

〈变频器项目〉

液晶键盘通讯（SCI）模块

软件详细设计说明书

拟制人	孙志鑫	日期	2014年11月4日
拟制部门		版本	V100
审核人		日期	
批准人		日期	

修订记录 Chang Record:

版本号 Version	日期 Date	修改内容及理由 Change and Reason	拟制人 Prepared by	审核人 Reviewed by	批准人 Approved by
V100	2014-11-25	新拟制	孙志鑫		

目录

目录

- 1. 文档介绍 4
 - 1.1 来源 4
 - 1.2 参考资料 4
- 2. 适用范围 4
- 3. 软件功能描述 4
- 4. MODBUS 协议简介 4
 - 4.1 报文格式 4
 - 4.2 功能码 4
- 5.软件功能描述 9
 - 5.1 任务调度模块 9
 - 5.2 数据解析 12
- 6 模块类型定义 16

1. 文档介绍

1.1 来源

项目

1.2 参考资料

协议

2. 适用范围

变频器项目液晶键盘

3. 软件功能描述

变频器项目液晶键盘与 CPUA 通讯（SCI）模块

4. MODBUS 协议简介

4.1 报文格式

一个标准的 MODBUS 报文包括起始标记、RTU 报文(Remote Terminal Unit, 远程终端装置)和结束标记。

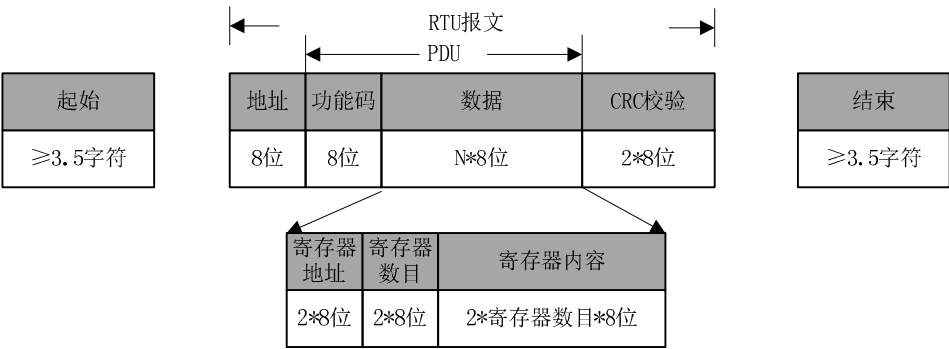


图 4 - 1 RTU 模式报文帧示意图

其中 RTU 报文包括地址码、PDU（Protocol Data Unit，协议数据单元）和 CRC 校验。PDU 包括功能码和数据部分（主要包括寄存器地址、寄存器数目和寄存器内容等，各功能码其详细定义各不相同。）

4.2 功能码

MODBUS 功能码分类如下图所示：



图 4 - 2 MODBUS 功能码分类

如表 4 - 1 所示，平台系列产品主要涉及公共类功能码。如 0x03 读多个寄存器或状态字功能码、0x06 写单个寄存器或命令功能码、0x08 诊断功能码、0x10 写多个寄存器或命令功能码和 0x17 读/写多个寄存器或命令功能码。

另外，为了完成一些特定的功能，如写寄存器（RAM）但不存 EEPROM，在用户定义功能码中自定义了 0x41 写单个寄存器或命令功能码和 0x42 写多个寄存器或命令功能码；如快速交换数据自定义了 0x6A 直接写 n 个寄存器内容功能码。

当从设备接收到异常有效数据时，会返回相关异常信息（详见 3.7 异常信息响应）。为与正常通讯数据区分，特定义异常功能码。与正常请求功能码相对应，异常功能码 = 请求功能码 + 0x80。

表 4 - 1 液晶键盘定义功能码

功能码 (0x)	异常功能码 (0x)	功能
03	83	读多个寄存器或状态字功能码
41	C1	写单个寄存器或命令功能码，不存 EEPROM
42	C2	写多个寄存器或命令功能码，不存 EEPROM
06	86	写单个寄存器或命令功能码
10	90	写多个寄存器或命令功能码
17	97	读/写多个寄存器或命令功能码

