| Revision History | | | |
| --- | --- | --- | --- |
| Date | Version | By | Description of Change |
| 21 Aug, 2022 | 0.01 | Kunming Yang | First version |
| 29 Aug, 2022 | 0.02 | Jerry Wang | Define some Req |
| 09 Sep,2022 | 0.03 | Kunming Yang | Update as review |
| 19 Sep,2022 | 0.04 | Kunming Yang | Update as review |
| 21 Sep, 2022 | 0.05 | Jerry Wang | Define Weld Result and Weld Signature work flow |
| 22 Sep, 2022 | 0.06 | Kunming Yang | Update as review |
| 29 Sep, 2022 | 0.07 | Kunming Yang | Change type of some Table ID to long long |
| 09 Oct, 2022 | 0.08 | Kunming Yang | Create all 19 tables |
| 11 Oct, 2022 | 0.09 | Kunming Yang | Update tables based on DB file;  Add table AlarmLog |
| 13 Oct, 2022 | 0.10 | Kunming Yang | Add table HeightCalibration |
| 14 Oct, 2022 | 0.11 | Kunming Yang | Add table DbVersion |
| 15 Oct, 2022 | 0.12 | Kunming Yang | Add table UserProfiles and PrivilegeConfiguration |
| 22 Oct, 2022 | 0.13 | Kunming Yang | Add table PowerSupply/TeachModeSetting/SystemConfigure |
| 26 Oct, 2022 | 0.14 | Kunming Yang | Change 64-bit ID to 32-bit  Fix code for WeldRecipe |
| 27 Oct, 2022 | 0.15 | Kunming Yang | Update code for UserProfiles and PrivilegeConfiguration |
| 06 Nov, 2022 | 0.16 | Kunming Yang | Records read from database are separated by semicolon  Add table ActiveRecipe  Update StoreWeldRecipe |
| 08 Nov, 2022 | 0.17 | Kunming Yang | Table WeldRecipe has unique RecipeName  Restore separator to comma |

# Schedule

|  |  |
| --- | --- |
| Day | Sprint 1st |
| 1 | Database concurrent mechanism |
|  | 1st Week |
| 6 | Database architecture review |
|  | 2nd Week |
| 11 | Database initialization workflow review |
| 14 | Database Interface & message queue definition |
| 15 | Recipe, WeldResult, Weld Signature |
|  | 3rd Week |
| 21 | Code Review |
|  | Sprint 2nd |
|  | 1st Week |
| 25 | Performance testing |
|  | 2nd Week |
| 1 | Test Report Review |
| 2 | Database daemon task |
|  | 3rd Week |
| 8 | Code Review |

# DataBase

## General requirements

1. The database file should be named as sample\_l20\_base.db
2. The database file should be stored in /mmc1
3. The database should include 20 tables

## Table ActiveRecipe

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | ActiveRecipe | | | | | | |
| Primary Key | | -- | | | | | | |
| Other fields | | CycleNumber, BatchSize, OperateType, RecipeID, SequenceID | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | CycleNumber | INTEGER | N | N | Y | 0 | INTEGER | 0 |
|  | BatchSize | INTEGER | N | N | Y | 0 | INTEGER | 10 |
|  | OperateType | ENUM | N | N | Y |  | INTEGER | 0 |
|  | RecipeID | INTEGER | N | N | Y |  | INTEGER | 1 |
|  | SequenceID | INTEGER | N | N | Y |  | INTEGER | 1 |
| Sqlite Script | | --Create the table  CREATE TABLE "ActiveRecipe" (  "CycleNumber" INTEGER DEFAULT 0 UNIQUE,  "BatchSize" INTEGER DEFAULT 0 UNIQUE,  "OperateType" INTEGER UNIQUE,  "RecipeID" INTEGER UNIQUE,  "SequenceID" INTEGER UNIQUE,  FOREIGN KEY("RecipeID") REFERENCES "WeldRecipe"("ID") ON DELETE CASCADE  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | The table shall have only one record so the basic operand should have INSERT and DELETE. | | | | | | |

