Technical

Note

广州广州炫通电气科技有限公司

工程技术笔记

ARM与DSP交互说明

TN01010101 V1.00 Date:2013/05/13

DSP开发文档

|  |  |
| --- | --- |
| 类别 | 内容 |
| 关键词 | DSP与ARM通信、DM8000 |
| 摘 要 | DSP与ARM通信协议，通信数据，接线方式 |

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 修改人 | 原因 |
| V1.00 | 2013/5/13 | 郭安琪 | 创建文档 |
|  |  |  |  |

目 录

[1. 适用范围 1](#_Toc358037969)

[2. ARM与DSP通信 2](#_Toc358037970)

[2.1 功能码及描述 2](#_Toc358037971)

[2.2 数据帧结构 2](#_Toc358037972)

[2.3 DSP参数设置 3](#_Toc358037973)

[2.3.1 数据帧定义 3](#_Toc358037974)

[2.3.2 设置错误码定义 3](#_Toc358037975)

[2.3.3 默认参数值 3](#_Toc358037976)

[2.4 获取DSP信息 4](#_Toc358037977)

[2.5 有效值、谐波、功率、序量等计算 4](#_Toc358037978)

[2.6 报文离散度计算 7](#_Toc358037979)

[2.7 波形生成计算 8](#_Toc358037980)

[2.8 常量定义 9](#_Toc358037981)

[2.8.1 限制常量定义 9](#_Toc358037982)

[2.8.2 标识常量定义 10](#_Toc358037983)

[3. DSP处理流程 11](#_Toc358037984)

[3.1 总体处理流程 11](#_Toc358037985)

[4. 交互时间 12](#_Toc358037986)

[4.1 DSP运算时间级别 12](#_Toc358037987)

[4.2 ARM发送时间频率 12](#_Toc358037988)

# 适用范围

该文档提供ARM和DSP数据交换的定义，方便开发。DSP型号为OAMP3530。ARM端以数据帧的形式向DSP端发送计算指令和参数，由DSP计算完成后再以数据帧的形式向ARM端返回计算结果。本文档中描述的各种数据交换的定义适用于DM800光数字分析仪中的各种数据计算。

# ARM与DSP通信

## 功能码及描述

功能码用于区分不同的数据计算，分为两种类型，一种是ARM到DSP的功能码，用于区分ARM向DSP发送的不同的计算命令，另一种是DSP到ARM的功能码，用以区分从DSP端返回的不同的计算结果。

表 2.1功能码及描述ARM-to-DSP

|  |  |
| --- | --- |
| 功能码 | 描述 |
| ARM2DSP\_CONTROLCODE\_CONFIG | 对DSP进行参数配置 |
| ARM2DSP\_CONTROLCODE\_SMV\_DATA | 给DSP传送采样值数据 |
| ARM2DSP\_CONTROLCODE\_RESET\_SMV | 重置采样值数据缓存 |
| ARM2DSP\_CONTROLCODE\_RESET\_TIME | 重置报文数据缓存 |
| ARM2DSP\_CONTROLCODE\_SMV\_GEN | 通知DSP产生特定类型波形数据 |
| ARM2DSP\_CONTROLCODE\_TIME\_DATA | 给DSP传送报文时间数据 |
| ARM2DSP\_CONTROLCODE\_INFO | 得到DSP信息 |
| ARM2DSP\_CONTROLCODE\_QUIT | 通知DSP退出 |

表 2.2功能码及描述DSP-to-ARM

|  |  |
| --- | --- |
| 功能码 | 描述 |
| DSP2ARM\_CONTROLCODE\_INFO | DSP返回硬件信息 |
| DSP2ARM\_CONTROLCODE\_SMV\_DATA | DSP返回的实际采样值 |
| DSP2ARM\_CONTROLCODE\_SMV\_CALC\_RT | DSP返回计算有效值/谐波等数据 |
| DSP2ARM\_CONTROLCODE\_SMV\_GEN\_RT | DSP返回生成波形数据 |
| DSP2ARM\_CONTROLCODE\_MSG\_DISP\_RT | DSP返回报文离散度数据 |
| DSP2ARM\_CONTROLCODE\_ERROR | DSP返回参数设置结果信息 |

