文件编号(File No)：**Mxx-SRxxx**

NIBP软件详细设计

说明书

****

上海三埃弗电子有限公司研发中心

2015年 12 月

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **3F_logo1** 上海三埃弗电子有限公司 | | | | | | | |
| 主 题  Title | | NIBP软件详细设计说明书 | | 文 件 编 号  File No. | | Mxx-SRxxx | |
| 制订部门  By | | 研发中心 | | 版本号/修改状态  Revision | | V1.0 | |
| 发放部门(Issue to)：  ■研发中心 (R&D Center) | | | | 受控印章： | | | |
| **修 改 记 录** | | | | | | | |
| **版号/状态**  Revision | **修改内容**  Change Summary | | **制定/日期**  Author/Date | | **复核/日期**  Review/Date | | **批准生效/日期**  Approved/Date |
| V1.0 | 首次发布  Initial release | |  | |  | |  |
|  |  | |  | |  | |  |
|  |  | |  | |  | |  |
|  |  | |  | |  | |  |
|  |  | |  | |  | |  |
|  |  | |  | |  | |  |

**目 录**

[1 目的 5](#_Toc467600019)

[2 相关文档 5](#_Toc467600020)

[3 缩略语和定义 5](#_Toc467600021)

[4 范围 5](#_Toc467600022)

[5 框架设计 6](#_Toc467600023)

[5.1 NIBP参数层次图 6](#_Toc467600024)

[5.2 NIBP参数类图 7](#_Toc467600025)

[6 子部件描述 8](#_Toc467600026)

[6.1 TN3Provider 8](#_Toc467600027)

[6.2 NIBP设备通信协议 11](#_Toc467600028)

[6.3 NIBPParam 12](#_Toc467600029)

[6.4 NIBPState 14](#_Toc467600030)

[6.4.1 *NIBPStandbyState* 14](#_Toc467600031)

[6.4.2 *NIBPStartingState* 15](#_Toc467600032)

[6.4.3 *NIBPMeasureState* 15](#_Toc467600033)

[6.4.4 *NIBPGetResultState* 15](#_Toc467600034)

[6.4.5 *NIBPStopState* 16](#_Toc467600035)

[6.4.6 *NIBPSafeWaitTime* 16](#_Toc467600036)

[6.5 NIBPEventTrigger 16](#_Toc467600037)

[6.6 NIBPTrendWidget 17](#_Toc467600038)

[6.7 NIBPMenu 18](#_Toc467600039)

[6.8 NIBP Alarm 18](#_Toc467600040)

[7 任务设计 20](#_Toc467600041)

[7.1 指令收发功能 20](#_Toc467600042)

[7.2 数据解析功能 20](#_Toc467600043)

[7.3 超时功能 21](#_Toc467600044)

[8 业务逻辑处理 21](#_Toc467600045)

[8.1 状态转换 21](#_Toc467600046)

[8.1.1 *手动测量状态转换路径：* 21](#_Toc467600047)

[8.1.2 *连续测量状态转换路径：* 22](#_Toc467600048)

[8.1.3 *自动测量状态转换路径：* 24](#_Toc467600049)

[8.2 指令和数据流动 26](#_Toc467600050)

[9 错误监控 27](#_Toc467600051)

[10 附录 28](#_Toc467600052)

[10.1 NIBP状态切换详细列表 28](#_Toc467600053)

[10.1.1 *手动模式* 28](#_Toc467600054)

[10.1.2 *自动模式* 29](#_Toc467600055)

[10.1.3 *STAT模式* 31](#_Toc467600056)

[10.1.4 *额外测量状态* 33](#_Toc467600057)

[10.2 NIBP按钮触发方式 35](#_Toc467600058)

[10.2.1 *STAT使能* 35](#_Toc467600059)

[10.2.2 *STAT未使能* 38](#_Toc467600060)

# 目的

该文档详细描述iDM软件NIBP参数接入的详细设计，该设计将指导NIBP参数的编码开发与验证。

# 相关文档

|  |  |
| --- | --- |
| **文件** | **版本** |
| iDM Detailed Software Specification(Vital Sign- NIBP部分) | V1.0 |
| NIBP设备控制协议 | V1.0 |
| iDM软件架构设计说明书 | V1.0 |

# 缩略语和定义

|  |  |
| --- | --- |
| **缩略语和定义** | **说明** |
|  |  |
|  |  |
|  |  |
|  |  |

# 范围

# 框架设计

## NIBP参数层次图

Figure 5‑1描述了NIBP参数在iDM系统中的层次图。图中蓝色部分为数据层，实现数据的收发和协议封装和解析；黄色部分为业务逻辑层，负责指令处理和状态切换；绿色部分为UI层，实现NIBP功能控制和参数设置，以及可视化数据展示。



Figure 5‑1 NIBP Param Architecture Hierarchy Chart

Menu：提供用户操纵、设置NIBP参数的UI接口，包含载入配置、执行UI布局、更改测量模式、更改测量间隔、更改单位、更改PR显示等。

Widget：提供用户操纵、设置NIBP参数的UI接口，包含显示实时测量结果、设置实时袖带压、设置倒计时、更改PR显示等。

Alarm：提供用户显示和设置NIBP生理报警的UI接口，包含设置和获取报警上下限、使能报警、设置报警级别、保存报警波形等。

NIBPParam：NIBP参数主控制模块，执行或反馈UI层发出的各种指令，将TN3Provider上传的数据转发给NIBPStateMachine进行具体处理。

NIBPStateMachine：NIBP参数模块的状态机，NIBP参数模块共有6种工作状态，对于不同的指令和数据有不同的处理方式，状态机实现了指令和状态间的解耦。