0x03 读多个寄存器或状态字功能码

在一个远程设备中，使用该功能码读取保持寄存器连续块的内容。请求 PDU 说明了起始寄存器地址和寄存器数量。

将响应报文中的寄存器数据分成每个寄存器有两字节，对于每个寄存器，第一个字节包括高位比特，并且第二个字节包括低位比特

● 请求 PDU

功能码	1 个字节	0x03
起始地址	2 个字节	0x0000~0xFFFF
寄存器数量	2 个字节	1~16

● 响应 PDU

功能码	1 个字节	0x03
-----	-------	------

字节数	1 个字节	$2 \times N^*$
寄存器值	$N^* \times 2$ 个字节	

0x41 写单个寄存器或命令功能码（不存 EEPROM）

在一个远程设备中，使用该功能码写单个非保持寄存器。

请求 PDU 说明了被写入寄存器的地址。

正常响应是请求的应答，在写入寄存器内容之后返回这个正常响应。

● 请求 PDU

功能码	1 个字节	0x41
寄存器地址	2 个字节	0x0000~0xFFFF
寄存器值	2 个字节	0x0000~0xFFFF

● 响应 PDU

功能码	1 个字节	0x41
寄存器地址	2 个字节	0x0000~0xFFFF
寄存器值	2 个字节	0x0000~0xFFFF

● 错误 PDU

0x42 写多个寄存器或命令功能码（不存 EEPROM）

在一个远程设备中，使用该功能码写连续非保持寄存器块(1 至 16 个寄存器)。

在请求数据域中说明了请求写入的值。每个寄存器将数据分成两字节。

正常响应返回功能码、起始地址和被写入寄存器的数量。

● 请求 PDU

功能码	1 个字节	0x42
起始地址	2 个字节	0x0000~0xFFFF
寄存器数量	2 个字节	1~16
字节数	1 个字节	$2 \times N^*$
寄存器值	$N^* \times 2$ 个字节	

N^* =寄存器数量

● 响应 PDU

功能码	1 个字节	0x42
起始地址	2 个字节	0x0000~0xFFFF
寄存器数量	2 个字节	1~16

0x06 写单个寄存器或命令功能码

在一个远程设备中，使用该功能码写单个保持寄存器。

请求 PDU 说明了被写入寄存器的地址。

正常响应是请求的应答，在写入寄存器内容之后返回这个正常响应。

● 请求 PDU

功能码	1 个字节	0x06
寄存器地址	2 个字节	0x0000~0xFFFF
寄存器值	2 个字节	0x0000~0xFFFF

● 响应 PDU

功能码	1 个字节	0x06
寄存器地址	2 个字节	0x0000~0xFFFF
寄存器值	2 个字节	0x0000~0xFFFF

0x10 写多个寄存器或命令功能码

在一个远程设备中，使用该功能码写连续寄存器块(1 至 16 个寄存器)。

在请求数据域中说明了请求写入的值。每个寄存器将数据分成两字节。

正常响应返回功能码、起始地址和被写入寄存器的数量。

● 请求 PDU

功能码	1 个字节	0x10
起始地址	2 个字节	0x0000~0xFFFF
寄存器数量	2 个字节	1~16
字节数	1 个字节	2×N*
寄存器值	N*×2 个字节	

N*=寄存器数量

● 响应 PDU

功能码	1 个字节	0x10
起始地址	2 个字节	0x0000~0xFFFF
寄存器数量	2 个字节	1~16

0x17 读/写多个寄存器或命令功能码（键盘专用功能码）

在一个单独 MODBUS 事务中，这个功能码实现了一个读操作和一个写操作的组合。

请求说明了被读取的保持寄存器起始地址、数量和写保持寄存器的起始地址、数量以及写入的数据。在写数据域中，字节数说明随后的字节数量。

正常响应包括被读出的寄存器组的数据。在读数据域中，字节数域说明随后的字节数量。

● 请求 PDU

功能码	1 个字节	0x17
读起始地址	2 个字节	0x0000~0xFFFF
读的数量	2 个字节	1~16
写起始地址	2 个字节	0x0000~0xFFFF
写的数量	2 个字节	1~16
写字节数	1 个字节	2×N*
写寄存器值	N*×2 个字节	