## 数据帧结构

在本系统中，无论是从ARM-to-DSP的数据帧，还是DSP-to-ARM的数据帧，都采用统一的一种帧结构，即帧头+数据的形式，帧头是一个FrameHead结构体，而数据则根据不同的功能码有不同的形式。比如对于ARM2DSP\_CONTROLCODE\_CONFIG，数据将会是一个Config的结构体。

表 2.3 数据帧结构

|  |  |
| --- | --- |
| 帧头 | 数据 |
| FrameHead | 与具体计算相关的数据 |

///

/// @brief

/// FrameHead结构定义ARM和DSP之间进行数据交互的数据帧的帧头，数据帧为帧头+数据

///

typedef struct \_FrameHead

{

uint8\_t nCommandID; ///< ARM对DSP的控制命令（功能码）

uint32\_t nDataLength; ///< 数据长度

}FrameHead, \*PFrameHead;

在ARM与DSP通信时，需要构造一个完整的数据帧，并在帧头填入数据类型和数据长度，然后再发送和帧数据。

## DSP参数设置

### 数据帧定义

在DSP进行计算之前，需要先设置DSP计算参数，在设置参数之后，DSP端会返回一个设置状态信息，包括设置操作的错误码。ARM发送的数据为一个Config结构体，ARM收的数据为一个Error结构体。这两个结构体的定义如下：

///

/// @brief

/// DSP配置结构体

///

typedef struct \_Config

{

uint16\_t nSmvCalcCount;

uint16\_t nMsgTimeCalcCount;

uint32\_t nDefaultTimeDif; ///< 额定报文差时间（单位：μs）

}Config, \*PConfig;

///

/// @brief

/// 错误信息结构体

///

typedef struct \_Error

{

uint8\_t nErrno; ///< 设置错误码

}Error, \*PError;

### 设置错误码定义

设置计算参数时，DSP会对设置参数进行检查，然后对检查结果通过错误码的形式返回。

|  |  |
| --- | --- |
| 错误码（nErrno） | 描述 |
| 0 | 设置成功 |
| 1 | 谐波计算所需数据数目超过限制 |
| 2 | 报文离散度计算所需数据数目超过限制 |

### 默认参数值

在DSP计算之前，应该首先对DSP中相关的计算参数进行设置，以保证计算结果的正确。否则DSP就会按照默认的参数进行计算。默认计算参数值如下表：

|  |  |
| --- | --- |
| 参数名 | 默认值 |
| nSmvCalcCount | 800 |
| nMsgTimeCalcCount | 4680 |
| nDefaultTimeDif | 0 |

## 获取DSP信息

如果要得到DSP的信息，比如版本信息，ARM端需要发送ARM2DSP\_CONTROLCODE\_INFO命令码的数据帧，只需在帧头指明命令码即可，不需要其他额外信息。DSP会返回一个DspInfo的结构体，该结构体定义如下：

///

/// @brief

/// DSP信息结构体

///

typedef struct \_DspInfo

{

uint8\_t nVersion; ///< DSP版本号

}DspInfo, \*PDspInfo;

## 有效值、谐波、功率、序量等计算

对于有效值、谐波，功率，序量等计算，需要得到一定数目的采样值数据，因此ARM端需要发送ARM2DSP\_CONTROLCODE\_SMV\_DATA命令码，后面加上一个SmvData的结构体的数据帧，DSP计算之后会返回一个SMV\_DATA\_CALC\_RT的结构体，用以表示各种计算结果。这两个结构体的定义如下：