NIBPTrigger：处理测量模式、病人类型、时间测量的触发，对触发后的状态进行处理；对安全间隔期的时间进行处理。

TN3Provider：管理数据包和指令的收发，解析设备协议数据，利用其内部的RingBuffer缓冲接收数据。

## NIBP参数类图

Figure 5‑2描述了NIBP参数的类关系图。

ParamManager为核心管理类，它维护着Provider对象和Param对象的关联信息。

Provider和Param是参数管理的两个顶层接口。

TN3Provider提供解析NIBP设备数据以及下发控制指令的功能。

NIBPParam对象是NIBP参数管理的核心。负责执行或反馈UI层发出的各种指令，将TN3Provider上传的数据转发给NIBPStateMachine状态对象进行具体处理。

NIBPState对象是状态机接口，定义了所有操作NIBP参数的公有方法，并提供了默认的实现。

NIBPEventTrigger对NIBPParam中的触发事件进行处理，将处理结果发送给NIBPState执行。

NIBPState对象是状态对象，继承自NIBPState，根据各自的状态重载NIBPState的公有方法。

NIBPTrendWidget对象继承自QWidget，实质是封装了一个Qt窗口对象，实现测量结果的文字显示和趋势图绘制，并提供控制NIBP参数的功能接口。



Figure 5‑2 NIBP Param Class diagram

# 子部件描述

## TN3Provider

TN3Provider承载主机与NIBP模块交互的重任，由ParamManager维护和管理。其主要作用有：管理数据包和指令的收发，解析设备数据，利用其内部的RingBuffer缓冲接收数据。TN3Provider提供了数据包下发的协议封装，使用者只需调用TN3Provider的相关接口便可以方便地将数据下发到主机，无需关心下发是如何实现的。TN3Provider的另外一个重要功能是读取串口来的数据包，并解析这些数据包。TN3Provider判断数据是否接收完整，如果接收完整无误，会调用NIBPParam对象的解包函数对该数据包进行处理。

TN3Provider继承自Provider和NIBPProviderIFace。并实现了Provider和NIBPProviderIFace定义的接口。

Provider对象是iDM系统所有数据提供对象（Provider）的父类，定义了数据提供对象需要实现的通用接口。

public:

// 发送协议命令

virtual void sendCmdData(unsigned char cmdId, const unsigned char \*data, unsigned int len);

// 构造与析构。

TN3Provider();

~TN3Provider();

protected:

// 读取数据到RingBuff中。

virtual void handlePacket(unsigned char \*data, int len);

private:

void \_sendACK(unsigned char type);

当TN3Provider实例被构造时，调用父类Provider的有参构造函数，并初始化Uart Port。Provider的构造函数初始化内部数据接收缓冲区RingBuffer，设置数据提供对象的名称，并利用Qt提供的信号槽机制，连接数据接收信号和槽函数，并在槽函数dataArrived()中实现具体的数据接收工作。

// TN3Provider类构造函数

TN3Provider::TN3Provider() : BLMProvider("BLM\_TN3"), NIBPProviderIFace()

{

UartAttrDesc portAttr(115200, 8, 'N', 1);

initPort(portAttr);

setDisconnectThreshold(5);

}

Provider::Provider(const QString &name) : QObject(), ringBuff(ringBuffLen), \_name(name)

{

connect(&uart, SIGNAL(activated(int)), this, SLOT(dataArrived()));

}

NIBPProviderIFace对象定义TN3Provider数据提供对象需要实现的特有接口方法。

public:

// 起停测量。

virtual void startMeasure(void) { }

virtual void stopMeasure(void) { }

// 设置预充气压力值。

virtual void setInitPressure(short /\*pressure\*/) { }

// 设置智能压力使能。

virtual void enableSmartPressure(bool /\*enable\*/) { }

// 设置病人类型。

virtual void setPatientType(unsigned char /\*type\*/) { }

// 发送启动指令是否有该指令的应答。

virtual bool needStartACK(void) { return false; }

virtual bool isStartACK(unsigned char \*/\*packet\*/) {return false;}

// 发送停止指令是否有该指令的应答。

virtual bool needStopACK(void) { return false; }

virtual bool isStopACK(unsigned char \*/\*packet\*/) {return false;}

// 压力数据，不是压力数据返回-1。

virtual int cuffPressure(unsigned char \*/\*packet\*/) {return -1;}

// 测量是否结束。

virtual bool isMeasureDone(unsigned char \*/\*packet\*/) {return false;}

// 是否为错误数据包。

virtual NIBPOneShotType isMeasureError(unsigned char \*/\*packet\*/)

{

return NIBP\_ONESHOT\_NONE;

}

// 发送获取结果请求。

virtual void getResult(void) { }

// 是否为结果包。

virtual bool isResult(unsigned char \*/\*packet\*/,short &/\*sys\*/,

short &/\*dia\*/, short &/\*map\*/, short &/\*pr\*/, NIBPOneShotType &/\*err\*/)

{

return false;

}

## NIBP设备通信协议

NIBP通信协议是iDM NIBP参数同NIBP设备进行指令和数据收发的桥梁，所有指令和数据都被封装进遵守协议格式的数据包内，然后提交给Uart端口发送和接收。NIBP参数利用TN3Provider类实现遵守协议格式的数据封装和解析。

