迪文 DGUS 屏用户开发指南

(Ver4.1　2015.01)

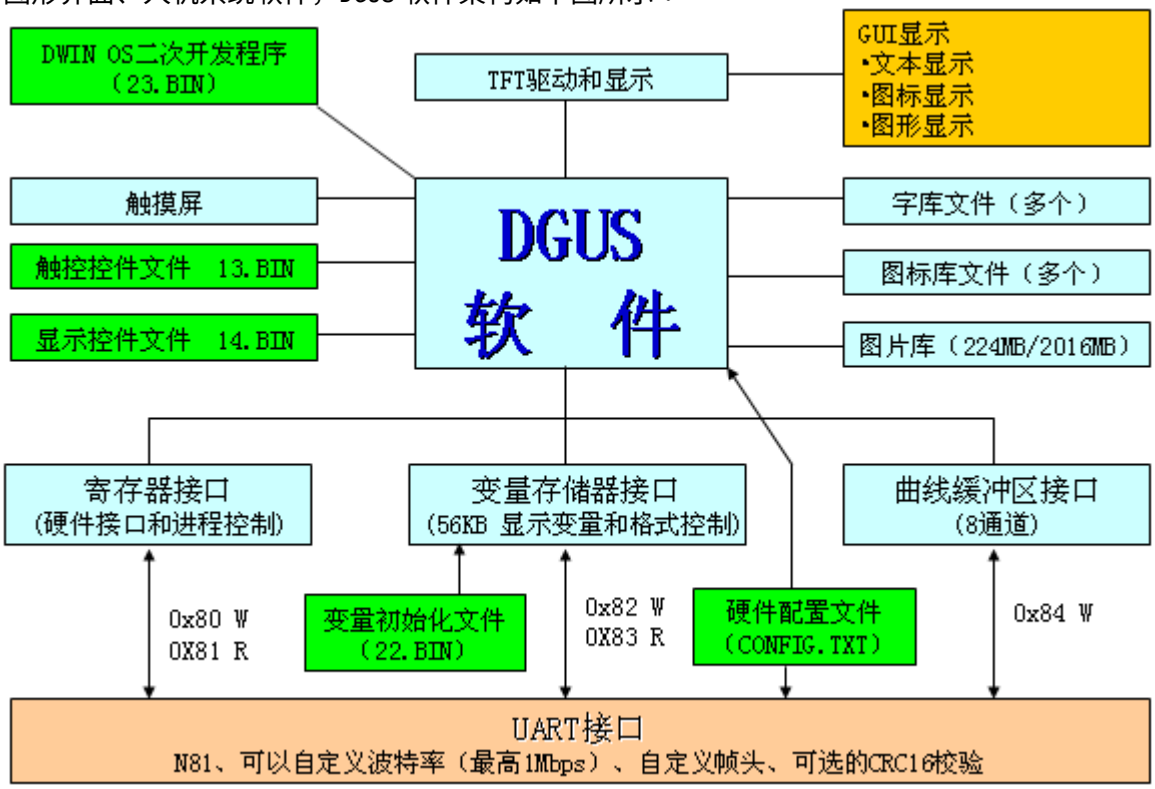
北京迪文科技有限公司

目 录

1 DGUS 概述

1.1 认识迪文 DGUS 屏

DGUS（DWIN Graphic Utilized Software）是北京迪文科技有限公司基于 K600+内核迪文屏所设计的智慧型、图形界面、人机系统软件，DGUS 软件架构如下图所示：



出厂预装 DGUS 软件的屏称为 DGUS 屏，一个典型的 DGUS 屏（DMT80480T070_07WT）硬件组成如下图所示：

1.2 DGUS 屏主要特点

DGUS 屏的主要特点包括：

- 把 GUI 分解成控件并按页面来配置，控件显示直接由变量控制；
  在通过 PC 软件配置好控件文件（14.BIN）下载到 DGUS 屏后，用户仅仅需要通过串口改写变量值即可实现控件显示的相应改变。
  比如，要在某个页面显示两位小数（25.00）的温度值，分两步开发完成：
  Step1 配置：借助迪文提供的 PC 组态开发软件，在相应页面位置添加一个数据变量控件，设置好显示格式（字体大小、显示颜色、单位、小数点长度、数据源、数据类别），预览 OK 后通过 SD 卡把生成的控件文件（14.BIN）下载到 DGUS 屏。
  Step2 运行：用户软件只需要定时（或者参数变化时）把温度值通过串口刷新到对应的数据源地址，当显示切换到对应页面时，屏幕就自动按照预先设定显示格式显示出来。
- 触摸屏或键盘录入过程，通过 PC 软件按照页面定义的触控文件（13.BIN）来控制，用户软件仅需要定时（或者参数改变时的串口中断触发）来读取录入变量值即可。
  比如，要在某个页面用触摸屏录入显示两位小数（25.00）的设定温度值，分两步开发完成：
  Step1 配置：借助迪文提供的 PC 组态开发软件，在相应页面位置添加一个变量数据录入控件，设置好录入格式（字体大小、光标模式、显示颜色、小数点长度、数据源、数据类别），预览 OK 后通过 SD 卡把生成的控件文件（13.BIN）下载到 DGUS 屏。
  Step2 运行：当切换到对应页面，并按压触摸屏触发相应控件，DGUS 会自动完成录入过程。用户软件只需要定时（或者配置为录入完成自动串口下发给用户）查询录入值即可。
- 56KB 变量空间，8 通道曲线趋势图存储器，极快（最快 80mS）的变量显示响应速度；
- 256 字节配置寄存器空间，串口指令读写，用于硬件控制和操作；
- 256MB（可以扩展到 1GB、2GB）Flash 存储器，海量图片、图标、字库存储；
- 每页可设置多达 128 个显示控件（支持显示控件叠加）和任意多的触控控件。
- SD/SDHC 接口，FAT32 文件格式，可以使用 SD 卡来实现 DGUS 屏硬件参数配置、图片数据下载、软件升级，批量生产时尤其方便，并便于生产档案管理。
- 集成了 RTC（公历/农历）、集成背光亮度调节、背光自动待机、触控蜂鸣器伴音功能；
- 支持语音播放功能、支持电容触摸屏、可以在图片存储器空间构造高可靠性用户数据库；
- 集成的 DWIN OS 平台，丰富的指令，可以允许用户把一部分代码放到 DGUS 屏上运行，让用户二次开发变得简单，也提供了 DGUS 屏做为系统主控设备的可能。
  迪文 OS 平台集成了数学运算（包括 MAC、CRC）、数据存储（包括 Flash 数据库读写）、串口通信、常用通信协议处理（比如 Modbus 协议、DL/T645 电力抄表协议等）、串口外设（比如打印机）驱动、DGUS 进程控制等指令，典型的应用案例包括 Modbus 总线管理、电力抄表、票据打印、POS 设备等。
- 可靠的硬件平台（基于迪文 ASIC 的 HMI 平台架构，已经历了将近 10 年的工业应用考验）、迪文自主知识产权的软件设计（DGUS 软件采用汇编代码设计，总代码量约 50KB），使 DGUS 屏不仅性能优越，运行也极其稳定可靠。
- 通过 TUV CE 和 RoHS 认证。

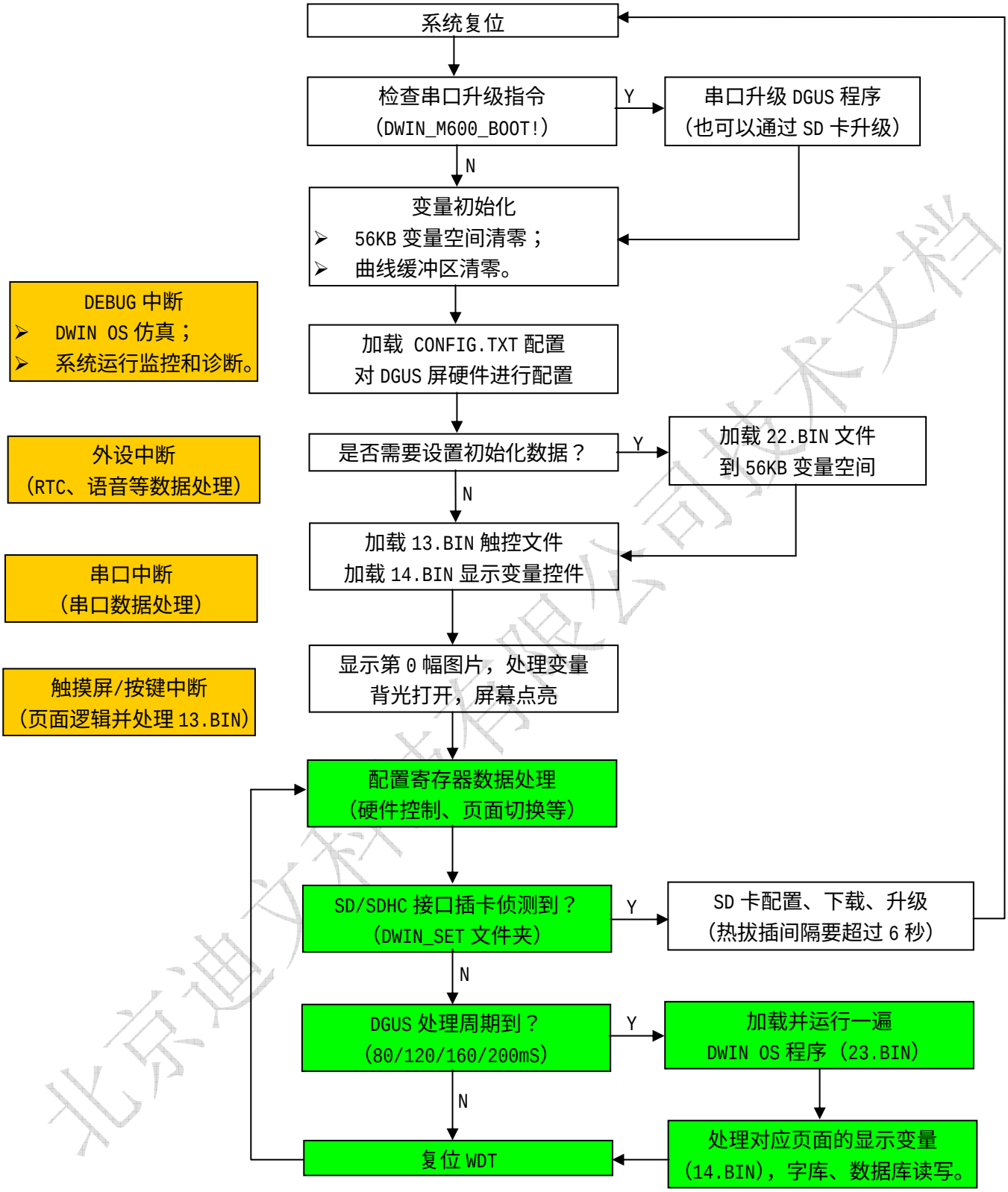
1.3 DGUS 屏数据格式

由于主要面向MCU等嵌入式系统应用，为了用户处理的方便，DGUS屏使用的数据采用整数（字）、无符号整数（字）、长整数（双字）、超长整数（4个字）表示，相关表示范围如下：

整数：-32768（0x8000）到+32767（0x7FFF）
无符号整数：0（0x0000）到65535（0xFFFF）
长整数：-2147483648（0x80000000）到+2147483647（0x7FFFFFFF）
超长整数：-9223372036854775808 到 9223372036854775807
小数采用定点小数表示，用户自定义小数位数，比如0x4D2（1234），规定小数为2位时，表示12.34。
DGUS屏使用65K色颜色系统，调色板定义如下：

DGUS 使用的 65K 设调色板位定义																	
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
	红色 0xF800					绿色 0x07E0						蓝色 0x001F				

1.4 DGUS 软件处理流程图



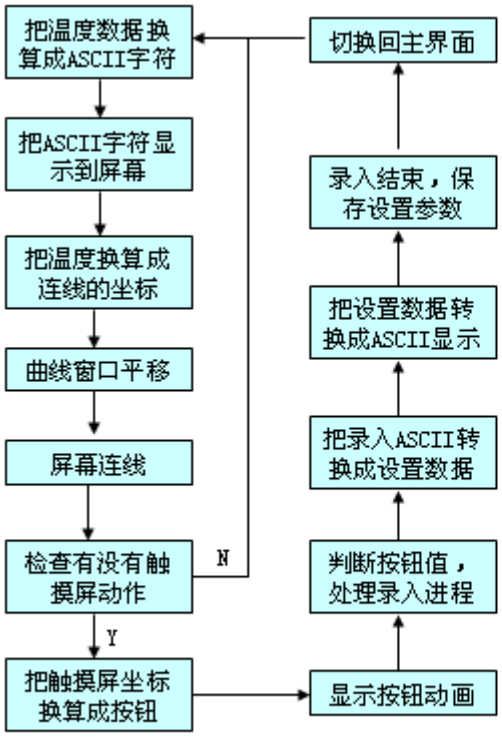
说明：

DWIN OS 程序在每个 DGUS 周期（80/120/160/200mS）都完整运行一遍，所以 DWIN OS 程序中不能出现死循环或者通过指令循环的延时。

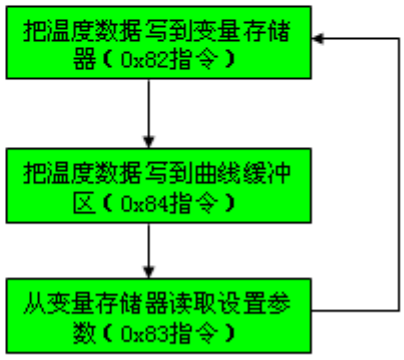
1.5 DGUS 屏开发流程

与传统的 LCM 通过时序或者指令控制显示不同，DGUS 屏采用直接变量驱动显示方式，所有的显示和操作都是基于预先设置好的变量配置文件来工作的。两种不同的工作方式导致用户应用时的软件架构和二次开发难度完全不同。

举例，假设做一个简单的触摸屏温控仪，要在当前页面显示测量温度，点击触摸屏切换到设置页面进行参数设置。两种不同开发方式下的软件流程图如下：



基于普通LCM开发温控仪流程图



基于DGUS屏开发温控仪流程图

上面的例子，只是一个两个参数、两个页面的最简单 GUI，如果对于实际应用中稍微“高级”一点的产品，几十个参数，几十个页面，还要考虑动画、图标等等吸引眼球的 GUI，前一种方式需要 1 个优秀工程师加班加点干 1 年，而使用 DGUS 屏开发，3-4 个工程师（可以并行协同做）2-3 天就搞定了。

在有些中、小型工业自动化项目应用中，当整个系统由一些相对独立、功能完善的组件（比如支持 Modbus 协议的温控仪或其它二次仪表）构成时，用户可以直接把 DGUS 屏当成主机使用，用 485 网络把设备和 DGUS 屏组成一个网络，基于 DGUS 屏上搭载的 DWIN OS 二次开发平台来开发用户主控软件直接在 DGUS 屏上运行，替代用户 CPU 的工作。

总的来说，DGUS 屏是基于配置文件来工作的，所以整个开发过程也就是通过 PC 软件辅助设计完成变量配置文件的过程，基本开发流程如下：

第1步：变量规划

推荐客户开发过程中用 Excel 表格来记录、整理好变量分配记录，便于将来的修改、升级维护。

第2步：界面设计

利用PS（或者其它绘图软件）进行界面及界面相关元素（图标、字库）设计。设计过程中，请选择调色板系统为65K色，确保最终显示效果和设计效果一致。如果想让您的产品至少看起来很有价值，建议委托专业美工或者工业设计公司来设计UI和相关界面，一般收费在200人民币/1个页面左右。

第3步：界面配置

利用迪文提供的工具软件进行界面的配置，生成触控配置文件(13.BIN)和变量配置文件(14.BIN)。

第4步：测试修改

把配置文件、图片、字库、图标库等借助SD卡下载到DGUS屏，进行界面测试和修改（第2-3步）。
把串口连上用户MCU系统，进行数据联调。

第5步：定版归档

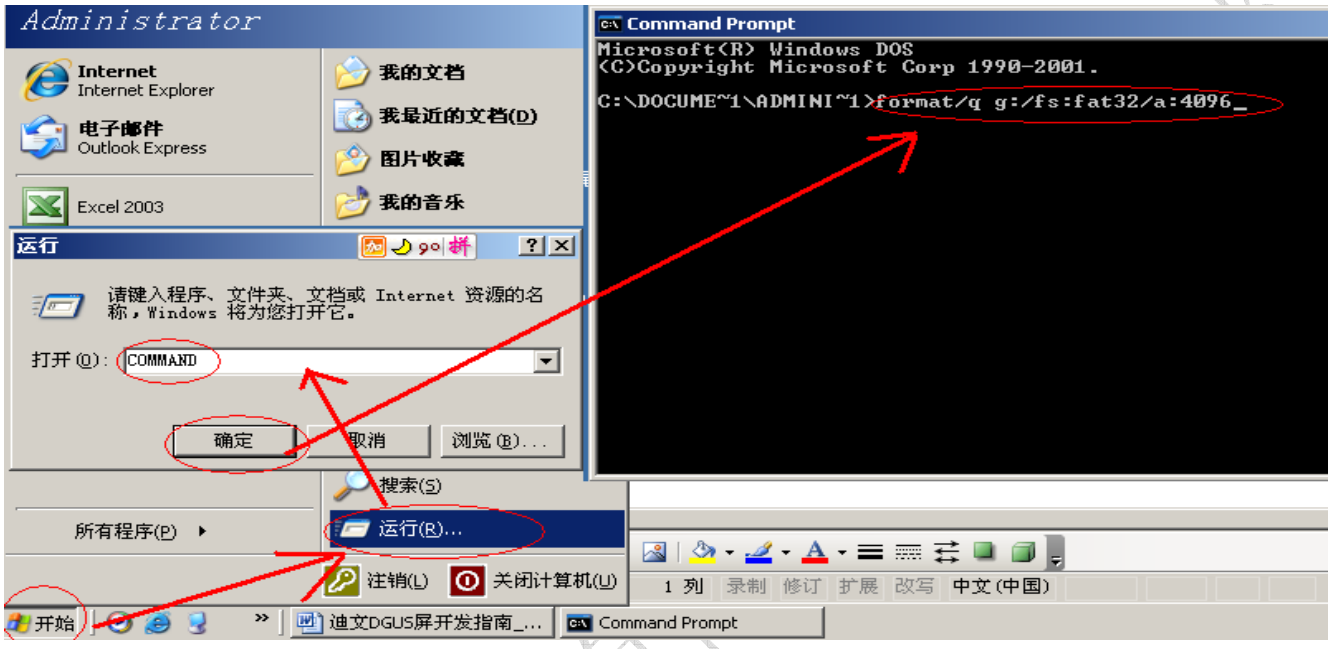
定版后，把配置文件、图片、字库、图标库等DGUS屏涉及的文件保存在一张SD卡转生产即可。
如果不希望最终客户通过SD卡接口改变或者导出内部数据，可以对SD卡接口加密锁死(加密后用户必须保管好SD卡的开锁密码，一旦SD卡锁死，没有开锁密码，只能返厂更换内核CPU才能使SD卡接口再次启用)。

2 DGUS 屏配置

2.1 SD/SDHC 配置接口

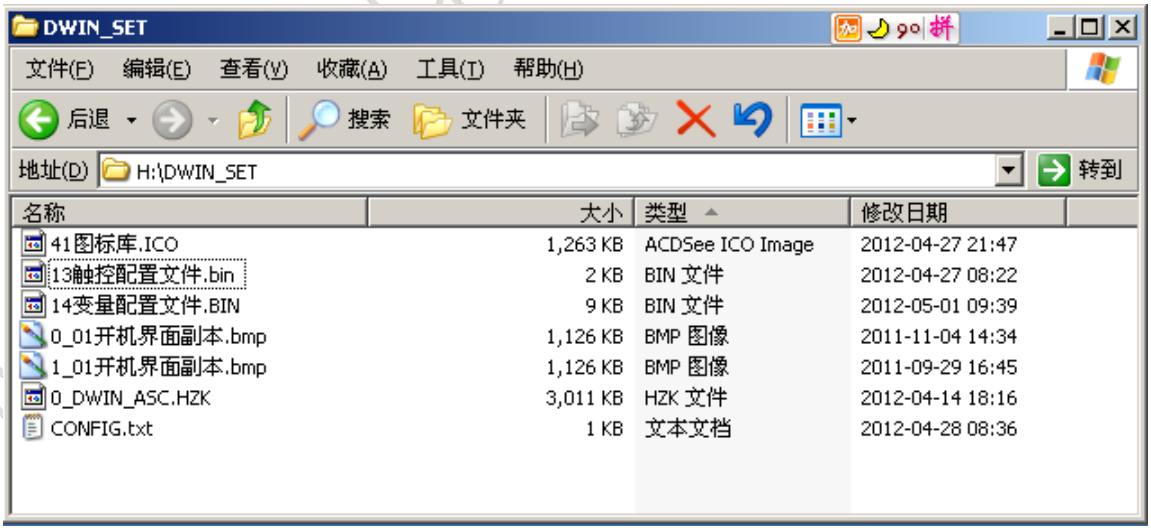
DGUS 屏的所有硬件参数设置和资料下载，都通过屏上的 SD/SDHC 接口来完成，文件必须使用 FAT32 文件格式。

第一次使用 SD 卡前，推荐先格式化一次，流程如下：
第 1 步：在 windows 的　开始//运行,键入 command 运行 DOS 系统；
第 2 步：把 SD 卡格式化，键入指令：　　format/q g:/fs:fat32/a:4096
其中 g 是 SD 卡的盘符，不同的电脑用对应的盘符（比如 h,i）替换即可；如下图所示：



使用 SD 卡下载数据的流程

- 在 SD 卡根目录下建立 DWIN_SET 文件夹；
- 把需要下载的图片、字库、配置文件都放在 DWIN_SET 文件夹中，如下图所示：



- 把 SD 卡插到 DGUS 屏的 SD 卡接口上，DGUS 屏检测到 SD 卡后，会显示蓝屏提示用户检测到 SD 卡，然后开始屏参数配置，或将数据下载到屏上的 Flash 中；
  两次 SD 卡热拔插之间必须间隔至少 6 秒，不然 DGUS 屏会认为是同一张卡而不会启动 SD 卡操作。如果用户已经禁止 SD 卡接口，要启用 SD 卡接口，除非事先解锁或在 SD 卡的 CONFIG.TXT 文件中有解锁指令。

- SD 卡下载完成，DGUS 屏会自动复位一次，用户拔出 SD 卡，下载结束。

2.2 SD/SDHC 下载文件格式说明

为防止误操作，DGUS 屏对 SD/SDHC 配置文件有严格的命名和格式要求，不然会导致出错。

2.2.1 图片下载

图片文件必须是和 DGUS 屏分辨率相同的 24 位色 BMP 格式文件，文件的命名必须是表示图片存储位置的阿拉伯数字开头。

比如，要把一副图片用 SD 卡存储到 DGUS 屏的第 20 个图片存储位置，图片文件可以命名成“20 测试.BMP”、“20.BMP”或者“020 测试.BMP”，但不能命名成“测试 20.BMP”。

不同分辨率 DGUS 屏最大存储图片数量如下表所示：

类 别	存储器空 间	字库	不同分辨率 DGUS 屏最大存储图片数量						
			320*240	480*272	640*480	800*480	800*600	1024*600	1024*768
标 准	256MB	32MB	836	836	278	278	209	167	139
扩展到 1GB	1GB	32MB	3728	3728	1242	1242	932	745	621
扩展到 2GB	2GB	32MB	7584	7584	2528	2528	1896	1516	1264

2.2.2 字库下载

DGUS 屏一共有 32MB 字库空间，分割成 128 个容量固定为 256KB 的字库空间。

和下载图片类似，字库文件的命名也必须是表示字库存储位置（0-127）的阿拉伯数字开头。

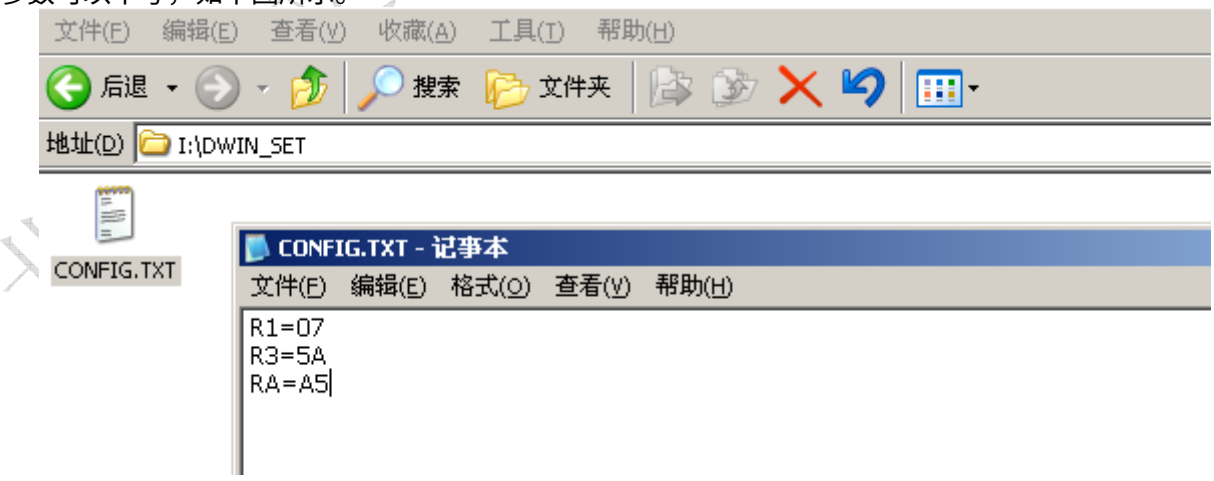
下载的字库文件包括字库、输入法词库、配置文件、图标文件；

字库文件格式说明如下表所示：

SD 卡文件格式说明			
文件类型	命名规则	举 例	说 明
字库文件	字库存储位置+（可选的）文件名.BIN/HZK/DZK	32_GBK12 汉 字库.DZK	可以由 TS3 字库提取软件生成
图标库	字库存储位置+（可选的）文件名.ICO	41 图标库.ICO	迪文工具箱生成
专用 ASCII 字库	0*.HZK，固定位置 0-11	0_DWIN_ASC.HZK	迪文工具箱生成
输入法词库	12*.BIN，固定位置 12	12_PY_DGUS.BIN	迪文预装
触控配置	13*.BIN，固定位置 13	13 触控文件.BIN	DGUS 组态软件生成
变量配置	14*.BIN，固定位置 14-21	14 变量文件.BIN	DGUS 组态软件生成
变量初始化	22*.BIN，固定位置 22	22 变量初始化.BIN	
OS 代码	23*.BIN，固定位置 23	23 水处理.BIN	基于 DWIN OS
0-23 号字库（6MB）被 DGUS 系统预留做为将来系统升级使用，用户字库建议从 24 位置开始使用。			