///

/// @brief

/// 定义ARM发送到DSP的采样值数据结构

///

typedef struct

{

uint8\_t nChannelCount; ///< 实际传输的通道数目

uint16\_t nSmvPerCircle; ///< 实际传输的数据一个周波的采样数

uint8\_t nChannelVCType[MAX\_SMV\_CHANNELS]; ///< 每个通道的电压/电流类型

float fScaleFactor[MAX\_SMV\_CHANNELS]; ///< 每个通道的比例因子

float fChannelData[MAX\_SMV\_CHANNELS][MAX\_SMV\_PER\_CIRCLE];

} SmvData, \*PSmvData;

///

/// @brief

/// 定义计算谐波返回的数据帧格式

///

typedef struct {

// 通道数据

uint8\_t nChannelCount; ///< 实际传输的通道数目

uint8\_t nChannelVCType[MAX\_SMV\_CHANNELS]; ///< 每个通道的电压/电流类型

// 频率和有效值

float fFreq[MAX\_SMV\_CHANNELS]; ///< 每个通道的信号频率

float fRmsValue[MAX\_SMV\_CHANNELS]; ///< 每个通道的有效值

// 谐波

float fHarmContent[MAX\_SMV\_CHANNELS][MAX\_HARM\_COUNT];

float fHarmAngle[MAX\_SMV\_CHANNELS][MAX\_HARM\_COUNT];

float fDCValue[MAX\_SMV\_CHANNELS]; ///< 直流有效值

float fSumHarmTHD[MAX\_SMV\_CHANNELS]; ///< 各通道总谐波含有率

float fHarmTHD[MAX\_SMV\_CHANNELS][MAX\_HARM\_COUNT];

// 序量

float fPhaseSeqContent[2][3]; ///< 零序、正序、负序向量大小

float fPhaseSeqAngle[2][3]; ///< 零序、正序、负序相位角

// 功率

float fActivePower[3]; ///< 有功功率，分别对应ABC三相

float fReactivePower[3]; ///< 无功功率，分别对应ABC三相

float fApparentPower[3]; ///< 视在功率，分别对应ABC三相

float fPowerFactor[3]; ///< 功率因数，分别对应ABC三相

float fTotalActivePower; ///< 总功率

float fTotalReactivePower; ///< 总无功功率

float fTotalApparentPower; ///< 总视在功率

float fTotalPowerFactor; ///< 总功率因数

// 核相

float fSPAngleDif[3]; ///< 同相相角差

float fSPVectorDifContent[3]; ///< 同相向量差大小

float fSPVectorDifAngle[3]; ///< 同相向量差角度

float fIPVectorDifContent[6]; ///< 相间向量差大小

float fIPVectorDifAngle[6]; ///< 相间向量差角度

} SMV\_DATA\_CALC\_RT;

需要说明的是，DSP需要接收到一定数目的采样值数据才会计算结果，而具体接收多少采样值数据计算结果则需要由用户在计算前设置计算参数，即Config中nSmvCalcCount的值，默认情况下，DSP会每800个点计算一次有效值和谐波等。大致流程如下：