具体的协议细节请参考《NIBP设备控制协议》，下表列出了当前协议支持的指令类型和指令传输方向。

|  |  |  |
| --- | --- | --- |
| 指令名称 | 指令码 | 指令方向 |
| 获取版本信息 | 0x10，0x11 | Host <-> Module |
| 病人类型 | 0x12，0x13 | Host <-> Module |
| 启动测量 | 0x14，0x15 | Host <-> Module |
| 停止测量 | 0x16，0x17 | Host <-> Module |
| 获取测量结果 | 0x06，0x07 | Host <-> Module |
| 启动帧 | 0x40 | Host <- Module |
| 测量结束帧 | 0x41 | Host <- Module |
| 测量模式下的压力帧 | 0x5C | Host <- Module |
| 原始数据帧 | 0x5D | Host <- Module |
| 进退维护模式 | 0x80，0x81 | Host <-> Module |
| 进入升级模式 | 0xF6，0xF7 | Host <-> Module |
| 测量错误警告帧 | 0x76 | Host <- Module |
| 文件片段传输 | 0xF8，0xF9 | Host <-> Module |
| 升级状态 | 0xFD | Host <- Module |
| 进退校准模式控制 | 0x82，0x83 | Host <-> Module |
| 校准点压力值输入 | 0x84，0x85 | Host <-> Module |
| 进退压力计模式控制 | 0x86，0x87 | Host <-> Module |
| 进退压力操控模式控制 | 0x88，0x89 | Host <-> Module |
| 维护模式下的压力帧 | 0xDB | Host <- Module |
| 压力控制 | 0x8A，0x8B | Host <-> Module |
| 放气控制 | 0x8C，0x8D | Host <-> Module |
| 保活帧 | 0x5B | Host <- Module |
| 数据应答 | 0x01，0x00 | Host -> Module |

## NIBPParam

NIBPParam对象是NIBP参数管理的核心。负责执行或反馈UI层发出的各种指令，将TN3Provider传递的数据进行具体处理。

NIBPParam继承自Param类，由ParamManager构造，关联TN3Provider、NIBPState、NIBPEventTrigger和NIBPTrendWidget。

Param类是iDM系统所有参数对象的父类，定义了参数对象需要实现的通用接口：

public:

// 初始化参数。

virtual void initParam(void);

// 处理DEMO数据。

virtual void handDemoWaveform(WaveformID id, short data);

virtual void handDemoTrendData(void);

// 构造与析构。

Param(ParamID id);

virtual ~Param();

除了实现以上的接口，NIBPParam为控制NIBP参数实现了如下成员函数：

// 短按NIBP触发 测量起停控制，起停反操作：如果当前处于测量则停止，当前为停止则开始测量。

void toggleMeasureShort(void);

//长按NIBP触发

void toggleMeasureLong(void);

// 停止测量。

void stopMeasure(void);

NIBPParam为UI层提供控制和反馈信息实现了如下成员函数：

void setTrendWidget(NIBPTrendWidget \*trendWidget); // 设置界面对象。

void setResult(short sys, short dia, short map, short pr, NIBPOneShotType err); // 设置测量结果。

void setResultTime(void); // 设置测量结果的时间。

void invResultData(void); // 清除显示值。

void setCuffPressure(int pressure); // 袖带压力。

void setCountdown(int t); // 更新倒计时信息，秒为单位。

void setText(QString text); // NIBP状态信息显示。

void setShowMeasureCount(bool lastMeasureCountflag); //上次测量时间的显示

void setMeasurMode(NIBPMode mode); // 设置测量模式

NIBPMode getMeasurMode(void); // 获取测量模式

void setAutoInterval(NIBPAutoInterval interv); // 设置取自动测量时的时间间隔

NIBPAutoInterval getAutoInterval(void); // 获取自动测量时的时间间隔

void setUnit(UnitType unit); // 设置单位

UnitType getUnit(void); // 获取单位

NIBPParam为从TN3Provider获取数据实现了如下成员函数：

// 设置数据提供对象。

void setProvider(NIBPProviderIFace \*provider);

// 解析数据包。

void unPacket(unsigned char \*packet, int len);

NIBPParam为操纵状态机实现了如下成员函数：

// 状态机运行。

void machineRun(void);

// 切换虚拟机状态。

void switchState(NIBPStateType state);

## NIBPState

NIBPState对象是状态机接口，定义了所有操作NIBP参数的公有方法，并提供了默认的实现。

NIBP\*State对象是状态对象，继承自NIBPState，根据各自的状态重载实现NIBPState的公有方法。NIBP参数模块共有6种工作状态：Standby（待机状态）、Starting（启动测量状态）、Measuring（正在测量状态）、Stoping（停止状态）、Get Results（获取结果状态）、SafeWaitTime（安全时间间隔），每个状态对于相同的指令和数据有不同的处理方式，状态机实现了指令和状态的解耦。

enum NIBPStateType

{

NIBP\_STATE\_STANDBY, // 待机状态。

NIBP\_STATE\_STARTING, // 启动测量状态。

NIBP\_STATE\_MEASURING, // 正在测量状态。

NIBP\_STATE\_STOPING, // 停止状态。

NIBP\_STATE\_GET\_RESULT, // 获取结果状态。

NIBP\_STATE\_SAFEWAITTIME, // 安全等待时间

NIBP\_STATE\_NR

};

NIBPState状态对象在NIBPParam对象构建时构造，并添加到NIBPParam对象内部的StateMap集合中。NIBPState状态对象在NIBPParam对象析构时被释放，遍历StateMap释放所有状态对象。

NIBPState对象是状态机接口，定义了所有操作NIBP参数的公有方法，如下所示：

public:

virtual void unPacket(unsigned char \*, int ) { } // 解析该数据包。

virtual void run(void) { } // 主运行。

virtual void enter(void) { } // 进入该状态。

6个状态对象根据自己的特点分别重载上述方法，若不需要进行处理，也可以放弃重载

### *NIBPStandbyState*

NIBPStandbyState对象表示待机状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void enter(void); | 进入本状态，显示测量模式，测量时间等信息 |

### *NIBPStartingState*

NIBPStartingState对象表示启动测量状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void run(void); | 等待超时，返回NIBPStandbyState状态 |
| void enter(void); | 进入本状态，向模块发送启动测量指令，并进入NIBPMeasureState状态 |
| void unPacket(unsigned char\*, int ); | 检测启动应答包，如检测到，进入NIBPMeasureState状态 |

