



Customization in NvRAM

Ver 1.2



Customization in NvRAM

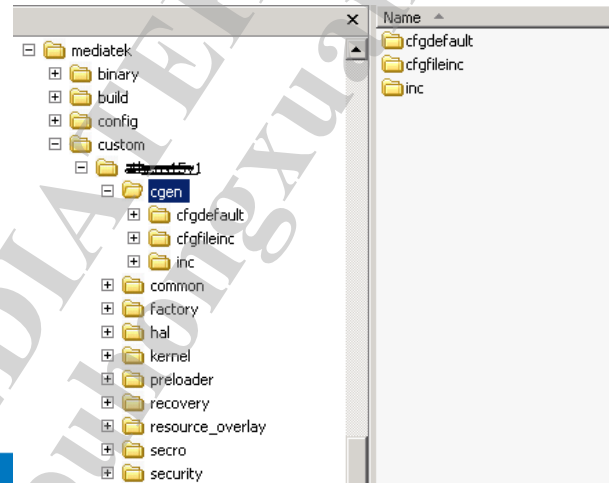
- **Version 1.2**
 - Jian Lin (WCP2/OSS3/SS6)
- **Version 1.1**
 - Yuchi Xu (WCP2/OSS3/SS6)
- **Version 1.0**
 - Koshi Chiu (WCP2/OSS1/SS1)

Customization in NvRAM

- **For the different requirements of projects, NvRAM modules also need to provide the supports of customization configurations, including default value and record data structure of NvRAM files.**
- **There are two parts of NvRAM data**
 - Common
 - For MTK platform NvRAM used
 - Customer can see the definition of related NVRAM record structure
 - But should not modify them
 - Customized for different projects
 - For customer NvRAM used
 - Customer can see the definition of related NVRAM record structure
 - Can modify them according to the requirements

Customization in NvRAM

- The folder of NvRAM customization is located in the path
 - mediatek\custom\ [\$PROJECT]\ cgen
 - There are three folders in this customization folder
 - Cfgdefault
 - Used to define the default value of NvRAM files
 - Cfgfileinc
 - Used to define the record data structure of NvRAM file
 - Inc
 - Used to support general NvRAM module functionalities



Customization in NvRAM

- **Should modify the file**

- mediatek\custom\ [\$PROJECT]\ cgen\inc\CFG_file_info_custom.h
- Data structure of g_akCFG_File_Custom

- **The information of NvRAM file**

- File path
 - The file path that the NvRAM files should be store
- File version
- Record size
- Record numbers
- The type of the default value
- The default value
- type of processing when version not match (Convert/Reset)
- Data-convert function

Customization in NvRAM

- The data structure of *g_akCFG_File_Custom*

```
TCFG_FILE g_akCFG_File_Custom[] =
```

```
{ "/data/nvram/APCFG/APRDCL/Audio_Sph", VER(AP_CFG_RDCL_FILE_AUDIO_LID), CFG_FILE_SPEECH_REC_SIZE,
  CFG_FILE_SPEECH_REC_TOTAL, SIGNLE_DEFAULT_REC, |(char *)&speech_custom_default, DataReset, NULL},

{ "/data/nvram/APCFG/APRDEB/GPS", VER(AP_CFG_CUSTOM_FILE_GPS_LID), CFG_FILE_GPS_CONFIG_SIZE,
  CFG_FILE_GPS_CONFIG_TOTAL, SIGNLE_DEFAULT_REC, (char *)&stGPSConfigDefault, DataReset, NULL},
```

- The default value of *stGPSConfigDefault*

```
ap_nvram_gps_config_struct stGPSConfigDefault =
{
    /* "/dev/ttyMT1" */
    {'/', 'd', 'e', 'v', '/', 't', 't', 'y', 'M', 'T', '1', 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
    /* 0:s/w, 1:none, 2:h/w */
    1,

    /* 16.368MHz */
    16368000,
    /* 500ppb */
    500,
    /* 0:16.368MHz TCXO */
    0,

    /* 0:mixer-in, 1:internal-LNA */
    0,

    /* 0:none */
    0
};
```

Note: the sequence of *a_akCFG_File_Custom* must be the same as LID definition table(Page 9)

Reset to Default

Type	Descriptions
<i>SINGLE_DEFAULT_REC</i>	<p>If multiple records have same default value, this type should be used to minimize the Ram size.</p> <p>It only need define the default value of one record, NvRAM module will use the default value of this record to initialize all of records</p>
<i>MULTIPLE_DEFAULT_REC</i>	<p>If NvRAM has different default value for different records, this type should be used.</p> <p>It will use default value which is define in the cfg_file, then writes to NvRAM file</p>
<i>DEFAULT_ZERO</i>	The default value is 0 , the property of default value will not be cared
<i>DEFAULT_FF</i>	The default value is 0xff , the property of default value will not be cared

Step by Step to Add NvRAM Data

1. Add one **header file** which describes the definition of its record **data structure**, **record size** and **record numbers**
 - In the path of *mediatek\custom\[\$PROJECT]\cgen\cfgfileinc*

```
#ifndef _CFG_CUSTOM1_FILE_H
#define _CFG_CUSTOM1_FILE_H

typedef struct
{
    unsigned int Array[1];
} File_Custom1_Struct;

#define CFG_FILE_CUSTOM1_REC_SIZE    sizeof(File_Custom1_Struct)
#define CFG_FILE_CUSTOM1_REC_TOTAL  1

#endif
```