N*=写的数量

● 响应 PDU

功能码	1 个字节	0x17
字节数	1 个字节	2×N*
读寄存器值	N*×2 个字节	

帧数据长度定义

虽然理论上一个 MODBUS 报文 RTU 帧 PDU 部分可传输 110+个寄存器数据，但因接收缓存区大小限制，实际 PDU 部分读/写寄存器数量被限制在 1~16 范围内。针对不同功能码，其 RTU 帧实际长度会有不同，详细见表 4-2 所示。

表 4 - 2 RTU 帧长度与功能码对照表

功能码（0x）	RTU 帧长度（字节）			最大长度（字节）
	请求	正常响应	异常响应	
03	8	5+2N _r ^[4]	5	37
06（41）	8	8	5	8
08	8	8	5	8
10（42）	9+2N _w ^[5]	8	5	41
17 ^[6]	10+2 N _w	5+2N _r	5	42 ^[7]
6A	6+2 N _w	6+2 N _r	5	38

[4]: N_r≤16，表示请求读寄存器的数量；[5]: N_w≤16，表示请求写寄存器的数量；[6]: N_w+N_r≤16；

5.软件功能描述

SCI 通信采用任务管理机制进行通信，系统首先判断当前任务状态，然后根据任务类型进行不同的通信。整体结构为：

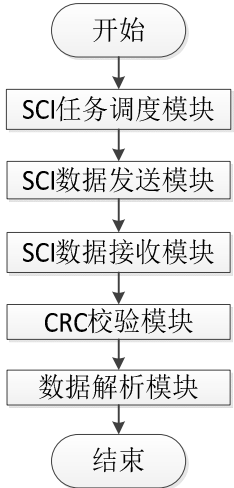


图 5-1 SCI 模块整体流程

5.1 任务调度模块

SCI 通信模块根据当前是否有要通信的任务和任务类型进行通信数据准备，组帧和发送。通信的任务来源是键盘，SCI 模块和 IIC 模块。当他们有通信任务时，发出通信命令保存在任务缓冲区中。SCI 任务调度模块根据该缓冲的任务类型和 SCI 总线的状态决定 SCI 在下一个执行周期要执行的任务。

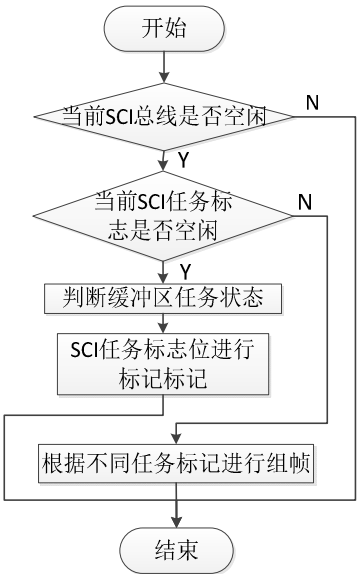


图 5-2 任务调度模块

5.1.1 键盘产生的任务标志

ENTER 按键在 0 级菜单下的任务标志

状态	目的	产生命令	组帧类型
----	----	------	------

选中 A 模式	读选中参数值属性	Task_KeyAttr	17
选中 U 模式	读 6 个用户功能码	User_Read	03
选中 C 模式	读 6 个非出厂值功能码	NFactory_Read	17
选中 F 模式	读 5 个故障功能码	Task_KeyFault	17

ENTER 按键在 1 级菜单下的任务标志

变频器旧版模式下，监控界面

状态	目的	产生命令	组帧类型
监视界面 0	读监视参数属性	.Task_MonitRead2	17
监视界面 1 快速定位	读监视参数属性	Task_MonitRead2	17
监视界面 1 修改参数	保存修改参数	Task_Save	06

变频器新版模式下，监控界面

状态	目的	产生命令	组帧类型
监视界面 0	读 F12.33 到 F12.37 值命令	Task_MonitRead1	17
监视界面 1 快速定位	读监视参数属性	Task_MonitRead2	17
监视界面 1 修改参数	保存监视参数地址到 F13.33-F12.37	Task_SaveMonit	10

变频器新版模式下 A, U, C, F, V

状态	目的	产生命令	组帧类型
A, C	读选中参数值命令	Task_KeyAttr	17
U, 快速定位	读 6 个用户功能码	User_Read	03
U, 非快速定位	读选中参数值命令	Task_KeyAttr	17