## Table AlarmLog

Max count 1000000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | AlarmLog | | | | | | |
| Primary Key | | ID | | | | | | |
| Sort other fields | | DateTime, AlarmType, PresetID, UserID, IsReset, WeldResultID | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER | 1 |
|  | DateTime | DATETIME | Y | N | N |  | TEXT | 12-12-2022 12:12:21 345 |
|  | AlarmType | ENUM | Y | N | N |  | INTEGER | 1 |
|  | RecipeID | INTEGER | N | N | N |  | INTEGER | 1 |
|  | UserID | INTEGER | Y | N | N |  | INTEGER | 1 |
|  | IsReseted | INTEGER | Y | N | N | 0 | INTEGER | 0 |
|  | WeldResultID | INTEGER | N | N | N |  | INTEGER | 1 |
| Sqlite script | | 1. --Create table   CREATE TABLE "AlarmLog" (  "ID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,  "DateTime" TEXT NOT NULL,  "AlarmType" INTEGER NOT NULL,  "RecipeID" INTEGER,  "WeldResultID" INTEGER,  "UserID" INTEGER NOT NULL,  "IsReset" INTEGER NOT NULL,  FOREIGN KEY("WeldResultID") REFERENCES "WeldResult"("ID") ON DELETE CASCADE  );   1. --Insert a record   (DateTime,AlarmType, RecipeID, UserID,IsReset,WeldResultID) VALUES (”2021-11-26 12:12:12 999”,1,1,1,0,999) | | | | | | |
| Tips | |  | | | | | | |
| Notes: | | The table operands should include INSERT for a whole record, DELETE oldest records using quantity, UPDATE “IsReset” field following ID, QUERY records using “IsReset”, Date and quantity. | | | | | | |

## Table Connectivity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | Connectivity | | | | | | |
| Primary Key | | -- | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example value |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | TEXT | Y | N | Y | 0 | TEXT |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 1.00 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0.00 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 4000 | INTEGER |  |
| Sqlite Scripts | | --Create the table  CREATE TABLE "Connectivity" (  "EthernetType" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "SignatureOption" INTEGER NOT NULL UNIQUE,  "ServerPort" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "DeviceIP" TEXT NOT NULL DEFAULT '127.0.0.1' UNIQUE,  "WeldResultOption" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "WeldRecipeOption" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "WeldSignatureOption" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "SystemConfigureOption" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "GatewayID" INTEGER NOT NULL DEFAULT 0 UNIQUE  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Please ignore it. | | | | | | |

## Table DBVersion

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | DBVersion | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | VersionNo | | | | | | |
| NO. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | Major | INTEGER | Y | N | Y |  | INTEGER | 1 |
|  | Minor | INTEGER | Y | N | Y |  | INTEGER | 1 |
|  | Build | INTEGER | Y | N | Y |  | INTEGER | 1 |
| Sqlite Scripts | | --Create the table  CREATE TABLE "DBVersion" (  "Major" INTEGER NOT NULL UNIQUE,  "Minor" INTEGER NOT NULL UNIQUE,  "Build" INTEGER NOT NULL UNIQUE  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | It should be a database key attribute for the whole system. We will manage database using it. | | | | | | |

## Table EventLog

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | EventLog | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | Datetime, EventType, EventName, UserID, Old, New, Comment | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER | 1 |
|  | Datetime | TEXT | Y | N | N |  | TEXT | 12-12-2021 12:12:12 |
|  | EventType | ENUM | Y | N | N |  | INTEGER | 1 |
|  | EventName | TEXT | Y | N | N |  | TEXT | “Modify Parameter” |
|  | UserID | INTEGER | Y | N | N |  | INTEGER | 1 |
|  | Old | TEXT | Y | N | N |  | TEXT | 5 |
|  | New | TEXT | Y | N | N |  | TEXT | 4 |
|  | Comment | VARCHAR(200) | N | N | N |  | TEXT | “Modify the param” |
| Sqlite script | | 1. --Create the table   CREATE TABLE "EventLog" (  "ID" INTEGER NOT NULL UNIQUE,  "Datetime" TEXT NOT NULL,  "EventType" INTEGER NOT NULL,  "EventName" TEXT NOT NULL,  "UserID" INTEGER NOT NULL,  "Old" TEXT NOT NULL,  "New" TEXT NOT NULL,  "Comment" TEXT,  PRIMARY KEY ("ID" AUTOINCREMENT)  );   1. --Insert a record   INSERT INTO EventLog (Datetime,EventType,EventName, UserID,Old,New,Comment) VALUES ("2021-11-11 12:12:12 666", 1, "Modify parameter", 1, "old", "New", "Changed the parameter") | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Same with Alarm Table, please ignore it firstly. | | | | | | |

## Table GatewayServer

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | GatewayServer | | | | | | |
| Primary | | ID | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y | 0 | INTEGER | 1 |
|  |  | TEXT | Y | N | Y |  | TEXT | 0 |
|  |  | INTEGER | Y | N | Y |  | INTEGER | 0 |
|  |  | TEXT | Y | N | Y |  | TEXT | 0 |
| Sqlite Scripts | | --Create the table  CREATE TABLE "GatewayServer" (  "ID" INTEGER NOT NULL UNIQUE,  "MachineName" TEXT NOT NULL UNIQUE,  "ServerPort" INTEGER NOT NULL UNIQUE,  "ServerIP" TEXT NOT NULL UNIQUE,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Please ignore it. | | | | | | |