2.2.3 CONFIG.TXT 配置文件

CONFIG.TXT 文件采用 windows 的文本文档格式，类似脚本语言的方式来描述参数寄存器，每一行描述一个参数(格式必须是 R?=HH 其中？是寄存器序号，HH 是寄存器配置值的 16 进制（HEX）值，**必须大写**)，不用的参数可以不写，如下图所示。



上例中，RA=A5 表示把 RA 寄存器内容配置成 0xA5。

不能写成 ra=5a 或者 RA=5a，并且每个寄存器配置后必须换行写下一个寄存器配置。

CONFIG.TXT 文件相关参数说明请参考“2.3 DGUS 屏硬件参数配置”。

2.2.4 声音文件下载

有些 DGUS 屏（具体看硬件规格书说明）支持 128 段语音播放，需要事先下载声音文件存储到屏里面。

和下载字库类似，声音文件的命名也必须是表示声音存储位置（0-127）的阿拉伯数字开头（比如 12 请刷卡.WAV），声音文件的后缀是.WAV，采用 32KHz 采样，16bit 单声道 WAV 文件格式。

声音文件采用额外的 Flash 存储，不占用 DGUS 的 FLASH 空间，下载存储速度约为 32KB/S。

2.2.5 DGUS 应用程序升级

DGUS 屏支持 SD 卡升（降）级应用程序，把应用程序（DGUS_V*.BIN）放到 SD 卡 DWIN_SET 目录下即可。

2.2.6 用户数据库 SD 卡导出

用户数据库是在 DGUS 屏的图片存储器中拿出一块空间进行数据存储，空间大小和位置由用户控制，不同内核其 Flash 大小不同，导致图片存储器空间和可用数据库空间也不相同，如下表所示：

类 别	图片空间最大容量	数据库空间最大容量	不同分辨率 DGUS 屏最大存储图片数量						
			320*240	480*272	640*480	800*480	800*600	1024*600	1024*768
标 准	210MB	89MB	836	836	278	278	209	167	139
扩展到 1GB	932MB	450MB	3728	3728	1242	1242	932	745	621
扩展到 2GB	1896MB	960MB	7584	7584	2528	2528	1896	1516	1264

用户数据库读写过程中，DGUS 对数据进行了加密和前向纠错（FEC）操作，确保数据存储的可靠性。

用户数据库导出方法如下：

- 先计算出导出数据库的起始页位置

  假设要导出的数据库首地址为 ADR，那么地址的 最高两个字节+256 即为对应的 SD 卡导出页 ID。
- 用起始页位置命名创建一个和导出数据库等大（对齐到 128KB）的 DAT 文件：

  字库起始页位置+（可选的）文件名.DAT
- 把这个 DAT 文件放到 SD 卡 DWIN_SET 文件夹下面，插入 DGUS 屏 SD 卡接口，DGUS 屏会自动把指定的数据库内容读取出来覆盖 SD 卡上的 DAT 文件。

举例，

假设要导出数据库空间 0x00 10 00 00 到 0x00 17 FF FF 共 1MB（512KW）数据，

那么对应的 SD 导出页 ID 是： 0x00 10+256=272

在 SD 卡的 DWIN_SET 目录下放置 1 个大小是 1MB 的文件 272 数据库记录.DAT（或者其它 272***.DAT 的文件名都可以），把 SD 卡插入 DGUS 屏 SD/SDHC 接口即可把数据库内容导出。

数据库导出速度大约为 180KB/S，对于很大的数据库导出，可以分成几个文件来处理。

关于用户数据库的详细说明请见“**4.5 数据库读写**”。

2.3 DGUS 屏参数配置

DGUS 屏参数配置通过在 CONFIG.TXT 文件中写好寄存器参数，然后用 SD 卡下载到 DGUS 屏中实现。

2.3.1 屏幕物理分辨率设置（R0）

显示屏物理分辨率由 R0 寄存器设置，如下表所示。

R0 设置	分辨率设置（H*V）	典型 DGUS 屏	备　注
00	640*480	DMT64480T056_03W	
01	640*480	DMT64480T057_01W	
02	800*480	DMT80480T070_07W	
03	800*600	DMT80600T080_07W	
04	1024*768	特殊定制屏	
05	1024*768	DMT10768T057_01W	
06	800*600	特殊定制屏	
07	800*600	特殊定制屏	
08	800*600	MVGA01、MDVI01	
09	1024*768	DMT10768T150_02W	
0A	1280*800	未使用	
0B	1024*600	DMT10600T102_02W	
0C	1366*768	未使用	
0D	240*320	特殊定制屏	
0E	320*240	特殊定制屏	DMT32240T035_02W 早期也使用此模式
0F	480*272	DMT48270T043_03W	
10	480*272	特殊定制屏	
11	800*480	特殊定制屏	
12	320*240	DMT32240T035_02W	

R0 寄存器出厂已经设置好，用户无须再配置。R0 配置错误将导致显示异常。

2.3.2 显示位时钟相位选择（R4）

DGUS 使用的液晶屏，由于 TCON 不同，起显示数据和显示位时钟的相位关系也有两种，由 R4 设置：

R4=00　显示数据在显示位时钟下降沿锁存

R4=FF　显示数据在显示位时钟上升沿锁存

R4 寄存器出厂已经设置好，用户无须再配置。R4 配置错误将导致显示画面抖动或者出现毛边。

2.3.3 串口波特率设置（R1、R5、R9）

DGUS 屏用户接口的串口波特率由 R1、R5、R9 寄存器设置。

- 当 R1 取值在 00-10 时，R5、R9 无效，可以选择 17 档固定波特率之一，如下表(波特率单位为 Kbps)：

R1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
波特率	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

- 当 R1 取值为 11 时，此时波特率由 R5、R9 决定，并由下式计算：

R5:R9=6250000/波特率　R5:R9 表示一个双字节参数，R5 为高字节，R9 为低字节。

比如，设定 10000bps 波特率，R5:R9=6250000/10000＝625=0x0271　R5=02　R9=71

DGUS 屏出厂波特率预设值 R1=7，波特率为 115200bps。

2.3.4 串口通信帧头设置（R3、RA）

DGUS 屏的串口数据帧由 5 个数据块组成，如下表：

数据块	1	2	3	4	5
定　义	帧 头	数据长度	指令	数据	指令和数据的 CRC 校验
数据长度	2	1	1	N	2
说　明	R3:RA 定义	数据长度包括指令、数据和校验	0x80-0x84		R2.4 决定是否启用

通信帧头的设置主要达到以下两个目的：

（1）用于串口数据帧的识别和同步；

（2）多台 DGUS 屏并联工作时，把帧头做为设备地址加以区分；

假设设置 R3=AA RA=BB，那么串口指令必须以 0xAA 0xBB 开头（比如读寄存器指令 AA BB 03 81 00 10），DGUS 屏才会接收。

DGUS 屏出厂通信帧头预设值 R3=5A RA=A5 ，帧头为 0x5A A5。

2.3.5 软件工作模式配置寄存器（R2、RC）

R2、RC 寄存器按位（bit）定义，用于配置 DGUS 屏软件工作模式，如下表所示（阴影表示出厂设置值）。

➢ R2 寄存器定义

位	权重	定义	说明
.7	0x80	VDS	0=正常显示 1=偏转 90° 显示
.6	0x40	HDS	0=正常显示 1=偏转 180°（反视角）显示
.5	0x20	TP_LED	0=背光不受触摸屏状态控制 1=背光受触摸屏状态控制，控制参数由 R6、R7、R8 寄存器设定
.4	0x10	FCRC	0=不启用串口通信的 CRC16 帧校验 1=启用串口通信的 CRC16 帧校验
.3	0x08	TPSAUTO	0=触摸屏录入参数后不自动上传（用户查询） 1=触摸屏录入参数后是否自动上传到串口由相应触控变量的配置决定
.2	0x04	L22_Init_En	0=56KB 变量存储器上电初始化数据为 0x00 1=56KB 变量存储器上电初始化数据由 22 字库文件加载
.1	0x02	FRS1	设置 DGUS 周期，DGUS 周期越小则变量响应越灵敏，但处理变量的能力越低。 DGUS 周期 / 80mS / 120mS / 160mS / 200mS FRS1 / 1 / 1 / 0 / 0 FRS0 / 1 / 0 / 1 / 0 对于 1024*768 分辨率，建议 DGUS 周期设置成 120mS 以上。 DGUS 周期会影响动画图标显示的动画速度。
.0	0x01	FRS0	

➢ RC（AUX_CFG 配置字）说明

位	权重	定义	说明
.7	0x80	系统保留	必须写 0
.6	0x40	RUN_OS_EN	0=不运行 DWIN OS 程序 1=运行 DWIN OS 程序
.5	0x20	TP_BUZZ_EN	0=点击触摸屏有效区域时有蜂鸣器提示音； 1=点击触摸屏有效区域时无蜂鸣器提示音，但仍旧可以通过向 0x02 寄存器写入数据控制蜂鸣器鸣叫。
.4	0x10	PAGE128_EN	0=每页显示变量数目为 64 个，必须在 DGUS PC 软件平台对应选择 64 变量模式。 1=每页显示变量数目为 128 个，必须在 DGUS PC 软件平台对应选择 128 变量模式。
.3	0x08	CRC_ACK_EN	0=启动 CRC 帧校验后，不自动应答帧校验结果。 1=启动 CRC 帧校验后，自动应答帧校验结果。
.2	0x04	TP_CAL_MOD	0=触摸屏采用 3 点校准模式。 1=触摸屏采用 5 点校准模式。
.1	0x02	未定义	写 0
.0	0x01	未定义	写 0

2.3.6 屏幕显示方向设置（R2.7 R2.6）

DGUS 屏有 4 种不同显示方向（下图所示）可以选择，由 R2.7（VDS）、R2.6（HDS）设置。

VDS=0 HDS=0　VDS=1 HDS=0　VDS=0 HDS=1　VDS=1 HDS=1

2.3.7 触摸屏控制背光（R2.5 R6 R7 R8）

当设置 R2.5=1 时，背光亮度将受触摸屏状态控制（背光待机后，第一次点击触摸屏不会触发动作）。

R#	取值范围	说 明
R6	0x00-0x40	触摸屏控制背光启动后，点击触摸屏后背光点亮亮度。
R7	0x00-0x40	触摸屏控制背光启动后，一段时间不点击触摸屏，背光关闭的亮度。
R8	0x01-0xFF	触摸屏控制背光启动后，触摸屏背光点亮时间，单位为 1.0 秒。

举例，设置 R2.5=1 R6=40 R7=10 R8=1E，30 秒（0x1E）不点击触摸屏，背光亮度将自动降低到 0x10（25%）；点击触摸屏后，背光亮度将自动调节到 0x40（100%）。

2.3.8 触摸屏校准

DGUS 屏有 3 种方式进入触摸屏校准模式。

- 校准方式 1

开机状态下，如果 4 秒内快速点击触摸屏的非触控区域超过 20 次，则进入触摸屏校准模式，步骤如下：

（1）4 秒内快速点击触摸屏非触控区域超过 20 次；

（2）蜂鸣器长鸣 1 秒，听到蜂鸣器鸣叫时停止点击；

（3）进入校准模式，按照十字交叉线的提示点击触摸屏的指定位置校准触摸屏；

（4）校准结束，返回进入校准前的画面。

- 校准方式 2

在 CONFIG.TXT 文件中，写入一行特殊文本“ TP_CORRECT ”将启动一次触摸屏校准过程。

- 校准方式 3

通过串口向 0xEA 寄存器写入 0x5A 将启动一次触摸屏校准过程。

DGUS 屏会自动检测校准是否有效，当校准无效时（比如误操作导致进入校准模式），DGUS 屏不会被被错误的设置（校准）。如果多次校准触摸屏都无效，往往是触摸屏已经被物理损伤，比如触摸屏破裂。

当 SD 卡接口被禁止后，除非解锁，否则将不能进行触摸屏校准。

2.3.9 SD/SDHC 接口禁止和解锁

- SD/SDHC 接口禁止

在客户测试完成正式量产后，为了防止在应用中通过 SD 卡进行错误的升级或下载操作，导致工作不正常。可以通过在 CONFIG.TXT 文件中，增加一行特殊文本来禁止 SD 卡接口，说明如下：

CONFIG.TXT 文档中禁止 SD 接口文本的说明		
第 1 部分	SD_LOCK_	固定
第 2 部分	1000	用来重新启用 SD 接口的密码保存在变量存储器空间的地址，0000-6FF8。
第 3 部分	ABCD1234	重新启用 SD 接口的 8 位密码。

举例：

假设禁止 SD/SDHC 接口后的重新启用密码为 12345678，密码保存在变量存储空间的 0x6000 位置。

禁止 SD/SDHC 接口的步骤：

（1）在 CONFIG.TXT 文档中增加指令： SD_LOCK_6000_12345678

（2）把 CONFIG.TXT 用 SD 卡配置 DGUS 屏；

（3）之后 DGUS 将禁止 SD/SDHC 接口。

- SD/SDHC 接口解锁（取消禁止）

以下 3 种方法可以对 SD/SDHC 接口解锁（取消禁止），以解锁上面的禁止过程为例说明如下：

方法 1：

通过串口发送正确的密码到正确的存储空间位置，SD 卡将被激活一次。

假设用户设置的帧头为（0xA55A）：A5 5A 0B 82 60 00 31 32 33 34 35 36 37 38。

方法 2：

使用触摸屏 ASCII 文本录入功能来设置一个“ 解锁 ”操作菜单，也可以激活一次 SD 卡。

方法 3：

CONFIG.TXT 文档中写入取消 SD 卡禁止的命令 SD_UNLOCK_密码 ，存入 SD 卡去重新激活 SD/SDHC 接口。比如，SD_UNLOCK_12345678 。

如果 SD 卡被禁止，用户务必妥善保管好启用密码，否则 DGUS 屏将不能更新数据、资料和校准触摸屏。

3 串口操作

迪文DGUS屏采用异步、全双工串口（UART），串口模式为8n1，即每个数据传送采用10个位：1个起始位，8个数据位，1个停止位。

串口波特率通过SD卡来配置。

串口的所有指令或者数据都是16进制（HEX）格式；对于字型（2字节）数据，总是采用高字节先传送（MSB）方式。比如0x1234传送时先传送0x12。

DGUS屏的串口接收FIFO为4KB，即1个DGUS周期（80/120/160/200mS）内可以传送至少4KB数据（约等于230400-691200bps波特率连续发送）；一个DGUS周期能够传送的最大数据长度取决于用户界面的复杂程度；推荐客户在一个DGUS周期内不要发送超过4KB的数据给DGUS屏。

3.1 数据帧架构

迪文 DGUS 屏的串口数据帧由 5 个数据块组成，如下表所述：

数据块	1	2	3	4	5
定　义	帧 头	数据长度	指令	数据	指令和数据的 CRC 校验
数据长度	2	1	1	N	2
说　明	CONFIG.TXT 配置文件的 R3:RA 定义。	数据长度包括指令、数据和校验	0x80-0x84		CONFIG.TXT 配置文件的 R2.4 决定是否启用
举例	5A A5	05	81	00 10	20 24

CRC 校验不包括帧头和数据长度，仅针对指令和数据，采用 ANSI CRC-16(X16+X15+X2+1) 格式。

当启用 CRC 帧校验应答（R2.4=1 RC.3=1）后，DGUS 屏会在 CRC 校验后自动应答校验情况：

帧头+02+（DGUS 屏接收的）指令+数据（0xFF 表示 CRC 校验正确　0x00 表示 CRC 校验错误）+CRC

3.2 指令集

DGUS 屏采用变量驱动模式工作，屏的工作模式和 GUI 状态完全由数据变量来控制。相应的，串口指令也只需要对变量进行读、写即可，指令集非常简单，一共只有 5 条指令。

功能	指令	数　据	说　明
访问寄存器	0x80	下发：寄存器地址(0x00-0xFF)+写入数据	指定地址写寄存器数据
	0x81	下发：寄存器地址(0x00-0xFF)+读取字节长度(0x00-0xFF)	指定地址开始读指定字节长度的寄存器数据
		应答：寄存器地址(0x00-0xFF)+字节数据长度+读取的寄存器数据	读寄存器的 DGUS 屏应答
	DGUS 屏有 256Byte 的寄存器，主要用于硬件操作的软件接口，按照**字节（Byte）**寻址操作。		
访问变量存储器	0x82	下发：变量存储器地址(0x0000-0x6FFF)+写入的变量数据	指定变量地址开始写入数据串(字数据)到变量存储区
	0x83	下发：变量存储器地址(0x0000-0x6FFF)+读取变量数据字长度(0x00-0x7F)	从变量存储区指定地址开始读入 RD_LEN 长度字数据
		应答：变量存储器地址+变量数据字长度+读取的变量数据	读数据存储器的 DGUS 屏应答
	DGUS 屏有 28K word（56K Byte）的变量存储器，主要用于 GUI 变量数据存储，按照**字（Word）**寻址操作。		
写曲线缓冲区	0x84	CH_Mode（Byte）+DATA0（Word）+…+DATAn	写曲线缓冲区数据。 CH_Mode 定义了后续数据的通道排列顺序： ➢ CH_Mode 的每个位（bit）对应 1 个通道； CH_Mode.0 对应 0 通道，.7 对应 7 通道； 对应位置 1 表示对应的通道数据存在； 对应位置 0 表示对应的通道数据不存在。 ➢ 数据按照低通道数据在前排列。 比如 CH_Mode=0x83（10000011B），表示后续数据格式为：（通道 0＋通道 1＋通道 7）＋…＋（通道 0＋通道 1＋通道 7）。
	DGUS 屏有一个 8K Word，可以存储 8 条曲线趋势图的曲线缓冲区，用于用户简单、快速显示曲线。 曲线缓冲区的数据都是 16 位无符号数。		

3.3 串口 CRC 校验 C 程序参考

DGUS 屏的 CRC 校验采用 ANSI CRC-16(X16+X15+X2+1）格式，相应 C 程序参考如下：

```
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] 是发送数据
                    crch=crcl^CRCTABH[index];
                    crcl=CRCTABL[index];}
```

4 DGUS 寄存器

4.1 DGUS 寄存器一览表

寄存器地址	定 义	R/W	字节长度	说 明
0x00	Version	R	1	DGUS 版本号，BCD 码表示，0x10 表示 V1.0
0x01	LED_SET	W	1	LED 亮度控制寄存器，0x00-0x40
0x02	BZ_TIME	W	1	蜂鸣器鸣叫控制寄存器，单位为 10mS
0x03	PIC_ID	R/W	2	读：当前显示页面 ID　　写：切换到指定页面
0x05	TP_Flag	R/W	1	0x5A＝触摸屏坐标有更新； 其它＝触摸屏坐标未更新。 用户读取数据后未清零本标记，则触摸屏数据不再更新。
0x06	TP_Status	R	1	0x01=第一次按下　0x03=一直按压中　0x02＝抬起　其它＝无效
0x07	TP_Position	R	4	触摸屏按压坐标位置：X_H:L Y_H:L
0x0B	TPC_Enable	R/W	1	0x00＝触控不启用　其它＝触控启用（上电默认为 0xFF）。
0x0C-0x0F	RUN_TIME	R	4	上电后运行时间，BCD 码时分秒，其中小时为两个字节，最大 9999:59:59。
0x10-0x1C	R0-RC	R	13	SD 卡配置寄存器的映射，串口只读，串口写无效。
0x1D	保留		1	未定义
0x1E	LED_STA	R	1	背光当前亮度返回值
0x1F	RTC_COM_ADJ	W	1	0x5A 表示用户串口改写了 RTC 数据，DGUS 修改 RTC 后清零。
0x20	RTC_NOW	R/W	16	YY:MM:DD:WW:HH:MM:SS+农历 YY:MM:DD+天干地支＋生肖
0x30-0x3F	保留		16	未定义
0x40	En_Lib_OP	R/W	1	0x5A 表示用户申请进行读字库存储器操作，DGUS 操作完后清零。 每个 DGUS 周期执行一次读操作。
0x41	Lib_OP_Mode	W	1	0xA0：把指定字库空间的数据读入变量存储器空间。
0x42	Lib_ID	W	1	指定的字库空间,0x40-0x7F,每个字库 128KW,对应最大 Flash 空间为 8MW(16MB)。
0x43	Lib_Address	W	3	指定字库空间的数据操作首（字）地址，0x00:00:00-0x01:FF:FF
0x46	VP	W	2	指定变量存储器空间的数据操作首（字）地址，0x00:00-0x6F:FF
0x48	OP_Length	W	2	数据操作的（字）长度，0x00:01-0x6F:FF。
0x4A	Timer0	R/W	2	16bit 软件定时器，单位为 4mS，自减到零停止。
0x4C	Timer1	R/W	1	8bit 软件定时器，单位为 4mS，自减到零停止。
0x4D	Timer2	R/W	1	8bit 软件定时器，单位为 4mS，自减到零停止。
0x4E	Timer3	R/W	1	8bit 软件定时器，单位为 4mS，自减到零停止。
				设置值和实际运行值之间有 +/-4mS 误差，比如设置为 2，实际运行值在 4-12mS 之间。
0x4F	Key_code	W	1	用户键码，用于触发 0x13 触控文件；0x01-0xFF，0x00 表示无效。 DGUS 处理键码后会自动清零键码寄存器。
0x50	Play_Music_Set	W	3	0x5A:Play_Strat:Play_Num，音乐播放设定值。 Play_Start 为播放起始段，Play_Num 为连续播放段数（0x00 将停止播放）。
0x53	Volume_Adj	W	2	写入 0x5A:VOL 将调整播放音乐的音量，音量=VOL/64，上电默认值是 0x40。
0x55	保留		1	未定义
0x56	En_DBL_OP	R/W	1	0x5A 表示用户申请进行数据库存储器操作，DGUS 操作完后清零。 每个 DGUS 周期执行一次数据库读或写操作。
0x57	OP_Mode	W	1	0x50：把变量存储器空间数据写入数据库空间。 0xA0：把数据库空间的数据读入变量存储器空间。
0x58	DBL_Address	W	4	数据库空间字地址，0x00:00:00:00-1D:FF:FF:FF,最大 480MW（960MB，取决于内核 Flash 情况）数据库空间。数据库从物理存储空间的第 64MB 开始存储，和图片存储器空间有重合，每 1Byte 数据库存储器占据 2Byte 物理存储器。使用 SD 卡导出数据库时，每个字库大小为 64KW（128KB），编号从 256 开始，960MB 数据库对应字库 ID 范围为 256-7935，每个字库写寿命为 10 万次。 读写时，DGUS 会自动处理跨字库的情况。
0x5C	VP	W	2	指定变量存储器空间的数据库操作首（字）地址，0x00:00-0x6F:FF
0x5E	OP_Length	W	2	数据库操作的（字）长度，0x00:01-0x6F:FF。
0x60-0xE8	保留		137	未定义
0xE9	Scan_Status	R	1	0x01=触摸屏录入状态　0x00=触摸屏未处于录入状态
0xEA	TPCal_Triger	W	1	写入 0x5A 启动一次触摸屏校准，校准完成后会被 DGUS 清零。
0xEB	Trendline_Clear	W	1	写入特殊定义的数值以清除对应的曲线缓冲区数据。 0x55:清除全部 8 条曲线缓冲区数据； 0x56-0x5D：分别清除 CH0-CH7 通道的曲线缓冲区数据。 曲线缓冲区数据清除后，本寄存器会被 DGUS 清零。
0xEC-0xED	保留		2	保留
0xEE-0xEF	Reset_Triger	W	2	写入 0x5AA5 导致 DGUS 屏软件复位一次。
0xF0-0xFF	保留		16	保留

4.2 读写 RTC

0x1F	RTC_COM_ADJ	W	1	0x5A 表示用户串口改写了 RTC 数据，DGUS 修改 RTC 后清零。
0x20	RTC_NOW	R/W	16	YY:MM:DD:WW:HH:MM:SS+农历 YY:MM:DD+天干地支＋生肖

➢ **串口读取 RTC**

0x20 寄存器开始保存了当前 RTC 值，使用 0x81 指令读取。

读取日历（YY:MM:DD:WW:HH:MM:SS）：5A A5 03 81 20 07

读取时间（HH:MM:SS）：5A A5 03 81 24 03

➢ **串口修改（写）RTC**

用 0x80 指令改写 0x1F 寄存器为 0x5A，并给 0x20 开始的寄存器写入需要修订的时间，即改写了 RTC。

举例：

把 RTC 设置为 2013-11-08 18:56:00，串口下发

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

注意，改写 RTC 时，只需要改写公历的 年、月、日、时、分、秒 即可，星期和农历 DGUS 会自动换算。上面例子中，改写星期位置数据就随便写的是 **00**。

