Android 与图像处理板（IPS）通信协议

1 通讯协议

根据公司自研自动售货机的控制需求，制定 Android 与图像处理板(IPS）之间的通讯协议，为了满足通信的实时性需求，尽量缩简协议的帧长度，减少冗余，从而减少传输时间，同时采用位域的定义方式，提高带宽利用率。

Android 板与图像处理板（IPS）通信采用主从通信方式，Android 作为主机，IPS 作为从机。

1.1 生成货道坐标列表（0x02）

设备补货后，Android 发送该指令给 Linux 板，进行货架扫描，生成货架数据表，并把货架分区情况和分层情况反馈给 Android。

命令帧格式

命令	长度	数据域
CMD	LEN	Resvd
0x02	0x01	0x00

【说明】

1. CMD：命令码，取值 0x02。
2. Resvd：保留。
3. 超时响应时间：300S。

响应帧格式

响应	长度	数据域							
RSP	LEN	ErrCode		MchSta	FloorCnt		Floor1		
							Area	Floor	Count
0x02	5+3*N								

数据域			
……	Floor N		
	Area	Floor	Count

【说明】

1. RSP：响应码，取值与命令码相同。
2. ErrCode：错误码，占两个字节。错误码列表参见附录。
3. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

4. FloorCnt：货架分层的总数量,取值范围 0 到 65535。
5. Floort(x)：货架分层情况，包括 Area 数量，Floor 数量，每层的货道数量 Count。

注意：当响应帧分多帧上报时，只有第一帧包含 ErrCode、MchSta 和 ErrorCnt。

sdk 代码示例:

```
/******************************** start:生成货道坐标列表命令(异步)*********************************/

//Machine machine =已经初始化的 Machine

/**
```

```
* 1.构造命令(异步)

*/

Command cmd = Command.create(ID_LOCATION_SCAN, null, Command.TYPE_ASYNC);

/**

* 2.监听回调

*/

cmd.setOnCompleteListener(new ICallbackListener() {

@Override

public boolean callback(Command cmd) {

if (cmd.isReplyTimeout() || cmd.isTaskTimeout()) {

return true;

}


if (null != cmd.getResult()) {

byte[] data = cmd.getResult().getData();

if (null != data) {

RspGenerateLocationCoordinateData rspGenerateLocationCoordinateData = RspGenerateLocationCoordinateData.valueOf(data);

if (null != rspGenerateLocationCoordinateData) {

//TODO 处理相关业务 比如可抛出获取货道坐标事件等操作

}

}

}


return true;

}

});

/**

* 3.执行命令

*/

machine.executeCommand(cmd);

/********************************* end:生成货道坐标列表命令(异步)*********************************/
```

1.2 出货（0x04）

Linux 接收到该指令后，控制小车运动到指定货道，执行取货、出货、打开取货门等一系列动作。

注：执行出货指令之前，首先判断托盘是否有商品，如果有商品先运动到出货口让客户取走，然后再进行取货。

命令帧格式

命令	长度		数据域					
CMD	LEN	Mode	Count	1st.Goods				……
				Index	Area	Floor	Num	
0x04	2+4*Count							

【说明】

1. Mode ：出货模式，0 表示一次出单件商品，1 表示一次出多件商品。
2. Count ：购买商品总数，一次最多可指定 3 件商品。
3. Index ：出货商品序号，取值 0~2。
4. Area ： 商品所在区域。
5. Floor：商品所在层号。
6. Num：商品所在货道号。
7. 超时响应时间：20S。

（取货补充）通知帧格式

通知	长度	数据域				
ATT	LEN	Flag	Goods			
			Index	Area	Floor	Num
0x04	0x05	0x03				

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：取货补充编码，取值 0x03。
3. Index：补充的商品序号，取值 0~2。
4. Area ： 商品所在区域。
5. Floor：商品所在层号。
6. Num：商品所在货道号。

（待取货）通知帧格式

通知	长度	数据域
ATT	LEN	Flag
0x04	0x01	0x01

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：待取货码，取值 0x01。

（取货失败）通知帧格式

通知	长度	数据域	
ATT	LEN	Flag	Index
0x04	0x02	0x02	

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：取货失败码，取值 0x02。
3. Index：失败的商品编号，取值 0~2。