## Table HeightCalibration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | HeightCalibration | | | | | | |
| Primary Key | | PSI | | | | | | |
| Other fields | |  | | | | | | |
| NO. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  |  | INTEGER | Y | N | N |  | INTEGER |  |
|  |  | INTEGER | Y | N | N |  | INTEGER |  |
| Sqlite Scripts | | --Create the table  CREATE TABLE "HeightCalibration" (  "PSI" INTEGER NOT NULL,  "Count" INTEGER NOT NULL,  PRIMARY KEY("PSI")  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Please ignore it | | | | | | |

## Table MaintenanceCounter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | MaintenanceCounter | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  |  | INTEGER | Y | Y | N |  | INTEGER |  |
|  |  | INTEGER | Y | N | N |  | INTEGER |  |
|  |  | INTEGER | Y | N | N | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | N | 0 | INTEGER |  |
|  |  | REAL | Y | N | N | 0 | REAL |  |
|  |  | INTEGER | Y | N | N | 0 | INTEGER |  |
|  |  | TEXT | Y | N | N |  | TEXT |  |
|  |  | INTEGER | Y | N | N | 0 | INTEGER |  |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "MaintenanceCounter" (  "ID" INTEGER NOT NULL UNIQUE,  "ToolingType" INTEGER NOT NULL,  "CurrentCounter" INTEGER NOT NULL DEFAULT 0,  "EnergySum" INTEGER NOT NULL DEFAULT 0,  "CounterLimit" REAL NOT NULL DEFAULT 0,  "EnergyLimit" INTEGER NOT NULL DEFAULT 0,  "StartDate" TEXT NOT NULL,  "AlarmOption" INTEGER NOT NULL DEFAULT 0,  PRIMARY KEY("ID" AUTOINCREMENT)  );   1. --Insert a record   INSERT INTO MaintenanceCounter VALUES (?,?,…) | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Please ignore it. | | | | | | |

## Table MaintenanceLog

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | MaintenanceLog | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGERT | Y | Y | Y | -- | INTEGER | 1 |
|  | MaintenanceType | INTEGER | Y | N | N | -- | INTEGER | 1(Calibration) |
|  | DateTime | TEXT | Y | N | N | -- | TEXT | 12-12-2021 12:12:11 |
|  | UserID | INTEGER | Y | N | N | -- | INTEGER | 1 |
| Sqlite Scripts | | --Create the table  CREATE TABLE "MaintenanceLog" (  "ID" INTEGER NOT NULL UNIQUE,  "MaintenanceType" INTEGER NOT NULL,  "DateTime" TEXT NOT NULL,  "UserID" INTEGER NOT NULL,  PRIMARY KEY ("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes: | | Shall be same with Alarm Log | | | | | | |

## Table PowerSupply

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table names | | PowerSupply | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | SequenceName, CreatedDate, UserID | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | Frequency | INTEGER | Y | N | N |  | INTEGER |  |
|  | Power | INTEGER | Y | N | N |  | INTEGER |  |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "PowerSupply" (  "ID" INTEGER NOT NULL UNIQUE,  "Frequency" INTEGER NOT NULL,  "Power" INTEGER NOT NULL,  PRIMARY KEY("ID" AUTOINCREMENT)  );   1. Insert a record | | | | | | |
| Tips | |  | | | | | | |
| Notes | |  | | | | | | |

## Table PrivilegeConfiguration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table names | | PrivilegeConfiguration | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | PermissionLevel | BOOLEAN | Y | N | N | 2 | INTEGER |  |
|  | ScreenIndex | BOOLEAN | Y | N | Y |  | INTEGER |  |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "PrivilegeConfiguration" (  "ID" INTEGER NOT NULL UNIQUE,  "PermissionLevel" INTEGER NOT NULL DEFAULT 2,  "ScreenIndex" INTEGER NOT NULL UNIQUE,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | | The ID means the screen number and the details is shown below table. There needs to insert 20 records when create the table. | | | | | | |
| Notes: | | Please ignore it | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Screen** | **ID** | **Screen** | **ID** | **Screen** | **ID** | **Screen** |
| 1 | Create Splice | 2 | Create Sequence | 3 | Edit Splice | 4 | Edit Sequence |
| 5 | Operate splice | 6 | Operate Sequence | 7 | Test | 8 | Teach Mode |
| 9 | Calibration | 10 | Advanced Maintenance | 11 | Maintenance Counter | 12 | Maintenance Log |
| 13 | Weld Result History | 14 | Statistical Trend | 15 | Alarm/Error Log | 16 | Library |
| 17 | Version Information | 18 | Permission Setting | 19 | Data Communication | 20 | Operator Library |