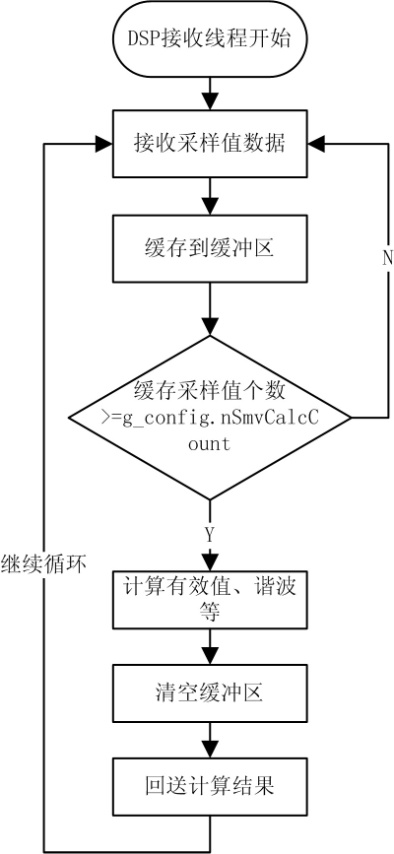


图 2.1 DSP谐波等计算流程

以上数据结构中含有多维数组。有关谐波的多维数组的含义如下：

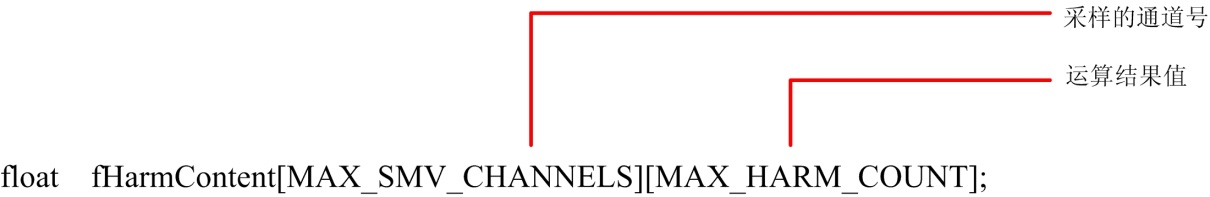


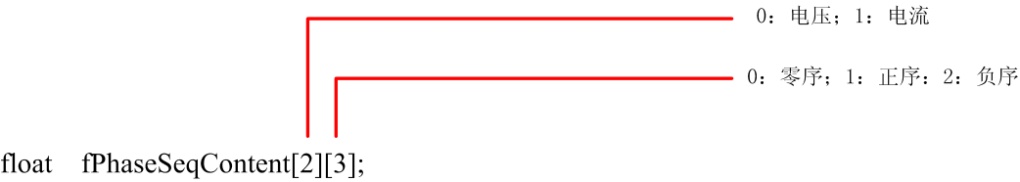
图2.2 字段意义说明

例如：

fHarmContent[1][25]的是代表：第2通道的26次谐波有效值；

fHarmContent[2][49]的是代表：第3通道的50次谐波有效值；

有关序量的多维数组含义如下：



有关核相的数组含义如下：



float fSPVectorDifContent[3];和float fSPVectorDifAngle[3];与float fSPAngleDif[3]含义相同。



float fIPVectorDifAngle[6];与float fIPVectorDifContent[6];含义相同

## 报文离散度计算

报文离散度的计算和谐波等数据的计算类似，也需要传输一定个数的报文时间时间数据才能计算报文离散度。从ARM端要发送ARM2DSP\_CONTROLCODE\_TIME\_DATA命令码的数据帧和MsgTimeData结构体数据，DSP计算之后会返回一个11个元素的浮点数组，用来表示各范围的报文离散度统计结果。MsgTimeData的定义如下：

///

/// @brief

/// 报文时间结构体

///

typedef struct \_MSGTIME

{

uint32\_t nSecond; ///< 秒

uint32\_t nMicrosecond; ///< 微秒

}MSGTIME;

///

/// @brief

/// 报文时间的数据帧格式

///

typedef struct

{

uint16\_t nMsgTimeCount; ///< 每次传输的报文时间数

MSGTIME msgTimes[MAX\_MSG\_TIME\_PER\_CIRCLE]; ///< 每次传输的报文时间数据

}MsgTimeData, \*PMsgTimeData;

从DSP返回的浮点数组各元素表示的范围如下：

|  |  |
| --- | --- |
| 下标 | 范围 |
| 0 | <=-250 |
| 1 | -249~-50 |
| 2 | -49~-10 |
| 3 | -9~-2 |
| 4 | -1 |
| 5 | 0 |
| 6 | 1 |
| 7 | 2-9 |
| 8 | 10-49 |
| 9 | 50-249 |
| 10 | >=250 |

## 波形生成计算

波形生成可以生成一个带有多次谐波的复杂波形数据，由DSP计算之后以采样值的形式返回给ARM。首先由ARM端发送一个ARM2DSP\_CONTROLCODE\_SMV\_GEN类型的数据帧和SmvGenData结构体数据，DSP计算之后会返回一个SMV\_GEN\_DATA\_CALC\_RT结构体，用以表示生成的波形结果，这两个结构体定义如下：

