the last 2 bits definition of register R2 in CONFIG.TXT to set DGUS cycle as 200ms/160ms/120ms/80ms. 2. Data Display (0x5A10) supports 64-bit integer, 32-bit unsigned integer and 16-bit unsigned integer now. 3. Data Input (FE00) supports 64-bit integer now. 4. Touchscreen calibration will be disabled when SD card is disabled.	V4.3
2012-11-12	1. Add the command in CONFIG.TXT to activate touchscreen calibration once: TP_CORRECT. 2. Add the command in CONFIG.TXT to re-able SD card: SD_UNLOCK_8-bit code.	V4.5
2012-12-04	1. Add the function in Word Art (5A03): right-aligned. 2. Add the function in Basic Graphic Display (5A21): Segment Display (0x000A). 3. Add running time after power on, saved in registers 0C – 0F.	V4.7
2012-12-08	Add DGUS reset register in register space (0xEE-0xEF).	V4.9
2013-01-15	Adjusted the error when data written in com2 which may result in blurred screen	V5.0
2013-01-18	1. 0x000B for line segment line released in Command 0x5A21; 2. BCD format support in Command 0x5A22; 3. Character display was added in Command 0x5A21	V5.0
2013-03-05	1. Updated to V53,debugged the error for SD card lock-out&in without Config.txt, and add maximum variable in one page to 128+32;64+32/128+32 can be selected via RC.4 2. 0x000D Command XOR was added in 0x5A21 for color apply; Besides, 0x000E for bio-images display,0x000F bitmap display command were updated for real time icon display; 3. 0x5A23 was cancel off and updated to ODM service which is closer to actual application	V5.3
2013-04-02	1. Supported cleaning function of curve buffer; 2. 64bit unsigned square root calculation available in DWIN OS	V5.5
2013-05-18	1. Added Command SCANADD in DWIN OS which is used for character string adding in input buffer, in order to Associable input; 2. Added variable 0x1C in 0xFE02 slider adjustment to support one-step/continuous adjusts of button pressing.	V5.6
2013.08.16	1.negative coordinate of centre of circle in Command 5A21_05 2. music play added in Reg. 0x50-0x54 for 128 segment of audio.	V5.8
2013.11.22	1.database added in Reg.0X56-0X5F, as maximum 960MB to export via SD card. 在 0x56-0x5F 2.Take off No.32-128 Font File exporting via SD Card. 3.Command MOVXL in DWIN OS support database operation 4.Command 5A21 added for area zoom-in.	V6.0
2014.01.06	1. Add definition of RC.3 In Register for on-and-off CRC checksum 2.5 points calibration, set via RC.2 in RC register. 3. Calibration via serial under writing of 0x5A to Reg. 0xEA.	V6.2

If any doubts or questions are still existing in operation, as well as you hope to learn more information about DWIN Technology，please feel free to click our website www.dwin.com.cn or welcome to mail us dwinhmi@dwin.com.cn. Thanks for your supports as always.