## Table SequenceRecipe

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | SequenceRecipe | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | SequenceID, Order, Quantity, PresetID | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite type | Example value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | SequenceID | INTEGER | Y | N | N |  | INTEGER |  |
|  | RecipeID | INTEGER | Y | N | N |  | INTEGER |  |
|  | Order | INTEGER | Y | N | N |  | INTEGER |  |
|  | Quantity | INTEGER | Y | N | N | 1 | INTEGER |  |
| Sqlite Scripts | | 1. --Create the table   CREATE TABLE "SequenceRecipe" (  "ID" INTEGER NOT NULL UNIQUE,  "SequenceID" INTEGER NOT NULL,  "RecipeID" INTEGER NOT NULL,  "Order" INTEGER NOT NULL,  "Quantity" INTEGER NOT NULL DEFAULT 1,  FOREIGN KEY("SequenceID") REFERENCES "WeldSequence"("ID") ON DELETE CASCADE,  FOREIGN KEY("RecipeID") REFERENCES "WeldRecipe"("ID") ON DELETE CASCADE,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes: | |  | | | | | | |

## Table SystemConfigure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | SystemConfigure | | | | | | |
| Primary Key | | -- | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Values |
|  |  | INTEGER | Y | N | Y | 0 | INTERGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTERGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTERGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTERGER |  |
|  |  | INTEGER | Y | N | Y | 1 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | N | 72 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 0 | INTEGER |  |
|  |  | INTEGER | Y | N | Y | 38000 | INTEGER |  |
| Sqlite script | | 1. --Create the table   CREATE TABLE "SystemConfigure" (  "FootPedalAbort" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "LockOnAlarm" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "HeightEncoder" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "CoolingOption" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "CoolingDuration" INTEGER NOT NULL DEFAULT 1.00 UNIQUE,  "CoolingDelay" INTEGER NOT NULL DEFAULT 0.00 UNIQUE,  "Language" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "AmplitudeUnit" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "PressureUnit" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "HeightUnit" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "MaxAmplitude" INTEGER NOT NULL DEFAULT 72,  "TeachModeID" INTEGER NOT NULL DEFAULT 0 UNIQUE,  "HomePositionCount" INTEGER NOT NULL DEFAULT 38000 UNIQUE); | | | | | | |
| Tips | |  | | | | | | |
| Notes: | | Please ignore it firstly. | | | | | | |

## Table TeachModeSetting

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | TeachModeSetting | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | |  | | | | | | |
| NO. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  |  | ENUM | Y | Y | Y | 0 | INTEGER |  |
|  |  | INT | Y | N | N | 0 | INTEGER |  |
|  |  | INT | Y | N | N | 40 | INTEGER |  |
|  |  | INT | Y | N | N | 40 | INTEGER |  |
|  |  | INT | Y | N | N | 25 | INTEGER |  |
|  |  | INT | Y | N | N | 25 | INTEGER |  |
|  |  | INT | Y | N | N | 15 | INTEGER |  |
|  |  | INT | Y | N | N | 15 | INTEGER |  |
|  |  | INT | Y | N | N | 10 | INTEGER |  |
|  |  | INT | Y | N | N | 10 | INTEGER |  |
|  |  | INT | Y | N | N | 15 | INTEGER |  |
| Sqlite Scripts | | --Create the table  CREATE TABLE "TeachModeSetting" (  "ID" INTEGER NOT NULL UNIQUE,  "TeachModeType" INTEGER NOT NULL DEFAULT 0,  "TimePLRG" INTEGER NOT NULL DEFAULT 40,  "TimeMSRG" INTEGER NOT NULL DEFAULT 40,  "PowerPLRG" INTEGER NOT NULL DEFAULT 25,  "PowerMSRG" NUMERIC NOT NULL DEFAULT 25,  "PreHeightPLRG" INTEGER NOT NULL DEFAULT 15,  "PreHeightMSRG" INTEGER NOT NULL DEFAULT 15,  "HeightPLRG" INTEGER NOT NULL DEFAULT 10,  "HeightMSRG" INTEGER NOT NULL DEFAULT 10,  "Quantity" INTEGER NOT NULL DEFAULT 15,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes: | | Please ignore it | | | | | | |