### *NIBPMeasureState*

NIBPMeasureState对象表示正在测量状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void run(void); | 等待超时，返回NIBPStandbyState状态 |
| void enter(void); | 进入本状态，设置测量时限 |
| void unPacket(unsigned char\*, int ); | 检测数据包类型  1：压力数据，nibpParam向窗口UI设置压力  2：测量结束，进入NIBPGetResultState状态  3：异常数据，报警模块设置报警 |

### *NIBPGetResultState*

NIBPGetResultState对象表示获取结果状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void run(void); | 等待超时，返回NIBPStandbyState状态 |
| void enter(void); | 进入本状态，设置测量时限，向模块发送获取测量结果指令 |
| void unPacket(unsigned char\*, int ); | 检测数据包类型  1：测量结果数据，nibpParam向窗口UI设置测量结果  手动测量模式：返回NIBPStandbyState状态  其他测量模式：进入NIBPTimingState状态  2：其他数据，直接返回 |

### *NIBPStopState*

NIBPStopState对象表示停止状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void run(void); | 等待超时，返回NIBPStandbyState状态 |
| void enter(void); | 进入本状态，向模块发送停止测量指令，并进入NIBPStandbyState状态，设置等待超时 |
| void unPacket(unsigned char\*, int ); | 检测停止应答包，如检测到，进入NIBPStandbyState状态 |

### *NIBPSafeWaitTime*

NIBPTimingState对象表示测量倒计时状态。

|  |  |
| --- | --- |
| **重载方法** | **说明** |
| void run(void) | nibpParam更新倒计时，如果超时，进入NIBPStandbyState状态 |
| void enter(void); | 进入本状态，向模块发送停止测量指令，并进入倒计时 |

## NIBPEventTrigger

NIBPEventTrigger对各种触发事件进行统一的处理，触发事件包括：NIBP按键触发、修改病人类型触发、修改测量模式触发和stop触发，将各个处理状态处理完成后，发送到NIBPState进行相应状态的处理操作。

public:

void triggerbutton(void); //NIBP按键触发

void triggerpatienttype(void); //修改病人类型触发

void triggermodel(bool STAT\_Flag); //修改测量模式触发

void stop(void) { triggerbutton(); } //stop触发

int safeWaitTime(void); //安全时间间隔

private:

NIBPEventTrigger();

NIBPState \*\_currentState;

int \_timing;

int \_safeWaitTiming; //安全间隔倒计时时间

void \_switchToAuto(void); //状态转为AUTO

void \_switchToManual(void); //状态转为MANUAL

void \_safeWaitTimeSTATStop(void); //STAT在安全间隔内被关闭

## NIBPTrendWidget

NIBPTrendWidget类由windowManager管理，在构建NIBPParam对象后构建，在windowManager析构时被释放。

NIBPTrendWidget对象封装了Qt窗口对象，实现测量结果的实时文字显示和趋势图绘制，并提供控制NIBP参数的UI接口。用户能够利用该窗口操纵、设置NIBP参数、设置实时袖带压、设置倒计时、更改测量模式显示、测量时间的显示等。

NIBPTrendWidget对象实现如下成员函数：

public:

// 设置测量结果的数据。

void setResults(short sys, short dia, short map, unsigned time);

// 设置实时袖带压。

void setCuffPressure(int p);

// 设置倒计时。

void setCountdown(int t);

// 单位更改。

void setUNit(UnitType unit);

// 是否报警

void isAlarm(int id, bool flag);

// 显示血压值

void showValue();

// 显示字

void showText(QString text);

// NIBP显示框设置

void setShowStacked(int num);

//上次测量时间的显示

void setShowMeasureCount(bool lastMeasureCountflag);

// 显示STAT模式

void showModelText(QString text);

## NIBPMenu

NIBPMenu继承自SubMenu，扩展了Qt的菜单组建。该类主要提供NIBP各项控制参数的设置和载入，如下：

1. 测量模式的设置：手动、自动
2. 自动模式测量间隔的设置：2.5 min, 5 min, 10 min, 15 min, 20 min, 30 min, 45 min, 60 min, 90 min, 120 min
3. 启动STST测量

## NIBP Alarm

NIBP报警分为生理报警和技术报警，报警又分成高中低三个等级。NIBP报警功能由NIBPLimitAlarm和NIBPOneShotAlarm完成。

NIBPLimitAlarm继承自AlarmLimitIFace，用于处理生理超限报警；

NIBPOneShotAlarm继承自AlarmOneShotIFace，用于生理、技术和生命报警；

AlarmLimitIFace、AlarmOneShotIFace都继承自AlarmParamIFace。

AlarmParamIFace定义参数报警基本信息的接口：

public:

// 报警源的名字。

virtual const char \*getAlarmSourceName(void) = 0;

// 获取对应的参数ID。

virtual ParamID getParamID(void) = 0;

// 报警源的个数。

virtual int getAlarmSourceNR(void) = 0;

// 获取报警对应的波形ID，该波形将被存储。

virtual WaveformID getWaveformID(int id) = 0;

// 生理参数报警级别。

virtual AlarmPriority getAlarmPriority(int id) = 0;

AlarmLimitIFace定义参数超限报警信息的接口：

public:

// 获取指定的生理参数测量数据。

virtual int getValue(int id) = 0;

// 生理参数报警使能。

virtual bool isAlarmEnable(int id) = 0;

// 生理参数报警上限。

virtual int getUpper(int id) = 0;

// 生理参数报警下限。

virtual int getLower(int id) = 0;

// 将报警ID转换成字串。

virtual const char \*toString(int paramID, bool isHigh) = 0;

// 获取id对应的子参数ID。

virtual SubParamID getSubParamID(int id) = 0;

AlarmOneShotIFace定义生理、技术和生命报警的接口：

public:

// 该报警是否为生命报警，技术报警和生理/命报警分开存放。

virtual AlarmType getAlarmType(int id)

{

id = id;

return ALARM\_TYPE\_TECH;

}

virtual const char \*toString(int id) = 0; // 将报警ID转换成字串。

void clear(void); // 清除所有的报警。

virtual void setOneShotAlarm(int id, bool newStatus);

virtual bool isAlarmed(int id); // 报警条件是否满足，满足则返回true。

NIBPLimitAlarm类和NIBPOneShotAlarm类由Alarm类管理，在构建NIBPParam对象后构建，在Alarm析构时被释放。

NIBPLimitAlarm类和NIBPOneShotAlarm类根据NIBP参数的特有要求，实现其父类公开的接口。

# 任务设计

NIBP参数的所有功能全部在主线程Main Thread中完成。

主线程Main Thread是iDM的核心，大部分的任务将在此线程中完成。NIBP参数中，Main Thread负责NIBP参数的指令收发、数据解析和超时功能。

## 指令收发功能

iDM系统UI层接收用户对NIBP模块的操作指令。然后调用NIBPParam类中的成员函数，如：NIBPParam::toggleMeasure()执行适当的操作。然后将指令传递给NIBPState状态机，由当前状态机对象负责解析指令，并作出适当的状态切换或指令下发（发送指令至NIBP模块）。

如果需要下发指令，则调用TN3Provider的成员函数，如：TN3Provider.startMeasure()，该函数将指令封装成符合协议（7.4节）格式的数据包，并向Uart端口发送指令。

## 数据解析功能

槽函数dataArrived()将接收到的数据保存在RingBuffer缓冲区。当完整的数据包接收完毕，调用NIBPParam的unPacket进行数据的解析和处理。最终，NIBPParam会把数据包交给当前的状态机对象进行适当的处理和反馈。

## 超时功能

在系统初始化时，Main Thread利用Qt计时器，定义每隔500ms执行一次NIBP参数运行任务。该任务辅助完成NIBP参数的超时功能。

该任务每隔500ms执行一次nibpParam.machineRun()，machineRun()调用NIBPParam的当前状态机对象的run()成员函数完成等待超时控制。

除Standby状态之外，其他状态均有等待超时控制，一旦发生超时，便返回Standby状态。

# 业务逻辑处理

## 状态转换

### *手动测量状态转换路径：*

Figure 8‑1为NIBP参数在手动模式下的状态机图，系统默认初始状态为Standby，共有9条状态转换路径：



Figure 8‑1 NIBP State Machine（Manual mode）

|  |  |  |
| --- | --- | --- |
| **转换路径** | **状态变化** | **转换条件** |
| 1 | Standby | 系统默认初始状态为Standby |
| 2 | Standby->Starting | 启动测量，面板NIBP测量按键按下 |
| 3 | Starting->Measuring | Starting状态中向模块成功发送测量指令  1、有应答模块，等待应答进入Measuring状态  2、无应答模块，直接进入Measuring状态 |
| 4 | Starting->Stoping | 1、有应答模块，如果在设定的时间（2s）未得到应答则跳转到Stoping状态  2、面板NIBP测量按键按下 |
| 5 | Measuring->Stop | 1、Measuring状态的时间超过了3分钟，强行转到Stop状态  2、模块发送了测量异常信息，比如过压、超时等  3、面板NIBP测量按键按下 |
| 6 | Measuring->GetResults | Measuring状态中收到了模块发送上来的测量结束指令 |
| 7 | Stoping->Standby | Stop态中下发停止测量指令   1. 有应答模块，如果在设定的时间（2s）未得到应答则跳转到Standby状态 2. 有应答模块，如果在设定的时间（2s）内得到应答则跳直接跳转到Standby状态   3、无应答模块，直接转到Standby状态 |
| 9 | GetResults->Standby | 1. 等待2S未获取到结果则进入Standby状态 2. 手动模式下，2S内获得测量结果，则进入Standby状态 |

### *连续测量状态转换路径：*

Figure 8‑2为NIBP参数在连续模式下的状态机图，系统默认初始状态为Standby，共有11条状态转换路径：



Figure 8‑2 NIBP State Machine（AUTO, Continuous mode）

|  |  |  |
| --- | --- | --- |
| **转换路径** | **状态变化** | **转换条件** |
| 1 | Standby | 系统默认初始状态为Standby |
| 2 | Standby->Starting | 启动测量，面板NIBP测量按键按下 |
| 3 | Starting->Measuring | Starting状态中向模块成功发送测量指令  1、模块支持应答，并收到应答进入Measuring状态  2、模块不支持应答，直接进入Measuring状态 |
| 4 | Measuring->GetResults | Measuring状态中收到了模块发送上来的测量结束指令 |
| 5 | GetResults->Timing | 1、获取到测量结果，进入该倒计时状态  2、面板NIBP测量按键按下 |
| 6 | Timing->Starting | 倒计时超时后再次进入Starting状态 |
| 7 | Starting->Stoping | 1、面板NIBP测量按键按下  2、病人类型被切换或测量模式被切换 |
| 8 | Measuring-> Stoping | 1、Measuring状态的时间超过了3分钟，强行转到Stop状态  2、面板NIBP测量按键按下  3、病人类型被切换或测量模式被切换 |
| 9 | Stoping->Standby | Stop态中下发停止测量指令  1、模块支持应答，如果在设定的时间（2s）内得到应答则跳直接跳转到Standby状态  2、模块不支持应答，直接转到Standby状态  3、面板NIBP测量按键按下  4、测量模式被切换 |
| 10 | GetResults-> Standby | 1、病人类型被切换或测量模式被切换 |
| 11 | Timing-> Standby | 1、在连续测量模式中测量超过5分钟  2、面板NIBP测量按键按下  3、病人类型被切换或测量模式被切换 |
| 12 | Stoping-> Timing | 在测量时，病人类型被切换，停止后进入倒计时 |