4.3 字库读取

0x40	En_Lib_OP	R/W	1	0x5A 表示用户申请进行读字库存储器操作，DGUS 操作完后清零。 每个 DGUS 周期执行一次读字库操作。
0x41	Lib_OP_Mode	W	1	0xA0：把指定字库空间的数据读入变量存储器空间。
0x42	Lib_ID	W	1	指定字库空间,0x40-0x7F,每个字库 128KW,对应最大 Flash 空间为 8MW(16MB)。
0x43	Lib_Address	W	3	指定字库空间的数据操作首（字）地址，0x00:00:00-0x01:FF:FF
0x46	VP	W	2	指定变量存储器空间的数据操作首（字）地址，0x00:00-0x6F:FF
0x48	OP_Length	W	2	数据操作的（字）长度，0x00:01-0x6F:FF。

DGUS 的第 64-127 字库（64 个字库，16MB），可以通过串口指令操作，把字库数据读取到变量存储器中（如果用户系统需要使用，可以使用 0x82 指令再从变量存储器中读取）。

举例：

从第 80 号字库的 0x 00 00 00 地址开始读取 4KW （0x10 00）数据到变量存储器 0x10 00 开始的位置

串口下发指令：5A A5 0C 80 40 5A A0 50 00 00 00 10 10 10 00

注意，读取数据不能超过字库空间，即 Lib_Address+OP_Length<= 0x02 00 00 。

4.4 128 段音乐播放

0x50	Play_Music _Set	W	3	0x5A:Play_Strat:Play_Num，音乐播放设定值。 Play_Start 为播放起始段，Play_Num 为连续播放段数（0x00 将停止播放）。
0x53	Volume_Adj	W	2	写入 0x5A:VOL 将调整播放音乐的音量，音量=VOL/64，上电默认值是 0x40。

某些 DGUS 屏自带 128 段（每段 1.024 秒）音乐播放功能，通过 SD 卡把音乐（32K 采样 16bit 单声道 WAV 文件）下载到屏里面后，可以用 0x80 指令写相关寄存器控制音乐播放和进行音量调节。

举例，一段提示音（比如“ 欢迎光临北京迪文 ”）长度为 3.5 秒，保存在第 6 段语音开始位置，占据的语音段是第 6-9，一共 4 段语音，要以 100%音量播放这段提示音，串口下发：

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

要停止当前语音播放，只需要把播放指令中的播放段数设置为 0x00 即可，比如

5A A5 05 80 50 5A 06 00

要把音量提升到 150%（64*1.5=96 0x60），串口下发指令：

5A A5 04 80 53 5A 60

语音播放过程中，未保存语音的空白段将直接被“ 略过 ”。

4.5 数据库读写

0x56	En_DBL_OP	R/W	1	0x5A 表示用户申请进行数据库存储器操作，DGUS 操作完后清零。 每个 DGUS 周期执行一次数据库读或写操作。
0x57	OP_Mode	W	1	0x50：把变量存储器空间数据写入数据库空间。 0xA0：把数据库空间的数据读入变量存储器空间。
0x58	DBL_Address	W	4	数据库空间字地址，0x00:00:00:00-1D:FF:FF:FF,最大 480MW（960MB，取决于内核 Flash 情况）数据库空间。数据库从物理存储空间的第 64MB 开始存储，和图片存储器空间有重合，每 1Byte 数据库存储器占据 2Byte 物理存储器。使用 SD 卡导出数据库时，每个字库大小为 64KW（128KB），编号从 256 开始，960MB 数据库对应字库 ID 范围为 256-7935，每个字库写寿命为 10 万次。 读写时，DGUS 会自动处理跨字库的情况。
0x5C	VP	W	2	指定变量存储器空间的数据库操作首（字）地址，0x00:00-0x6F:FF
0x5E	OP_Length	W	2	数据操作的（字）长度，0x00:01-0x6F:FF。

用户数据库是在 DGUS 屏的图片存储器中拿出一块空间进行数据存储，空间大小和位置由用户控制，不同内核其 Flash 大小不同，导致图片存储器空间和可用数据库空间也不相同，如下表所示：

类 别	图片空间最大容量	数据库空间最大容量	不同分辨率 DGUS 屏最大存储图片数量						
			320*240	480*272	640*480	800*480	800*600	1024*600	1024*768
标 准	210MB	89MB	836	836	278	278	209	167	139
扩展到 1GB	932MB	450MB	3728	3728	1242	1242	932	745	621
扩展到 2GB	1896MB	960MB	7584	7584	2528	2528	1896	1516	1264

用户数据库读写过程中，DGUS 对数据进行了加密和纠错操作，确保数据存储的可靠性。

用户数据库在物理上是由若干个大小为 64KW（128KB）的数据库页构成，每个页面写寿命是 10 万次（启动 1 次写操作减少 1 次写寿命），但读写操作中地址是连续的，不受分页影响，DGUS 会自动处理分页问题。

➢ 数据库首地址（0x00 00 00 00，对应第 64MB 物理存储器）对应的图片 ID 和存储系数 K1

分辨率	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"表示如果从 0x00 地址开始使用数据库，则第 128 幅图片位置开始不能保存图片；“ 42-43 ”表示 42、43 都不能使用。							

➢ 图片空间到数据库空间的计算

假设有 N（N 大于上表中的 PIC_ID）幅图片需要存储，那么数据库可以使用的最小首地址为：

数据库最小首地址=（(N*K1) -128）*64*1024 已经取整到 64KW(128KB)。

举例，480*272 分辨率下，需要预留出 200 幅图片，那么数据库的最小起始地址 Adr_Min：

Adr_Min= ((200*1) -128) *64*1024=0x00 48 00 00

➢ 数据库空间数据 SD 卡导出

数据库内容可以使用 SD 卡导出，请参考“ **2.2.6 用户数据库 SD 卡导出** ”。

4.6 按键触发

0x4F	Key_code	W	1	用户键码，用于触发 0x13 触控文件；0x01-0xFF，0x00 表示无效。 DGUS 处理键码后会自动清零键码寄存器。

DGUS 屏没有键盘接口，但很多应用需要使用键盘或者按键操作。DGUS 的 0x4F 寄存器提供了用户使用键盘来控制 DGUS 屏 GUI 触控进程（13 触控文件）的接口，使得用户只需把键码写入 0x4F 寄存器，就可以由 DGUS 按照 13 文件的描述来处理相关 GUI。

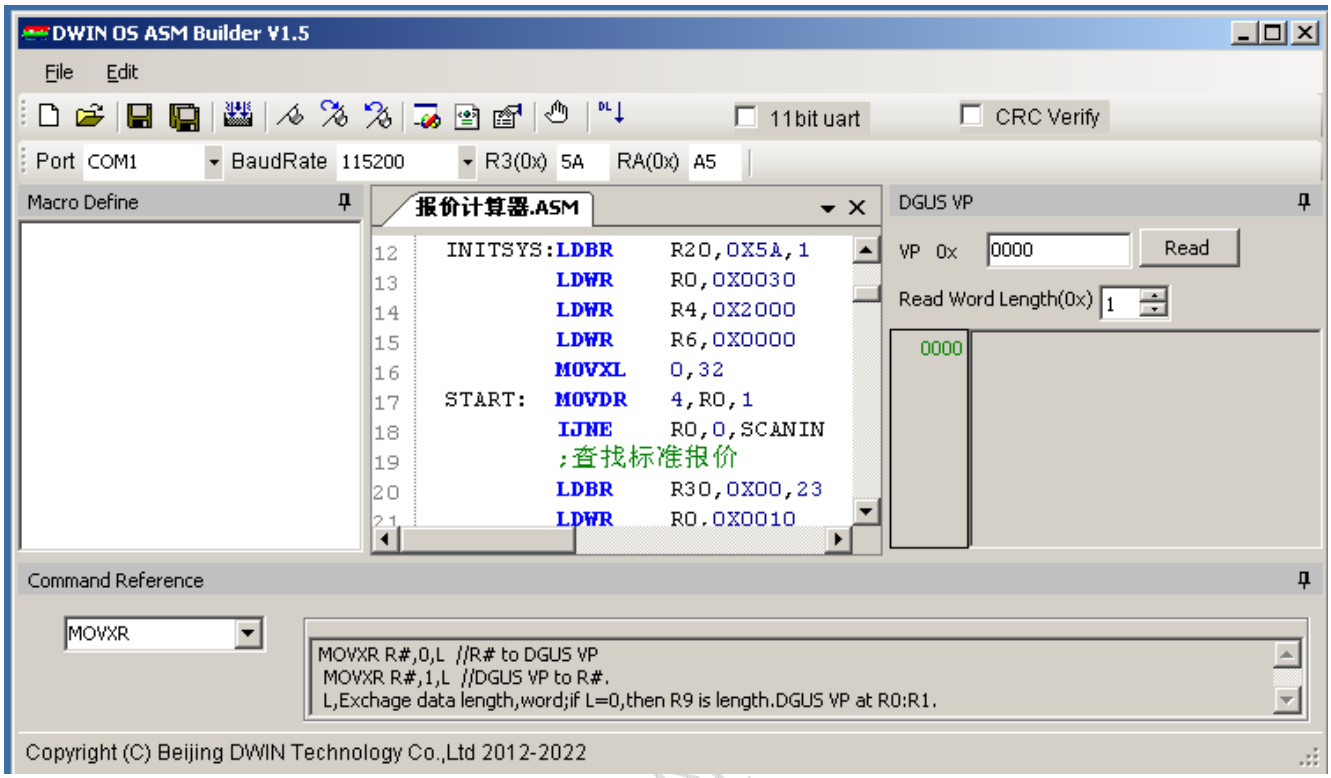
比如，在 13 触控文件中，在 10 号页面定义 0xF1 键码将进入参数录入界面，那么，当用户在 10 号页面下发送指令：5A A5 03 80 4F F1 就启动了一次按键触发，DGUS 会自动进入参数录入界面。

按键触发和触摸屏触发是并行触发的，可以混在一起使用（即触摸屏界面下也可以同时使用按键）。

5 DWIN OS 用户程序设计

DWIN OS 平台采用类似汇编程序编写规范，在 DGUS 稳定的 GUI 平台下，方便用户针对自己的特殊需求快速、可靠的进行二次开发。

DWIN OS 的 PC 软件编译界面如下图所示。



基于 DGUS 的 DWIN OS 平台，用户最大代码空间是 256KB （32764 行代码）。DWIN OS 程序在每个 DGUS 周期（80/120/160/200mS）都完整运行一遍，所以 DWIN OS 程序中不能出现死循环。

DGUS 平台上 DWIN OS 的常见应用是使用 DWIN OS 来解析用户自定义数据协议和进行数据处理，代替标准 HMI 或工控机，不仅降低成本（DGUS 屏价格只有标准 HMI 或工控机的几分之一），也大大提升了系统可靠性（标准 HMI 多是基于 PC 或者工控机架构，软件在 Windows CE 之类的通用操作系统平台下开发）。

5.1 基本约定

DWIN OS 寄存器变量： R0-R255，256 Byte；
DGUS 寄存器： 对应 DGUS 0x80/0x81 指令访问的寄存器变量空间（0x00-0xFF）；
DGUS 变量： 对应 DGUS 0x82/0x83 指令访问的变量存储器空间（0x0000-0x6FFF）
字库空间： 对应 32-127（0x20-0x7F）号汉字库，24MB。

伪汇编指令

- **EQU** 替换，编译时直接替换

比如，

```
PICID  EQU  3
WORD   EQU  2
MOVDR  PICID,R10,WORD      ;等效成 MOVDR 3,R10,2
```

- **DB** 定义 1 个字节或字（定义数据小于 255 将自动定义为字节）的 ROM 数据

```
LDADR TAB1    ;把 TAB1 的 24bit 地址保存到 R5:R6:R7 地址指针寄存器

TAB1: DB 1,2,3,4
      DB 1000,2000,3000,4000,-100
      DB "北京迪文 DGUS"
```

- 注释用的无效标记，使用**;** 。

5.2 DWIN OS 汇编指令集

- R#表示 DWIN OS 的 256 个寄存器之任意一个或一组，R0-R255；
- < >表示立即数，汇编代码中，100，0x64，64H，064H 都是表示 10 进制数据 100。
- COM2 串口仅 DGUS+内核硬件平台支持。

指令功能	操作码	操作数	说　　明
空操作	NOP		不执行任何操作。 NOP
DGUS 变量和寄存器数据交换	MOVXR	R#,<MOD>,<NUM>	R#：寄存器或寄存器组。 <MOD>：0=寄存器到变量　1=变量到寄存器。 <NUM>：交换的数据字（Word）长度，0x00-0x80； 当<NUM>为 0x00 时，数据长度由 R9 决定。 DGUS 变量指针由 R0:R1 寄存器定义。 MOVXR R20,0,2
装载 N 个 8bit 立即数到寄存器组	LDBR	R#,<DATA>,<NUM>	R#：寄存器或寄存器组。 <DATA>：要装载的数据。 <NUM>：要装载的寄存器个数，0x00 表示 256 个。 LDBR　R8,0x82,3
装载 1 个 16bit 立即数到寄存器组	LDWR	R#,<DATA>	R#：寄存器组。 <DATA>：要装载的数据。 LDWR　R8,1000 LDWR　R8,-300
程序空间查表（程序空间数据到寄存器）	MOVC	R#,<NUM>	R#：寄存器或寄存器组。 <NUM>：查表返回的字节数据长度。 表地址指针由 R5:R6:R7 寄存器定义。 MOVC R20,10
寄存器和寄存器数据交换	MOV	R#S,R#T,<NUM>	R#S：源寄存器或寄存器组。 R#T：目标寄存器或寄存器组。 <NUM>：交换的字节数据长度，0x00 表示长度由 R9 寄存器定义。 MOV R8,R20,3
寄存器到 DGUS 寄存器	MOVRD	R#,D#,<NUM>	R#：寄存器或寄存器组； D#：DGUS 的寄存器或寄存器组； <NUM>：交换的字节数据长度。 MOVRD R10,3,2
DGUS 寄存器到寄存器	MOVDR	D#,R#,<NUM>	R#：寄存器或寄存器组； D#：DGUS 的寄存器或寄存器组； <NUM>：交换的字节数据长度。 MOVDR 3,R10,2
DGUS 变量和字库、数据库 数据交换	MOVXL	<MOD>,<NUM>	<MOD>：0=字库数据读到 DGUS 变量； 1=DGUS 变量数据写到字库； 2=用户数据库数据读到 DGUS 变量； 3=DGUS 变量数据写到用户数据库。 <NUM>：交换数据(字，Word)长度；NUM=0，则由 R9:R10 定义。 DGUS 变量地址由 R0:R1 寄存器定义。 **字库读写模式（MOD=0、1）：** 字库由 R4(0x20-0x7F)寄存器指定，R5:R6:R7 为字库内的数据操作首地址，越界取消。 **用户数据库读写模式（MOD=2、3）：** 用户数据库首地址由 R4:R5:R6:R7 寄存器指定。 MOVXL 0,300
DGUS 变量之间交换数据	MOVXX	<NUM>	<NUM>：交换（字，Word）数据长度。 <NUM>为 0 表示长度由 R8:R9 寄存器定义。 DGUS 源变量地址由 R0:R1 寄存器定义。 DGUS 目标变量地址由 R2:R3 寄存器定义。 MOVX 100
寄存器变址寻址	MOVA		R2 规定了源寄存器（组）地址； R3 规定了目标寄存器（组）地址； R9 规定了交换的数据长度，字节数。 MOVA
32bit 整形数加法	ADD	R#A,R#B,R#C	C=A+B,A、B 为 32bit 整数；C 为 64bit 整数。 ADD R10,R20,R30
32bit 整形数减法	SUB	R#A,R#B,R#C	C=A-B,A、B 为 32bit 整数；C 为 64bit 整数。

			SUB R10,R20,R30
64bit 长整数 MAC	MAC	R#A,R#B,R#C	C=(A*B+C)，A、B 是 32bit 整数，C 是 64Bit 整数。 MAC R10,R20,R30
64bit 整形数除法	DIV	R#A,R#B,<MOD>	A/B，商是 A，余数是 B。 A 和 B 都是 64bit 寄存器。 <MOD>：0=商不进行四舍五入，1=商进行四舍五入。 DIV R10,R20,1
变量扩展成 32bit	EXP	R#S,R#T,<MOD>	把 R#S 指向的数据转成 32bit 整数保存到 R#T。 R#S：源寄存器或寄存器数。 R#T：32bit 目标寄存器。 <MOD>：R#S 数据类型，0=8bit 无符号 1=8bit 带符号 2=16bit 无符号数 3=16bit 整数。 EXP R10,R20,2
32bit 无符号 MAC	SMAC	R#A,R#B,R#C	C=A*B+C。 A、 B 是 16bit 无符号数，C 是 32bit 无符号数。 SMAC R10,R20,R30
寄存器自增量	INC	R#，<MOD>,<NUM>	R#=R#＋NUM，无符号数自增计算，<NUM>为 0x00-0xFF。 <MOD>：R#数据类型，0=8bit 1=16bit。 INC R10,1,5
寄存器自减量	DEC	R#,<MOD>,<NUM>	R#=R#-NUM，无符号数自减计算，<NUM>为 0x00-0xFF。 <MOD>：R#数据类型，0=8bit 1=16bit。 DEC R10,0,1
加载地址	LDADR	<Address>	把<Address>加载到 R5:R6:R7 LDADR TAB LDADR 0x123456
与逻辑运算	AND	R#A,R#B,<NUM>	A=A AND B，序列与逻辑运算。 <NUM>：R#A、R#B 变量字节数目。 AND R10,R20,1
或逻辑运算	OR	R#A,R#B,<NUM>	A=A OR B，序列或逻辑运算。 <NUM>：R#A、R#B 变量字节数目。 OR R10,R20,1
异或逻辑运算	XOR	R#A,R#B,<NUM>	A=A XOR B，序列异或逻辑运算。 <NUM>：R#A、R#B 变量字节数目。 XOR R10,R20,1
解整数线性方程	ROOTLE		由 16bit 整数（X0,Y0）、（X1,Y1）两点确定的直线上的 X，求解对应的 Y 值。 输入：X=R10 X0=R14 Y0=R16 X1=R18 Y1=R1A。 输出：Y=R12。 ROOTLE
ANSI CRC-16 计算	CRCA	R#S,R#T,R#N	对序列数据计算 ANSI CRC-16(X16+X15+X2+1) 。 R#S：输入的寄存器组； R#T：输出 CRC 结果，16bit，LSB 模式保存。 R#N：保存计算 CRC 数据字节长度寄存器，8bit。 CRCA R10,R80,R9
CCITT CRC-16 计算	CRCC	R#S,R#T,R#N	对序列数据计算 CCITT CRC-16(X16+X12+X5+1) 。 R#S：输入的寄存器组； R#T：输出 CRC 结果，16bit，MSB 模式保存。 R#N：保存计算 CRC 数据字节长度寄存器，8bit。 CRCC R10,R80,R9
到 COM1_Rx_FIFO 读取 MODBUS 数据帧	RMODBUS	R#A,R#T,R#C	检查 COM1 接收 FIFO 是否有满足要求的 MODBUS 数据帧，如果有则读取数据到寄存器并清空接收 FIFO。 R#A：保存 MODBUS 接收数据包前 3 个字节（地址:指令:长度）的寄存器组。 如果长度为 0x00，那么表示不进行长度匹配，紧随其后的数据（第 4 个字节）表示了地址、指令、校验和之外的数据长度。 R#C：返回状态的寄存器，寄存器保存了返回数据；0x00 表示未接收到匹配的 MODBUS 数据帧；0xFF 表示接收到匹配的 MODBUS 数据帧并把数据读取到 R#T 寄存器组。 R#T：匹配后，保存 MODBUS 数据的寄存器组。 RMODBUS R10,R20,R13
位分解	BITS	R#,<VP>	把 R#的 8 个比特分解到 VP 指向的 8 个 DGUS 字变量，MSB 方式，bit 1 分解为 0x0001，bit 0 分解为 0x0000。

			R#：需要进行位分解的寄存器，8bit。 <VP>：DGUS 变量地址。 BITS R10,0x2000
位组合	BITI	R#,<VP>	把 VP 指向的 8 个 DGUS 字变量组合成 1 个字节位变量,MSB 方式，0x0000 为 bit 0，非 0x0000 数据为 bit 1。 R#：存储位组合数据的寄存器，8bit。 <VP>：DGUS 变量地址。 BITI R10,0x2000
HEX 转 ASC	HEXASC	R#S,R#T,<MOD>	R#S：需要转换的 32bit 整数； R#T：转换后的 ASCII 字符串寄存器组； <MOD>：转换模式控制，高 4bit 为整数位长度，低 4bit 为小数位数目。 转换的 ASCII 串带符号，右对齐，空位用 0x20 填充。 对于数据 0x12345678， <MOD>=0x62 转换结果为+054198.96 <MOD>=0xF2 转换结果为 +3054198.96 HEXASC R20,R30,0x62
序列比较	TESTS	R#A,R#B,<NUM>	依次比较 A、B 两个寄存器序列的值： 值不同时，返回 A 序列此时的地址到 R0 寄存器； 如果 A、B 相同则返回 0x00 到 R0 寄存器。 R#A：A 序列寄存器。 R#B：B 序列寄存器。 <NUM>：最大比较数据字节长度。 TESTS R10,R20,16
COM 配置	COMSET	<MODE>,<BS>	设置串口模式： <MODE>: 高 4bit 选择要配置的串口，0=COM1 1=COM2 低 4bit 选择模式 0x*0=N81 模式； 0x*1=E81 模式； 0x*2=O81 模式； 0x*3=N82 模式。 <BS>:波特率设置值 对于 COM1， 设置值=6250000/设置的波特率。 对于 COM2， 设置值高字节选择波特率因子 00=5.208Mbps 01=15.625Mbps 02=1.302Mbps 设置值低字节设置波特率 设置值低字节=波特率因子/要设置的波特率。 每次设置会自动清空串口接收 FIFO 一次。 COMSET 0,54
位测试、跳转	JB	R#,<Bit>,<TAB>	测试 R#寄存器的第<Bit>位，1 跳转，0 继续执行下一条代码，跳转范围+/-127 条指令。 R#：位测试的寄存器，16bit。 <Bit>：位测试位置，0x00-0x0F，MSB 方式。 <TAB>：跳转位置。 JB R10,15,TEST1 NOP TEST1:ADD R8,R12,R16
变量比较、不相等跳转	CJNE	R#A,R#B,<TAB>	比较 A、B 两个 8bit 寄存器的内容，相等则执行下一条指令，不等则跳转，跳转范围+/-127 条指令。 TEST1:NOP INC R10,0,1 CJNE R10,R11,TEST1
整形数比较，小于跳转	JS	R#A,R#B,<TAB>	比较 A、B 两个 16bit 整数的大小，A>=B 则执行下一条指令，A<B 则跳转，跳转范围+/-127 条指令。 JS R10,R12,TEST1 NOP TEST1:NOP
变量和立即数比较、不相等跳转	IJNE	R#,<INST>,<TAB>	比较 8bit 寄存器和立即数<INST>的内容，相等则执行下一条指令，不等则跳转，跳转范围+/-127 条指令。 IJNE R10,100,TEST1

			NOP TEST1:NOP
强制结束当前输入法	EXIT	R#A,R#B	强制结束当前的输入法。 R#A：控制是否切换页面，0x00=不切换，0x01=切换； R#B：要切换回的页面 ID（16bit）。 EXIT R10,R11
子程序调用返回	RET		CALL 调用指令返回。 RET
子程序调用	CALL	<PC>	调用子程序，最多支持 32 级程序嵌套。 CALL TEST
直接跳转	GOTO	<PC>	程序跳转 GOTO TEST1 NOP TEST1:NOP
串口发送	COMTXD	<COM>,R#S,R#N	把数据发送到指定的串口。 <COM>：选择串口，0=COM1（DGUS 用户串口） 1=COM2（DGUS 保留串口）。 R#S：要发送的数据寄存器组。 R#N：要发送的字节数寄存器，8bit，寄存器数据 0x00 表示发送 256 字节数据。 COMTXD 0,R10,R9
串口打印	CPRTS	<COM>,<VP>	检查 VP 指向的 DGUS 变量地址有没有打印信息，有则打印到串口。VP 为 DGUS 的 0xFE07_05 打印指令对应的变量 VP 值,打印后清除 VP 地址的打印标记。 <COM>：选择串口，0=COM1（DGUS 用户串口） 1=COM2（DGUS 保留串口）。 CPRTS 0,0x2000
检查 COM_Rx_FIFO	RDXLEN	<COM>,R#	返回 COM 接收缓冲区（FIFO）接收数据字节长度（0-253）到 R#寄存器，0x00 表示没有数据。 <COM>：选择串口，0=COM1 1=COM2。 RDXLEN 0,R10
读取 COM_Rx_FIFO	RDXDAT	<COM>,R#A,R#B	从 COM 接收缓冲区（FIFO）中读取 R#B 个字节（01-253）到 R#A 寄存器组；读取后 FIFO 长度自动调整。 <COM>：选择串口，0=COM1 1=COM2。 RDXDAT 0,R11,R10
直接串口发送	COMTXI	<COM>,R#,<NUM>	把 R#指向的<NUM>个寄存器内容发送到 COM。 <COM>：选择串口，0=COM1 1=COM2。 COMTXI 0,R20,16
读取当前输入法内容到寄存器	SCAN	R#,<NUM>	把当前输入法下已经录入的最多<NUM>个字符加载到 R#+1 开始的寄存器，R#保存数据长度，字符个数从当前输入法光标位置往前计算。 SCAN R20,6
写指定通道动态曲线缓冲区	WRLINE	R#S,R#I,<CH>	把 R#S 指向的 N 个 16bit 无符号整数，加上 16bit 无符号整数 V_BIAS 后写到<CH>（0x00-0x07）指定的动态曲线缓冲区，R#I 寄存器指向一个 3 字节变量：N，V_BIAS。 WRLINE R80,R10,2
汉字库匹配搜索	LIBSCH	R#A,R#B,R#C	到指定汉字库搜索匹配字符串的数据。 ➢ R#A 的 2 个寄存器规定了匹配字符串格式： R#A：字符串长度（0x00-0x1F），0x00 表示字符串由 0x00 或 0xFF 结尾。 R#A+1：字符串数据，最多 31 个字符。 ➢ R#B 的 11 个寄存器规定了数据库（DATA[M][N] 字数组）格式和搜索格式： R#B：字库 ID（0x20-0x7F，0x00 表示不用重新加载），每个 DGUS 周期调用一次后，只要没有对其它汉字库做过操作就不用重新加载。 R#B+1：二维数组的行维度 M，0x0001-0x0FFFF； R#B+3：二维数组的列维度 N，0x01-0x80，字数目； R#B+4：搜索匹配模式，0x00=左对齐，0x01=任意位置匹配； R#B+5：搜索开始行，0x0000-0xFFFF； R#B+7：每行中的搜索开始列，0x00-0xFF，一列中的字节