ENTER 按键在 2 级菜单下的任务标志

状态	目的	产生命令	组帧类型
A, U, C	保存修改后的参数值	Task_Save	06

ESC 按键在 0 级菜单下的任务标志

状态	目的	产生命令	组帧类型
变频器旧版	发出读 F12.04-F12.08 监视参数命令	Task_Read_Monit	03
变频器新版	读 F12.38 到 F12.40 值命令	Task_MonitRead1	17

ESC 按键在 1 级菜单下的任务标志

状态	目的	产生命令	组帧类型
监视界面 1 非快速定位 非修改参数	发出读 F12.04-F12.08 监视参数命令	Task_Read_Monit	03
修改参数	发出恢复参数命令	Task_Restore	41

ESC 按键在 2 级菜单下的任务标志

状态	目的	产生命令	组帧类型
A, U, C 参数改变	参数恢复命令	Task_Restore	41

UP 按键按下在 1 级菜单下的任务标志

状态	目的	产生命令	组帧类型
M, 监视界面 0 上次非按下	发出 UP 命令	Task_UPDown	41
M, 监视界面 1 修改参数	发出参数加命令	Task_KeyAdd	41
A, 非快速定位	读选中参数值命令	Task_KeyAttr	17
U, 非快速定位	第一个 F11 组参数	User_Read_up	03
C, 光标位不在最上面	读选中参数值命令	Task_KeyAttr	17
C, 光标位在最上面	读上一个非出厂值功能码	NFactory_Read_up	17
F	读故障功能码	Task_KeyFault	03

UP 按键按下在 2 级菜单下的任务标志

状态	目的	产生命令	组帧类型
A, U, C 参数改变	向 CPUA 发出改变后的参数	Task_KeyAdd	41

UP 按键弹起任务标志

状态	目的	产生命令	组帧类型
上次按下 处于监视界面 0	发出弹起命令	Task_UPDown	41

DOWN 按键和 UP 按键类似，但是操作相反。

SHIFT 按键在 1 级菜单下的任务标志

状态	目的	产生命令	组帧类型
旧版 监视界面 0	读监视参数属性	Task_MonitRead2	17

M.K 按键按下

状态	目的	产生命令	组帧类型
上次处于 非按下状态	读该键功能	Task_ReadJogFunc	03

M.K 按键弹起任务标志

状态	目的	产生命令	组帧类型
处于点动运行	发出停车命令	Task_JOGStop	41

RUN 按下任务标志

状态	目的	产生命令	组帧类型
/	发出运行命令	Task_KeyRun	41

STOP 按键按下

状态	目的	产生命令	组帧类型
变频器故障 没改动参数	发出恢复参数命令	Task_Restore	41
变频器故障	读 6 个用户功能码	Task_FaultRst	
其他	发出停车命令	Task_Stop	41

5.1.2 IIC 产生的任务标志

IIC 一共产生三个任务，在选择参数下载时，如果软件匹配成功则发送保存参数命令 Task_Save。

在参数下载状态下，当从 EEPROM 读取出一组参数后发送 Task_Iic_Down

命令，开始下载到变频器控制芯片中。

在参数上传状态下，开始阶段要从变频器中读取出一组参数，发送 Task_Ilc_Up 命令，开始从变频器中读取一组参数。

5.1.3 SCI 产生的任务标志

SCI 部分会在接收到数据进行数据解析过程中产生任务。具体详见数据解析。

5.2 数据解析

数据帧经过校验后如果接收正确，就按照 Modus 协议进行数据解析，提取数据帧中有用的数据信息，给键盘其它模块使用。解析完成后，置 SCI 任务状态为空并释放 SCI 总线。