（出货前托盘非空）通知帧格式

通知	长度	数据域
ATT	LEN	Flag
0x04	0x01	0x04

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：待取货码，取值 0x04。

（结束补充）通知帧格式

通知	长度	数据域
ATT	LEN	Flag
0x04	0x01	0x05

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：结束补充标志，取值 0x05。

（取货成功）通知帧格式

通知	长度	数据域	
ATT	LEN	Flag	Index
0x04	0x01	0x06	

【说明】

1. ATT：通知码，取值与命令码相同。
2. Flag：取货成功标志，取值 0x06。
3. Index：失败的商品编号，取值 0~2。

响应帧格式

响应	长度	数据域						
RSP	LEN	ErrCode		MchSta	FailedCount	Index(1)	……	Index(N)
0x04	4+FailedCount							

【说明】

1. ErrCode：错误码，占两个字节。错误码列表参见附录。
2. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

3. FailedCoun：出货失败总数。
4. Index(x)：出货失败商品的序号，取值 0~2。

sdk 代码示例：

```
/*********************************start: 出货（0x04）(异步)*********************************/

/**

* 1.生成出货业务数据(一组出货最多出三个货道,如果单组出货超过一个货道,必须遵循智能出货规则,如果单组出货不超过一个则无需遵循智能出货规则)

*/

List<Location> locations=new ArrayList<>();

locations.add(location1);

locations.add(location2);

locations.add(location3);

final ReqOutGoods reqOutGoods = new ReqOutGoods(locations);

final byte[] data = ReqOutGoods.valueOf(reqOutGoods);



/**

* 2.构造出货命令(异步)

*/

Command cmd = Command.create(CommandDef.ID_SELLOUT, data, Command.TYPE_ASYNC);

cmd.setDesc(businessInfo);




/**

* 3.监听相关回调

*/

cmd.setOnReplyListener(new ICallbackListener() {

@Override

public boolean callback(Command cmd) {

String log=null;

Attention attention = cmd.getLastAttentionData();

SingleOutGoodsInfo singleOutGoodsInfo = null;

//出货失败，需要补发一条命令。

byte[] attData = attention.getData();

switch (attData[0]) {

case ATT_SELLOUT_NOT_TAKEN_AWAY:

log="上个用户没有把货取走，请先取走商品再继续出货";
```

```
Log.i(TAG,log);

break;

case ATT_SELLOUT_PICK_UP_COMPLETE:

log="第" + attData[1] + "个商品取货成功。";

Log.i(TAG,log);

break;

case ATT_SELLOUT_PICK_UP_FAILED:

ByteBuffer bb = new ByteBuffer(5);

log="第" + attData[1] + "个商品取货失败。";

Log.i(TAG,log);

int outGoodsFailedIndex = attData[1] & BmtMsgConstant.BYTE_MASK;

Location retryLocation = onOutGoods.getRetryLocation(outGoodsFailedIndex);

if (null == retryLocation) {

bb.append(ATT_SELLOUT_PICK_UP_NO_OPTION);

byte[] retry = cmd.sendAttention(bb.copy());

log="无重试货道";

Log.i(TAG,log);

} else {

String nextLocation = StringUtil.getLocation(retryLocation);


if (false == TextUtils.isEmpty(nextLocation)) {

Logger.info(TAG, "重试位置:" + JsonUtil.toJson(retryLocation));

Logger.info(TAG, "重试位置:" + nextLocation);

bb.append(ATT_SELLOUT_PICK_UP_RETRY).append(attData[1]).append(nextLocation);

byte[] retry = cmd.sendAttention(bb.copy());

if (null != retry) {

Logger.info(TAG, "重试指令:" + DataUtil.bytesToHexString(retry));

}


} else {

bb.append(ATT_SELLOUT_PICK_UP_NO_OPTION);

byte[] retry = cmd.sendAttention(bb.copy());

log="无重试货道";
```

```
Log.i(TAG,log);

}

}




break;

case ATT_SELLOUT_WAIT_TO_GET_AWAY:

log="取货口已打开，等待用户取走。";

Log.i(TAG,log);

break;

default:

}

return true;

}

});




cmd.setOnCompleteListener(new ICallbackListener() {

@Override

public boolean callback(Command cmd) {

Response response = cmd.getResult();

if(null!=response){

RspOutGoods rspOutGoods = RspOutGoods.valueOf(response.getData());

//TODO 根据出货结果,进行其它业务处理

}




return true;

}

});




cmd.setOnErrorListener(new ICallbackListener() {
```

```
@Override

public boolean callback(Command cmd) {



Response response = cmd.getResult();

if(null!=response){

RspOutGoods rspOutGoods = RspOutGoods.valueOf(response.getData());

//TODO 根据出货结果,进行其它业务处理

}



return true;

}

});



/**

* 4.执行出货命令

*/

machine.executeCommand(cmd);

/**********************************end: 出货（0x04）(异步)**********************************/



/**********************************start: 出货（0x04）(同步)**********************************/

/**

* 1.生成出货业务数据(一组出货最多出三个货道,如果单组出货超过一个货道,必须遵循智能出货规则,如果单组出货不超过一个则无需遵循智能出货规则)

*/

List<Location> locations=new ArrayList<>();

locations.add(location1);

locations.add(location2);

locations.add(location3);

ReqOutGoods reqOutGoods = new ReqOutGoods(locations);

byte[] data = ReqOutGoods.valueOf(reqOutGoods);

/**

* 2.构造命令(同步)

*/

Command cmd = Command.create(CommandDef.ID_SELLOUT, data, Command.TYPE_SYNC);
```

```
cmd.setDesc(businessInfo);


/**
* 3.执行命令
*/
machine.executeCommand(cmd);


/**
* 4.解析命令响应数据
*/
if (cmd.isError()){
Log.i(TAG,cmd.getErrorDescription());
}
if(null!=cmd.getResult()) {
byte[] rspData = cmd.getResult().getData();
if (null != rspData) {
RspOutGoods rspOutGoods  = RspOutGoods.valueOf(rspData);
//根据出货结果,进行其它业务处理
}
}
/*********************************start: 出货（0x04）(同步)*********************************/
```