			位置； R#B+8：每行中的搜索终止列，0x00-0xFF，仅当搜索匹配模式是 0x01（任意位置匹配）时有效； R#B+9：搜索匹配后，返回数据在搜索列的起始位置，0x00-0xFF； R#B+A：搜索匹配后，返回的数据字节长度，0x01-0xFF。 ➢ R#C 的 4 个寄存器规定了返回变量： R#C：搜索标记，0x00=未匹配，0xFF=匹配； R#C+1：匹配时的(行维度＋1)值； R#C+3：匹配时，回传数据保存的寄存器首地址。 LIBSCH R10,R12,R23
擦除指定的字库	ERASE	<L_ID>	<L_ID>:要擦除的汉字库 ID，0x20-0x7F； 如果<L_ID>为 0x00，表示字库位置由 R9 寄存器指定。 ERASE 40
字节累加校验和计算	SUMADD	R#S,R#T,R#N	对序列数据计算字节累加和校验 R#S：输入的寄存器组； R#T：输出 1 字节累加和结果，8bit。 R#N：序列长度寄存器，8bit。 SUMADD R10,R80,R9
带进位字节累加校验和计算	SUMADDC	R#S,R#T,R#N	对序列数据计算带进位的字节累加和校验 R#S：输入的寄存器组； R#T：输出 1 字节累加和结果，8bit。 R#N：序列长度寄存器，8bit。 SUMADDC R10,R80,R9
异或校验和计算	SUMXOR	R#S,R#T,R#N	对序列数据计算字节异或校验 R#S：输入的寄存器组； R#T：输出 1 字节异或结果，8bit。 R#N：序列长度寄存器，8bit。 SUMXOR R10,R80,R9
HEX 转换成压缩 BCD 码	HEXBCD	R#S,R#T,<MOD>	把 HEX 数据转换成压缩的 BCD 码，比如数据 1000 将转换成 0x10,0x00。 R#S：输入 HEX 数据的寄存器组首地址 R#T：输出压缩 BCD 码数据的寄存器首地址 <MOD>：高 4bit 表示输入 HEX 字节数，0x01-0x08 低 4bit 表示输出 BCD 码字节数，0x01-0x0A HEXBCD R10,R80,0x23
压缩 BCD 码转换成 HEX	BCDHEX	R#S,R#T,<MOD>	把压缩的 BCD 码转换成 HEX 数据，比如数据 0x1000 将转换成 0x3E8（1000）。 R#S：输入压缩 BCD 码的寄存器组首地址 R#T：输出 HEX 数据的寄存器首地址 <MOD>：高 4bit 表示输入 BCD 码字节数，0x01-0x0A 低 4bit 表示输出 HEX 字节数，0x01-0x08 BCDHEX R10,R80,0x32
ASCII 字符串转 HEX 字符	ASCHEX	R#S,R#T,<LEN>	把 ASCII 字符串转换成 64bit 带符号 HEX 数据。 R#S：输入 ASCII 字符串寄存器首地址； R#T：输出 HEX 数据，64bit 寄存器； <LEN>：ASCII 字符串数据长度，包括符号位和小数点，0x01-0x15。 ASCHEX R10,R80,0x05
到 COM1_Rx_FIFO 读取 DL/T645 数据帧	RD645	R#A,R#T,R#C	检查 COM1 接收 FIFO 是否有满足要求的 DL/T645 的数据帧，如果有则读取数据到寄存器并清空接收 FIFO。 R#A：保存 6 字节地址（LSB 排列，压缩 BCD 码）寄存器组。 R#C：返回状态的寄存器，寄存器保存了返回数据；0x00 表示未接收到匹配的 DL/T645 数据帧 ，0xFF 表示接收到匹配的 DL/T645 数据帧并把数据读取到 R#T 寄存器组。 R#T：结果寄存器，匹配后，保存 DL/T645 数据的寄存器组,数据格式如下：控制码+数据长度+数据 RD645 R10,R20,R16
时间计算函数	TIME	R#A,R#B,<MOD>	R#A、R#B：保存 6bytes 时间变量的寄存器，时间变量为 BCD 格式； MOD=0，计算 A=A-B，计算两个时间的之间的相对值。 A 必须大于 B，当 A<B 时，不计算并返回 R#A 第一个字

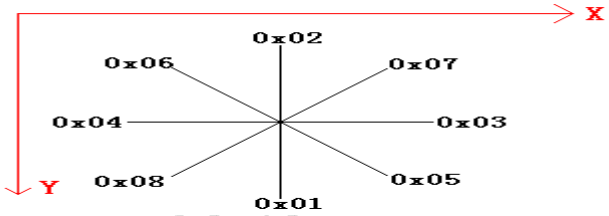
			节为 0xFF。 MOD=1，计算 A=B-RTC； MOD=2，计算 A=RTC-B。 TIME R0,R10,0
增加显示变量	ADDL14	R#A,R#B,<MOD>	R#A：保存 1 条显示变量（32Bytes）的寄存器； R#B：显示变量的添加位置，0x00-0x1F，最多添加 32 个显示变量。 <MOD>： 0x5A=添加到指定位置 其它=删除指定位置，此时 R#A 无定义。 ADDL14 R80,R81,0x5A
平方根计算	SQRT	R#A,R#B	计算一个 64 位无符号数 R#A 的平方根并保存到 R#B 中。 R#A：保存了 8 Byte 无符号数； R#B：保存了 4 Byte 无符号数结果。 SQRT R80,R90
输入法缓冲区添加	SCANADD	R#A,R#B	文本输入法状态下，把 R#A 指向、**字节长度**由 R#B 定义的字符串由当前光标位置添加到输入字符缓冲区，字符串中的 0xFF 将会被忽略。 SCANADD R80，R90 只在文本输入法状态下有效。
FEC 编码	FECEN	R#A,R#B,R#C	对 R#A 指向，**字节（Byte）长度**为 R#C 的数据串进行 FEC 编码，编码输出保存在 R#B 指针位置。 FECEN R80，R100，R10 注意，FEC 编码会把 1Byte 原始数据编码为 2Byte 编码数据。
FEC 解码	FECDE	R#A,R#B,R#C	对 R#A 指向，**字（Word）长度**为 R#C 的数据串进行 FEC 解码，解码输出保存在 R#B 指针位置。 FECDE R80，R100，R10
程序结束	END		DWIN OS 程序运行结束指令。 END

6 触控/键控配置文件（13.BIN）说明

触控配置文件由N条按照页面配置的触控指令组成，每条触控指令固定占用16、32或者48字节存储空间。一条触控指令由以下6部分组成：

序号	定义	数据长度	说　明
1	Pic_ID	2	页面ID
2	TP_Area	8	触控按钮区域：左上角坐标（Xs,Ys），右下角坐标（Xe,Ye） 当Xs=0xFFFF（启动按键语音伴音时是0x5FFF）时，表示触发控制由0x4F寄存器的键码值触发，此时Ys_H为设定的触发键码值（Ys_L,Xe,Ye值未定义，可任意写）；由键码值触发时，请把按钮按压效果设置为无效。 触摸屏手势动作识别键码固定为0x01-0x08，识别出手势后自动写入0x4F寄存器。 当Xs=0x5***时，表示按键伴音使用语音提示代替。 一个127段伴音的起始ID将分别存放在Xe、Ye的高4bit，播放段数保存在Ys的高4bit（0x00-0x0F表示0x01-0x10）。 一旦启用语音按键伴音，蜂鸣器按键伴音将关闭。
3	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
4	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
5	TP_Code	2	触控键码： 0xFF**表示无效的键码 0xFE**（或者0xFD**）表示触控功能按键，比如0xFE00表示启动变量数据触摸屏录入。 0xFE**的功能按键可以由R2.3设置成变量改变后是否自动上传，0xFD**的功能按键始终禁止变量改变后自动上传。 其它表示触控键码，用ASCII表示；比如0x0031表示按键“1”。
6	TP_FUN	32	当TP_Code=0xFE**时，用来对触控功能按键进行描述。

说明，DGUS 屏触摸屏手势识别代码和手势方向关系如下图所示：



6.1 触控/键控功能一览表

序号	触控键码	功　能	说　明
01	00	变量数据录入	录入整数、定点小数等各种数据到指定变量存储空间。
02	01	弹出菜单选择	点击触发一个弹出菜单，返回菜单项的键码。
03	02	增量调节	点击按钮，对指定变量进行+/-操作，可设置步长和上下限。 设置 0-1 范围循环调节可以实现栏目复选框功能。
04	03	拖动调节	拖拉滑块实现变量数据录入，可设置刻度范围。
05	04	RTC 设置	DGUS 屏触摸键盘设置 RTC 组件，需要完整录入公历 年月日时分秒。
06	05	按键值返回	点击按键，直接返回按键值到变量，支持位变量返回。
07	06	文本录入	文本方式录入各种字符，录入过程支持光标移动、编辑。 直接支持 ASCII 字符、GBK 中文、繁体注音输入法录入； 修改字库和 0#字库可以支持所有类似 ASCII 字符的 8bit 编码文本录入； 配合 DWIN OS 可以实现 Unicode 或多语种混合录入。
08	07_00	寄存器写到变量空间	提供了触摸屏改写寄存器空间的方法，来间接控制硬件。 比如把背光寄存器内容读取到变量，调节变量后再回写来调节背光亮度。
09	07_01	变量空间写到寄存器	
10	07_02	图像转成单色位图（纵向）	把指定区域的彩色位图转换成单色位图并保存在 VP 指定的变量区域。 主要用于当前屏幕显示内容的打印输出。
11	07_05	图像转成单色位图（横向）	
12	07_03	发送数据块到 COM1	点击触摸屏，把指定 VP 区域的数据发送到用户串口（COM1）。
13	07_04	发送数据块到 COM2	点击触摸屏，把指定 VP 区域的数据发送到扩展串口（COM2）。 COM2 用于 DGUS 屏功能扩展，并没有引出。
14	07_06	发送触摸屏坐标到 COM2	点击触摸屏，将把点击位置坐标发送到扩展串口（COM2）。 COM2 用于 DGUS 屏功能扩展，并没有引出。

6.2 变量数据录入（0x00）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE00
0x10	0xFE	1	0xFE
0x11	*VP	2	录入数据对应的变量地址指针
0x13	V_Type	1	返回变量类型： 0x00=2 字节变量，整数-32768 到 32767，无符号整数 0-65535 0x01=4 字节变量，长整数-2147483648 到 2147483647 无符号长整数 0-4294967295 0x02=*VP 高字节,无符号数 0 到 255 0x03=*VP 低字节，无符号数 0 到 255 0x04=超长整数(8 字节) -9223372036854775808 到 9223372036854775807
0x14	N_Int	1	录入的整数位数。比如录入1234.56，则N_Int=0x04。
0x15	N_Dot	1	录入的小数位数。比如录入1234.56，则N_Dot=0x02。
0x16	(x,y)	4	输入过程显示位置：右对齐方式，(x,y) 是字符串输入光标的右上角坐标。
0x1A	Color	2	输入字体显示颜色。
0x1C	Lib_ID	1	显示使用的 ASCII 字库位置，0x00＝默认字库
0x1D	Font_Hor	1	字体大小，X 方向点阵数目
0x1E	Cusor_Color	1	光标颜色，0x00=黑色 其它＝白色
0x1F	Hide_En	1	0x00=输入遮挡，显示为"*"；其它，输入直接显示
0x20	0xFE	1	0xFE
0x21	KB_Source	1	0x00=键盘在当前页面；其它＝键盘不在当前页面。
0x22	PIC_KB	2	键盘所在页面 ID，仅当 KB_Source 不等于 0x00 时有效。
0x24	AREA_KB	8	键盘区域：(Xs,Ys) 为左上角、，(Xe,Ye) 为右下角坐标。 仅当 KB_Source 不等于 0x00 时有效。
0x2C	AREA_KB_Position	4	键盘在当前页面显示位置，左上角坐标；仅当 KB_Source 不等于 0x00 时有效。
0x30	0xFE	1	0xFE
0x31	Limits_En	1	0xFF：表示启用输入范围限制，输入越界无效（等同取消）； 其它：输入无范围限制。
0x32	V_min	4	输入下限，4 字节（长整数或无符号长整数）。
0x36	V_max	4	输入上限，4 字节（长整数或无符号长整数）。
0x3A	Return_Set	1	0x5A：录入过程中，向 Return_VP 地址加载 Return_Daata，结束自动恢复。 0x00：录入过程中，不加载数据。 加载数据功能， 主要用于和变量显示的 SP 修改结合，实现对多参数录入过程自动标示，比如修改字体颜色、大小、启动一个（位）变量图标或者区域反色。 也可以作为录入过程的标记位，配合 DWIN OS 开发实现特殊需求。
0x3B	Return_VP	2	录入过程中加载数据的 VP 地址。
0x3D	Return_DATA	2	录入过程中，加载到 Return_VP 的数据。
0x3F	保留	1	写 0x00

输入过程中有效键码：

0x0030-0x0039，0x002E（.），0x002D(+/-),0x00F0（取消），0x00F1（确认），0x00F2（退格）。

键盘和输入启动按钮在一个页面（KB_Source=0x00）

键盘不在当前界面上（KB_Source=0x01）：触发输入法后

键盘不在当前界面上（KB_Source=0x01）：键盘所在页面

6.3 弹出菜单选择（0x01）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE01
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针，返回数据由VP_Mode决定。
0x13	VP_Mode	1	0x00＝把 0x00**键码写入 VP 字地址（整型数）； 0x01＝把**键码写入 VP 字地址的高字节地址（VP_H）； 0x02＝把**键码写入 VP 字地址的低字节地址（VP_L）； 0x10-0x1F：把**键码最低位(1bit)变量并写入 VP 字地址的指定位（0x10 修改 VP.0，0x1F 修改 VP.F）
0x14	Pic_Menu	2	弹出菜单的图片位置
0x16	AREA_Menu	8	菜单区域：左上角坐标（Xs,Ys）,右下角坐标（Xe,Ye）
0x1E	Menu_Position_X	2	菜单在当前页面显示的位置:左上角 X 坐标
0x20	0xFE	1	固定
0x21	Menu_Position_Y	2	菜单在当前页面显示的位置:左上角 Y 坐标
0x23	NULL	13	写 0x00

输入过程中有效键码：0x0000-0x00FF，其中 0x00FF 为取消（不选择参数直接返回）。

上图中：弹出的菜单在另外的界面上，"开启"和"关闭"两个按钮配置的键码（0x0000-0x00FE）将被返回给 0x01 触控按钮所指向的变量；取消按钮设置键码为 0x00FF，点击时不会返回值。

下拉式菜单也可以使用本指令设计。

6.4 增量调节（0x02）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys)（Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。必须为0xFF**。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE02
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针，返回数据由VP_Mode决定。
0x13	VP_Mode	1	0x00＝调节 VP 字地址（整型数）； 0x01＝调节 VP 字地址的高字节地址（1 字节无符号数，VP_H）； 0x02＝调节 VP 字地址的低字节地址（1 字节无符号数，VP_L）； 0x10-0x1F：对 VP 字地址的指定位（0x10 对应 VP.0，0x1F 对应 VP.F）进行调节，调节范围必须设置为 0-1。
0x14	Adj_Mode	1	调节方式：0x00=--　其它＝++
0x15	Return_Mode	1	逾限处理方式：0x00=停止（等于门限）　其它＝循环调节
0x16	Adj_Step	2	调节步长，0x0000-0x7FFF
0x18	V_Min	2	下限：2 字节整数（VP_Mode=0x01 或 0x02 时，仅低字节有效）
0x1A	V_Max	2	上限：2 字节整数（VP_Mode=0x01 或 0x02 时，仅低字节有效）
0x1C	Key_Mode	1	0x00：按住按键时连续调节； 0x01：按住按键时只调节 1 次。
0x1D	NULL	3	写 0x00

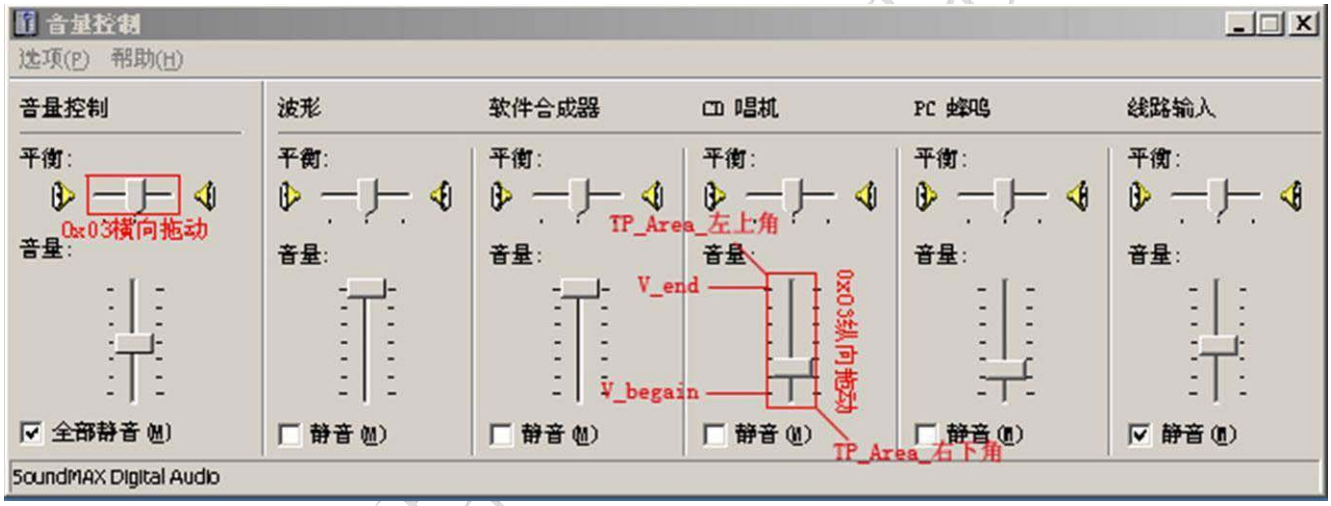
上图中，"＋"　"－" 两个触控按钮分别被配置为＋＋增量（Adj_mode=0x01）和－－增量（Adj_mode=0x00）。

当把范围设置为 0-1 时，配合图标变量显示可以方便的设计出复选功能（点击 1 下选中，再点击取消）。

6.5 拖动调节（0x03）

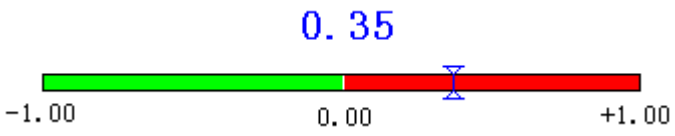
地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。必须为0xFF**。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。必须为0xFF**。
0x0E	TP_Code	2	0xFE03
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针。
0x13	Adj_Mode	1	➢ 高 4 比特定义了数据返回格式： 0x0*＝调节 VP 字地址（整型数）； 0x1*＝调节 VP 字地址的高字节地址（1 字节无符号数，VP_H）； 0x2*＝调节 VP 字地址的低字节地址（1 字节无符号数，VP_L）。 ➢ 低 4bit 定义了拖动方式： 0x*0=横向拖动，0x*1=纵向拖动。
0x14	Area_Adj	8	有效调节区域：Xs,Ys,Xe,Ye；必须和 TP_Area（触控区域）一致。
0x1C	V_begain	2	起始位置对应的返回值，整数。
0x1E	V_end	2	终止位置对应的返回值，整数。

为防止误操作，须按压有效拖动区域超过 0.5 秒后拖动才启动。



上图中，刻度滑块是用滑块刻度指示（0x02 变量）实现的。

拖动录入的优点是直观、快捷，而且参数不会越界。当需要更精确的拖动录入时，可以把拖动修改的变量同时用数据变量显示方式（0x10 变量）显示出来，如下图所示：



拖动调节不支持按键（0x4F 寄存器保存的键码）控制。

6.6 RTC 设置（0x04）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE04
0x10	0xFE	1	0xFE
0x11	0x00 00 00	3	0x00 00 00
0x14	(x,y)	4	输入过程显示位置,右对齐方式， (x,y)是字符串右上角坐标。
0x18	Color	2	输入字体显示颜色。
0x1A	Lib_ID	1	显示使用的 ASCII 字库位置，0x00＝默认字库
0x1B	Font_Hor	1	字体大小，X 方向点阵数目
0x1C	Cusor_Color	1	光标颜色，0x00=黑色 其它＝白色
0x1D	KB_Source	1	0x00=键盘在当前页面；其它＝键盘不在当前页面
0x1E	PIC_KB	2	键盘所在页面 ID，仅当 KB_Source 不等于 0x00 时有效。
0x20	0xFE	1	0xFE
0x21	AREA_KB	8	键盘区域：左上角坐标（Xs,Ys），右下角坐标（Xe,Ye）； 仅当 KB_Source 不等于 0x00 时有效。
0x29	AREA_KB_Position	4	键盘在当前页面显示位置，左上角坐标；仅当 KB_Source 不等于 0x00 时有效。
0x2D	NULL	3	写 0x00

设计方法和　0x00 触控变量_键盘不在当前界面　基本一致。

键盘不在当前界面上（KB_Source=0x01）：触发输入法后

键盘不在当前界面上（KB_Source=0x01）：键盘所在页面

6.7 按键值返回（0x05）

地址	定义	数据长度	说 明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE05
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针
0x13	VP_Mode	1	0x00＝返回键值保存在 VP 字地址（整型数）； 0x01＝返回键值低字节保存在 VP 字地址的高字节地址（VP_H）； 0x02＝返回键值低字节保存在 VP 字地址的低字节地址（VP_L）； 0x10-0x1F：把返回键值的最低位（1bit）写入 VP 字地址的指定位（0x10 修改 VP.0，0x1F 修改 VP.F）
0x14	Key_Code	2	返回键值。
0x16	NULL	10	写 0x00

6.8 文本录入（0x06）

输入文本键盘码表

在文本录入的触控文件中，两字节键码的低字节表示普通键码，高字节表示大写键码。典型的文本录入键盘定义如下表所示：

键码	普通	大写	键码	普通	大写	键码	普通	大写	键码	普通	大写
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	\	|						

注：文本键盘键码须小于 0x80（ASCII 码）。0x0D 键码录入会自动转换成 0x0D 0x0A；0x00 和 0xFF 键码禁用。

键盘功能键码定义

键码	定义	说明
0x00F0	Cancel	取消录入返回，不影响变量数据。
0x00F1	Return	确认录入返回，录入文本保存到指定变量位置。
0x00F2	Backspace	向前（退格）删除一个字符。
0x00F3	Delete	向后删除 1 个字符。
0x00F4	CapsLock	大写锁定。如果启用，对应按钮必须定义按钮按下的效果。
0x00F7	Left	光标前移一个字符；GBK 汉字录入中用于翻页。
0x00F8	Right	光标后移一个字符；GBK 汉字录入中用于翻页。

使用键盘（0x4F 寄存器保存的键码）做文本录入时，如果使用 CapsLock 键，请把按钮的动画区域定义在需要提示“ CapsLock ”的区域；这样定义后，发送 CapsLock 键时，屏幕的相应位置会自动显示“ CapsLock ”的区域图标提示。

6.8.1 ASCII 文本录入

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE06
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针
0x13	VP_Len_Max	1	文本变量最大长度，字 (Word) 数目，0x01-0x7B； 文本保存到指定地址时，自动在文本结束处加上 0xFFFF 作为结束符； 录入的文本变量实际可能占用最大变量空间＝VP_Len_Max＋1。
0x14	Scan_Mode	1	录入模式控制：0x00=重新录入　0x01=打开原来文本再修改
0x15	Lib_ID	1	显示使用的 ASCII 字库位置，0x00＝默认字库
0x16	Font_Hor	1	字体大小，X 方向点阵数目
0x17	Font_Ver	1	字体大小，Y 方向点阵数目 (Lib_ID＝0x00 时，Y 方向点阵数目必须为 2*X)
0x18	Cusor_Color	1	光标颜色，0x00=黑色 其它＝白色
0x19	Color	2	文本显示颜色。
0x1B	Scan_Area_Start	4	录入文本显示区域左上角坐标 (Xs,Ys)
0x1F	Scan_Return_Mode	1	0x55:在*(VP-1)位置保存输入结束标记和有效数据长度： *(VP-1)高字节，输入结束标记：0x5A 表示输入结束，输入过程为 0x00。 *(VP-1)低字节，有效输入数据长度，字节单位。 0x00:不返回输入结束标记和长度；
0x20	0xFE	1	
0x21	Scan_Area_End	4	录入文本显示区域右下角坐标 (Xe,Ye)
0x25	KB_Source	1	键盘页面位置选择：0x00=键盘在当前页面；其它＝键盘不在当前页面。
0x26	PIC_KB	2	以下数据，仅当 KB_Source 不为 0x00 时有效。键盘所在页面 ID
0x28	AREA_KB	8	键盘页面上键盘区域坐标：左上角 (Xs,Ys) 、右下角 (Xe,Ye)
0x30	0xFE	1	
0x31	AREA_KB_Position	4	键盘区域粘贴在当前页面显示的位置，左上角坐标。
0x35	DISPLAY_EN	1	0x00：输入过程正常显示 0x01：输入过程显示"*"，用于密码输入
0x36	NULL	10	写 0x00

