| 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 |
|  |  |  |  |
|  |  |  |  |

# 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 19 tables, with names
   1. AlarmLog
   2. SystemConfiguration
   3. EventLog
   4. MaintenanceCounter
   5. UserProfile
   6. PrivilegeConfiguration
   7. PrivilegeLevelName
   8. Sequence
   9. SequencePreset
   10. WeldRecipe
   11. LastOperateConfiguration
   12. Communication
   13. GlobalSetting
   14. WeldResult
   15. WeldResultSignature
   16. HeightCalibration
   17. MaintenanceLog
   18. TeachmodeConfiguration
   19. DatabaseVersion
4. To be supplemented

## 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 | 1 |
|  | RecipeName | VARCHAR(100) | Y | N | Y |  | TEXT | “aaa” |
|  | DateTime | TEXT | Y | N | Y |  | TEXT | 2022-01-31 01:02:03 |
|  | UserID | INTEGER | Y | N | N |  | INTEGER | 1 |
|  | PresetPicPath | VARCHAR | N | N | N |  | TEXT | "D: \picture” |
|  | IsValidate | BOOLEAN | Y | N | N | 0 | INTEGER | 0(false) |
|  | Amplitude | INTEGER | Y | N | N | 18 | INTEGER | 18(um) |
|  | Width | INTEGER | Y | N | N | 2000 | INTEGER | 2000(micrometer) |
|  | WeldPressure | INTEGER | Y | N | N | 20 | INTEGER | 20(psi) \* 1000 |
|  | TriggerPressure | INTEGER | Y | N | N | 20 | INTEGER | 20(psi) \* 1000 |
|  | TimePlus | INTEGER | Y | N | N | 5000 | INTEGER | 5000(ms) |
|  | TimeMinus | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | PeakPowerPlus | INTEGER | Y | N | N | 4800 | INTEGER | 4800(W) |
|  | PeakPowerMinus | INTEGER | Y | N | N | 0 | INTEGER | 0(W) |
|  | TriggerHeightPlus | INTEGER | Y | N | N | 15000 | INTEGER | 15000(micrometer) |
|  | TriggerHeightMinus | INTEGER | Y | N | N | 0000 | INTEGER | 0000(micrometer) |
|  | WeldHeightPlus | INTEGER | Y | N | N | 15000 | INTEGER | 15000(micrometer) |
|  | WeldHeightMinus | INTEGER | Y | N | N | 0000 | INTEGER | 0000(micrometer) |
|  | WeldMode | ENUM | Y | N | N | 0 | INTEGER | 0(Energy) |
|  | ModeValue | INTEGER | Y | N | N | 0 | INTEGER | 100(J) |
|  | PreBurst | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | HoldTime | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | SqueezeTime | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | AfterBurstDelay | INTEGER | Y | N | N | 1000 | INTEGER | 1000(ms) |
|  | AfterBurstDuration | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | AfterBurstAmplitude | INTEGER | Y | N | N | 0000 | INTEGER | 0000(ms) |
|  | WeldHeight | INTEGER | Y | N | N | 50 | INTEGER | 50(micrometer) |
|  | MeasuredHeight | INTEGER | Y | N | N | 30 | INTEGER | 30(micrometer) |
|  | StepWeldMode | ENUM | N | N | N | -1 | INTEGER | -1(disable) |
|  | 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]} |
| Sqlite scripts | | 1. --Create the table   CREATE TABLE "Preset" (  "ID" INTEGER NOT NULL UNIQUE,  "RecipetName" TEXT NOT NULL UNIQUE,  "DateTime" TEXT NOT NULL,  "UserID" INTEGER,  "PresetPicPath" TEXT,  "IsVerified" INTEGER NOT NULL DEFAULT 0,  "Amplitude" INTEGER NOT NULL DEFAULT 18,  "Width" INTEGER NOT NULL DEFAULT 2000,  "WeldPressure" INTEGER NOT NULL DEFAULT 20000,  "TriggerPressure" INTEGER NOT NULL DEFAULT 20000,  "TimePlus" INTEGER NOT NULL DEFAULT 5000,  "TimeMinus" INTEGER NOT NULL DEFAULT 0,  "PowerPlus" INTEGER NOT NULL DEFAULT 4800,  "PowerMinus" INTEGER NOT NULL DEFAULT 0,  "TriggerHeightPlus" INTEGER NOT NULL DEFAULT 15000,  "TriggerHeightMinus" INTEGER NOT NULL DEFAULT 0,  "WeldHeightPlus" INTEGER NOT NULL DEFAULT 15000,  "WeldHeightMinus" INTEGER NOT NULL DEFAULT 0,  "WeldMode" INTEGER NOT NULL DEFAULT 0,  “ModeValue” INTEGER NOT NULL DEFAULT 100,  "PreBurst" INTEGER NOT NULL DEFAULT 0,  "HoldTime" INTEGER NOT NULL DEFAULT 0,  "SqueezeTime" INTEGER NOT NULL DEFAULT 0,  "AfterBurstDelay" INTEGER NOT NULL DEFAULT 1000,  "AfterBurstDuration" INTEGER NOT NULL DEFAULT 0,  "AfterBurstAmplitude" INTEGER NOT NULL DEFAULT 0,  "WeldHeight" INTEGER NOT NULL DEFAULT 0,  "MeasuredHeight" INTEGER NOT NULL DEFAULT 0,  "StepWeldMode" INTEGER DEFAULT 0,  "EnergyToStep" BLOB DEFAULT 0,  "TimeToStep" BLOB DEFAULT 0,  "PowerToStep" BLOB DEFAULT 0,  FOREIGN KEY("UserID") REFERENCES "UserProfiles"("ID"),  PRIMARY KEY("ID" AUTOINCREMENT)  );   1. --Insert a record   INSERT INTO Preset (PresetName,CreatedDate,UserID,PresetPicPath,Verified,..) VALUE ("aaa","2021-11-11 12:12:12 666",1,"D:\yz\Other\picture",0,..) | | | | | | |
| Tips | | Json format: {"0":[Order1,Order2,Order3,Order4,Order5],"1":[energy1,energy2,energy3,energy4,energy5],"2":[amplitude1,amplitude2,amplitude3,amplitude4,amplitude5]]}  {“Order”: [energy, amplitude]} 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 | 1 |
|  | 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,  "SequenceID" INTEGER,  "RecipeID" INTEGER,  "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,  "CycleCounter” INTEGER,  PRIMARY KEY("ID" AUTOINCREMENT)  );   1. --Insert a record   INSERT INTO WeldResult (PartID, DateTime, SequenceID, RecipeID, WeldEnergy, TriggerPressure, WeldPressure, WeldAmplitude, WeldTime, WeldPeakPower, TriggerHeight, WeldHeight, AlarmFlag, CycleCounter) VALUES (“1”, “2021-11-11 12:12:12”,1,1,100,20000,20000,18,1500,100,100,4250,4250,0,200) | | | | | | |
| 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 | 1 |
|  | WeldResultID | INTEGER | Y | N | Y |  | INTEGER | 1 |
|  | WeldGraph | BLOB | N | N | N |  | TEXT | {"0": [1, 2, 34, 5, 6, 6],  "1": [1.5, 2, 1, 1.9, 2.0]} |
| Sqlite script | | 1. --Create the table   CREATE TABLE "WeldResultGraph" (  "ID" INTEGER NOT NULL UNIQUE,  "WeldResultID" INTEGER NOT NULL UNIQUE,  "WeldGraph" TEXT NOT NULL,  FOREIGN KEY("WeldResultID") REFERENCES "WeldResult"("ID") ON DELETE CASCADE,  PRIMARY KEY("ID" AUTOINCREMENT)  );   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”. | | | | | | |