1.3 新电源板继电器控制（0x0E）

Android 发送该指令给图像处理板（IPS）, 由 IPS 通知 ACCS 电源控制板控制继电器。

命令帧格式

命令	长度	数据域				
CMD	LEN	Index	Relay	Action	ReseTim	Reserve
0x0E	0x05					

【说明】

1. Index：指定的柜门号，取值 0~15。
2. Relay：继电器号 0~7，默认功能如下表：

0	1	2	3	4	5	6	7
预留	电子锁	门控	小车	一体机	LED	风机	压缩机

3. Action：动作码　1：ON　0： OFF
4. ReseTim：动作恢复时间　0：不需要恢复，1：1 秒后恢复，依此类推
5. Reserve：保留，默认为 0

6. 超时响应时间：10S。

响应帧格式

响应	长度	数据域							
RSP	LEN	ErrCode		MchSta	Index	Relay	Action	ReseTim	Reserve
0x0E	0x08								

【说明】

1. ErrCode：错误码，占两个字节。错误码列表参见附录。
2. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

3. Index：指定打开的柜门号，取值与命令帧一致。
7. Relay：继电器号 取值与命令帧一致
8. Action：动作码 取值与命令帧一致
9. ReseTim：动作恢复时间 取值与命令帧一致
10. Reserve：保留，取值与命令帧一致

```
/******************************** start:新电源板继电器控制（0x0E）同步)********************************/

/**

* 1.构造命令数据部分

*/

int areaIndex=0;

int relayIndex=0;

int action=0;

int resetTime=0;

ReqRelayControl reqRelayControl=new ReqRelayControl(  areaIndex, relayIndex, action, resetTime);

byte[] reqData= ReqRelayControl.valueOf(reqRelayControl);

/**

* 2.构造命令

*/

Command cmd = Command.create(CommandDef.ID_ACCS_CONTORL,reqData, Command.TYPE_SYNC);

/**

* 3.执行命令

*/

machine.executeCommand(cmd);

/**

* 4.解析命令响应数据

*/

if (cmd.isError()){

Log.i(TAG,cmd.getErrorDescription());

}


if(null!=cmd.getResult()) {

byte[] rspData = cmd.getResult().getData();

if(null!=reqData) {

RspRelayControl rspRelayControl = RspRelayControl.valueOf(rspData);

//TODO 处理相关业务

}

}
```