注：迪文预装的 0#字库包含 4*8-64*128 点阵的所有 ASCII 字符。

6.8.2 GBK 汉字文本录入

地址	定义	数据长度	说 明
0x00	Pic_ID	2	页面ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE06
0x10	0xFE	1	0xFE
0x11	*VP	2	变量地址指针
0x13	VP_Len_Max	1	文本变量最大长度，字 (Word) 数目，0x01-0x7B； 文本保存到指定地址时，自动在文本结束处加上 0xFFFF 作为结束符； 录入的文本变量实际可能占用最大变量空间＝VP_Len_Max＋1。
0x14	Scan_Mode	1	录入模式控制：0x00=重新录入 0x01=打开原来文本再修改
0x15	Lib_GBK1	1	汉字字符显示使用的 GBK 字库 ID，ASCII 字符默认使用 0x00 字库。
0x16	Lib_GBK2	1	录入过程，汉字字符显示使用的 GBK 字库 ID
0x17	Font_Scale1	1	Lib_GBK1 字体大小，点阵数目
0x18	Font_Scale2	1	Lib_GBk2 字体大小，点阵数目
0x19	Cusor_Color	1	光标颜色，0x00=黑色 其它＝白色
0x1A	Color0	2	录入文本显示颜色。
0x1C	Color1	2	录入过程中文本显示颜色
0x1E	PY_Disp_Mode	1	录入过程中，拼音提示和对应汉字的显示方式： ➢ 0x00=拼音提示显示在上边，对应的汉字显示另起一行显示在下面； 拼音提示和汉字显示左对齐，行间距为 Scan_Dis。 ➢ 0x01=拼音提示显示在左边，对应的汉字提示在右边显示； 汉字提示起始显示 x 位置在 Scan1_Area_Start＋3×Font_Scale2+Scan_Dis。
0x1F	Scan_Return_Mode	1	0xAA:在*(VP-1)位置保存输入结束标记和有效数据长度： *(VP-1)高字节，输入结束标记：0x5A 表示输入结束，输入过程中为 0x00。 *(VP-1)低字节，有效输入数据长度，字节单位。 0xFF:不返回输入结束标记和长度。
0x20	0xFE	1	
0x21	Scan0_Area_Start	4	录入文本显示区域左上角坐标 (Xs,Ys)
0x25	Scan0_Area_End	4	录入文本显示区域右下角坐标 (Xe,Ye)
0x29	Scan1_Area_Start	4	录入过程中拼音提示文本显示区域的左上角坐标
0x2D	Scan_Dis	1	录入过程显示中，每个汉字显示的间距。每行固定显示最多 8 个汉字。
0x2E	0x00	1	
0x2F	KB_Source	1	键盘页面位置选择：0x00=键盘在当前页面；其它＝键盘不在当前页面。
0x30	0xFE	1	
0x31	PIC_KB	2	以下数据，仅当 KB_Source 不为 0x00 时有效。键盘所在页面 ID
0x33	AREA_KB	8	键盘页面上键盘区域坐标：左上角 (Xs,Ys) 、右下角 (Xe,Ye)
0x3B	AREA_KB_Position	4	键盘区域粘贴在当前页面显示的位置，左上角坐标。
0x3F	SCAN_MODE	1	0x02:拼音输入法 0x03:注音输入法 (台湾地区繁体录入)

注：

- 拼音“bd”对应所有 GBK 编码的全角标点符号录入；
- 迪文预装的 0#字库包含 4*8-64*128 点阵的所有 ASCII 字符。
- 不使用触摸屏，使用键盘 (0x4F 寄存器保存的键码) 来做 GBK 录入时，必须用 0x01-0x08 键码来选择对应的汉字。
- 注音输入法的键码 (键码低字节) 按照下表定义：

注音	ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ	ㄐ
键码	0xc5	0xc6	0xc7	0xc8	0xc9	0xca	0xcb	0xcc	0xcd	0xce	0xcf	0xd0
注音	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄧ	ㄨ	ㄩ
键码	0xd1	0xd2	0xd3	0xd4	0xd5	0xd6	0xd7	0xd8	0xd9	0xe7	0xe8	0xe9
注音	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ
键码	0xda	0xdb	0xdc	0xdd	0xde	0xdf	0xe0	0xe1	0xe2	0xe3	0xe4	0xe5
注音	ㄦ	ㄧ	ˊ	ˇ	ˋ	˙						
键码	0xe6	0x99	0x40	0x98	0x41	0x42						

GBK 汉字文本录入的两种模式说明：

6.9 硬件参数配置（0x07）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面 ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys)（Xe,Ye）
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE07
0x10	0xFE	1	0xFE
0x11	Mode	1	操作模式选择，见“操作模式表”说明。
0x12	DATA_PACK	14	操作模式数据包，见“操作模式表”说明。

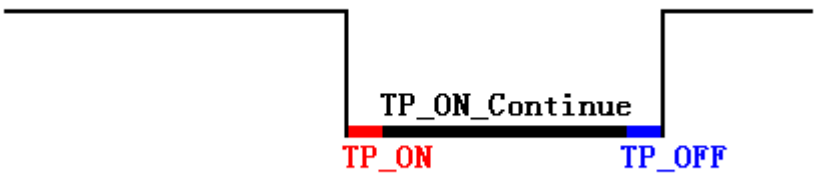
操作模式表

<table>
<tr><th>Mode</th><th>Data_Pack</th><th>Data_Pack 说明</th><th>功　能</th></tr>
<tr><td>0x00</td><td>无</td><td>无</td><td>加载寄存器变量区数据到 0x6F00-0x6FFF 变量存储器空间（占据低字节）；</td></tr>
<tr><td>0x01</td><td>无</td><td>无</td><td>加载 0x6F00-0x6FFF 变量存储器空间（占据低字节）数据到寄存器变量区；同时改写对应的 R1-R3,R5-RA　SD/SDHC 接口配置变量。</td></tr>
<tr><td rowspan="3">0x02</td><td>Tran_Area</td><td>将要转换的区域坐标：左上角、右下角</td><td rowspan="3">将指定区域的内容转换成单色位图（纵向取模打印位图格式），并保存到 VP 指针指向的数据存储器。
1.区域宽度（Xe-Xs+1）必须是偶数；
2.区域高度（Ye-Ys+1）必须是 8 的倍数；
3.*VP 指针保存数据格式如下：
*VP：状态位，处理完成后设置成 0x5555；
*VP+1：横向字长度＝（Xe-Xs＋1）&0xFFFE/2；
*VP+2：数据段个数＝（Ye-Ys＋1）&0xFFF8/8；
*VP+3：位图数据开始，MSB 方式。
如果启用了“参数自动上传功能”(R2.3=1)，那么转换完成后，会按照 *VP 内容被修改成 0x5555 而自动上传一条提示信息。
本指令主要用于屏幕内容的打印输出。</td></tr>
<tr><td>*VP</td><td>保存转换位图数据的缓冲区首地址</td></tr>
<tr><td colspan="2"><table>
<tr><td></td><td>X=0</td><td>X=1</td><td>X=2</td><td>X=3</td><td>…</td><td>X=126</td><td>X=127</td></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>数据指针</td><td rowspan="2">把*VP 指针位置开始的，Tx_Len 字节长度的数据发送到用户串口。
Tx_Len 是一个字变量，长度从 0x0001-0xFFFF。</td></tr>
<tr><td>Tx_LEN</td><td>要发送的数据长度</td></tr>
<tr><td>0x04</td><td colspan="3">功能同 0x03，区别在于数据发送到 COM2（系统保留串口）。</td></tr>
<tr><td rowspan="2">0x05</td><td>Tran_Area</td><td>将要转换的区域坐标：左上角、右下角</td><td rowspan="2">将指定区域的内容转换成单色位图（横向取模打印位图格式），并保存到 VP 指针指向的数据存储器。
1.区域宽度（Xe-Xs+1）必须是 16 的倍数；
2.*VP 指针保存数据格式如下：
*VP：状态位，处理完成后设置成 0x5555；
*VP+1：横向字长度＝（Xe-Xs＋1）&0xFFF0/16；
*VP+2：数据段个数＝（Ye-Ys＋1）；
*VP+3：位图数据开始，MSB 方式。
如果启用了“参数自动上传功能”(R2.3=1)，那么转换完成后，会按照 *VP 内容被修改成 0x5555 而自动上传一条提示信息。
本指令主要用于屏幕内容的打印输出。</td></tr>
<tr><td>*VP</td><td>保存转换位图数据的缓冲区首地址</td></tr>
<tr><td rowspan="2">0x06</td><td>Frame_Head</td><td>帧头（2 字节）</td><td rowspan="2">把当前点击位置的触摸屏坐标发到 COM2（系统保留串口），格式如下：
Frame_Head+X＋Y＋Check(X,Y 的 1 字节累加和)＋Frame_end。</td></tr>
<tr><td>Frame_End</td><td>帧尾（2 字节）</td></tr>
</table>

6.10 触摸屏按压状态同步数据返回（0x08）

地址	定义	数据长度	说　明
0x00	Pic_ID	2	页面 ID
0x02	TP_Area	8	触控按钮区域：(Xs,Ys) (Xe,Ye)
0x0A	Pic_Next	2	目标切换页面，0xFF**表示不进行页面切换。
0x0C	Pic_On	2	按钮按压效果图所处的页面，0xFF**表示没有按钮按压效果。
0x0E	TP_Code	2	0xFE08
0x10	0xFE	1	0xFE
0x11	TP_ON_Mode	1	触摸屏第一次按压下去时，数据返回模式： 0x00=不返回数据 0x01=读取*VP1S 指向的 LEN1 长度数据到*VP1T 指向的存储空间。 0x02=读取*VP1S 指向的 LEN1 长度数据发送到串口。 0x03=读取*VP1S 指向的 LEN1 长度数据到*VP1T 指向的寄存器空间。
0x12	VP1S	2	触摸屏第一次按压时，读取数据的地址。
0x14	VP1T	2	触摸屏第一次按压时，写入数据的地址。
0x16	0x00	1	0x00
0x17	LEN1	1	返回数据长度，字节数。TP_ON_Mode=0x01 时，LEN1 必须为偶数。
0x18	0xFE	1	0xFE
0x19	TP_ON_Continue_Mode	1	触摸屏第一次按压后，持续按压下时，数据返回模式： 0x00=不返回数据 0x01=读取*VP2S 指向的 LEN2 长度数据到*VP2T 指向的存储空间。 0x02=读取*VP2S 指向的 LEN2 长度数据发送到串口。 0x03=读取*VP2S 指向的 LEN2 长度数据到*VP2T 指向的寄存器空间。
0x1A	VP2S	2	触摸屏持续按压时，读取数据的地址。
0x1C	VP2T	2	触摸屏持续按压时，写入数据的地址。
0x1E	0x00	1	0x00
0x1F	LEN2	1	返回数据长度，字节数。TP_ON_Continue_Mode=0x01 时，LEN2 必须为偶数。
0x20	0xFE	1	0xFE
0x21	TP_OFF_Mode	1	触摸屏松开时，数据返回模式： 0x00=不返回数据 0x01=读取*VP3S 指向的 LEN3 长度数据到*VP3T 指向的存储空间。 0x02=读取*VP3S 指向的 LEN3 长度数据发送到串口。 0x03=读取*VP3S 指向的 LEN3 长度数据到*VP3T 指向的寄存器空间。
0x22	VP3S	2	触摸屏松开时，读取数据的地址。
0x24	VP3T	2	触摸屏松开时，写入数据的地址。
0x26	0x00	1	0x00
0x27	LEN3	1	返回数据长度，字节数。TP_OFF_Mode=0x01 时，LEN3 必须为偶数。
0x28	0x00	8	保留，写 0x00

触摸屏按压的 3 个状态，如下图所示：



7 显示变量配置文件（14.BIN）说明

显示变量配置文件由N条按照页面配置的变量指令组成，每条变量指令固定占用32字节存储空间。

每个页面固定分配2KB或4KB（0x0800或0x1000）变量存储空间，每个页面最多可以设置64或128个变量(64/128变量选择由CONFIG.TXT配置文件中的RC.4选择)。

显示变量配置文件最大2MB，可以配置最多1024个页面（128变量模式下为512页面）。

相同类型的变量，存储位置越靠后，显示优先级越高。

一条显示变量配置指令由以下6部分组成：

序号	定义	数据长度	说　明
1	0x5A	1	固定
2	Type	1	变量类别
3	*SP	2	变量描述文件从Flash加载后存储到数据存储区的地址指针，0xFFFF表示不转存到数据存储区。
4	Len_Dsc	2	变量描述内容的字长度
5	*VP	2	变量地址，0x0000-0x6FFF，有些无需指定地址的变量，写0x0000即可。 当变量地址高字节为0xFF时，本条指令将被取消。
6	Description	N	变量描述内容

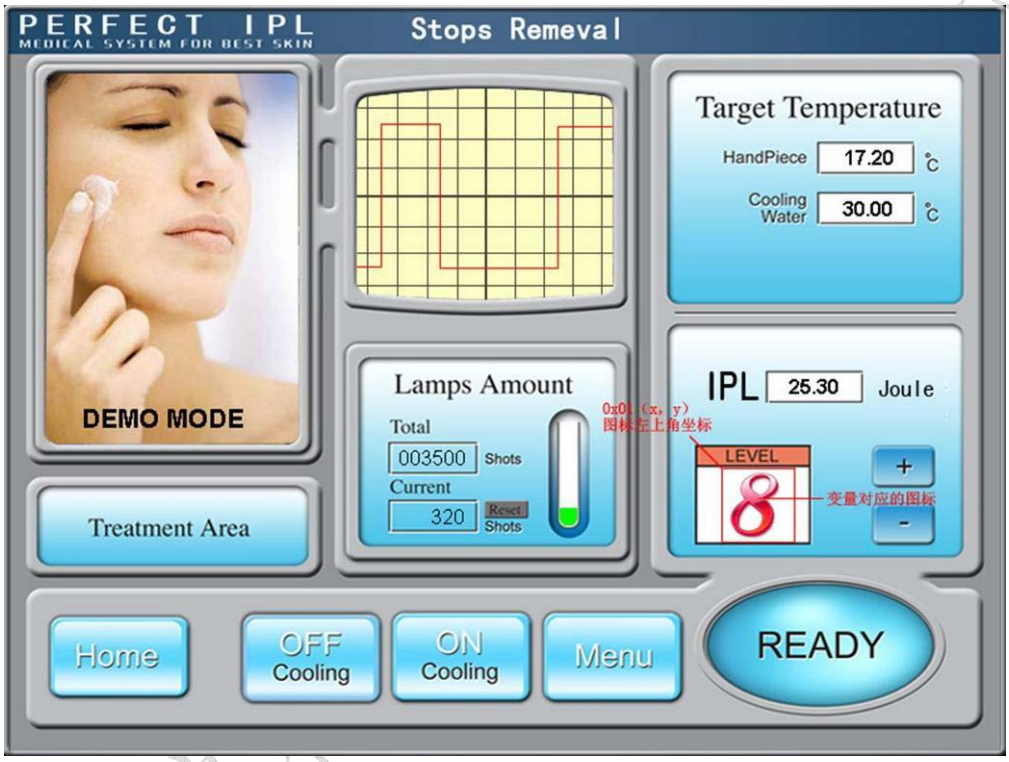
7.1 显示变量功能一览表

序号	功能代码	功　能	说　明
01	00	变量图标显示	将一个数据变量的变化范围线性对应一组 ICON 图标显示；当变量变化时，图标也自动相应切换。多用于精细的仪表板、进度条显示。
02	01	动画图标显示	将一个定值数据变量对应了 3 种不同的图标指示状态：不显示、显示固定图标、显示动画图标。多用于变量的报警提示。
03	02	滑块刻度指示	将一个数据变量的变化范围对应一个图标（滑块）的显示位置变化。 多用于液位、刻度盘、进度表的指示。
04	03	艺术字变量显示	用 ICON 图标取代字库来显示变量数据。
05	04	图片动画显示	将一组全屏图片按照指定速度播放。多用于开机界面或屏保。
06	05	图标旋转显示	把一个数据变量的变化范围线性对应角度数据，然后把一个 ICON 图标按照对应的角度数据旋转后显示出来。多用于指针仪表板显示。
07	06	位变量图标显示	把一个数据变量的每个位（bit）的 0/1 状态对应 8 种不同显示方案中的两种，用 ICON 图标（或图标动画）来对应显示。 多用于开关状态显示，比如风机的运转（动画）、停止（静止图标）。
08	10	数据变量显示	把一个数据变量按照指定格式（整数、小数、是否带单位）用指定字体和大小的阿拉伯数字显示出来。
09	11	文本显示	把字符串按照指定的格式（选择字库决定），在指定的文本框显示区域显示。
10	12_00	文本格式 RTC 显示	按照用户编辑的格式把公历 RTC 用文本显示出来。
11	12_01	表盘格式 RTC 显示	采用 ICON 图标旋转，用指针表盘方式把公历 RTC 显示出来。
12	13	HEX 数据显示	把变量数据按照字节 HEX 方式间隔用户指定的 ASCII 字符显示出来。 多用于计时显示，比如把 1234 显示成 12:34。
13	20	实时曲线（趋势图）	结合 0x84 串口写曲线缓冲区数据来自动匹配显示实时曲线（趋势图）。可以指定显示区域、中心轴坐标、显示比例（放大/缩小）可控。
14	21_01	绘图_置点	置点（x,y,color）
15	21_02	绘图_端点连线	端点连线（color,(x0,y0),…(xn,yn)）
16	21_03	绘图_矩形	显示矩形，颜色和位置、大小可控。
17	21_04	绘图_矩形填充	填充指定的矩形区域，填充颜色和位置、大小可控。
18	21_05	绘图_画圆	显示整圆弧，颜色和位置、大小可控。
19	21_06	绘图_图片剪切粘贴	从指定图片上剪切一个区域粘贴到当前显示页面上。
20	21_07	绘图_ICON 图标显示	ICON 图标显示，图标库可以选择。
21	21_08	绘图_封闭区域填充	封闭区域填充，种子点坐标、填充颜色可控。
22	21_09	绘图_频谱显示	根据变量数据显示频谱（垂直线条），线条颜色、位置可控。
23	21_0A	绘图_线段显示	根据变量数据连接线段，端点、颜色可控。
24	21_0B	绘图_圆弧显示	显示圆弧，半径、颜色、起止角度可控。
25	21_0C	绘图_字符显示	根据变量数据进行单个字符显示。
26	21_0D	绘图_矩形域 XOR	对指定的矩形域位图数据用指定颜色进行 XOR 操作，多用于高亮显示。
27	21_0E	绘图_双色位图显示	变量存储器数据看成双色位图数据，0/1 对应颜色可指定，多用于自定义光标。
28	21_0F	绘图_位图显示	变量存储器数据位 65K 色位图数据，多用于实时图标（照片）下载显示。
29	21_10	绘图_区域放大粘贴	把指定区域放大 1 倍粘贴到指定位置，多用于配合 0F 指令实现照片实时显示。
30	22	列表显示	把按照二维数组定义的数据用表格分栏显示出来。

7.2 图标变量

7.2.1 变量图标显示（0x00）

地址		定义	数据长度	说　明
0x00		0x5A00	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x0008	2	
0x06	0x00	*VP	2	变量指针，变量为整数格式。
0x08	0x01	(x,y)	4	变量显示位置，图标左上角坐标位置
0x0C	0x03	V_Min	2	变量下限，越界不显示
0x0E	0x04	V_Max	2	变量上限，越界不显示
0x10	0x05	Icon_Min	2	V_Min 对应的图标 ID
0x12	0x06	Icon_Max	2	V_max 对应的图标 ID
0x14	0x07:H	Icon_Lib	1	图标库存储位置
0x15	0x07:L	Mode	1	ICON 显示模式，0x00=透明（不显示背景） 其它＝显示图标背景



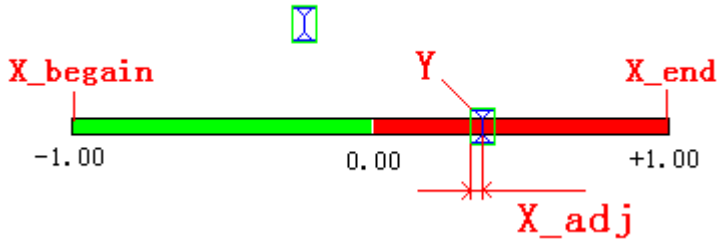
7.2.2 动画图标显示（0x01）

地址		定义	数据长度	说　明
0x00		0x5A01	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000A	2	
0x06	0x00	*VP	2	初始图标变量指针，变量为双字，低位字保留，高位字无符号数（0x0000-0x0FFFF）用户数据控制动画图标显示。
0x08	0x01	(x,y)	4	变量显示位置，图标左上角坐标位置。
0x0C	0x03	0x0000	2	固定
0x0E	0x04	V_Stop	2	变量为该值时显示固定图标
0x10	0x05	V_Start	2	变量为该值时自动显示动画图标
0x12	0x06	Icon_Stop	2	变量为 V_STOP 值时固定显示的图标
0x14	0x07	Icon_Start	2	变量为 V_Start 值时，自动从 Icon_Start 到 Icon_End 显示图标，形成动画。
0x16	0x08	Icon_End	2	
0x18	0x09:H	Icon_Lib	1	图标库存储位置
0x19	0x09:L	Mode	1	ICON 显示模式，0x00=透明

当变量不等于 V_Stop 或者 V_Start 时，不显示图标或者动画。

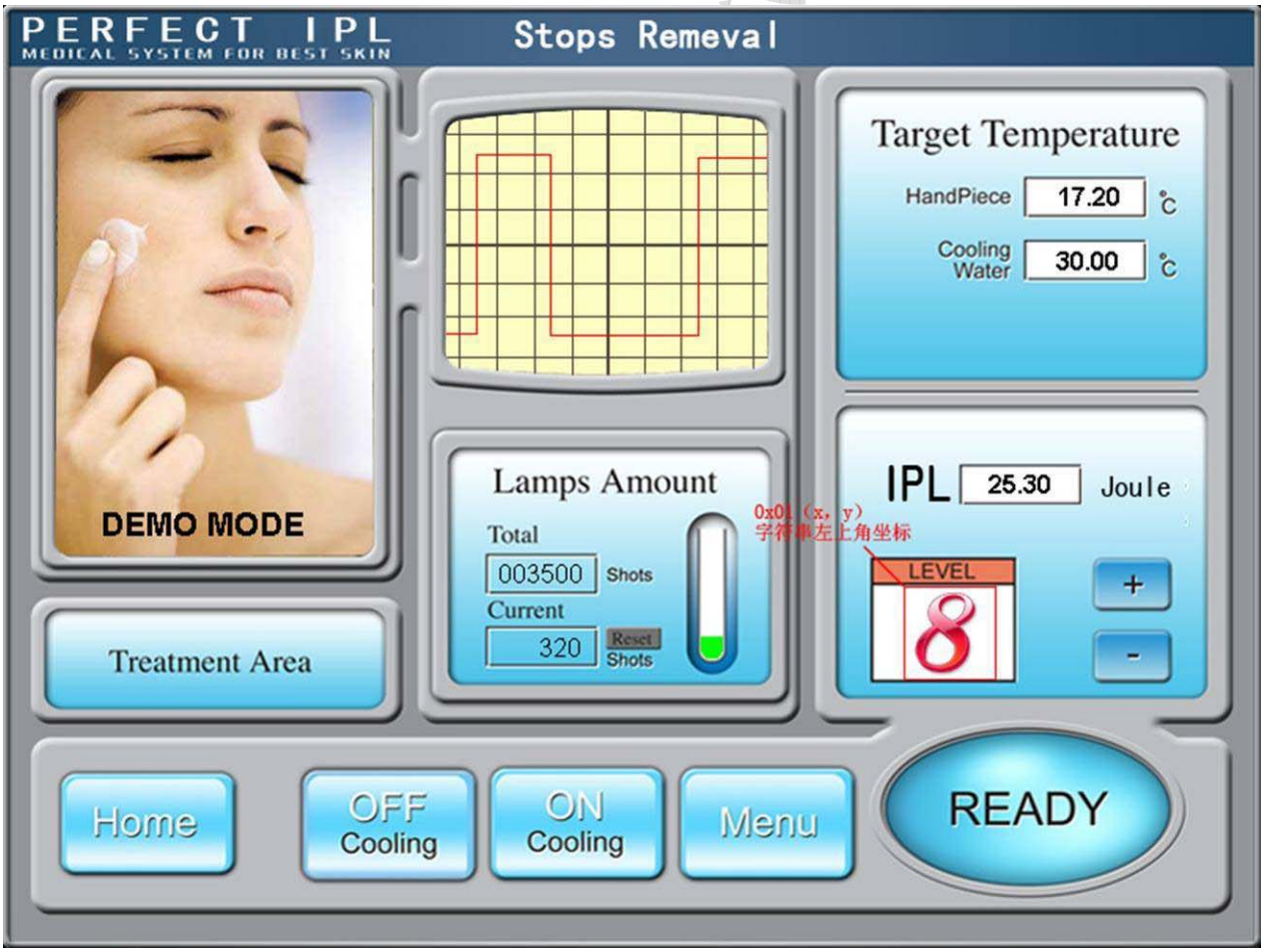
7.2.3 滑块刻度指示（0x02）

地址		定义	数据长度	说　明
0x00		0x5A02	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000A	2	
0x06	0x00	*VP	2	变量指针，变量格式由 VP_DATA_Mode 决定。
0x08	0x01	V_begain	2	对应起始刻度的变量值
0x0A	0x02	V_end	2	对应终止刻度的变量值
0x0C	0x03	X_begain	2	起始刻度坐标（纵向为 Y 坐标）
0x0E	0x04	X_end	2	终止刻度坐标（纵向为 Y 坐标）
0x10	0x05	Icon_ID	2	刻度滑动块的图标 ID
0x12	0x06	Y	2	刻度指示图标显示的 Y 坐标位置（纵向为 X 坐标）
0x14	0x07:H	X_adj	1	刻度指示图标显示的 X 坐标前移偏移量（纵向为 Y），0x00-0xFF
0x15	0x07:L	Mode	1	刻度模式：0x00=横向刻度条　01＝纵向刻度条
0x16	0x08:H	Icon_Lib	1	图标库存储位置
0x17	0x08:L	Icon_mode	1	ICON 显示模式，0x00=透明（不显示背景）　其它＝显示图标背景
0x18	0x09:H	VP_DATA_Mode	1	0x00：*VP 指向一个整型变量 0x01：*VP 指向一个整型变量的高字节数据 0x02：*VP 指向一个整型变量的低字节数据