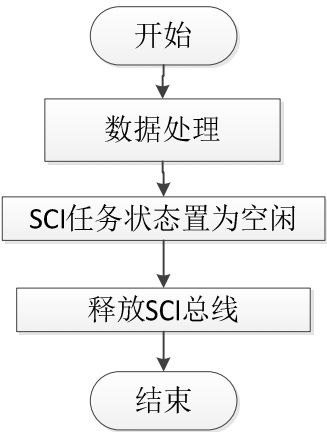


图 5-3 数据解析处理

数据处理的方式因任务类型的不同而不同，总体可以分成以下几类：控制命令、读功能码命令、读参数命令、写参数命令。

发送控制命令的任务类型所发送的任务有：

表 5-2 控制命令数据解析

名称	发送变量	组帧类型	数据处理
正转运行	CMD_R_RUN	41	无
点动正转	CMD_JOG_R_RUN	41	无
点动反转	CMD_JOG_F_RUN	41	无
减速停车	CMD_STOP	41	无
快速停车	CMD_STOP_QUICK	41	无
自由停车	CMD_STOP_FREE	41	无
故障复位	CMD_FAULT_RST	41	无
正负切换	CMD_FR	41	无
JOG 停车	CMD_JOG_STOP	41	无
参数恢复	CMD_ARG_RST	41	无

读功能码命令：

F11 组是用户自选参数组，功能码保存的内容是用户自设定的想要查看的其他组的功能码，在键盘中选择用户自定义模式可以查看并修改用户选择的功能码。在内存中的意义如下表所示：

表 5-3 F11 组功能命令

地址	值	指向功能码地址	功能码的值
F1100	0000	F0000	0
F1101	0001	F0001	0
F1102	0002	F0002	0

在用户自定义模式下要查看用户自选功能码过程如下：

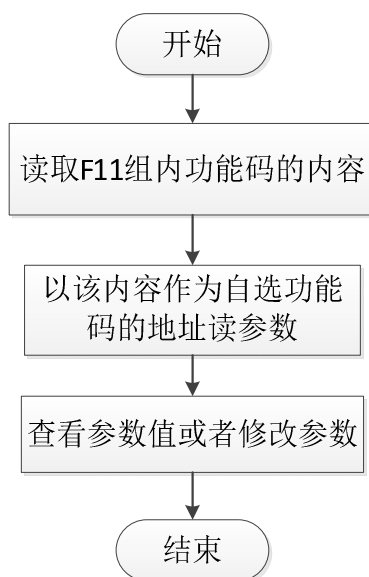


图 5-4 F11 组数据处理流程

在液晶键盘中，一屏可以查看六个功能码，并且显示选中的功能码的值，因此操作流程为：

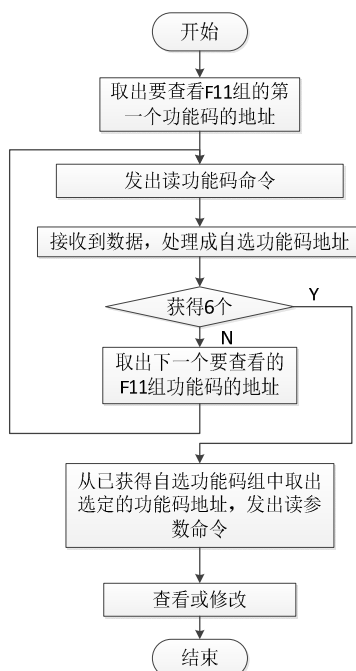


图 5-5 自定义模式数据处理流程

非出厂值模式与用户自定义模式类似。从上图中可以看出，对于读功能码类型的命令，在获得数据后需要进行的处理为：

表 5-4 读功能码数据处理过程

名称	发送变量		数据处理	产生命令
ENTER 按键发出读用户自选功能码命令	前五个	F11 组地址	将接收数据处理为地址 再次发出读 F11 组功能码命令	User_Read
	第六个	F11 组地址	将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr
UP 按键发出读用户自选功能码命令	要显示的第一个 F11 组地址		将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr
UP 按键发出读用户自选功能码命令	要显示的第六个 F11 组地址		将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr
ENTER 按键发出读非出厂值功能码命令	前五个	当前非出厂值地址	将接收数据处理为地址 再次发出读非出厂值功能码命令	NFactory_Read
	第六个	当前非出厂值地址	将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr
UP 按键发出读非出厂值功能码命令	要显示的第一个非出厂值地址		将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr
UP 按键发出读非出厂值功能码命令	要显示的第六个非出厂值地址		将接收数据处理为地址 发出读功能码参数值和属性命令	Task_KeyAttr

