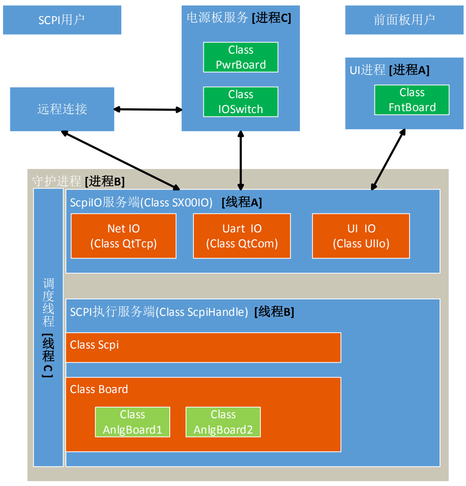

C1(AC链路连接)备注：

C1连接不足：

1、增加A/C进程通信复杂度

2、C进程已经有内部进程接口，单独调试A/C可直接通信，没必要连接

3、A进程通信职责只是发送SCPI指令给进程B；B应该负责所有的SCPI指令解析或转达

4、C进程职责应该只接受用户和SCPI服务的SCPI指令信息

C1不连接不足：

1、A/C之间通信不直接

设计思路：

SX00主程序设计方案采用多进程方案，分为A、B、C三个进程，每个进程独立运行在ARM上，进程之间交互采用socket方式。进程A为UI进程，负责UI展示以及前面板按键旋钮相关所有功能；进程B为守护进程，负责SCPI指令服务相关所有功能，是主程序中所有进程的服务端；进程C为电源板服务进程，负责电源板上业务相关所有功能(包括电源板上的IO)。

采用多进程方案便于将功能模块化独立出来，方便后期维护，也便于各模块独立调试。

进程阐述：

进程A：UI进程，所有UI展示、控件相关、前面板相关服务均在UI进程中实现。

进程B：守护进程，所有SCPI实际服务相关所有功能均在守护进程中实现，是设备工作的核心进程，单独进程能和其他非UI界面设备直接对接。

进程C：电源服务进程，所有电源板相关业务逻辑、IO切换均在电源服务进程，单独进程能适应没有电源板服务的设备也能正常使用A、B进程工作。

类申明：

```
Class ProcessB
{
	class threadA;
	class threadB;
	class threadC;

	public:
		/* IO服务、SCPI执行服务 */
		processServerListen();				//启动进程服务
		processDataTransfer();				//进程数据传输
		processServerClose();				//关闭进程服务

		/* 特殊命令交互 */
		scpiServerRead();				//读取scpi io命令
		scpiServerWrite();				//向scpi io写入数据

	private:
		Quequ mInQue;					//输入命令队列
		Quequ mOutQue;					//输出结果队列
};

Class threadA
{
	class QtTcp;					//网络用户
	class QtCom;					//电源板服务
	class UiIo;					//UI用户

	public:
		/* 与用户IO通信 */
		netServerListen();
		CommListen();
		UiIoListen();

		netClose();
		CommClose();
		UiIoClose();

		UiReadUserData();
		UiWriteToUser();

		netReadUserData();
		netWriteToUser();

		CommReadUserData();
		CommWriteToUser();

		/* 与SCPI执行服务端通信 */
		clientReadData();				//读取执行服务端数据
		clientWriteData();				//向执行服务端写入数据

		/* 与线程C通信 */
		cmdWrite();					//特殊指令
	private:
		Quequ mInQue;					//输入命令队列
		Quequ mOutQue;					//输出结果队列
};

Class threadB
{
	class scpi;					//scpi指令解析
	class board;					//功能子板

	public:
		/* 执行服务 */
		serverListen();				//启动scpi执行服务
		serverClose();				//关闭scpi执行服务
		serverRead();				//读取scpi io数据
		serverWrite();				//向scpi io写入数据

		/* 子板通信 */
		boardRead();				//读子板返回数据
		boardWrite();				//向子板写入数据

	private:
		Quequ mInQue;					//输入命令队列
		Quequ mOutQue;					//输出结果队列
};

Class threadC
{
	public:
		startThread();				//启动线程
		stopThread();				//停止线程

		/* 与scpi io通信 */
		cmdRead();				//读取特殊指令
};
```

进程B流程：

```
ProcessB main()
{
	pthread_t handlePid = 0, sx00IoPid = 0;

	/* 当前主线程为C, 负责调度和管理线程A、B*/

	pthread_create(handlePid);

	pthread_create(sx00IoPid);

	while(1)
	{
		/* 特殊管理命令处理 */
		switch(cmd)
		{
			case 0: restart(handlePid);

			case 1: stop(handlePid);

			case 2: restart(sx00IoPid);

			case 3: stop(sx00IoPid);

			default: break;
		}
	}

	pthread_join(sx00IoPid);

	pthread_join(handlePid);
}


handle_thread()
{
	while(1)
	{
		/* 数据处理 */
		scpi_handle();

		/* 功能执行 */
		board_handle();

		/* 结果返回 */
		result_out();
	}
}


sx00io_thread()
{
	/* 网络监听 */
	net_io_listen();

	/* 串口监听 */
	uart_io_listen();

	/* UI IO监听 */
	ui_io_listen();

	while(1)
	{
		/* UI数据传输 */
		transfer_ui_data();

		/* 网络数据传输 */
		transfer_net_data();

		/* 串口数据传输 */
		transfer_uart_data();
	}
}
```