7.2.4 艺术字变量显示（0x03）

地址		定义	数据长度	说　明
0x00		0x5A03	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x0007	2	
0x06	0x00	*VP	2	变量指针
0x08	0x01	X,Y	4	起始显示位置： 左对齐模式，坐标为显示字符串左上角坐标； 右对齐模式，坐标为显示字符串的右上角坐标。
0x0C	0x03	Icon0	2	0 对应的 ICON_ID，排列顺序为 0123456789-.
0x0E	0x04:H	Icon_Lib	1	Icon 库位置
0x0F	0x04:L	Icon_Mode	1	ICON 显示模式，0x00=透明（不显示背景） 其它＝显示图标背景
0x10	0x05:H	整数位数	1	显示的整数位数
0x11	0x05:L	小数位数	1	显示的小数位数
0x12	0x06:H	变量数据类型	1	0x00=整数(2 字节)， -32768 到 32767 0x01=长整数(4 字节) -2147483648 到 2147483647 0x02=*VP 高字节,无符号数 0 到 255 0x03=*VP 低字节，无符号数 0 到 255 0x04= 超 长 整 数 (8 字 节) -9223372036854775808 到 9223372036854775807 0x05=无符号整数(2 字节) 0 到 65535 0x06=无符号长整数(4 字节) 0 到 4294967295
0x13	0x06:L	对齐模式	1	0x00=左对齐 0x01=右对齐



7.2.5 图片动画显示（0x04）

地址		定义	数据长度	说　明
0x00		0x5A04	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x0004	2	
0x06	0x00	0x0000	2	固定
0x08	0x01	Pic_Begain	2	起始图片位置
0x0A	0x02	Pic_End	2	终止图片位置
0x0C	0x03:H	Frame_Time	1	一帧（一幅图片）显示的时间，单位为 8mS

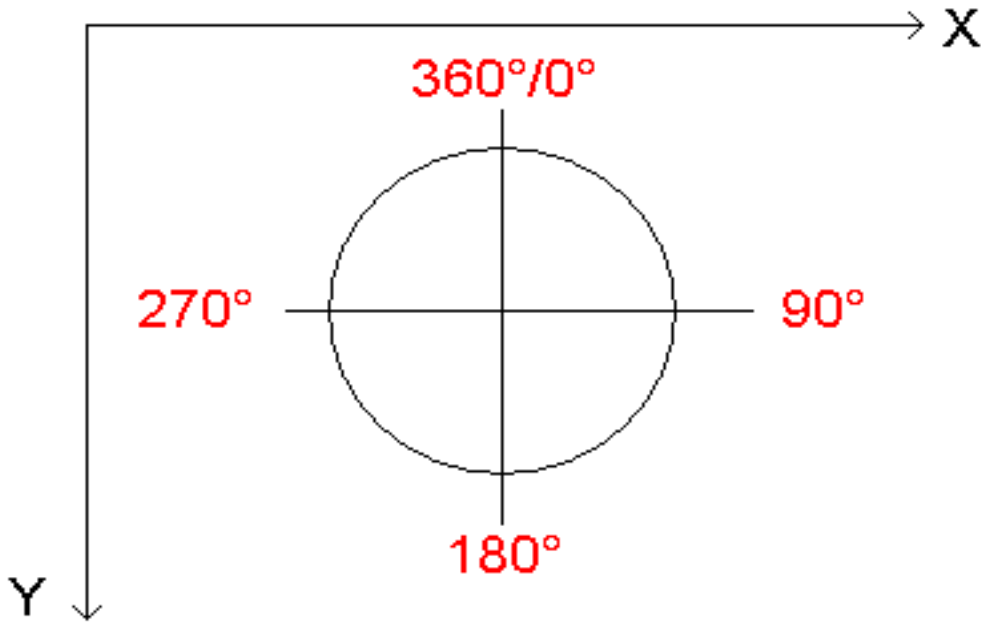
起始图片位置必须小于终止图片位置。
如果在 Pic_End 页面也设置图片动画变量，将可以实现不断重播。
串口指令切换图片或者触控指令切换图片可以结束重播。

7.2.6 图标旋转指示（0x05）

地址		定义	数据长度	说　明
0x00		0x5A05	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000C	2	
0x06	0x00	*VP	2	变量指针,变量格式由 VP_Mode 决定。
0x08	0x01	Icon_ID	2	指定的图标 ID。
0x0A	0x02	Icon_Xc	2	ICON 图标上的旋转中心位置：X 坐标。
0x0C	0x03	Icon_Yc	2	ICON 图标上的旋转中心位置：Y 坐标。
0x0E	0x04	Xc	2	ICON 显示到当前屏幕的旋转中心位置：X 坐标。
0x10	0x05	Yc	2	ICON 显示到当前屏幕的旋转中心位置：Y 坐标。
0x12	0x06	V_Begain	2	对应起始旋转角度的变量值，整型数，越界不显示
0x14	0x07	V_End	2	对应终止旋转角度的变量值，整型数，越界不显示
0x16	0x08	AL_Begain	2	起始旋转角度，0-720（0x000-0x2D0），单位 0.5°。
0x18	0x09	AL_End	2	终止旋转角度，0-720（0x000-0x2D0），单位 0.5°。
0x1A	0x0A:H	VP_Mode	1	0x00：*VP 指向一个整型变量 0x01：*VP 指向一个整型变量的高字节数据 0x02：*VP 指向一个整型变量的低字节数据
0x1B	0x0A:L	Lib_ID	1	ICON 图标库 ID。
0x1C	0x0B	Mode	1	ICON 显示模式，0x00=透明（不显示背景）　其它＝显示图标背景

本指令主要用于仪表刻度盘的指针指示。

旋转始终假定为“ 顺时针 ”转动，即 AL_End 必须大于 AL_Begain(如果 AL_End 小于 AL_Begain，系统处理时会自动加上 360°)。



7.2.7 位变量图标显示（0x06）

地址		定义	数据长度	说　明
0x00		0x5A06	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000C	2	
0x06	0x00	*VP	2	位变量指针,字变量
0x08	0x01	*VP_AUX	2	辅助变量指针，双字，用户软件不能访问。
0x0A	0x02	Act_Bit_Set	2	为 1 的 bit 位置说明*VP 对应位置需要显示。
0x0C	0x03:H	Display_Mode	1	定义了显示模式： Display_Mode | 变量位（Bit）值 0 | 1 0x00 | ICON0S | ICON1S 0x01 | ICON0S | 不显示 0x02 | ICON0S | ICON1S-ICON1E 动画 0x03 | 不显示 | ICON1S 0x04 | 不显示 | ICON1S-ICON1E 动画 0x05 | ICON0S-ICON0E 动画 | ICON1S 0x06 | ICON0S-ICON0E 动画 | 不显示 0x07 | ICON0S-ICON0E 动画 | ICON1S-ICON1E 动画 比如设置 Display_Mode=0x02，那么： *VP 对应的变量某个位为“0”时，显示 ICON0S 图标；
0x0D	0x03:L	Move_mode	1	位图图标排列方式： 0x00=X++，Act_Bit_Set 指定的不处理 bit 不保留位置； 0x01=Y++，Act_Bit_Set 指定的不处理 bit 不保留位置； 0x02=X++，Act_Bit_Set 指定的不处理 bit 保留 DIS_MOV 位置； 0x03=Y++，Act_Bit_Set 指定的不处理 bit 保留 DIS_MOV 位置；
0x0E	0x04:H	Icon_Mode	1	ICON 显示模式：0x00=透明 0x01=不透明
0x0F	0x04:L	Icon_Lib	1	图标库存储位置
0x10	0x05	ICON0S	2	不显示动画模式，bit 0 图标 ID 显示动画模式，bit 0 图标动画起始 ID 位置
0x12	0x06	ICON0E	2	显示动画模式，bit 0 图标动画结束 ID 位置
0x14	0x07	ICON1S	2	不显示动画模式，bit 1 图标 ID 显示动画模式，bit 1 图标动画起始 ID 位置
0x16	0x08	ICON1E	2	显示动画模式，bit 1 图标动画结束 ID 位置
0x18	0x09	(x,y)	4	起始位变量显示位置，图标左上角坐标位置。
0x1C	0x0B	DIS_MOV	2	下一个图标坐标移动坐标间隔
0x1E			2	写 0x00

7.3 文本变量

7.3.1 数据变量显示（0x10）

地址		定义	数据长度	说　明
0x00		0x5A10	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000D	2	
0x06	0x00	*VP	2	变量指针
0x08	0x01	X,Y	4	起始显示位置，显示字符串左上角坐标。
0x0C	0x03	COLOR	2	显示颜色
0x0E	0x04:H	Lib_ID	1	ASCII 字库位置
0x0F	0x04:L	字体大小	1	字符 X 方向点阵数
0x10	0x05:H	对齐方式	1	0x00=左对齐　0x01=右对齐 0x02=居中
0x11	0x05:L	整数位数	1	显示整数位
0x12	0x06:H	小数位数	1	显示小数位
0x13	0x06:L	变量数据类型	1	0x00=整数(2 字节)，　-32768 到 32767 0x01=长整数(4 字节)　-2147483648 到 2147483647 0x02=*VP 高字节,无符号数　0 到 255 0x03=*VP 低字节，无符号数　0 到 255 0x04=超长整数(8 字节)　-9223372036854775808 到 9223372036854775807 0x05=无符号整数(2 字节)　0 到 65535 0x06=无符号长整数(4 字节)　0 到 4294967295
0x14	0x07:H	Len_unit	1	变量单位（固定字符串）显示长度，0x00 表示没有单位显示
0x15	0x07:L	String_Unit	Max11	单位字符串，ASCII 编码

整数位数和小数位数之和不能超过 20。

7.3.2 文本显示（0x11）

地址		定义	数据长度	说　明
0x00		0x5A11	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000D	2	
0x06	0x00	*VP	2	文本指针
0x08	0x01	X,Y	4	起始显示位置，显示字符串左上角坐标。
0x0C	0x03	Color	2	显示文本颜色
0x0E	0x04	Xs Ys Xe Ye	8	文本框
0x16	0x08	Text_length	2	显示字节数量,遇到 0xFFFF、0x0000 数据或者显示到文本框尾将不再显示。
0x18	0x09:H	Font0_ID	1	编码方式 0x01-0x04 时 ASCII 字库位置。
0x19	0x09:L	Font1_ID	1	编码方式 0x00、0x05，以及 0x01-0x04 的非 ASCII 字符使用的字库
0x1A	0x0A:H	Font_X_Dots	1	字体 X 方向点阵数（0x01-0x04 模式，ASCII 字符 X 按照 X/2 计算）
0x1B	0x0A:L	Font_Y_Dots	1	字体 Y 方向点阵数目
0x1C	0x0B:H	Encode_Mode	1	.7 定义了文本显示的字符间距是否自动调整： .7=0　字符间距自动调整； .7=1　字符间距不自动调整，字符宽度固定为设定的点阵数。 .6-.0 定义了文本编码方式： 0=8bit 编码 1=GB2312 内码 2=GBK 3=BIG5 4=SJIS 5=UNICODE
0x1D	0x0B:L	HOR_Dis	1	字符水平间隔
0x1E	0x0C:H	VER_Dis	1	字符垂直间隔
0x1F	0x0C:L	未定义	1	写 0x00

注意，文本显示时，字库中字体的 Y 方向点阵数目必须为偶数。

DGUS 屏预装的 0#字库，包含 4*8－64*128 点阵的所有 ASCII 字符。

7.3.3 RTC 显示（0x12）

➢ **文本 RTC 显示**

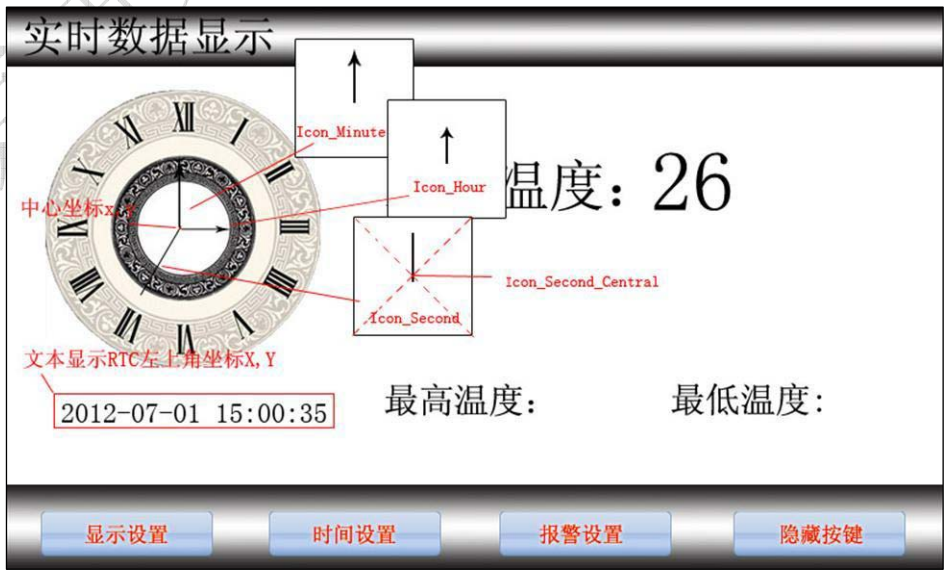
地址		定义	数据长度	说 明
0x00		0x5A12	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000D	2	
0x06	0x00	0x0000	2	
0x08	0x01	X,Y	4	显示位置，显示字符串左上角坐标。
0x0C	0x03	Color	2	字体颜色
0x0E	0x04:H	Lib_ID	1	字库位置
0x0F	0x04:L	字体大小	1	X 方向点阵数目
0x10	0x05	String_Code	MAX16	编码字符串，使用 RTC 编码表和 ASCII 字符构成。 假设当前时间是 2012-05-02 12:00:00 星期三，那么 ➢ Y-M-D H:Q:S 0x00 将显示为 2012-05-02 12:00:00 ➢ M-D W H:Q 0x00 将显示为 05-02 WED 12:00

RTC 编码表：

说 明	编码	显示格式
公历_年	Y	2000-2099
公历_月	M	01-12
公历_日	D	01-31
公历_小时	H	00-23
公历_分钟	Q	00-59
公历_秒	S	00-59
公历_星期	W	SUN MON TUE WED THU FRI SAT
编码结束	0x00	

➢ **表盘时钟显示**

地址		定义	数据长度	说 明
0x00		0x5A12	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000D	2	
0x06	0x00	0x0001	2	
0x08	0x01	X,Y	4	时钟表盘的指针中心。
0x0C	0x03	Icon_Hour	2	时针 ICON 的 ID，0xFFFF 表示时针不显示。
0x0E	0x04	Icon_Hour_Central	4	时针 ICON 的旋转中心位置。
0x12	0x06	Icon_Minute	2	分针 ICON 的 ID，0xFFFF 表示分针不显示。
0x14	0x07	Icon_Minute_Central	4	分针 ICON 的旋转中心位置。
0x18	0x09	Icon_Second	2	秒针 ICON 的 ID，0xFFFF 表示秒钟指针不显示。
0x1A	0x0A	Icon_Second_Central	4	秒针 ICON 的旋转中心位置。
0x1E	0x0C:H	ICON_Lib	1	指针图标所在的 ICON 库文件 ID
0x1F		未定义	1	写 0x00



7.3.4 HEX 变量显示（0x13）

地址		定义	数据长度	说　明
0x00		0x5A13	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000D	2	
0x06	0x00	*VP	2	变量指针数据串首地址，变量为 BCD（HEX）编码。 比如： 数据 0x32 显示为 32。 数据 0xBF 将显示为 BF。
0x08	0x01	X,Y	4	显示起始位置，显示字符串左上角坐标。
0x0C	0x03	Color	2	字体颜色
0x0E	0x04:H	Byte_Num	1	*VP 指针高字节开始显示的字节数目，0x01-0x0F
0x0F	0x04:L	Lib_ID	1	字库位置；字库必须是半角方式。 如果 Lib_ID 不为 0，字库必须使用 8bit 编码。
0x10	0x05:H	Font_x	1	X 方向点阵数目。
0x11	0x05:L	String_Code	MAX15	编码字符串，用来和时间变量组合出客户需要的显示格式。 每显示一个 BCD 时间码后，会到编码字符串顺序取出一个 ASCII 字符来间隔显示。 编码字符串中，特殊字符定义如下： 0x00：无效，本字符不显示，两个 BCD 时间码将连在一起显示； 0x0D：换行显示，即 X=Xs，Y=Y+Font_X*2。

7.3.5 文本滚屏显示（0x14）

地址		定义	数据长度	说　明
0x00		0x5A14	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000B	2	
0x06	0x00	*VP	2	文本指针。 文本指针前 3 个字必须保留，用户显示文本内容从（VP+3）开始存放。文本必须以 0xFF 或 0x00 结尾。
0x08	0x01:H	Rolling_Mode	1	滚屏模式：0x00=从右向左滚屏。
0x09	0x01:L	Rolling_Dis	1	滚屏间距，每个 DGUS 周期文本滚动的像素点阵数。
0x0A	0x02:H	Adjust_Mode	1	0x00=左对齐　0x01=右对齐　0x02=居中。 文本显示内容不足文本框时滚屏停止，此时显示对齐模式方有效。
0x0B	0x02:L	未定义	1	写 0x00
0x0C	0x03	Color	2	显示文本颜色
0x0E	0x04	Xs Ys Xe Ye	8	文本框
0x16	0x08:H	Font0_ID	1	编码方式为 0x01-0x04 时：ASCII 字符显示的字库位置。 编码方式为 0x00、0x05 时：本参数不用设置，写 0x00 即可。
0x17	0x08:L	Font1_ID	1	编码方式为 0x01-0x04 时：非 ASCII 字符显示的字库位置。 编码方式为 0x00、0x05 时：显示字符使用的字库位置。
0x18	0x09:H	Font_X_Dots	1	字体 X 方向点阵数（0x01-0x04 模式，ASCII 字符 X 将自动按照 X/2 计算）。
0x19	0x09:L	Font_Y_Dots	1	字体 Y 方向点阵数目。
0x1A	0x0A:H	Encode_Mode	1	.7 定义了文本显示的字符间距是否自动调整： .7=0　字符间距自动调整； .7=1　字符间距不自动调整，字符宽度固定为设定的点阵数。 .6-.0 定义了文本编码方式： 0=8bit 编码 1=GB2312 内码 2=GBK 3=BIG5 4=SJIS 5=UNICODE
0x1B	0x0A:L	Text_Dis	1	字符间隔
0x1C	0x0A:H	未定义	4	写 0x00

注意，文本显示时，字库中字体的 Y 方向点阵数目必须为偶数。

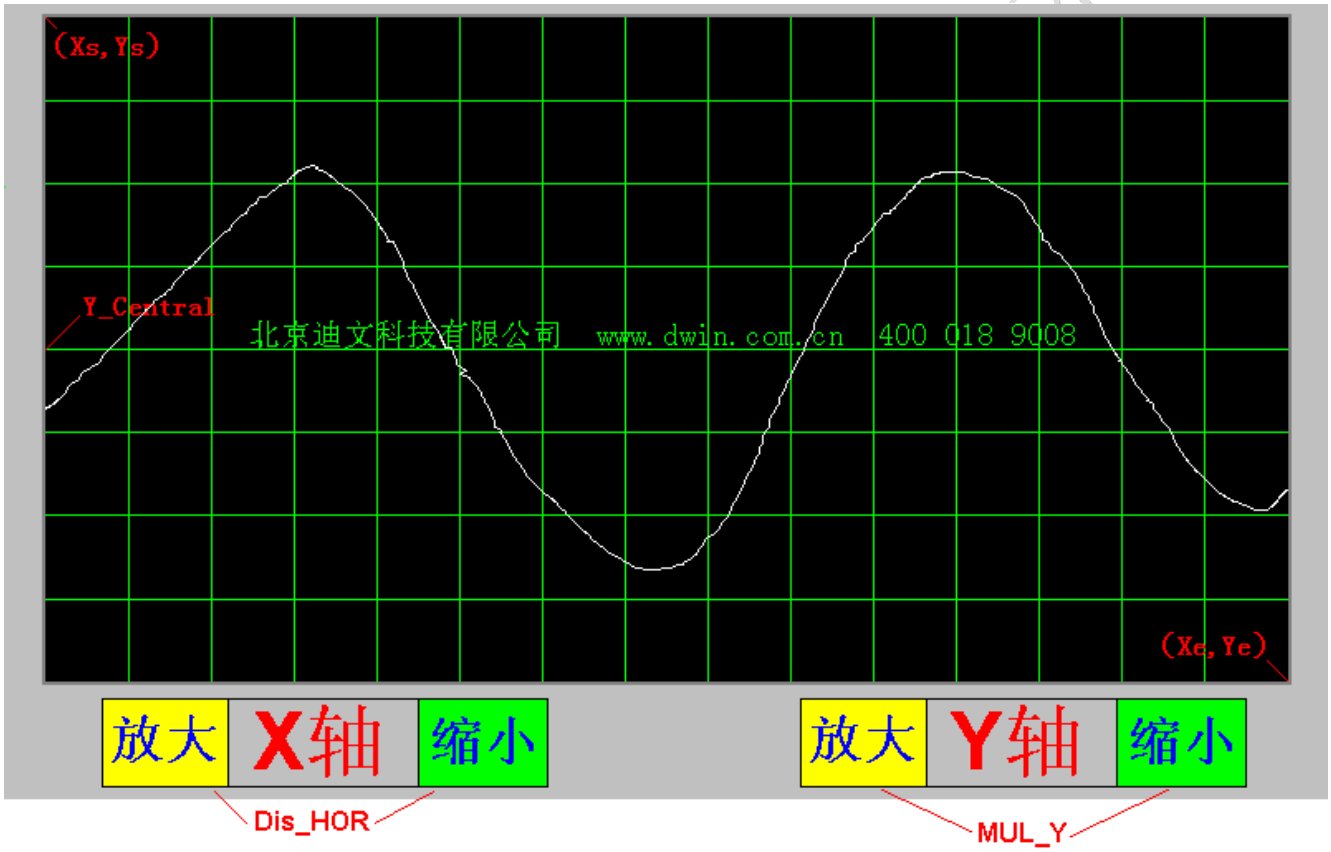
DGUS 屏预装的 0#字库，包含 4*8－64*128 点阵的所有 ASCII 字符。

7.4 图形变量

7.4.1 实时曲线（趋势图）显示（0x20）

地址		定义	数据长度	说 明
0x00		0x5A20	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000A	2	
0x06	0x00	0x0000	2	无定义
0x08	0x01	Xs:Ys:Xe:Ye	8	曲线窗口：左上角坐标（Xs,Ys），右下角坐标（Xe,Ye）； 曲线越界将不显示。
0x10	0x05	Y_Central	2	曲线中心轴位置
0x12	0x06	VD_Central	2	中心轴对应的的曲线数据值，一般取最大数据和最小数据和的 50%。
0x14	0x07	Color	2	曲线颜色
0x16	0x08	MUL_Y	2	纵轴放大倍数,单位是 1/256，0x0000-0x7FFF。
0x18	0x09:H	CHANEL	1	数据源通道，0x00-0x07
0x19	0x09:L	Dis_HOR	1	横轴间隔，0x01-0xFF。

曲线数据用 0x84 指令发送，请参考 3.2 指令集 说明。



如果把变量描述内容存储在数据存储空间（*SP 指定存储位置），那么：

- 结合 0x02 增量触控指令，可以实现无需要用户代码干预的曲线自动缩放；
- 结合 0x03 拖动触控指令修改 Y_Central 值，可以实现无需用户代码干预的曲线上下移动。

如果需要显示曲线线条比较粗，可以在同一个位置放置多个上下（Y 轴）平移的曲线变量并应引用同一个数据源通道即可实现。

满量程曲线的纵轴放大倍数计算：

MUL_Y=（Ye-Ys）*256/（Vmax-Vmin） Ye,Ys 为曲线窗口的 Y 坐标，Vmax,Vmin 为曲线数据的最大,最小值。

比如，一个 12bit A/D 采集数据（Vmax=4095 Vmin=0）要对应在 Ys=50,Ye=430 的屏幕区域满量程显示，那么：

MUL_Y=(430-50)*256/(4095-0)=23.7 向下舍入取 23。

7.4.2 基本图形显示（0x21）

地址		定义	数据长度	说　明
0x00		0x5A21	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x0008	2	
0x06	0x00	*VP	2	变量数据指针。
0x08	0x01	Area	8	绘图显示区域定义：指定区域的左上角、右下角坐标；绘图越界将不显示。仅对 0x0001-0x0005、0x0009、0x000A、0x000B 指令有效。
0x10	0x05:H	Dashed_Line_En	1	0x5A:使用线段的绘图指令（0x02、0x03、0x09、0x0A 指令）将使用虚线或者点划线显示线段； 其它:使用线段的绘图指令使用实线显示线段。
0x11	0x05:L	Dash_Set	4	4 个字节依次设置了虚线（点划线）格式： 第 1 段实线点阵数、第 1 段虚线点阵数、第 2 段实线点阵数、第 2 段虚线点阵数。 比如，设置 0x10 0x04 0x10 0x04 将显示虚线；设置 0x10 0x04 0x02 0x04 将显示点划线。
0x15			13	保留，写 0x00