///

/// @brief

/// 定义所需要产生的波形数据结构体

///

typedef struct

{

uint8\_t nChannelCount; ///< 要生成波形的通道数

uint16\_t nSmvCount; ///< 要生成的各通道的采样点数

float fSampleRate; ///< 采样率

uint8\_t nChannelVCType[MAX\_SMV\_GEN\_CHANNEL\_COUNT];

uint8\_t nHarmCount[MAX\_SMV\_GEN\_CHANNEL\_COUNT];

float fDCValue[MAX\_SMV\_GEN\_CHANNEL\_COUNT];

float fScaleFactor[MAX\_SMV\_GEN\_CHANNEL\_COUNT];

float fRmsValue[MAX\_SMV\_GEN\_CHANNEL\_COUNT][MAX\_HARM\_COUNT];

float fPhaseAngle[MAX\_SMV\_GEN\_CHANNEL\_COUNT][MAX\_HARM\_COUNT];

float fFreq[MAX\_SMV\_GEN\_CHANNEL\_COUNT][MAX\_HARM\_COUNT];

} SmvGenData;

///

/// @brief

/// 计算波形生成返回的数据帧格式

///

typedef struct{

uint8\_t nChannelCount; ///< 计算的通道数

uint16\_t nSmvCount; ///< 要生成的各通道的采样点数

uint8\_t nChannelVCType[MAX\_SMV\_GEN\_CHANNEL\_COUNT]; ///< 每个通道的电压/电流类型

float fSmvGenData[MAX\_SMV\_GEN\_CHANNEL\_COUNT][MAX\_SMV\_GEN\_COUNT];

} SMV\_GEN\_DATA\_CALC\_RT;

## 常量定义

### 限制常量定义

///

/// @brief

/// 定义ARM和DSP之间共享内存的大小

///

#define ARM\_DSP\_BUF\_SIZE (32\*1024)

///

/// @brief

/// SV协议的最大采样值通道数

///

#define MAX\_SMV\_CHANNELS 30

///

/// @brief

/// 一个周波内采样点的最大值，256对应采样率为256\*50=12800

///

#define MAX\_SMV\_PER\_CIRCLE 256

///

/// @brief

/// 采样周波数的最大值

///

#define MAX\_SMV\_CALC\_COUNT (MAX\_SMV\_PER\_CIRCLE\*10)

///

/// @brief

/// 计算的最高谐波次数

///

#define MAX\_HARM\_COUNT 50

///

/// @brief

/// 一次传输的最大报文时间数

///

#define MAX\_MSG\_TIME\_PER\_CIRCLE 468

///

/// @brief

/// 计算报文时间的最大次数

///

#define MAX\_MSG\_TIME\_CALC\_COUNT (MAX\_MSG\_TIME\_PER\_CIRCLE\*10)

///

/// @brief

/// 波形生成的最大通道数

///

#define MAX\_SMV\_GEN\_CHANNEL\_COUNT 16

///

/// @brief

/// 波形生成的最大采样点数

///

#define MAX\_SMV\_GEN\_COUNT 256

### 标识常量定义

///

/// @brief

/// 采样值是电压/电流值

/// xxxx xxxx 一个字节分高4位和低四位，低4位用于标识是电压还是电流数据，

/// 高4位用于标识相别，十六进制的ABC分别代表第一个AD的ABC三相，DEF分别代表另一个

/// AD的ABC三相（用于核相计算）

/// 如0xA1就代表第一个AD的A相电压，0xA2代表第一个AD的A相电流，其他相别一次类推

///

#define SMV\_VCTYPE\_VOLTAGE 0x01 ///< 采样值是电压值

#define SMV\_VCTYPE\_CURRENT 0x02 ///< 采样值是电流值

#define SMV\_VCTYPE\_PHASE\_A 0xA0 ///< 代表AD0的A相

#define SMV\_VCTYPE\_PHASE\_B 0xB0 ///< 代表AD0的B相

#define SMV\_VCTYPE\_PHASE\_C 0xC0 ///< 代表AD0的C相

#define SMV\_VCTYPE\_PHASE\_D 0xD0 ///< 代表AD1的A相

#define SMV\_VCTYPE\_PHASE\_E 0xE0 ///< 代表AD1的B相

#define SMV\_VCTYPE\_PHASE\_F 0xF0 ///< 代表AD1的C相

这些标识常量用于标示采样值中各通道的电流/电压类型，即SmvData和SmvGenData 中的nChannelVCType，以便DSP进行序量、功率和核相计算。