## Table UserProfiles

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | UserProfiles | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | UserName,CreateDate,WhoCreateNewID,Password,PermissionLevel | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Values |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | UserName | VARCHAR(100) | Y | N | Y |  | TEXT |  |
|  | DateTime | DATETIME | Y | N | N |  | TEXT |  |
|  | WhoCreatedNewID | INTEGER | Y | N | N |  | INTEGER |  |
|  | Password | TEXT | Y | N | N | 000000 | TEXT |  |
|  | PermissionLevel | ENUM | Y | N | N |  | INTEGER |  |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "UserProfiles" (  "ID" INTEGER NOT NULL UNIQUE,  "UserName" TEXT NOT NULL UNIQUE,  "DateTime" TEXT NOT NULL,  "WhoCreateNewID" INTEGER NOT NULL,  "Password" TEXT NOT NULL DEFAULT 000000,  "PermissionLevel" INTEGER NOT NULL,  PRIMARY KEY("ID" AUTOINCREMENT)   1. ); --Insert a record   INSERT INTO UserProfiles (UserName, DateTime,WhoCreateNewID,Password,PermissionLevel) VALUES ("Def","9999","ADMIN","123456",1) | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Please ignore it. | | | | | | |

## Table WeldRecipe

Max count 1000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | WeldRecipe | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | RecipeName, DateTime, UserID, PresetPicPath, IsVerified, Amplitude, Width, WeldPressure, TriggerPresure, TimePlus, TimeMinus, PeakPowerPlus, PeakPowerMinus, TriggerHeightPlus, TriggerHeightMinus, WeldHeightPlus, WeldHeightMinus, WeldMode, ModeValue, PreBurst, HoldTime, SqueezeTime, AfterBurstDelay, AfterBurstDuration, AfterBurstAmplitude, WeldHeight, MeasuredHeight, StepWeldMode, EnergyToStep, TimeToStep, PowerToStep | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | RecipeName | VARCHAR(100) | Y | N | Y |  | TEXT |  |
|  | DateTime | TEXT | Y | N | Y |  | TEXT |  |
|  | UserID | INTEGER | Y | N | N |  | INTEGER |  |
|  | PresetPicPath | VARCHAR | N | N | N |  | TEXT |  |
|  | IsValidate | BOOLEAN | Y | N | N | 0 | INTEGER |  |
|  | Amplitude | INTEGER | Y | N | N | 18 | INTEGER |  |
|  | Width | INTEGER | Y | N | N | 2000 | INTEGER |  |
|  | WeldPressure | INTEGER | Y | N | N | 20 | INTEGER |  |
|  | TriggerPressure | INTEGER | Y | N | N | 20 | INTEGER |  |
|  | TimePlus | INTEGER | Y | N | N | 5000 | INTEGER |  |
|  | TimeMinus | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | PeakPowerPlus | INTEGER | Y | N | N | 4800 | INTEGER |  |
|  | PeakPowerMinus | INTEGER | Y | N | N | 0 | INTEGER |  |
|  | TriggerHeightPlus | INTEGER | Y | N | N | 15000 | INTEGER |  |
|  | TriggerHeightMinus | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | WeldHeightPlus | INTEGER | Y | N | N | 15000 | INTEGER |  |
|  | WeldHeightMinus | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | WeldMode | ENUM | Y | N | N | 0 | INTEGER |  |
|  | ModeValue | INTEGER | Y | N | N | 0 | INTEGER |  |
|  | PreBurst | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | HoldTime | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | SqueezeTime | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | AfterBurstDelay | INTEGER | Y | N | N | 1000 | INTEGER |  |
|  | AfterBurstDuration | NUMERIC | Y | N | N | 0000 | NUMERIC |  |
|  | AfterBurstAmplitude | INTEGER | Y | N | N | 0000 | INTEGER |  |
|  | StepWeldMode | ENUM | N | N | N | -1 | INTEGER |  |
|  | EnergyToStep | JSON | N | N | N | 0 | BLOB | {"0": [1, 4, 0, 0, 7], "1": [2, 5, 0, 0, 8], "2": [3, 6, 0, 0, 9]} |
|  | TimeToStep | JSON | N | N | N | 0 | BLOB | {"0": [1, 4, 0, 0, 7], "1": [2, 5, 0, 0, 8], "2": [3, 6, 0, 0, 9]} |
|  | PowerToStep | JSON | N | N | N | 0 | BLOB | {"0": [1, 4, 0, 0, 7], "1": [2, 5, 0, 0, 8], "2": [3, 6, 0, 0, 9]} |
|  | WeldHeight | INTEGER | Y | N | N | 50 | INTEGER |  |
|  | MeasuredHeight | INTEGER | Y | N | N | 30 | INTEGER |  |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "WeldRecipe" (  "ID" INTEGER NOT NULL UNIQUE,  "RecipeName" TEXT NOT NULL,  "DateTime" TEXT NOT NULL,  "UserID" INTEGER NOT NULL,  "PresetPicPath" TEXT,  "IsValidate" INTEGER NOT NULL,  "Amplitude" INTEGER NOT NULL,  "Width" INTEGER NOT NULL,  "WeldPressure" INTEGER NOT NULL,  "TriggerPressure" INTEGER NOT NULL,  "TimePlus" INTEGER NOT NULL,  "TimeMinus" INTEGER NOT NULL,  "PeakPowerPlus" INTEGER NOT NULL,  "PeakPowerMinus" INTEGER NOT NULL,  "TriggerHeightPlus" INTEGER NOT NULL,  "TriggerHeightMinus" INTEGER NOT NULL,  "WeldHeightPlus" INTEGER NOT NULL,  "WeldHeightMinus" INTEGER NOT NULL,  "WeldMode" INTEGER NOT NULL,  "ModeValue" INTEGER NOT NULL,  "PreBurst" INTEGER NOT NULL,  "HoldTime" INTEGER NOT NULL,  "SqueezeTime" INTEGER NOT NULL,  "AfterBurstDelay" INTEGER NOT NULL,  "AfterBurstDuration" NUMERIC NOT NULL,  "AfterBurstAmplitude" INTEGER NOT NULL,  "StepWeldMode" INTEGER NOT NULL,  "EnergyToStep" BLOB,  "TimeToStep" BLOB,  "PowerToStep" BLOB,  "WeldHeight" INTEGER,  "MeasuredHeight" INTEGER,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | | Json format:  {Order0: [energy0, amplitude0],  Order1: [energy1, amplitude1],  Order2: [energy2, amplitude2]}  for example, {“0”: [0,0], “1”: [1, 100]} | | | | | | |
| Notes: | | The table operands should include INSERT for a new record, DELETE oldest records using quantity, UPDATE the record following ID, QUERY ID, DateTime and RecipeName using quantity; QUERY a record using ID. | | | | | | |