基本图形显示先在 14.BIN 中定义一个“ 绘图板 ”功能，而具体的绘图操作则由*VP 指向的变量存储器内容决定。用户通过改变变量存储器功能来实现不同的绘图功能。

（变量存储空间的）变量数据格式说明

地　址	定　义	说明
VP	CMD	绘图指令
VP+1	Data_Pack_Num_Max	最大数据包数目：连线指令（0x0002），定义为连线线条数目（顶点数－1）；
VP+2	DATA_Pack	数据

绘图指令数据包说明

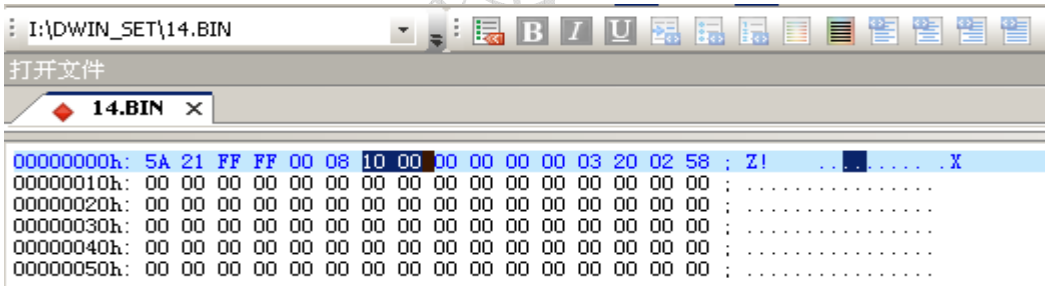
指　令（CMD）	操　作	绘图数据包格式说明（相对地址和长度单位均为字（word））			
		相对地址	长度	定　义	说　明
0x0001	置点	0x00	2	(x,y)	置点坐标位置，x 坐标高字节为判断条件。
		0x02	1	Color	置点颜色
0x0002	端点连线	0x00	1	Color	线条颜色
		0x01	2	(x,y) 0	连线顶点 0 坐标，x 坐标高字节为判断条件。
		0x03	2	(x,y) 1	连线顶点 1 坐标，x 坐标高字节为判断条件。
		0x01+2*n	2	(x,y) n	连线顶点 n 坐标，x 坐标高字节为判断条件。
0x0003	矩形	0x00	2	(x,y) s	矩形框左上角坐标，x 坐标高字节为判断条件。
		0x02	2	(x,y) e	矩形框右下角坐标。
		0x04	1	Color	矩形颜色
0x0004	矩形域填充	0x00	2	(x,y) s	矩形域左上角坐标，x 坐标高字节为判断条件。
		0x02	2	(x,y) e	矩形域右下角坐标。
		0x04	1	Color	矩形域填充颜色
0x0005	整圆弧显示	0x00	2	(x,y)	圆心坐标，x 坐标高字节为判断条件。
		0x02	1	Rad	半径。
		0x03	1	Color	圆颜色。
0x0006	图片区域剪切、粘贴	0x00	1	Pic_ID	剪切图片区域所在页面 ID；高字节为判断条件。
		0x01	2	(x,y) s	剪切图片区域左上角坐标。
		0x03	2	(x,y) e	剪切图片区域右下角坐标。
		0x05	2	(x,y)	剪切图片区域粘贴到当前页面的坐标位置，左上角坐标。
0x**07	ICON 图标显示	0x00	2	(x,y)	显示坐标位置，x 坐标高字节为判断条件。
		0x02	1	ICON_ID	图标 ID,图标库位置由指令高字节指定。 图标固定为不显示背景色。
0x0008	区域填充	0x00	2	(x,y)	种子点坐标，x 坐标高字节为判断条件。
		0x02	1	COLOR	填充颜色。
0x0009	频谱显示（垂直线条）	0x00	1	Color0	把(X0,Y0s)(X0,Y0e)用 Color0 颜色连线，X0 高字节为判断条件。
		0x01	3	X0,Y0s,Y0e	
0x000A	线段显示	0x00	1	Color	把（Xs,Ys）（Xe,Ye）用 Color 颜色连线，Xs 高字节为判断条件。
		0x01	2	(Xs,Ys)	
		0x03	2	(Xe,Ye)	
0x000B	圆弧显示	0x00	1	Color0	圆弧显示颜色
		0x01	2	(X,Y)0	圆心(X,Y)坐标,X 坐标高字节为判断条件。
		0x03	1	RAD0	半径

		0x04	1	DEG_S0	起始角度，单位 0.5°，0-720	
		0x05	1	DEG_E0	终止角度，单位 0.5°，0-720	
0x000C	字符显示	0x00	1	Color0	字符显示颜色	
		0x01	2	(X,Y)0	字符显示位置，字符左上角坐标，X 坐标高字节为判断条件。	
		0x03H	0.5	Lib_ID	字库位置	
		0x03L	0.5	En_Mode	字符编码模式：0=8bit 1=GB2312 2=GBK 3=BIG5 4=SJIS 5=UNICODE	
		0x04H	0.5	X_Dots	字符 X 方向点阵数	
		0x04L	0.5	Y_Dots	字符 Y 方向点阵数	
		0x05	1	Text0	字符数据，对 8bit 编码，仅高字节有效。当编码方式为 01-04 时，如果字符数据为 ASCII 字符，将自动使用 0#预装字库显示。	
0x000D	矩形域 XOR	0x00	2	(x,y) s	矩形域左上角坐标，x 坐标高字节为判断条件。	
		0x02	2	(x,y) e	矩形域右下角坐标。	
		0x04	1	Color	矩形域做 XOR 的颜色，0xFFFF 将进行反色操作。	
0x000E	双色位图显示	0x00	2	(x,y) s	位图显示矩形域左上角坐标，x 坐标高字节为判断条件。	
		0x02	1	X_Dots	位图 X 方向点阵数目	
		0x03	1	Y_Dots	位图 Y 方向点阵数目	
		0x04	1	Color1	“1”bit 对应的显示颜色	
		0x05	1	Color0	“0”bit 对应的显示颜色；如果设置 Color0 和 Color1 相同，表示“0”bit 不需要显示，直接跳过。	
		0x06	N	Data_Pack	显示数据，MSB 方式；为方便用户读写数据，每行数据必须对齐到一个字，即下一行数据总是从一个新数据字（Word）开始。	
0x000F	位图显示	0x00	2	(x,y) s	位图显示矩形域左上角坐标，x 坐标高字节为判断条件。	
		0x02	1	X_Dots	位图 X 方向点阵数目	受变量空间大小限制，最大显示位图为 196*146(4:3) 或 226*126 (16:9)。
		0x03	1	Y_Dots	位图 Y 方向点阵数目	
		0x04	N	Data_Pack	显示数据，每个像素点一个字（MSB，5R6G5B 数据格式）。	
0x0010	区域放大一倍粘贴显示	0x00	2	(x,y)	放大一倍后图像粘贴在屏幕左上角坐标，x 高字节为判断条件。	待放大区域位于放大后图像区域内时，必须右下角对齐。嵌套放大可以得到更大的放大倍数。
		0x02	2	(x,y) s	待放大矩形域左上角坐标。	
		0x04	2	(x,y) e	待放大矩形域右下角坐标。	

判断条件：0xFF 绘图操作结束 0xFE 本次操作跳过（忽略）。

基本图形显示举例（以 0x0006 图片区域剪切、粘贴指令为例）：

Step1 在 14.BIN 中定义一个绘图板变量，变量*VP 指向 0x1000 地址，如下图所示：



Step2 用 SD 卡把 14.BIN 下载到 DGUS 屏。

Step3 串口向 0x1000 地址（*VP）写入 0x0006 指令相关内容（把第 3 幅页面的（100,100）（512，256）区域剪切粘贴到当前页面的（0,0）位置）。

只要 VP 内容不变，DGUS 屏将在绘图板指令所在页面执行指令，显示剪切、粘贴内容。

7.4.3 列表显示（0x22）

地址		定义	数据长度	说 明
0x00		0x5A22	2	
0x02		*SP	2	变量描述指针，0xFFFF 表示由配置文件加载
0x04		0x000C	2	
0x06	0x00	*VP	2	表格内容指针，即 TAB[TAB_X_Num][TAB_Y_Num]数组的首地址。
0x08	0x01:H	TAB_X_Num	1	列数目，0x01-0xFF
0x09	0x01:L	TAB_Y_Num	1	行数目，0x01-0xFF
0x0A	0x02:H	TAB_X_Start	1	表格起始显示列位置，0x00-0xFF。
0x0B	0x02:L	TAB_Y_Start	1	表格起始显示行位置，0x00-0xFF。
0x0C	0x03:H	Unit_Data_Num	1	➢ 0x01-0x7F 所有单元格存储数据长度相同一个单元格所占的数据空间长度（Word，字长度）。 ➢ 0x00 由*VP 指针指向的变量存储空间定义了不同列单元格的数据长度（Word，字长度）。 当 Unit_Data_Num=0x00 时，表格数据内容存储位置相应后延（TAB_X_Num/2）向上取整个字地址。 比如，*VP=0x1000，TAB_X_Num=0x07，那么： 0x1000-0x1003 依次存储了第 0-6 列的表格数据长度，其中 1003 的低字节未使用。0x1004 地址开始存储表格内容。
0x0D	0x03:L	Encode_Mode	1	.7 定义了文本显示的字符间距是否自动调整： .7=0 字符间距自动调整； .7=1 字符间距不自动调整，字符宽度固定为设定的点阵数。 .6 定义了表格内容格式 .6=0 表格内容为文本格式； .6=1 表格内容格式由单元格数据的前两个字表示，见备注[1]； .5 定义了边框线条是否显示，.5=0 显示边框，0.5=1 不显示边框。 .4 未定义，写 0。 .3-.0 定义了文本编码方式： 0=8bit 编码 1=GB2312 内码 2=GBK 3=BIG5 4=SJIS 5=UNICODE
0x0E	0x04	Xs Ys Xe Ye	8	表格显示区域定义，表格左上角、右下角坐标； 表格总是总是从左上角位置开始显示，越界将结束显示。
0x16	0x08	Color_line	2	表格边框线条颜色
0x18	0x09	Color_text	2	表格文本显示颜色
0x1A	0x0A:H	Font0_ID	1	编码方式 0x01-0x04 时 ASCII 字库位置。
0x1B	0x0A:L	Font1_ID	1	编码方式 0x00、0x05，以及 0x01-0x04 的非 ASCII 字符使用的字库
0x1C	0x0B:H	Font_X_Dots	1	字体 X 方向点阵数（0x01-0x04 模式，ASCII 字符 X 按照 X/2 计算）
0x1D	0x0B:L	Font_Y_Dots	1	字体 Y 方向点阵数目
0x1E	0x0C:H	TAB_X_Adj_Mod	1	当设置 TAB_X_Start 不为零时，进行显示表头控制： 0x00=首列不显示； 0x01=首列显示。
0x1F	0x0C:L	TAB_Y_Adj_Mod	1	当设置 TAB_Y_Start 不为零时，进行显示表头控制： 0x00=首行不显示； 0x01=首行显示。

备注[1]:当 Encode_mode.6=1 时，每个单元格数据内容的前两个字定义了表格数据格式，说明如下：

第 1 个字高字节:Mode 选择数据类型；

0x00=整数(2 字节)， -32768 到 32767
0x01=长整数(4 字节) -2147483648 到 2147483647
0x02=*VP 高字节,无符号数 0 到 255
0x03=*VP 低字节，无符号数 0 到 255
0x04=超长整数(8 字节) -9223372036854775808 到 9223372036854775807
0x05=无符号整数(2 字节) 0 到 65535
0x06=无符号长整数(4 字节) 0 到 4294967295
0x10=时间格式 1，12:34:56 BCD 码串
0x11=时间格式 2，12-34-56 BCD 码串
0x12=时间格式 3，YYYY-MM-DD HH:MM:SS BCD 码串
0xFF=文本格式

第一个字低字节：

Mode=0x00-0x06 定义了变量数据的定点显示格式，高 4bit 表示整数位数，低 4bit 表示小数位数。
Mode=0x10-0X11 时间 BCD 码串的字节长度
Mode=其它 无定义

第 2 个字：定义单元格文本颜色。

如果表格实际内容短于 Unit_Data_Num 规定的长度时，使用 0xFFFF 做为单元格文本结束符。

对于特别大的表格，通过触摸屏修改 TAB_X_Start、TAB_Y_Start 值可以很方便的实现表格的定位和拖动。

8 DGUS 屏应用问答（FAQ）

➢ DGUS 如何简单？

举个例子：显示里面最麻烦的就是示波器了，基于 DGUS 开发示波器，用户单片机唯一要做的就是通过串口把 A/D 采集的数据送给迪文屏，其它的，比如曲线缩放、上下平移都可以用 DGUS 开发出来，不涉及单片机代码。

➢ 组态方式开发人机界面，快是快，但是做出来东西千篇一律缺少特点？

DGUS 的组态开发方式和传统人机界面根本区别在于：迪文屏有 256MB（最大可以扩展到 2GB）存储器，图形数据库是客户自定义的。意味着只要用 PS 能够设计出来的，迪文屏都可以支持，可以充分展示用户的创意。

➢ 相比传统 HMI，DGUS 的典型特点？

迪文 DGUS 和传统 HMI 的最大区别在于软件平台，传统 HMI 采用通用操作系统来设计，比如 WinCE、Linux、Andriod 等，而迪文 DGUS 是迪文自己独有的、固化在硬件中的专用软件，其典型特点是：

（a）可靠性、稳定性好，抗干扰能力强；
（b）没有版权费用导致性价比高；
（c）可以有效的保护用户知识产权，不会出现同行山寨的恶性竞争；
（d）开关机时间极短，并且可以随时开关机；
（e）始终表现如一，不会出现运行一旦时间后（因为内存需要清理）越跑越慢；

➢ 相比传统的液晶屏或者串口指令屏，DGUS 的典型特点？

迪文 DGUS 实质是硬件化的 GUI 平台，相比传统的液晶屏或者串口指令屏，其典型特点就是二次开发门槛低，开发质量高，生产、维护简单，并且很容易在用户通用的硬件平台上形成系列化产品。

➢ DGUS 系统的速度有多快？还需要用户单片机判忙吗？

DGUS 变量显示最小延迟是 80mS，也就是说 1 秒钟变量显示最少可以变化 12 次，能够完全满足实时性的要求。DGUS 采用了全新的设计思路，串口缓冲区不会溢出，用户不再需要判忙，DGUS 屏也没有 BUSY 信号输出。

➢ DGUS 一个页面的最多只能显示 128 个变量是不是不够用？

DGUS 的变量已经高度抽象化（比如 1 条曲线显示就是一个变量），再加上图形变量包含的信息量大，一般的应用，一个页面也就 10 来个变量了不起了：比如做个温控仪，真正的变量一共也就 4 个（当前温度、设定温度、报警上下限）。

另外，很多客户可能把键盘按钮也当做变量，触摸按钮在 DGUS 中是单独用触控文件来描述的，页面可以放置的触控按钮数量是没有限制的，不占用变量资源。

➢ DGUS 如何把当前显示屏幕内容打印到打印机上？

DGUS 上的 DWIN OS 平台内嵌了标准打印机驱动，可以直接驱动串口打印机打印指定区域屏幕内容。

➢ MODBUS 设备或者 PLC 能不能直接接 DGUS 屏？

可以，但需要借助 DGUS 屏内嵌的 DWIN OS 做一个简单的接口程序，迪文网站可以下载相关应用案例。

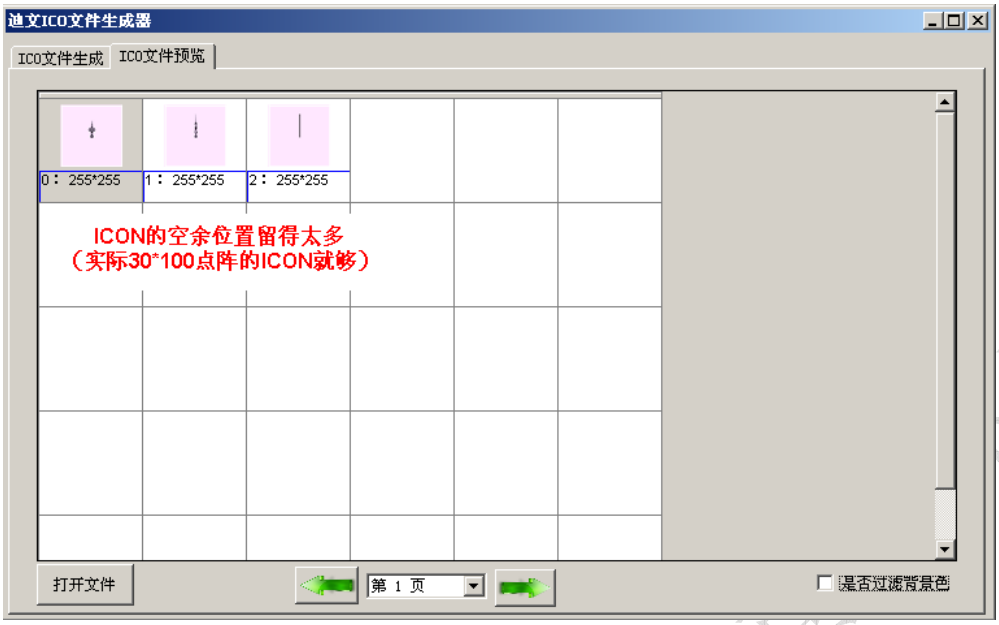
➢ DGUS 的变量存储区，要想上电时不是 0x0000 怎么办？

在 CONFIG.TXT 文件中，把 R2 寄存器的第 2 位（0x04，L22_EN）置位（R2=04）；
同时设计一个需要的变量初始化数据文件，命名为 22**.bin。
用 SD 卡下载 22**.bin 和 CONFIG.TXT 文件到迪文屏。
之后再上电时，DGUS 会自动把 22 号字库文件的前 56K 字数据加载到变量存储区做为初始化数据。

➢ 为什么我的 DGUS 操作起来感觉比较卡（反应迟钝、图标动画不连贯）？

这是 DGUS 因为处理数据量过大，处理不过来所致，原因可能有以下 3 个：

（1） 使用透明 ICON 显示，但是 ICON 设计时空余位置留得太多，导致艺术字显示、时钟表盘显示、滑块这些以 ICON 处理为主的指令处理慢。



改进办法：设计 ICON 时，尽可能去掉不需要的空余位置。

（2） 使用弹出键盘时，弹出区域选择过大（比如在 1024×768 的屏上弹出一个 800*600 的键盘），结果导致实时处理信息量大而影响显示速度。

改进办法：把弹出菜单区域选择得尽可能接近实际需要的区域大小。

（3） 客户一个页面显示的变量很多，导致处理任务多，处理慢。这种情况极少见，如果有，可以通过正确配置 R2 寄存器，提高 DGUS 的处理速度来解决，如下表所示：

DGUS 延时（处理能力）	R2.1	R2.0	说　明
200mS（100%）	0	0	标准模式
160mS（80%）	0	1	高速模式 1
120mS（60%）	1	0	高速模式 2
80mS（40%）	1	1	高速模式 3

➢ 我想在应用中实现一个用户软件控制弹出的报警菜单，菜单上设置有关闭按钮，操作者可以按钮关闭弹出菜单，用 DGUS 如何实现？

可以这么实现：

（1） 把弹出的报警菜单设置成一个 ICON，用变量图标显示（0x5A00），设置一个“ 报警变量 ”来控制 ICON 是否显示。

（2） 在每页的变量图标显示位置预先设计好触控，用按键值返回（0xFE05）来回传按钮值，保存在“ 按键变量 ”中；

（3） 当用户软件改变“ 报警变量 ”值时，即可实现报警菜单是否弹出；

如果操作者按了“ 关闭按钮 ”位置，用户软件会检查到“ 按键变量 ”值，根据“ 报警变量 ”的值，即可知道是不是有效按钮（弹出菜单后的按键），然后决定是否关闭显示。

➢ 参数录入状态下，长时间没有录入（或临时需要显示其它任务）需要用户 CPU 主动结束录入过程怎么办？

在 13.BIN 触控文件中相应录入界面增加一条“ 键控 ”退出的触控指令，当需要退出时，用户 CPU 发送一个“ 退出 ”键码到 0x4F 寄存器就让 DGUS 结束录入，自动退出了。

➢ DGUS 的触摸屏处理为了防止误操作，在一直按住触摸屏时，0x06 触摸屏状态寄存器要在按压 0.5 秒后再变成 0x03 状态，我应用中希望一按下就马上开始调节参数，DGUS 如何实现？

方法 1：用户代码稍加调整

```
READREG(0x06,0x01,Reg)      //定时读取触摸屏按压状态
if(Reg==0x01||Reg==0x03)
TPOK=1;
if(Reg==0x02)
TPOK=0;
if(TPOK)
P++;                        //如果触摸屏按压，就调整变量参数
```

方法 2：把上面这段代码用 DGUS 屏后台的 DWIN OS 代码实现。

➢ DGUS 屏功耗大，并且上面有 DC/DC，电流噪声大，对我的 A/D 采集带来了干扰，应该如何解决？

（a）在馈电设计上，一定注意 分开走线 供电，只在供电电源处集中；

（b）实在不便于分开走线供电的，一定注意先给干扰大的设备（比如 DGUS 屏）馈电，把信噪比要求高的电路放在最末级馈电；

（c）在给 DGUS 屏供电时，电源上串联一个 2.2mH 1A（取决于 DGUS 屏功率）以上的功率电感来平滑电流，减小电流噪声。

➢ 为什么我的 DGUS 屏上电后一闪一闪或者闪烁几次才能正常工作？如何解决？

这是因为给 DGUS 屏供电的电源功率储备不够，比如内阻（包括线路内阻）大、输出限流点偏低等。

可以通过在 DGUS 屏这侧并联一个大容量电解电容降低交流（动态）内阻来解决，电容按照下式计算：

C=1250/(DGUS 屏供电电压-DGUS 屏标称的最低工作电压) uF

不同供电电压下的电容选取如下表所示（供电电压是从 DGUS 屏上 VCC 接口处量得的电压值）：

供电电压（V）	DGUS 屏标称最低工作电压（V）	电容值（uF）	推荐值
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

➢ 我买的 DGUS 屏标称工作电压范围是 7-42V，电流是 300mA@12V，我现在应用中是一个 18V 0.5A 的本安电源通过 500 米电缆（电缆内阻 10R）来给屏供电，请问是否可以正常工作？

第一步，计算电源功率是否满足要求：

电源功率=18*0.5=9W 最小负载功率=（12*0.3）*2=7.2W 7.2<9 电源功率满足要求

第二步，计算最小负载功率点的电压是不是在 DGUS 屏正常工作范围

最小负载功率点的电压为 Vcc/2=9V 在 7V-42V 之间；

最小负载功率点的最大电流为 Vcc/(2*R)=900mA 低于 3.6/9=400Ma，这种工况下 DGUS 屏可以正常工作。

工作点（工作电流）的准确计算如下：

I=（V-Sqrt（V*V-4*R*P））/(2*R) V 为电源电压，R 为线路内阻，P 为负载功率。

代入 V=18 R=10 P=3.6 计算出 I=0.23A，计算此时屏的实际电压=18-0.23*10=15.7V。

➢ 我能不能向迪文定制特殊要求的 DGUS 软件功能？

软件功能的定制需要高人力成本的研发投入，只要订单的利润（大批量购买或者额外付一笔研发费用）足够支持这种成本投入，迪文很乐意和客户实现共赢。

事实上，DGUS 的功能已经相当完善，很多特殊要求也可以联络迪文应用工程师通过后台 DWIN OS 免费实现。

9 DGUS 特殊应用软件使用说明

9.1 基于 Modbus 接口的 DGUS 软件应用说明

- 升级程序名称：DGUS_V67_MODBUS.BIN，对应的 DGUS 版本是 V67，支持 K600+或 DGUS 内核。
- 主要功能：用户接口指令集为 Modbus RTU 主机模式，不再支持标准的 DGUS 指令集（包括 DWIN OS 仿真，串口图片、字库下载）。DWIN OS 中，以下指令不再支持：

指　令	说　明
RMODBUS	到 COM0_Rx_FIFO 读取 MODBUS 数据帧
COMSET	COM1 串口配置
COMTXD	串口数据发送的发送到 COM1
CPRTS	串口打印
RDXLEN	检查 COM0_Rx_FIFO
RDXDAT	读取 COM0_Rx_FIFO
COMTXI	直接串口发送
RD645	到 COM0_Rx_FIFO 读取 DL/T645 数据帧

- 通过 22.BIN 字库文件的 20KW-28KW（16KB 空间大小，文件字节地址 0x0A000-0x0DFFF）来定义 Modbus 运行参数，DGUS 上电会自动把这 8KW 配置文件解码加载到变量缓冲区的最后 8KW，定义如下表。