模块支持应答，如果在设定的时间（2s）未得到应答则跳转到Standby状态；

### *自动测量状态转换路径：*

Figure 8‑2为NIBP参数在连续模式下的状态机图，系统默认初始状态为Standby，共有11条状态转换路径：



Figure 8‑2 NIBP State Machine（AUTO, Continuous mode）

|  |  |  |
| --- | --- | --- |
| **转换路径** | **状态变化** | **转换条件** |
| 1 | Standby | 系统默认初始状态为Standby |
| 2 | Standby-> Timing | 当测量模式设置为自动测量模式，开始倒计时  1、在Timing状态更改倒计时时间，倒计时重新开始  2、在Timing状态更改病人类型，倒计时时间不重新计时 |
| 3 | Timing ->Starting | 1、倒计时超时后再次进入Starting状态  2、面板NIBP测量按键按下 |
| 4 | Starting->Measuring | Starting状态中向模块成功发送测量指令  1、模块支持应答，并收到应答进入Measuring状态  2、模块不支持应答，直接进入Measuring状态 |
| 5 | Measuring->GetResults | Measuring状态中收到了模块发送上来的测量结束指令 |
| 6 | GetResults->Timing | 1、在连续测量和自动测量有效，每次测量结束后进入该倒计时状态  2、面板NIBP测量按键按下 |
| 7 | Starting->Stoping | 1、面板NIBP测量按键按下  2、病人类型被切换或测量模式被切换 |
| 8 | Measuring->Stoping | 1、Measuring状态的时间超过了3分钟，强行转到Stop状态  2、面板NIBP测量按键按下  3、病人类型被切换或测量模式被切换 |
| 9 | Stoping-> Timing | Stop态中下发停止测量指令  1、模块支持应答，如果在设定的时间（2s）内得到应答则跳直接跳转到Timing状态  2、模块不支持应答，直接转到Timing状态  3、测量时，病人类型被切换 |
| 10 | Stoping-> Standby | 在Stoping 状态下，测量模式被切换后，进入Standby状态 |
| 11 | Timing -> Standby | 在Timing状态下，测量模式被切换后，进入Standby状态 |
| 12 | GetResults-> Standby | 病人类型被切换或测量模式被切换 |

模块支持应答，如果在设定的时间（2s）未得到应答则跳转到Standby状态；

## 指令和数据流动

Figure 8‑3描述了NIBP工作的整个流程。从用户按下NIBP测量按键，到NIBPParam对象进行指令处理，以及NIBP模块开启测量并返回结果，再次通过 NIBPParam对象处理后，用户在UI上观察到测量过程和最终结果。

1）用户按下NIBP测量按键，或其他控制、配置指令

2）UI层的Widget或Menu类接收到用户指令，调用NIBPParam类相应的成员函数处理指令，并调整状态机对象，图中所示为Standby->Starting

3）状态机对象决定开始测量，向NIBP模块发送开启测量指令，调用TN3Provider对象的相关成员函数，图示中所为startMeasure()

4）startMeasure()负责将指令封装成遵守协议格式的数据包，并调用父类Provider的writeData函数将数据发送到主机Uart端口

5）Uart对象收到数据后立即将数据发送给NIBP模块的Uart端口

6）NIBP模块在其Uart端口上收到指令数据后，利用相同的协议进行解包，并作出适当的指令反馈，将该反馈按协议封包后发送到主机的Uart端口

7）如果NIBP模块判定能够开始测量，会向主机发送测量启动指令

8）主机的主线程监听Uart端口，当端口收到数据，立即缓存至RingBuffer，并调用槽函数dataArrived()传递数据

9）dataArrived()判断数据包的完整性和正确性，一旦确认，将调用NIBPParam的unPacket()函数进行指令或数据解析

10）NIBPParam类的unPacket()并不真实解析数据，只是简单的调用当前状态机对象的unPacket()函数，由状态机对象的unPacket()函数来处理指令或数据，并进行可能的状态切换，图中所示为Starting->Measuring

11）NIBP模块在完成步骤7之后，会启动测量，测量过程中将袖带压实时发送给主机，如果测量结束，将发送测量结束指令给主机

12）主机的Uart端口收到数据后，依然由NIBPParam类当前状态机对象的unPacket()函数进行处理，判断数据是袖带压还是测量结束指令。如果数据是袖带压，则调用NIBP的setCuffPressure()函数，该函数将袖带压显示到UI层的widget组件上，用户能够观察到袖带压力值的变化

13）如果触发技术报警条件，则调用NIBPOneShotAlarm的setOneShotAlarm()函数，将报警数据信息交给Alarm进行处理

14）如果数据是测量结束指令，状态机将调用NIBPParam的getResult()函数请求NIBP模块发送测量结果，并切换状态，图中所示为Measuring -> Get Results

15）NIBP模块收到指令后将测量结果发给主机，状态机判断收到的数据是否为测量结果，如果是，则调用NIBPParam的setResult()函数，将测量结果显示到UI层的Widget组件上，用户能够观察到血压的测量结果