读参数命令：

读参数分为两中情况，第一个情况是读参数的值和属性，属性包括了精度，单位，是否可修改，操作/显示属性，符号，操作限定（具体解释见 MODBUS 协议）；参数的值包括当前值，最大值，最小值和出厂值。第二种情况读参数的过程是读参数的当前值。

在运行监视菜单选项下，一屏可以查看多个（3 或 7）监视参数，在处理的过程中就要分成两部分，第一部分依次读取这些监视参数的值和属性，第二部分是利用空闲状态循环读取这些参数的当前值进行刷新。流程为：

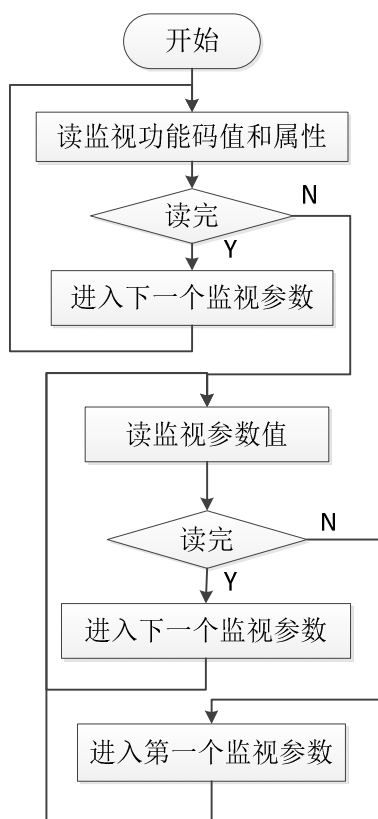


图 5-6 读监视参数流程

数据处理过程为：

表 5-5 读监视参数流程

名称	发送变量		数据处理	产生命令
ESC 监视功能 码值和属 性命令	前两个	监视参数地址	将接收到数据保存 产生读监视参数值和属性命令	Task_MonitRead2
	第三个	监视参数地址	将接收到数据保存	/
ENT/SFT 监视功能 码值和属 性命令	前 N-1 个	监视参数地址	将接收到数据保存 产生读监视参数值和属性命令	Task_MonitRead2
	第 N 个	监视参数地址	将接收到数据保存	/

在全菜单模式、用户自定义模式、非出厂值模式下，每次可以查看一个功能码的参数值，因此读参数值的流程为：

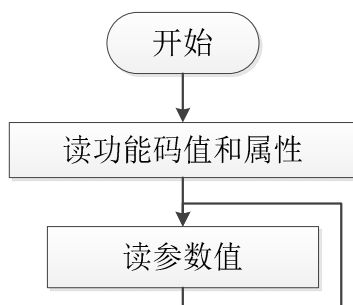


图 5-7 读功能码流程

在故障查询界面和软件版本界面都是采用依次读功能码的值和参数的命令，不同的是这两种情况不在实时更新参数值。

写参数命令：

发出写参数命令的情况有键盘加，键盘减，参数保存，参数下载这四种情况

表 5-6 写参数值属性

名称	发送变量	数据处理		产生命令
UP 键盘加	改变后的参数值	无		无
DOWN 键盘减	改变后的参数值	无		无
ENTER 参数保存	改变后的参数值	参数上传	置键盘状态为参数上传状态	无
		参数下载	置键盘状态为参数下载状态	无
		其他	无	无
参数下载	从 IIC 读出的参数值	制 IIC 模块状态为下载完成		无