## Table WeldResult

Max count 1000000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | WeldResult | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | UserID, DateTime, SequenceID, RecipeID, WeldEnergy, TriggerPressure, WeldPressure, WeldAmplitude, WeldTime, WeldPeakPower, TriggerHeight, WeldHeight,AlarmFlags, CycleCounter | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER | 4294967298 (64bit) |
|  | PartID | TEXT[50] | Y | N | N |  | TEXT | 1 |
|  | DateTime | TEXT | Y | N | N |  | TEXT |  |
|  | RecipeID | INTEGER | N | N | N |  | INTEGER | 1 |
|  | WeldEnergy | INTEGER | Y | N | N |  | INTEGER | 100 |
|  | TriggerPressure | INTEGER | Y | N | N |  | INTEGER | 20.0(psi) \* 1000 |
|  | WeldPressure | INTEGER | Y | N | N |  | INTEGER | 20.0(psi) \* 1000 |
|  | WeldAmplitude | INTEGER | Y | N | N |  | INTEGER | 18(um) |
|  | WeldTime | INTEGER | Y | N | N |  | INTEGER | 1500(ms) |
|  | WeldPeakPower | INTEGER | Y | N | N |  | INTEGER | 100(W) |
|  | TriggerHeight | INTEGER | Y | N | N |  | INTEGER | 4250(micrometre) |
|  | WeldHeight | INTEGER | Y | N | N |  | INTEGER | 4250(micrometre) |
|  | AlarmFlag | INTEGER | Y | N | N |  | INTEGER | 0 |
|  | SequenceID | INTEGER | N | N | N |  | INTEGER | 1 |
|  | CycleCounter | INTEGER | N | N | N |  | INTEGER | 1000 |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "WeldResult" (  "ID" INTEGER NOT NULL UNIQUE,  "PartID" TEXT NOT NULL,  "DateTime" TEXT NOT NULL,  "RecipeID" INTEGER NOT NULL,  "WeldEnergy" INTEGER NOT NULL,  "TriggerPressure" INTEGER NOT NULL,  "WeldPressure" INTEGER NOT NULL,  "WeldAmplitude" INTEGER NOT NULL,  "WeldTime" INTEGER NOT NULL,  "WeldPeakPower" INTEGER NOT NULL,  "TriggerHeight" INTEGER NOT NULL,  "WeldHeight" INTEGER NOT NULL,  "AlarmFlag" INTEGER NOT NULL,  "SequenceID" INTEGER,  "CycleCounter" INTEGER,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | | Should same with Alarm Log. | | | | | | |