# DSP处理流程

## 总体处理流程

DSP端的处理流程大致可以分为五个步骤：接收->任务分发->数据缓存->计算->回送。接收就是接收ARM端发来的数据帧，任务分发是将接收到的数据帧按照类型分发给不同的任务进行处理，数据缓存就是在计算之前先将接收到的数据缓存到DSP，从而减少了对ARM-DSP共享缓冲区的读写，减少共享缓冲区的错误率。计算就是在执行计算算法得到计算结果，回送就是将计算结果返回给ARM。

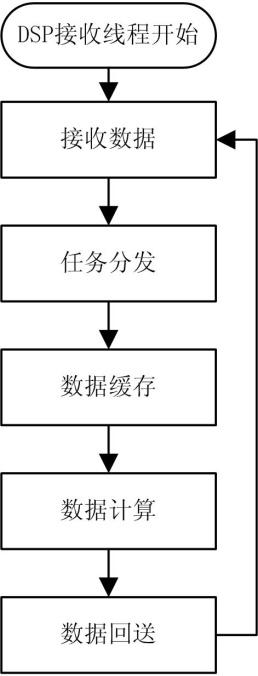


图 3.1 DSP处理流程

其中，对于任务分发，将按照算法的复杂程度进行，在DSP中内建有多个任务分级，不同级别的任务的优先级是不同的，所以任务的分发会根据需要处理的计算来进行，执行时间短的任务会分到优先级高的任务中去，而执行时间长的任务会分到优先级低的任务中去，以保证多任务计算时最优化DSP资源。

对于所有接收到的数据会先做缓存，缓存的大小由不同的计算决定，对于采样值和报文时间数据，会对数据做一定量的缓存，当达到计算所需的缓存数据量时才会进行计算。所以，当ARM一次传输的采样值或者报文时间数据尚未达到计算所需的数据量的时候，将只进行缓存而不进行计算，所以，对于这两种类型的计算，会接收多次数据回送一次数据。当缓存到计算所需的数据量并且计算出结果之后，缓冲区将会被清空，同时缓冲区指针会移到初始位置以继续缓存下一次数据，以防止缓冲区溢出。

# 交互时间

## DSP运算时间级别

DSP端是一个多任务系统，按照算法运行时间分为40ms级、200ms级、5120ms级、10min级4种级别，每种级别代表其中的算法运行时间在该级别对应的时间范围以内，在本系统中，算法对应的级别如下：

|  |  |  |
| --- | --- | --- |
| 时间级别 | 算法 | 说明 |
| 40ms | RealSmvCalc | 实际值计算 |
| SmvGenerate | 波形生成计算 |
| MsgDispCalc | 报文离散度计算 |
| 200ms | SmvCalc | 谐波等相关数据计算 |
| 5120ms | 无 |  |
| 10min | 无 |  |

## ARM发送时间频率

按照50Hz额定频率的信号来计算，每采样一个周期的数据就会发送一次采样值给DSP，所以发送时间间隔为20ms。

按照4000Hz采样频率计算，每采样一个数据得到一个报文，如果没得到一个报文就发送一次报文时间来计算报文离散度，发送时间间隔为1/4000=0.25ms。

但目前DSP由于计算能力有限，再加上数据传输时间，不能以太高的速率发送，以免计算不过来，所以规定ARM端给DSP端发送的时间间隔不小于10ms，以保证数据的可靠性和DSP的稳定性。