6 模块类型定义

typedef struct

{

u8 Task_IdleRead;
u8 Task_SVersionRead;
u8 Task_ReadProductnum;
u8 Task_WriteVersion;
u8 Task_KeyFault;
u8 Task_KeyAttr;
u8 Task_KeyRun;
u8 Task_Stop;
u8 Task_ReadJogFunc;
u8 Task_KeyJog;
u8 Task_JOGStop;
u8 Task_FaultRst;
u8 Task_Restore;
u8 Task_KeyAdd;
u8 Task_KeyDec;
u8 Task_Save;
u8 Task_MonitRead1;
u8 Task_MonitRead2;
u8 Task_Read_Monit;
u8 User_Read ;
u8 User_Read_up ;
u8 User_Read_down;
u8 NFactory_Read;
u8 NFactory_Read_up;
u8 NFactory_Read_down;
u8 Task_ReadSpeed;
u8 Task_ReadSpeed2;
u8 Task_IIC_Up;
u8 Task_IIC_Down;


```

u8 Task_contrast;
u8 Task_ReadGroup;
u8 Task_ReadGroupSize;
u8 Task_UPDown;
u8 Task_SaveMonit;
u8 Task_Language;
} TaskSCIType;typedef enum
{
    SCI_IDLE =0,
    SCI_SVERSION_READ=2,
    SCI_Productnum_READ=3,
    SCI_WRITE_VERSION=4,
    SCI_SEARCH_FAULT=5,
    SCI_RUN=6,
    SCI_STOP=7,
    SCI_READ_JOG_FUNC=8,
    SCI_JOG=9,
    SCI_JOGSTOP=10,
    SCI_FALTRST=11,
    SCI_ATTR=12,
    SCI_RESTORE=13,
    SCI_ADD=14,
    SCI_DEC=15,
    SCI_SAVE=16,
    SCI_MONITREAD1=17,
    SCI_MONITREAD2=18,
    SCI_READ_MOINT=31,
    SCI_USERREAD=19,
    SCI_USERREAD_UP=20,
    SCI_USERREAD_DOWN=21,
    SCI_NFACTORYREAD=22,
    SCI_NFACTORYREAD_UP=23,
    SCI_NFACTORYREAD_DOWN=24,
    SCI_IIC_UP=25,
    SCI_IIC_DOWN=26,
    SCI_READ_SPEED=27,
    SCI_IDLE_READ =28,
    SCI_READ_SPEED2=29,
    SCI_SET_CONTRAST=30,
    SCI_READ_GROUP =32,
    SCI_READ_GROUP_SIZE =33,
    SCI_UPDOWN=34,
    SCI_SaveMonit=35,
    SCI_Language=36,
}SCIWorkStatus;
typedef struct
{

```

```

    SINE_UInt8  Level;
    SINE_UInt8 PreMenuMode;
    SINE_UInt8 MenuMode;
    SINE_UInt8 MonitType;
    SINE_UInt8 MonitType1;
    SINE_UInt8 SCI_BUS_STATUS;
    SINE_UInt16 SCIBusyCnt;
    SINE_UInt16 StatusBusyCnt;
    SINE_UInt16 SCI_WorkStatus;
    SINE_UInt16 SCI_SchedCnt;
    SINE_UInt8 IdleReadState;
    SINE_UInt16 FaultNum;
    TYPE_VVVFState1 VVF_WorkStatus1;
    VVFStauts2Com VVF_WorkStatus2;
    TYPE_VVFStatus3Com VVF_WorkStatus3;
    DataType2Com Ctl_Cmd;
    SINE_UInt16 Passwordcompare;
    SINE_UInt8  Cursor;
    SINE_UInt8  Cursor1;
    SINE_UInt8  CursorMAX;
    SINE_UInt16 CommuCnt;
    SINE_UInt16 CPU_Version1[3];
    SINE_UInt16 CPU_Version2[3];
    SINE_UInt16 CPUA_Version11;
    SINE_UInt16 CPUA_Version21;
    SINE_UInt16 CPUB_Version11;
    SINE_UInt16 CPUB_Version21;
    SINE_UInt16 CPUC_Version11;
    SINE_UInt16 CPUC_Version21;
    SINE_UInt16 SerialNumber1;
    SINE_UInt16 SerialNumber2;
    SINE_UInt16 SerialNumber3;
    SINE_UInt8 Versionmatching;
    SINE_UInt8 LoctionDisp;
    SINE_UInt8 FreqFRState;//ÆµÂÊµÃ·û°ÃÏÔÊ¾
    SINE_UInt8 Testtype;
    SINE_UInt8 Testseg;
    SINE_UInt8 Keytype;
    SINE_UInt8 FuncSel;
    SINE_UInt8 FautType;
    SINE_UInt8 ReadGroupSize;
    SINE_UInt8 LCDMonitType;
    SINE_UInt8 Language;
    SINE_UInt8 RunWay;
    SINE_UInt8 MonitRead1;
} MenuUnionType;

```