## Table WeldResultSignature

Max count 5000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | WeldResultSignature | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | | WeldResultID, WeldGraph | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER | 4294967298 (64bit) |
|  | WeldResultID | INTEGER | Y | N | Y |  | INTEGER | 4294967298 (64bit) |
|  | WeldGraph | JSON | N | N | N |  | BLOB | {"0": [1, 2, 34, 5, 6, 6],  "1": [1.5, 2, 1, 1.9, 2.0]} |
| Sqlite script | | 1. --Create the table   CREATE TABLE "WeldResultSignature" (  "ID" INTEGER NOT NULL UNIQUE,  "WeldResultID" INTEGER NOT NULL UNIQUE,  "WeldGraph" BLOB,  PRIMARY KEY("ID" AUTOINCREMENT),  FOREIGN KEY("WeldResultID") REFERENCES "WeldResult"("ID") ON DELETE CASCADE);   1. --Insert a record   INSERT INTO WeldResultGraph (WeldResultID, WeldGraph) VALUES (1," {"0": [1, 2, 34, 5, 6, 6],"1": [1.5, 2, 1, 1.9, 2.0]}") | | | | | | |
| Tips | | WeldGraph data format: {“WeldGraphType”: curve data}  The WeldGraph Type is a Enum type and its detail is shown in the chapter5.3. | | | | | | |
| Notes | | Should include INSERT and QUERY record following “WeldResultID”. | | | | | | |

## Table WeldSequence

Max count 5000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table name | | WeldSequence | | | | | | |
| Primary Key | | ID | | | | | | |
| Other fields | |  | | | | | | |
| No. | Column | Type | Not Null | Auto Increment | Unique | Default | Sqlite Type | Example Value |
|  | ID | INTEGER | Y | Y | Y |  | INTEGER |  |
|  | SequenceName | INTEGER | Y | N | N |  | TEXT |  |
|  | DateTime | JSON | Y | N | N |  | TEXT |  |
|  | UserID | INTEGER | Y | N | N |  | INTEGER |  |
| Sqlite script | | 1. --Create the table   CREATE TABLE "WeldSequence" (  "ID" INTEGER NOT NULL UNIQUE,  "SequenceName" TEXT NOT NULL UNIQUE,  "DateTime" TEXT NOT NULL,  "UserID" INTEGER NOT NULL,  PRIMARY KEY("ID" AUTOINCREMENT)  ); | | | | | | |
| Tips | |  | | | | | | |
| Notes | |  | | | | | | |

# DataTask

## General requirements

1. When class DataTask is created, it should open database “sample\_l20\_base.db” using DBAccessL20DB::ConnectDB
2. When class DataTask is created, it should get message queue id of "/Control"
3. When class DataTask is created, it should get message queue id of "/Data"
4. When class DataTask is created, it should get message queue id of "/Request"
5. When class DataTask is destroyed, it should close database “sample\_l20\_base.db” using DBAccessL20DB::CloseDataBaseConnection

## Message Processing

The DataTask should employ 3 separate message queues based on priority to allow other tasks indirect access to the data storage for reading as well as writing.

1. DataTask should process all messages from the CONTROL queue first.
2. Then DataTask should process messages from the DATA queue, then it should check for new messages from the CONTROL queue.
3. Then DataTask should process messages from the REQUEST queue, then it should check for new messages from the CONTROL queue and the DATA queue.



1. The message processing flow of the data task is triggered by event.
2. The message struct should be defined in Common.h as

struct MESSAGE

{

UINT32 msgID;

char Buffer[MAX\_SIZE\_OF\_MSG\_LENGTH - sizeof(msgID)];

};

1. The length of structure should not be out of the Buffer range.

## Database Processing

1. DataTask should provide method to open the database.
2. DataTask should provide method to close the database.

### Table ActiveRecipe

1. DataTask should provide method to query the unique record from table ActiveRecipe.
2. DataTask should provide method to update the unique record for table ActiveRecipe.
3. Store the CycleNumber data from table into WeldResult.CycleCounter
4. store BatchSize into Recipe::ActiveRecipeSC.m\_BatchSize
5. store RecipeID into Recipe::ActiveRecipeSC.m\_RecipeID.

### Table AlarmLog

1. DataTask should provide method to insert new record for AlarmLog.
2. Data for table AlarmLog should be extracted from AlarmDataSC::AlarmData
3. DataTask should provide method to query block record from table AlarmLog.
4. The result queried from table AlarmLog should be parsed into vector AlarmDataSC::AlarmLog.

Jerry: Please hold on AlarmLog and WeldResult Block Query.

### Table DBVersion

1. DataTask should provide method to query record from table DBVersion.
2. The result queried from table DBVersion should be parsed into CommonProperty::SystemInfo.version\_DB.