# 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. Member Buffer in message struct is used for …
2. 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.
3. DataTask should provide method to insert new record.
4. Data for WeldRecipe should be extracted from message queue using struct type WeldRecipeSC

Jerry: With meeting discussion with WindRiver and Liu Qun, there are three solutions for the data exchange between control task and data task as following…

Keep global variables CommonProperty::WeldResult and CommonProperty::WeldSignatureVector as the data source for the data exchange between control task and data task. If we still stick to send data using message queue, the Weld Signature size will be out of message queue (1024).

In order to make sure the data integrity without any missing, there is the serial process between control task and data task using message queue. At the begin the control task send the request command to data task and wait for the response command from data task until the data is inserted into database.

It seems the size of message queue is not able to save the weld signature data. In order to keep the data process with high performance, there should be a round-robin queue between control task and data task so control task and data task can run each other parallel. All the data shall be saved in the round-robin queue temporarily.

In order to keep Branson be better policy, let’s still keep 1# to implement weld result and weld signature saving with data task.

1. Data for WeldResult should be extracted from message queue using struct type WELD\_RESULT
2. Data for WeldResultSignature should be extracted from CommonProperty::WeldSignatureVector
3. Other tables’ data should….
4. When the number of records exceeds the limit, the oldest record should be removed.
5. When the column “ID” reaches table limit, it should restart at 1

Jerry: We don’t need to restart incremental ID from 1 when it reaches out the table limit. With the table analysis, there is only weld result ID will be related in Alarm table and Weld Signature table. The code needs to handle with weld result ID using long long data type while the data processing for Alarm table and Weld Signature table.

1. DataTask should provide method to update the record following ID for table Recipe, meanwhile, the column DataTime should be update following current time stamp.
2. DataTask should provide method to query for a record using ID from table WeldResult.
3. DataTask should provide method to query for the records list using …?

Jerry: See CommonProperty::WeldResultForUI[50].

1. DataTask should provide method to query a WeldGraph using WeldResultID from table WeldResultSignature.
2. DataTask should provide method to query for a record using ID from table WeldRecipe.
3. DataTask should provide method to query for all the 1000 records from table WeldRecipe.
4. DataTask should provide method to query for…
5. The result from QUERY should be…