/******************************** end:新电源板继电器控制（0x0E）同步)*******************************/

1.4 查询设备参数（0x23）

设备处于空闲状态时，Android 可通过该指令读取设备参数。

命令帧格式

命令	长度	数据域
CMD	LEN	Resvd
0x23	0x01	0x00

【说明】

1. Resvd：保留。
2. 超时响应时间：100ms。

响应帧格式

响应	长度	数据域						
RSP	LEN	DeviceID（10 字符）			IPS_VER			
0x23	39+2*N	D[0]	……	D[9]				

MCS_VER				PCS_VER			

EXCS_VER				ACCS_VER			

RightLimit		TopLimit		PopPoint.x		PopPoint.y	

AearCnt	AreaPos_1		……	AreaPos_n	
N					

【说明】

1. DeviceID：设备号，由 10 个字符组成。
2. IPS_VER：图像处理板的固件版本号，由 4 个字节组成，版本号格式如下表所示。

IPS_VER	Bit[31]	0 表示发布版，1 表示测试版
	Bit[30:28]	硬件版本
	Bit[27:23]	主版本号
	Bit[22:16]	次版本号
	Bit[15:9]	发布日期（年），只表示十位和个位
	Bit[8:5]	发布日期（月）
	Bit[4:0]	发布日期（日）

3. MCS_VER：运动控制板的固件版本号，版本号格式同 IPS_VER。
4. PCS_VER：取货控制板的固件版本号，版本号格式同 IPS_VER。
5. EXCS _VER：外设控制板的固件版本号，版本号格式同 IPS_VER。
6. ACCS _VER：电气控制板的固件版本号，版本号格式同 IPS_VER。
7. RightLimit：右极限坐标值，单位 mm。
8. TopLimit：顶部极限坐标值，单位 mm。
9. PopPoint.x：出货口位置的 x 坐标，单位 mm。
10. PopPoint.y：出货口位置的 y 坐标，单位 mm。
11. AearCnt：货柜（或区域）的数量。
12. AreaPos_x：货柜（或区域）的起始坐标，单位 mm。

```
/*********************************start:查询设备参数（0x23）(同步)*********************************/

/**
 *1.构造命令(同步)
 */
Command cmd = Command.create(CommandDef.ID_GET_MAC_PARAMETER, Command.TYPE_SYNC);


/**
 * 2.执行命令
 */
machine.executeCommand(cmd);




/**
 * 3.解析命令响应数据
 */
if (cmd.isError()){
    Log.i(TAG,cmd.getErrorDescription());
}
if(null!=cmd.getResult()) {
    byte[] rspData = cmd.getResult().getData();
    if (null != rspData) {
        rspGetDeviceParameter = RspGetDeviceParameter.valueOf(rspData);
        //TODO 根据设备参数信息,进行其它业务处理
    }
}


/*********************************end:查询设备参数（0x23）(同步)*********************************/
```

1.5 查询货道坐标列表（0x24）

设备处于空闲状态时，Android 可通过该指令读取设备参数。

命令帧格式

命令	长度	数据域
CMD	LEN	Resvd
0x24	0x01	0x00

【说明】

1. CMD：命令码，取值 0x14。
2. Resvd：保留。
3. 超时响应时间：100ms。

响应帧格式

响应	长度	数据域							
RSP	LEN	ErrCode		MchSta	Count		Unit1		
							Area	Floor	Num
0x24	5+7*N								

数据域										
Unit1				……	Unit N					
PosX		PosY			Area	Floor	Num	PosX		PosY	

【说明】

1. RSP：响应码，取值与命令码相同。
2. ErrCode：错误码，占两个字节。错误码列表参见附录。
3. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

4. Count：货道总数量,取值范围 0 到 65535。
5. Area：货道所属区号，取值 0~16。
6. Floor：货道所属层号，取值 0~255。
7. Num：货道号，取值 0~255。
8. PosX：货道 X 坐标值，单位 mm。
9. PosY：货道 Y 坐标值，单位 mm。