### Table HeightCalibration

1. DataTask should provide method to query record from table HeightCalibration.
2. The result queried from table HeightCalibration should be parsed into ZeroCount of map HeightEncoder::HeightCalibratedMap based on PSI.
3. DataTask should provide method to update the record for table HeightCalibration.
4. The data to be updated into table HeightCalibration is from ZeroCount of map HeightEncoder::HeightCalibratedMap based on PSI.

### Table WeldRecipe

1. DataTask should provide method to insert new record with unique RecipeName for table WeldRecipe.
2. Data for WeldRecipe should be extracted from Recipe::RecipeSC
3. When the count of records reaches the limit of table “WeldRecipe”, the oldest one should be deleted.
4. DataTask should provide method to update the record for table Recipe, meanwhile, the column DataTime should be update following current time stamp.
5. The record to be updated for table Recipe should be extracted from Recipe::RecipeSC.
6. DataTask should provide method to query for a record using ID of WeldRecipeSC from table WeldRecipe.
7. The result queried from table WeldRecipe should be parsed into Recipe::RecipeSC.
8. DataTask should provide method to query for the last page from table WeldRecipe.
9. DataTask should provide method to query for the next page from table WeldRecipe.
10. The page queried from table WeldRecipe should be parsed into Recipe::WeldRecipeLibraryForUI.

### Table WeldResult

1. DataTask should provide method to insert new record for table WeldResult, with table WeldResultSignature and AlarmLog.
2. Data for WeldResult should be extracted from CommonProperty::WeldResult
3. When the count of records exceeds the limit of table “WeldResult”, the oldest record should be removed.
4. DataTask should provide method to query 50 records using ID from table WeldResult.
5. The result queried from table WeldResult should be parsed into CommonProperty::WeldResultForUI[50].

### Table WeldResultSignature

1. DataTask should provide method to insert new record for tableWeldResultSignature.
2. Data for WeldResultSignature should be extracted from CommonProperty::WeldSignatureVector
3. The size of vector in CommonProperty::WeldSignatureVector should not exceed 200.
4. When the count of records exceeds the limit of table “WeldResultSignature”, the oldest record should be removed.
5. DataTask should provide method to query record of WeldGraph from table WeldResultSignature.
6. The result queried from table WeldResultSignature should be...

### Table UserProfiles

1. DataTask should provide method to query all PermissionLevel and Password from table UserProfiles.
2. The PermissionLevel and Password queried from table UserProfiles should be paired into map<int, string>\* \_UserProfilesSC.
3. DataTask should provide method to update the Password for table UserProfiles using PermissionLevel.

### Table PrivilegeConfiguration

1. DataTask should provide method to query all ScreenIndex and PermissionLevel from table PrivilegeConfiguration.
2. The ScreenIndex and PermissionLevel queried from table PrivilegeConfiguration should be paired into map<int, int>\* \_UserPrivilegesSC.
3. DataTask should provide method to update PermissionLevel for table PrivilegeConfiguration using ScreenIndex.

### Table PowerSupply

1. DataTask should provide method to query all records from table PowerSupply.
2. The records queried from table PowerSupply should be parsed into vector<POWER\_SUPPLY\_TYPE> SystemConfiguration::PowerSupplyType.
3. DataTask should provide method to update all records for table PowerSupply.
4. The data to be updated into table PowerSupply is from vector<POWER\_SUPPLY\_TYPE> SystemConfiguration::PowerSupplyType.

### Table TeachModeSetting

1. DataTask should provide method to query all records from table TeachModeSetting.
2. The records queried from table TeachModeSetting should be parsed into vector<TEACH\_MODE\_SETTING> SystemConfiguration::TeachModeSetting.
3. DataTask should provide method to update all records for table TeachModeSetting.
4. The data to be updated into table TeachModeSetting is from vector<TEACH\_MODE\_SETTING> SystemConfiguration::TeachModeSetting.

### Table SystemConfigure

1. DataTask should provide method to query the unique record from table SystemConfigure.
2. The records queried from table SystemConfigure should be set into SYSTEMCONFIG\* SystemConfiguration::\_SystemConfig.
3. DataTask should provide method to update the unique record for table SystemConfigure.
4. The data to be updated into table SystemConfigure is gotten from SYSTEMCONFIG\* SystemConfiguration::\_SystemConfig.

### Table Connectivity

1. DataTask should provide method to query the unique record from table Connectivity.
2. The records queried from table Connectivity should be set into ETHERNET Connectivity::EthernetConfig
3. DataTask should provide method to update the unique record for table Connectivity.
4. The data to be updated into table Connectivity is gotten from ETHERNET Connectivity::EthernetConfig

### Table GatewaySever

1. DataTask should provide method to query the unique record from table GatewaySever.
2. The records queried from table Connectivity should be set into vector<struct GATEWAY\_SERVER>\* Connectivity::\_DIGServersUI