16）如果触发生理报警条件，则调用NIBPLimitAlarm的setLimitAlarm()函数，将报警数据信息交给Alarm进行处理

*Figure 8‑3 NIBP Param Command&Data Flow*

# 错误监控

# 附录

## NIBP状态切换详细列表

### *手动模式*

执行顺序为：待机时->测量中->安全间隔内->待机时。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | STAT | 无STAT |
| 手动模式 | 测量中 | 按NIBP键 | 测量停止->安全间隔5s；显示测量时间，reading stopped，5s后界面为空； | √ | √ |
| 切换病人类型 | 测量停止->安全间隔5s；显示测量时间，reading stopped，5s后界面为空； | √ | √ |
| 切换到自动测量模式 | 进入自动模式，本次测量完成后，等待30s的自动安全间隔（倒计时时间从切入自动模式开始） | √ | √ |
| 打开STAT模式/长按NIBP键>=2s | 开始STAT测量  ->安全间隔5s->（安全间隔内显示“wait”或“failed”，不显示上次测量时间） | √ | X |
| 安全间隔内 | 按NIBP键 | 不响应，显示“wait”（安全间隔期后消失） | √ | √ |
| 切换病人类型 | 不影响 | √ | √ |
| 切换到自动测量模式 | 进入自动模式并开始倒计时，按NIBP键不响应，显示“wait” | √ | √ |
| 打开STAT模式/长按NIBP键>=2s | 打开STAT模式，并显示STAT打开，安全间隔5s后->开始STAT测量（安全间隔内显示“wait”，并显示STAT打开，在安全间隔内，打开STAT后，按NIBP可关闭STAT） | √ | X |
| 打开STAT模式且STAT测量未开始时，关闭STAT模式 | 退出STAT模式，显示STAT停止,上次测量的安全间隔5s后，不会启动STAT | √ | X |
| 待机时 | 按NIBP键 | 开始手动测量 | √ | √ |
| 切换病人类型 | 不影响 | √ | √ |
| 切换到自动测量模式 | 进入自动模式并开始倒计时 | √ | √ |
| 只切换自动测量时间（不切换到自动测量模式） | 无影响 | √ |  |
| 打开STAT模式/长按NIBP键>=2s | 开始STAT测量 | √ | X |

### *自动模式*

执行顺序为：倒计时中->测量中->安全间隔内->倒计时中。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | STAT | 无STAT |
| 自动模式 | 测量中 | 按NIBP键 | 本次测量停止->安全间隔30, | √ | √ |
| 手动启动的一次测量中，按NIBP键 | 本次测量停止->安全间隔5s、,显示”MANUAL”,并显示测量时间，安全间隔5s后消失，显示显示”AUTO” | √ | √ |
| 切换病人类型 | 本次测量停止->安全间隔30s，显示“reading stopped” | √ | √ |
| 切换到手动测量模式 | 完成本次测量后->进入手动模式->安全间隔5s  （切换后显示“manual”） | √ | √ |
| 打开STAT模式/长按NIBP键>=2s | 直接进入STAT，并开始倒计时，安全间隔5s  （显示STAT时间，安全间隔内显示“wait”，） | √ |  |
| 切换时间 | 以上次测量开始的时间为起点开始倒计时，倒计时时间到开始下次自动测量（从5min改为2.5min，对已用时间取2.5的余数） | √ | √ |
| 安全间隔内 | 自动测量倒计时时间到 | 跳过该次测量，倒计时继续 | √ | √ |
| 按NIBP键 | 不响应，显示“wait”（安全间隔时间到消失） | √ | √ |
| 切换病人类型 | 不影响 | √ | √ |
| 切换到手动测量模式 | 切换为手动模式，按NIBP键，显示“wait” | √ | √ |
| 打开STAT模式/长按NIBP键>=2s | 安全间隔30s内，打开STAT，显示wait，安全间隔30s后->开始STAT测量（在安全间隔内，打开STAT后，按NIBP关闭STAT） | √ | X |
| 测量中或安全间隔内打开STAT模式且STAT测量未开始时，关闭STAT模式 | 退出STAT模式，安全间隔30s后不启动STAT，显示STAT停止 | √ | X |
| 手动测量的安全间隔5s内打开STAT模式且STAT测量未开始时，关闭STAT模式 | 退出STAT模式，安全间隔5s后不启动STAT，显示STAT停止 | √ | X |
| 切换时间 | 以测量开始的时间为起点开始倒计时  （测量时间+30s安全间隔时间 > 倒计时时间，则跳过该次测量） | √ | √ |
| 倒计时中  （安全间隔30s后） | 按NIBP键 | 开始手动测量（手动测量结束与自动测量开始之间需要5s安全间隔时间，5s内，倒计时时间到，则跳过该次测量） | √ | √ |
| 切换病人类型 | 不响应倒计时，下次自动测量时修改病人类型 | √ | √ |
| 切换到手动测量模式 | 按NIBP键，开始 |  |  |
| 打开STAT模式/长按NIBP键>=2s | 立即开始 | √ |  |
| 切换时间 | 以上次测量开始的时间为起点开始倒计时，倒计时时间到开始下次自动测量 | √ | √ |