注意：当响应帧分多帧上报时，只有第一帧包含 ErrCode、MchSta 和 ErrorCnt。

```
/*********************************start:查询货道坐标列表（0x24）(异步)*********************************/

        /**

         *1.构造命令(异步)

         */

        Command cmd = Command.create(ID_GET_LOCATION_DATA, null, Command.TYPE_ASYNC);

        /**

         * 2.监听相关回调

         */

cmd.setOnCompleteListener(new ICallbackListener() {
```

```
    @Override
    public boolean callback(Command cmd) {

        if (cmd.isReplyTimeout() || cmd.isTaskTimeout()) {
            return true;
        }

        if(null!=cmd.getResult()){
            byte[] data = cmd.getResult().getData();
            if(null!=data){
                RspGetLocationCoordinateData rspGetCargoRoadCoordinateData = RspGetLocationCoordinateData.valueOf(data);
                if (null != rspGetCargoRoadCoordinateData) {
                    //TODO  1.解析货道坐标
                    //TODO  2.货柜宽度=右极限+托盘外部宽度(300mm),货柜高度=顶部极限+托盘外部高度(210mm)
                    //TODO  3.根据 1 和 2 中数据解析货道宽度和高度

                }
            }

        }
        return true;
    }
});

/**
 * 3.执行命令
 */
machine.executeCommand(cmd);
/*********************************end:查询货道坐标列表（0x24）(异步)*********************************/
```

1.6 读写取货高度补偿值（0x29）

Android 通过该指令读取或写入取货高度补偿值

命令帧格式

命令	长度	数据域	
CMD	LEN	R/W	Compensation
0x29	0x02		

1、R/W :读写模式。0 表示读，1 表示写。
2、Compensation：取货高度补偿值
3、超时响应时间：10S。

响应帧格式

响应	长度	数据域				
RSP	LEN	ErrCode		MchSta	R/W	Compensation
0x0F	0x05					

【说明】
1. ErrCode：错误码，占两个字节。错误码列表参见附录。
2. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

3. R/W :读写模式。与命令帧取值一致。
4. Compensation：取货高度补偿值。

```
/********************************* start:读写取货高度补偿值（0x29）同步)*********************************/

/**

 * 1.构造命令数据部分

 */

byte opcode = 0;

byte compensation=0;

ReqRwPickOffset reqRwPickOffset=new ReqRwPickOffset(opcode,compensation);

byte[] reqData=ReqRwPickOffset.valueOf(reqRwPickOffset);



/**

 * 2.构造命令(同步)

 */

Command cmd = Command.create(CommandDef.ID_RW_PICK_OFFSET,reqData,Command.TYPE_SYNC);



/**
```

```
 * 3.执行命令

 */

machine.executeCommand(cmd);



if (cmd.isError()){

    Log.i(TAG,cmd.getErrorDescription());

}



/**

 * 4.解析命令响应数据

 */

if(null!=cmd.getResult()) {

    byte[] rspData = cmd.getResult().getData();

    if(null!=reqData) {

        RspRwPickOffset rspRwPickOffset = RspRwPickOffset.valueOf(rspData);

        //TODO 处理相关业务

    }

}

/********************************* end:读写取货高度补偿值（0x29）同步)*********************************/
```