DGUS 变量地址	定义	说明
0x5000	Modbus 启用标记	0x5AA5 表示启用 Modbus 通信。
0x5001:H	保存配置文件标记	0x5A：保存 DGUS 变量缓冲区的 Modbus 配置文件到 22 字库。
0x5001:L	加载配置文件标记	0x5A：从 22 字库加载 Modbus 配置文件到 DGUS 变量缓冲区。
0x5002	串口波特率	3.1 格式定点小数(115200bps=0x480)，单位为 kbps，最大 999.9kbps。
0x5003:H	串口模式	0x00=8N1（无校验） 0x01=8E1（偶检验，EVEN） 0x02=8O1（奇校验，ODD） 0x03=8N2（无校验，2 个停止位）
0x5004:H	变量写标记	0x5A 表示保存指定 DGUS 变量空间（PS:00-PE:00）到 22 字库。
0x5004:L	变量读标记	0x5A 表示从 22 字库相应位置读数据到 DGUS 指定变量空间。
0x5005	变量读写起止地址	PS:PE PS、PE 是 DGUS 变量空间开始、结束地址高字节，低字节固定为 0。
0x5006-0x5007	保留	写 0x0000
0x5008-0x500F	第 1 条 MODBUS 指令（16 字节）	0x**00**(0x5008H)：0x5A=本条指令有效，其它=本条指令无效 0x**01**(0x5008L)：读写的 Modbus 设备地址 0x**02**(0x5009H)：读/写使用的 Modbus 指令 0x**03**(0x5009L)：读写数据长度，0x00 表示本条指令无效 0x**04**(0x500A)： 本条指令处理定时时间，包含指令传送时间，4 位整数，单位为 mS，最大 9999mS。对于读指令，定时时间是从机响应的最长时间。 0x**06**(0x500B)： 4 个字节规定了 Modbus 读写指令的发送方式 0x0000:**** 所有页面下均执行指令 0x0001:Page_ID 仅在指定页面下执行指令 0x0002:VP 仅在 VP 指向的变量缓冲区低字节内容为 0x5A 才执行指令，所有相关指令执行完后自动清零 VP 指向的内容。 0x**0A**(0x500D)：本条指令读写数据在 DGUS 屏变量存储区的起始地址。如果地址高字节为 0xFF，表示读取的数据讲写入 DGUS 曲线缓冲区，此时低字节地址表示为曲线数据格式。 0x**0C**(0x500E)：本条指令读写的数据在 MODBUS 设备上的数据起始地址。 0x**0E**(0x500FH)：总线通信状态反馈，写指令始终为 0x00，读指令返回 0x00（失败）/0xFF(成功)。 0x**0F**（0x500FL）：保留，写 0x00。
……		
0x6FF8-0x6FFF	第 1023 条指令	最多支持 1023 条 Modbus 指令

MODBUS 指令操作对应表（注意，Modbus 的位（线圈）变量是按照 LSB 定义的，而 DGUS 是按照 MSB 定义）

MODBUS 指令	功　能	读写数据长度	MODBUS 起始地址
0x01	读取输入线圈状态	线圈个数/8	起始线圈位置
0x02	读取输入位变量状	位变量个数/8	起始输出位置
0x03	读取保存寄存器数据	寄存器个数*2	保存寄存器首地址
0x04	读取输入寄存器数据	寄存器个数*2	输入寄存器首地址
0x05	强置单个线圈	0x02	线圈地址
0x06	预置单个寄存器	0x02	寄存器地址
0x07	读取异常状态	0x01	任意值
0x0F	强置多个线圈	线圈数量	起始线圈位置
0x10	预置多个寄存器	寄存器个数*2	寄存器首地址
0x11	读取从机标示	从机标示字节数	任意值

10 DGUS 屏开发典型程序参考（ASM51、C51）

;DGUS 屏用户程序典型架构，GUI 和用户程序仅通过变量打交道，这样就可以方便的把串口数据交换放在后台中断处理
;为了举例方便，假设一个简单温控仪，通过 DGUS 屏设置温度，显示当前温度值和实时曲线
;这个程序只有不到 0.3KB（ASM51，C51 为 0.7KB）代码，256B RAM，通用的 8051 平台都可以运行。
;其它的应用，可以移植这个程序架构（灰度背景是和 DGUS 屏通信的核心代码），补充不同的用户应用层代码就可以了

//ASM51 代码

```
      $INCLUDE(REG52.H)
    JDQ1 BIT  P1.0                  ;温度控制继电器
    SYSFLG    EQU  20H
         RCVDATA   BIT  SYSFLG.7
         RCVOK     BIT  SYSFLG.6
    ;30-57 区用户可以自己定义
    RX5A      EQU  58H              ;DGUS 接收数据帧同步字符串
    RXA5      EQU  59H
    RXLEN     EQU  5AH
    RXCMD     EQU  5BH
    RXADRH    EQU  5CH
    RXADRL    EQU  5DH
    RXDLEN    EQU  5EH
    TXTIME    EQU  5FH              ;定时向 DGUS 读写数据的时间间隔，单位 10mS
    CFGBUF    EQU  60H              ;配置参数区（DGUS 0x83 指令）60-7FH 32B,对应 DGUS 0x1000-0x100F 地址
         TSET_H    EQU  CFGBUF      ;设定温度在 0x1000
         TSET_L    EQU  (CFGBUF+1)
    TXBUF     EQU  80H              ;发送给 DGUS 屏的数据读取指令和写指令缓冲区
              ;80H-86H 0x83 读 32B 数据指令
              ;87H-8DH 0x84 写曲线缓冲区指令
         LINE0_DH EQU  (TXBUF+12)   ;曲线显示数据
         LINE0_DL EQU  (TXBUF+13)
              ;8EH-92H 0x82 写 64B 数据指令
    DSPBUF    EQU  94H              ;显示变量区（DGUS 0x82 指令）94-D3H 64B,对应 DGUS 0x0000-0x001F 地址
         TNOW_H    EQU  DSPBUF      ;当前温度在 0x0000
         TNOW_L    EQU  (DSPBUF+1)

          ORG     0000H
          LJMP    RESET

         ORG      000BH
         LJMP     T10MS_INT         ;定时器中断

         ORG      0023H
         LJMP     UART_INT          ;串口中断

        ORG     0040H
RESET:  CLR     EA
        MOV     SP,#0D4H
        ORL     PCON,#80H           ;配置串口波特率为 115200bps，22.1184MHz 晶体
        MOV     SCON,#50H
        MOV     TMOD,#21H
        MOV     TH1,#255
        MOV     TL1,#255
        SETB    TR1
        MOV     TH0,#HIGH(47104)    ;10ms 定时器中断
        MOV     TL0,#LOW(47104)
        SETB    TR0
        CLR     JDQ1
        LCALL   INITHMI
        SETB    ES
        SETB    ET0
        SETB    EA
        JNB     RCVOK,$             ;上电读取到数据再开始运行，有 WDT 的注意复位 WDT
  START:LCALL   RDTMP               ;测量当前温度到 R7:R6
        MOV     R0,#TNOW_H          ;显示当前温度值
```

```
        MOV     A,R7
        MOV     @R0,A
        INC     R0
        MOV     A,R6
        MOV     @R0,A
        MOV     R0,#LINE0_DH        ;显示当前温度曲线
        MOV     A,R7
        MOV     @R0,A
        INC     R0
        MOV     A,R6
        MOV     @R0,A
        MOV     R0,#TSET_H          ;读取温度设置值到 R5:R4
        MOV     A,@R0
        MOV     R5,A
        INC     R0
        MOV     A,@R0
        MOV     R4,A
        CLR     C                   ;简单的开关控温
        MOV     A,R4
        SUBB    A,R6
        MOV     A,R5
        SUBB    A,R7
        MOV     JDQ1,C
       LJMP     START

;测量温度程序，用户自己添加，假设测量值在 R7:R6
RDTMP:MOV       R7,#HIGH(500)       ;显示 50.0
        MOV     R6,#LOW(500)
        RET

;初始化和 DGUS 屏相关的变量
INITHMI:MOV     SYSFLG,#00H
        MOV     TXTIME,#100         ;上电 1 秒后开始数据读写
        MOV     R0,#TXBUF
        MOV     DPTR,#CMD_83
        LCALL   RDROM
        MOV     DPTR,#CMD_84
        LCALL   RDROM
        INC     R0
        INC     R0
        MOV     DPTR,#CMD_82
        LCALL   RDROM
        RET
CMD_83:DB 5AH,0A5H,04H,83H,10H,00H,10H,0FFH ;7B，读 DGUS 屏 0x1000 开始的 16 字
CMD_82:DB 5AH,0A5H,43H,82H,00H,00H,0FFH     ;6B，写显示数据到 DGUS 屏 0x0000 开始的 32 字
CMD_84:DB 5AH,0A5H,04H,84H,01H,0FFH         ;5B，写通道 0 曲线缓冲区

;读取@DPTR 开始的数据到@R0 的位置，遇到 0xFF 结束
RDROM:CLR       A
        MOVC    A,@A+DPTR
        CJNE    A,#0FFH,RDROM1
        RET
RDROM1:MOV      @R0,A
        INC     R0
        INC     DPTR
        SJMP    RDROM

;10mS 定时器中断，用于定时 200mS 发送指令给 DGUS 屏
    T10MS_INT:PUSH ACC
        PUSH    PSW
        CLR     TF0
        MOV     TH0,#HIGH(47104)
        MOV     TL0,#LOW(47104)
```

```
        DJNZ    TXTIME,T10MSEND
        MOV     TXTIME,#20
        SETB    RS0
        SETB    RS1
        MOV     R1,#TXBUF
        MOV     R6,#84
        CLR     TI
        MOV     SBUF,@R1            ;启动数据发送
T10MSEND:POP    PSW
        POP     ACC
        RETI
```

```
    ;串口中断方式处理和 DGUS 屏的数据交换，从 DGUS 屏的 1000-100F 存储区读取 32Byte 数据保存到单片机的 CFGBUF
    ;把单片机的 DSPBUF 的 64B 内容写到 DGUS 屏的 0000-001FH 存储区
    UART_INT:PUSH ACC
        PUSH    PSW
        SETB    RS1                 ;固定使用 BANK3 R0-R7 作为串口中断专用
        SETB    RS0                 ;R0，R7 是 Rx 指针  R1，R6 是 Tx 指针
        JB      RI,UARTRCV
        CLR     TI
        DJNZ    R6,UARTTX
        SJMP    UARTEND
    UARTTX:INC  R1
        MOV     SBUF,@R1
        SJMP    UARTEND
    UARTRCV:MOV A,SBUF
        CLR     RI
        JB      RCVDATA,UARTR1
        MOV     RX5A,RXA5
        MOV     RXA5,RXLEN
        MOV     RXLEN,RXCMD
        MOV     RXCMD,RXADRH
        MOV     RXADRH,RXADRL
        MOV     RXADRL,RXDLEN
        MOV     RXDLEN,A
        MOV     A,RX5A              ;检查帧同步数据 5A A5 24 83 10 00 10 +32B DATA
        CJNE    A,#5AH,UARTEND
        MOV     A,RXA5
        CJNE    A,#0A5H,UARTEND
        MOV     A,RXLEN
        CJNE    A,#24H,UARTEND
        MOV     A,RXCMD
        CJNE    A,#83H,UARTEND
        MOV     A,RXADRH
        CJNE    A,#10H,UARTEND
        MOV     A,RXADRL
        CJNE    A,#00H,UARTEND
        MOV     A,RXDLEN
        CJNE    A,#10H,UARTEND
        SETB    RCVDATA
        MOV     R0,#CFGBUF
        MOV     R7,#32
        SJMP    UARTEND
    UARTR1:MOV  @R0,A
        INC     R0
        DJNZ    R7,UARTEND
        CLR     RCVDATA
        SETB    RCVOK
    UARTEND:POP PSW
        POP     ACC
        RETI
```

```
END
```

```
// C51 代码
#include <REG52.h>
sbit JDQ1=P1^0;    //温度控制继电器

//串口变量
unsigned char RCVOK,RCVDATA,RX5A,RXA5,RXLEN,RXCMD,RXADRH,RXADRL,RXDLEN,TXTIME;
unsigned char TX_P,RX_P;   //收发缓冲区的指针位置

//配置参数区（DGUS 0x83 指令）32B,对应 DGUS 0x1000-0x100F 地址
unsigned char CFGBUF[32];

//发送给 DGUS 屏的数据读取指令和写指令缓冲区
//TXBUF[12]:TXBUF[13]用于显示曲线 CH1 接口
//TXBUF[20]-TXBUF[84]0x82 写 DGUS 屏的 64B 变量数据，对应 DGUS 0x0000-0x001F 地址
unsigned char TXBUF[84]={0x5A,0xA5,0x4,0x83,0x10,0x0,0x10,
                         0x5A,0xA5,0x4,0x84,0x1,0x0,0x0,0x5A,0xA5,0x43,0x82,0x00,0x00};

void main(void)
{    int i,j;
     PCON=PCON|0x80;         //配置串口波特率为 115200bps，22.1184MHz 晶体
     SCON=0x50;
     TMOD=0x21;
     TH1=255;
     TL1=255;
     TR1=1;
     TH0=0xb8;      //;10ms 定时器中断
     TL0=0x00;
     TR0=1;
     JDQ1=0;
     JDQ1=0;        //关闭加热继电器
     RCVOK=0;       //清除数据接收 OK 标记
     RCVDATA=0;
     TXTIME=100;  //上电 1 秒后开始数据读写
     ES=1;
     ET0=1;
     EA=1;
     while(RCVOK==0x00);
     //用户测量和控制温度，显示温度值和实时温度曲线
    while(1)
 {   i=Readtemp();          //测量当前温度
     j=CFGBUF[0]*256+CFGBUF[1];  //设定温度，DGUS 0x1000 地址，整形变量
     if(i<j)
     {JDQ1=1;}
     else
     {JDQ1=0;}
    TXBUF[20]=i/256;    //显示当前温度值，DGUS 0x0000 地址，整形变量
     TXBUF[21]=i%256;
    TXBUF[12]=i/256;    //显示当前温度曲线，CH1 曲线通道
     TXBUF[13]=i%256;}
}

//温度测量函数
int Readtemp( )
{return(500);}

 //定时器 2 中断函数,10mS 定时器中断，用于定时 200mS 发送指令给 DGUS 屏
void Timer0_interrupt(void) interrupt 2
{    TF0=0;
     TH0=0xb8;
     TL0=0x00;
     TXTIME--;
     if(TXTIME==0x00)
     {TXTIME=20;
```

```
      TX_P=0;
      TI=0;
      SBUF=TXBUF[0];}}

  //串口中断方式处理和 DGUS 屏的数据交换
  //定时从 DGUS 屏的 1000-100F 存储区读取 32Byte 数据保存到单片机的 CFGBUF
  //定时把单片机的 DSPBUF 的 64B 内容写到 DGUS 屏的 0000-001FH 存储区
void Uart_interrupt(void) interrupt 4
{ unsigned char i;
if(RI)        //接收中断
 {  i=SBUF;
    RI=0;
     if(RCVDATA==0)
   { RX5A=RXA5;
      RXA5=RXLEN;
      RXLEN=RXCMD;
      RXCMD=RXADRH;
      RXADRH=RXADRL;
      RXADRL=RXDLEN;
      RXDLEN=i;
     //检查帧同步数据 5A A5 24 83 10 00 10 +32B DATA
     if((RX5A==0x5A)&&(RXA5==0xA5)&&(RXLEN==0x24)&&(RXCMD==0x83)&&(RXADRH=0x10)&&(RXADRL=0x00)&&(RXDLEN=0x10))
   { RCVDATA=0xff;
      RX_P=0;}}
    else
   { CFGBUF[RX_P]=i;
     RX_P++;
      if(RX_P==32)
   { RCVDATA=0;
     RCVOK=0xff;}}
}
 if(TI)   //发送中断
 { TI=0;
   TX_P++;
   if(TX_P<84)
 { SBUF=TXBUF[TX_P];}}
}
```

附录 修订记录

日 期	修订内容	DGUS 版本
2012.04.27	首次发布	V1.0
2012.05.25	1.在 CONFIG.TXT 配置文件中增加对 R2.2 的定义，允许从 22#字库加载变量初始化数据； 2.0x5A12 RTC 显示中增加了指针表盘时钟显示功能； 3.增加 0xFE07_02 打印位图处理触控指令，实现屏幕内容打印； 4.增加 0xFE07_03 变量数据发送到串口指令，支持无线遥控等应用。	V1.1
2012.06.08	1.修订基本图形显示指令 0x5A21 中 0xFF 结束符定义：由结束整个指令变更为结束本次操作； 2.增加了 0x5A21_08 封闭区域填充指令； 3.增加了 0x5A23_01 圆域交集显示指令，适应特殊行业防碰撞显示应用； 4.在寄存器空间（0x05-0x0B）增加了触摸屏坐标读取和触控使能控制功能，以方便设备运行时锁触摸屏、防止误操作的应用。	V1.2
2012.06.28	1.在 0xFE00 变量录入指令增加了字节（byte）类型数据返回和上下限设置； 2.在 0x5A10 数据变量显示中，增加了字节（byte）类型数据的显示； 3.增加了 SD 卡升级 DGUS 软件的功能; 4.在 0x5A21 基本图形显示中定义了判断条件：0xFF 为结束绘图操作，0xFE 为跳过本次操作； 5.在 0x5A21 基本图形显示中定义了垂直线条显示指令，用于频谱、刻度线的显示。	V2.0
2012.07.06	1.文档中增加指令说明，增加应用实例； 2.应用实例中加入指令说明； 3.增加 0x001F 寄存器定义，以允许用户通过串口修改 RTC。	V2.0
2012.07.12	增加 0x0040-0x0049 寄存器定义，允许用户在变量存储器和字库之间交换数据，可实现以下功能： ● 历史数据的存储和记录，最大 8MW（16MB）空间； ● 把配置参数等需要掉电保存的数据保存到字库空间，并可以方便的加载到变量存储器操作； ● 对字库内容的读取； ● 由于字库内容可以通过 SD 卡写入，间接实现了用户软件对 SD 卡数据的读取。	V2.2
2012.08.08	通过配置 CONFIG.TXT 文件： ● 增加 SD 卡升级禁止功能； ● 增加了基于 DWIN OS 的用户应用软件（23.BIN）执行功能。	V3.0
2012.08.22	增加了对键盘的支持，方便不使用触摸屏的用户快速开发 HMI，实现方法如下： ● 13 触控配置文件中触摸按钮位置换成键码（0x01-0xFF，最多 255 个键）； ● 用户键盘接口定义在 0x4F 寄存器。 用户软件只需要把键码通过串口传到 0x4F 寄存器，即可让 DGUS 按照 13 文件要求实现界面控制。	V3.2
2012.08.30	对 14.BIN 变量配置文件的*VP，增加了高字节为 0xFF 可以取消本条变量显示指令的功能以方便显示指令的用户控制开/关。 在 0xFE06 文本录入指令中，增加了返回输入结束标记和有效数据长度的定义选择。	V3.3
2012.09.20	1.增加了对位（bit）变量的支持，实现和 DCT100 基本 I/O 设备的无缝连接。 ● 0xFE01、0xFE02、0xFE05 参数录入中，增加了位（bit）变量录入功能； ● 增加了 0x5A06 位变量图标显示功能。 2.增加 0x5A13 BCD 码时间变量显示功能。 3.对 0x5A11、0x5A22 文本显示增加了字符间距自动调整 开/关 功能。定义 Encode_Mode.7 来控制是否启用字符间距自动调整。	V3.4
2012.09.27	在 0xFE06 触摸屏输入法的 ASCII 录入中，增加了录入过程中显示"*"的选择，用于文本密码输入。	V3.5
2012.11.05	1.修订了 CONFIG.TXT 文件 R2 寄存器（SYS_CFG 配置字）低两 bit 的定义，以设置 DGUS 周期为 200mS/160mS/120mS/80mS，提高 DGUS 响应的实时性。 2.0x5A10 数据变量显示中，增加了对 64 位整数、32 位无符号数、16 位无符号数的支持； 3.0xFE00 变量数据录入中，增加了对 64 位整数的支持； 4.SD 卡接口禁止后将不能校准触摸屏。	V4.3
2012.11.12	为方便用户现场应用，增加了以下功能： 1.增加了 SD 卡 CONFIG.TXT 文件触发触摸屏校准操作的指令：TP_CORRECT； 2.增加了 SD 卡 CONFIG.TXT 文件来输入密码取消 SD 卡禁止的指令：SD_UNLOCK_8 位密码； 3.PC 组态软件可以通过串口下载文件（字库、配置文件、用户程序），实现在线调试功能。	V4.5
2012.11.20	1.5A03 艺术字显示增加了右对齐功能; 2.5A21 绘图指令增加了 0x000A 线段显示指令； 3.增加了上电运行时间，保存在寄存器空间 0C-0F 寄存器。	V4.7
2012.12.28	1.在寄存器空间（0xEE-0xEF）增加了 DGUS 屏复位触发寄存器； 2.0x5A03 艺术字显示增加了对超长整数等变量类型的支持； 3.对 0x5A22 列表显示增加了对 HEX 单元格数据，以及单元格数据显示颜色指定的支持； 4.增加了 RC_AUX_CFG 辅助配置寄存器的定义，支持对触摸屏蜂鸣器伴音的关断/开启。	V4.9
2013.01.18	1.5A21 绘图指令增加了 0x000B 线段显示指令； 2.对 0x5A22 列表显示增加了对 BCD 格式时间数据的支持； 3.对 5A21 绘图指令增加了字符显示指令。	V5.0
2013.03.25	1.页最大显示变量数目由 64 个增加到 128 个（加上 OS 控制的 32 个，最大页显示变量可以达到	V5.3

	160 个），通过 RC AUX_CFG.4 位定义来选择页变量数目（64 或 128 个/页）。 2.5A21 绘图指令增加了 0x000D 区域 XOR 指令用于区域的渲染； 3.5A21 绘图指令增加了 0x000E 双色图显示、0x000F 位图显示指令用于实时图标的显示； 4.去掉了 0x5A23 特殊行业应用显示指令，变更为根据行业需求提供更贴近的软件 ODM 服务； 5.为方便远程维护，增加了串口下载字库、图片功能。	
2013.04.02	1.在 0xEB 寄存器增加了曲线缓冲区清空功能的定义； 2.DWIN OS 中增加了 64bit 无符号数平方根计算指令 SQRT；	V5.5
2013.05.18	1.在 GBK 输入法中增加了“ 注音输入法 ”，以支持台湾地区的繁体中文录入； 2.DWIN OS 中增加了对输入法缓冲区增加字符串的指令 SCANADD，以方便模糊或者联想输入应用； 3.0xFE02 增量调节增加对 0x1C 变量的定义，以支持按压触摸屏后单步/连续调节的选择。	V5.6
2013.05.21	5A21 绘图指令增加了对虚线、点划线线段显示的支持。	V5.7
2013.08.16	1.5A21_05 绘图指令增加了对负数圆心坐标的支持； 2.在 0x50-0x54 寄存器增加了音乐播放接口，用于控制播放板载的 128 段音乐信息。	V5.8
2013.11.22	1.在 0x56-0x5F 寄存器增加了数据库操作功能，数据库和图形存储空间重合，最大支持 960MB 数据库，可以用 SD 卡导出； 2.取消了 32-128 号字库 SD 卡导出功能； 3.DWIN OS 下的 MOVXL 指令增加了对用户数据库读写支持； 4.5A21 绘图指令中增加了指定区域放大（5A21_10）指令。	V6.0
2014.01.06	1.增加了 RC.3 配置寄存器定义，用于 关闭/打开 CRC 校验应答； 2.增加了触摸屏 5 点校准模式，通过 RC 寄存器的 RC.2 来选择； 3.增加了串口指令校准触摸屏功能，通过向 0xEA 寄存器写 0x5A 来实现。	V6.2
2014.03.05	在 13.BIN 触控文件中增加了按键伴音的支持。	V6.3
2014.10.08	Modbus 专用接口 DGUS 软件发布，增加了“ 9.1 基于 Modbus 接口的 DGUS 软件应用说明 ”。	V6.3
2014.11.22	1.升级内核硬件，在 0x1E 寄存器增加了当前背光亮度状态返回； 2.升级内核硬件，增加了触摸屏手势识别功能； 3.增加了 5A14 文本滚屏显示指令。	V6.5
2015.01.13	1.触控模式增加了 0xFE08：触摸屏按压同步数据返回功能。主要用于“ 点动 ”和参数设置时数据的自动保存和加载。 2.升级 Modbus 专用软件到 V67 版本。	V6.7

使用本文档或迪文 DGUS 屏过程中如存在任何疑问，或欲了解更多迪文 DGUS 屏的最新信息，欢迎 mail 到 dwinhmi@dwin.com.cn 联系我们。

感谢大家一直以来对迪文的支持，您的支持是我们进步的动力！

谢谢大家！

北京迪文科技有限公司是全球领先的从事工业串口屏技术研发和应用拓展、产品生产和营销的高新技术企业。公司总部位于被称为中国“硅谷”的北京中关村核心区，生产基地位于迪文湖南科技园，在北京、广州、上海、武汉设立区域营销和应用支持中心。深耕国内市场的同时，迪文也积极参与全球竞争，在北欧和美国设有驻外办事处，在以色列、韩国、印度、巴西等地建立了销售渠道。

自 2003 年成立至今，秉承“专业素养、诚实守信、追求卓越”的经营理念，强调为客户创造价值，实现共赢，使公司获得了快速发展和壮大。依托产品在功能、品质、服务、价格上的领先和竞争优势，迪文屏在工业用 TFT 显示屏市场，尤其在注重可靠性的工程机械、医疗仪器、出口机电产品上占据重要地位，赢得了众多客户的信任和支持。

迪文始终坚持为客户创造价值，专注科技创新，力求与人双赢、共同成长。

北京迪文科技有限公司

地址：北京市海淀区知春路 108 号豪景大厦 A 座 9 层
邮编：100086
电话：400 018 9008
传真：010-62628562
网站：www.dwin.com.cn
邮箱：dwinhmi@dwin.com.cn
企业 QQ：400 018 9008