### *STAT模式*

执行顺序为：测量中->安全间隔5s内->测量中->安全间隔5s内，5min时间到或退出STAT测量时，执行安全间隔30s内。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | STAT |
| 连续模式 | 安全间隔30s内 | 按NIBP键 | 不响应，显示“wait” | √ |
| 切换病人类型 | 不影响 | √ |
| 原为自动模式，设定为手动模式 | 进入手动模式->安全间隔30s，30s内按NIBP键，显示“wait” | √ |
| 设定为自动模式 | 启动SATA时，原来设定为自动，如果安全间隔30s内，自动倒计时时间到，则跳过 | √ |
| 启动SATA时，原来设定为手动，在安全间隔30s时设置为自动，则立即进入自动模式并开始倒计时； | √ |
| 打开STAT模式/长按NIBP键>=2s | 安全间隔30s后，启动SATA测量（安全间隔内显示“wait”，并显示STAT打开，短按NIBP可关闭STAT） | √ |
| 安全间隔内打开STAT模式且STAT测量未开始时，关闭STAT模式 | 安全间隔30s后进入设定的测量模式（安全间隔提示显示STAT关闭） | √ |
| 5min时间未到，测量中 | 按NIBP键 | 本次测量停止->退出连续模式->安全间隔30s->进入设定模式 | √ |
| 设定为自动模式 | 启动SATA时，原来设定为手动，在STAT期间设置为自动，则自动倒计时时间立即开始开始倒计时 | √ |
| 在菜单中或按NIBP键关闭STAT模式 | 本次测量停止->退出连续模式->安全间隔30s->进入设定模式 | √ |
| 切换病人类型 | 本次测量停止->退出连续模式->安全间隔30s->进入设定模式 | √ |
| 5min时间未到，安全间隔5s内 | 按NIBP键， | 不响应，显示“wait” | √ |
| 只能在菜单中关闭STAT模式 | 退出连续模式->安全间隔30s->进入设定模式 | √ |
| 设定为自动模式 | 从手动设置为自动，则自动倒计时时间立即开始；开始倒计时，当STAT退出后，倒计时时间到则开始自动测量 | √ |
| 修改自动测量时间 | 以测量开始的时间为起点开始倒计时；修改时间对STAT测量无影响 | √ |
| 设定为手动模式 | 停止自动倒计时，对STAT测量无影响 | √ |
| 切换病人类型 | 退出连续模式->安全间隔30->进入设定模式s | √ |
| 临时打开的状态 | 切换病人类型 | STAT停止，安全间隔不改变 | √ |
| 按NIBP键或在菜单中关闭 | STAT停止，安全间隔不改变 | √ |
|  |  |  |
|  |  |  |

### *额外测量状态*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 额外充气 | 额外充气前的安全间隔 | 切换病人类型 | 手动模式 | 显示“stopped” 等5s | √ |
| 自动模式下的手动触发 | 显示“stopped” 等5s | √ |
| 自动模式下的自动测量 | 显示“stopped” 等30s | √ |
| 修改测量模式 | 手动到自动 | 显示“stopped”，等5s | √ |
| 自动模式的自动触发到手动 | √ |
|  | 自动模式的手动触发到手动 | √ |
| 按NIBP键 | 手动模式下 | 显示“stopped” 等5s，显示手动模式，上次测量时间， | √ |
| 自动模式下手动触发 | 显示“stopped” 等5s，显示手动模式，上次测量时间 | √ |
| 自动模式下的自动测量 | 显示“stopped” 等30s | √ |
| 额外充气中 | 按  NIBP键 | 手动模式下 | 显示“stopped” 等5s | √ |
| 自动模式下的手动触发 | 显示“stopped” 等5s | √ |
| 自动模式下的自动测量 | 显示“stopped” 等30s | √ |
| 切换病人类型 | 手动模式 | 显示“stopped” 等5s | √ |
| 自动模式下的手动触发 | 显示“stopped” 等5s | √ |
| 自动模式下的自动测量 | 显示“stopped” 等30s | √ |
| 修改测量模式 | 手动到自动 | 继续本次测量，再一次判断是否测量失败 | √ |
| 自动模式的自动触发到手动 | 继续本次测量，再一次判断是否测量失败 | √ |
| 自动模式的手动触发到手动 | 继续本次测量，再一次判断是否测量失败 | √ |

## NIBP按钮触发方式

### *STAT使能*

当STAT使能时，使用弹起触发，触发方式如下：

短按触发：在小于2s的时间内弹起，触发短按事件，

长按触发：当按压时间大于等于2s时，在2s的时刻触发长按事件，大于2s后弹起不再触发任何事件。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测量模式 | 测量状态 | 触发动作 | 触发事件 |  |
| 手动 | 测量中 | 长按 | 转STAT，开始测量 | √ |
| 短按 | 停止 | √ |
| 安全间隔 | 长按 | 显示wait，STST打开，安全间隔后开始STAT | √ |
| 短按 | 显示wait，无响应（若STAT状态被打开，则关闭） | √ |
| Standby | 长按 | 开始STAT | √ |
| 短按 | 开始手动测量 | √ |
| 自动 | 测量中 | 长按 | 转STAT，开始测量 | √ |
| 短按 | 本次测量停止，安全间隔30s | √ |
| 安全间隔 | 长按 | 显示wait，STST打开，安全间隔后开始STAT | √ |
| 短按 | 显示wait，无响应（若STAT状态被打开，则关闭） | √ |
| 倒计时（Standby） | 长按 | 转STAT，开始测量 | √ |
| 短按 | 开始一次手动测量 | √ |
| STAT | 临时打开状态 | 长按 | STAT关闭，不影响安全间隔 | √ |
| 短按 | STAT关闭，不影响安全间隔 | √ |
| 测量中 | 长按 | 退出STAT，进入安全间隔30s | √ |
| 短按 | 停止测量，退出STAT，进入安全间隔30s | √ |
| 安全间隔30s期间 | STST打开时  长按 | 显示STAT Stopped，STAT关闭，安全间隔后不会启动STAT | √ |
| STST关闭时  长按 | 显示wait，STST打开，安全间隔后开始STAT | √ |
| STST打开时  短按 | 显示STAT Stopped，STAT关闭，安全间隔后不会启动STAT | √ |
| STST关闭时  短按 | 显示wait，无响应 | √ |
| 安全间隔5s期间 | 长按 | 显示wait，无响应 | √ |
| 短按 | 显示wait，无响应（若要关闭STAT，需在菜单中关闭） | √ |

### *STAT未使能*

当STAT未使能时：使用按下触发

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测量模式 | 测量状态 | 触发动作 | 触发事件 |  |
| 手动 | 测量中 | 按下 | 停止 | √ |
| 安全间隔 | 按下 | 显示wait，无响应 | √ |
| Standby | 按下 | 开始手动测量 | √ |
| 自动 | 测量中 | 按下 | 本次测量停止 | √ |
| 安全间隔 | 按下 | 显示wait，无响应 | √ |
| 倒计时（Standby） | 按下 | 开始一次手动测量 | √ |