1.7 读写出货口高度（0x2A）

Android 通过该指令读取或写入出货口高度

命令帧格式

命令	长度	数据域		
CMD	LEN	R/W	PopVal	
0x2A	0x02			

1、R/W :读写模式。0 表示读，1 表示写。
2、PopVal：出货口高度
3、超时响应时间：10S。

响应帧格式

响应	长度	数据域					
RSP	LEN	ErrCode		MchSta	R/W	PopVal	
0x0F	0x05						

【说明】

5. ErrCode：错误码，占两个字节。错误码列表参见附录。
6. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

7. R/W :读写模式。与命令帧取值一致。
8. PopVal：出货口高度。

```
/*********************************start:读写出货口高度（0x2A）(同步)*********************************/

        /**

         * 1.构造读写出货口高度（0x2A）命令数据部分

         */

        byte opcode=0;//0 表示读，1 表示写 可读 不可轻易写

        int popVal=0;//

        ReqRwPopVal reqRwPopVal=new ReqRwPopVal(opcode,popVal);

        byte[] reqData= ReqRwPopVal.valueOf(reqRwPopVal);



        /**

         * 2.构造读写出货口高度（0x2A）命令(同步)

         */

        Command cmd = Command.create(CommandDef.ID_RW_POP_VAL,reqData,Command.TYPE_SYNC);



        /**
```

```
        * 3.执行命令
        */
        machine.executeCommand(cmd);

        /**
        * 4.解析响应相关数据
        */
        if (cmd.isError()){
            printCmdErrorInfo(comId,cmd);
        }

        if(null!=cmd.getResult()){
            if(null!=cmd.getResult().getData()){
                byte[] data = cmd.getResult().getData();
                rspRwPopVal = RspRwPopVal.valueOf(data);
                //TODO 处理相关业务
            }
        }
/*********************************end:读写出货口高度（0x2A）(同步)*********************************/
```

1.8 重新找原点（0x51）

Android 通过该指令控制设备重新找原点，设备将完成计算 X 轴方向和 Y 轴方向的偏差值。

命令帧格式

命令	长度	数据域
CMD	LEN	Resvd
0x51	0x01	0x00

【说明】

1. CMD：命令码，取值 0x51。
2. Resvd：保留。
3. 超时响应时间：30S。

响应帧格式

响应	长度	数据域						
RSP	LEN	ErrCode		MchSta	ErrDistanceX		ErrDistanceY	
0x51	0x07							

【说明】

1. RSP：响应码，取值与命令码相同。

2. ErrCode：错误码，占两个字节。错误码列表参见附录。
3. MchSta：设备状态，占一个字节，状态定义如下表所示。

MchSta	Bit[7:6]	0 表示空闲，1 表示忙碌，2 表示设备故障。
	Bit[5]	0 表示托盘为空，1 表示托盘非空
	Bit[4]	0 表示小车非节能状态，1 表示节能状态
	Bit[3:0]	保留

4. ErrDistanceX：回到原点时在 X 轴方向的偏差（mm）。
5. ErrDistanceY：回到原点时在 Y 轴方向的偏差（mm）。

```
/*********************************start:重新找原点（0x51）(同步)*******************************/

/**

 * 1.构造命令(同步)

 */

Command cmd = Command.create(CommandDef.ID_GO_TO_ORIGIN, Command.TYPE_SYNC);



/**

 * 2.执行命令

 */

machine.executeCommand(cmd);



if (cmd.isError()){

   printCmdErrorInfo(comId,cmd);

}



/**

 * 3.解析命令响应数据

 */

if(null!=cmd.getResult()){

   if(null!=cmd.getResult().getData()){

      byte[] data = cmd.getResult().getData();

      RspGoToOrigin rspGoToOrigin = RspGoToOrigin.valueOf(data);

      //TODO 处理相关业务

   }

}

/*********************************end:重新找原点（0x51）(同步)*******************************/
```

2 附录

Table4-1 错误码列表

序号	错误码	错误描述	错误可能原因
1	0x0001	图像处理板与运动控制板通信超时	
2	0x0002	图像处理板与取货控制板通信超时	
3	0x0003	图像处理板与电池管理板通信超时	
4			
5	0x0101	小车电机不转	
6	0x0102	升降电机不转	
7	0x0103	小车电机堵转	
8	0x0104	升降电机堵转	
9	0x0105	小车电机编码器信号异常	
10	0x0106	升降电机编码器信号异常	
11	0x0107	X 轴零点传感器异常	
12	0x0108	Y 轴零点传感器异常	
13	0x0109	X 轴左限位传感器异常	
14	0x010A	X 轴右限位传感器异常	
15	0x010B	Y 轴上限位传感器异常	
16	0x010C	Y 轴下限位传感器异常	
17	0x010D	小车电机过流保护	
18	0x010E	升降电机过流保护	
19	0x0201	电磁铁伸出不到位	
20	0x0202	电磁铁缩回不到位	
21	0x0203	电极没有电压输出	
22	0x0204	电极过流保护	
23	0x0205	光感检测 H 传感器信号异常	
24	0x0206	光感检测 V 传感器信号异常	
25	0x0207	称重传感器信号异常	
26	0x0301	电池电量不足	
27	0x0302	电池充电电极接反	
28	0x0303	电池充电 MOS 无法接通	
29	0x0304	电池放电过流保护	
30	0x0305	电池电流检测信号异常	
31	0x0306	电池电压检测信号异常	