2. Add **header file** which define its **default value** of NvRAM file
 - In the path of *mediatek\custom\[\$PROJECT]\cgen\cfgdefault*

```
#ifndef _CFG_CUSTOM1_D_H
#define _CFG_CUSTOM1_D_H

File_Custom1_Struct stCustom1Default =
{
    1
};

#endif
```


Step by Step to Add NvRAM Data

3. Add one lid in the enum definition of “CUSTOM_CFG_FILE_LID” and define the version number of NvRAM file

- In the path of

mediatek\custom\ [\$PROJECT]\cgen\inc\Custom_NvRAM_LID.h

```
/* the definition of file LID */
typedef enum
{
    AP_CFG_RDCL_FILE_AUDIO_LID=AP_CFG_CUSTOM_BEGIN_LID, //AP_CFG_CUSTOM_BEGIN_LID: this lid must not be changed, it is reserved for system.
    AP_CFG_CUSTOM_FILE_GPS_LID,
    AP_CFG_RDCL_FILE_META_LID,
    AP_CFG_CUSTOM_FILE_CUSTOM1_LID,
    AP_CFG_CUSTOM_FILE_CUSTOM2_LID,

    AP_CFG_CUSTOM_FILE_MAX_LID,
} CUSTOM_CFG_FILE_LID;

/* verno of data items */
/* audio file version */
#define AP_CFG_RDCL_FILE_AUDIO_LID_VERNO "001"
/* META log and com port config file version */
#define AP_CFG_RDCL_FILE_META_LID_VERNO "000"

/* custom2 file version */
#define AP_CFG_CUSTOM_FILE_CUSTOM1_LID_VERNO "000"
/* custom2 file version */
#define AP_CFG_CUSTOM_FILE_CUSTOM2_LID_VERNO "000"
/* GPS file version */
#define AP_CFG_CUSTOM_FILE_GPS_LID_VERNO "000"
```

The sequence of lids in enum definition can't change. The newest lid must add at the end of the definition table.(before MAX_LID)

Step by Step to Add NvRAM Data

4. Add one **include path** which added in the step 1

- In the path of

mediatek\custom\ [\$PROJECT]\cgen\inc\custom_cfg_module_file.h

5. Add one **include path** which added in the step 2

- In the path of

mediatek\custom\ [\$PROJECT]\cgen\inc\custom_cfg_module_default.h

6. Add the related information of NvRAM file into the definition of “***g_akCFG_File_Custom***”

- In the path of

mediatek\custom\[\$PROJECT]\cgen\inc\CFG_file_info_custom.h

7. Add its related information, including record structure, NvRAM lid, and record number

- In the path of

mediatek\custom\[\$PROJECT]\cgen\inc\Custom_NvRAM_data_item.h

Add NvRAM File to Backup List

If your NvRAM File Need to **Backup to BinRegion**, add your module in **aBackupToBinRegion[]**

In the path of *mediatek\custom\ common\ cgen\CFG_file_info.c*

```

FileName aBackupToBinRegion[]=
{
    {"FILE_VER",AP_CFG_FILE_VER_INFO_LID},
    {"BT_Addr",AP_CFG_RDEB_FILE_BT_ADDR_LID},
    {"WIFI",AP_CFG_RDEB_FILE_WIFI_LID},
    {"AUXADC",AP_CFG_RDCL_FILE_AUXADC_LID},
    {"FACTORY",AP_CFG_RDCL_FACTORY_LID},
    {"BWCS",AP_CFG_RDCL_BWCS_LID},
    {"HWMON_ACC",AP_CFG_RDCL_HWMON_ACC_LID},
    {"HWMON_GYRO",AP_CFG_RDCL_HWMON_GYRO_LID},
    {"WIFI_CUSTOM",AP_CFG_RDEB_WIFI_CUSTOM_LID},
    {"GPS",AP_CFG_CUSTOM_FILE_GPS_LID},
    {"FileName In NvRAM",THE_LID_OF_YOUR_FILE}
};
  
```

Note: This Backup Mechanism can be triggered by Meta tool

NvRAM Library

■ NvRAM Interface For Module

- To provide related interface functions for modules to read, write NvRAM file
- Interface of opening and closing nvram file is provided by Nvram library

Interface Function	Description
NVM_GetFileDesc	Get the description of nvram file and the information of record size and number
NVM_CloseFileDesc	close the file description

NvRAM Library

■ NvRAM Interface For Module (For MT6575)

- typedef struct {
 int iFileDesc;
 Int ifile_lid;
} F_ID;
- **F_ID** NVM_GetFileDesc(int file_lid, int *pRecSize, int *pRecNum, bool IsRead)

DESCRIPTION:

this function is called to the desc of nvram file and the information of record size and number.

PARAMETERS:

file_lid:	[IN]	the lid of nvram file
pRecSize:	[OUT]	Pointer of the record size
pRecNum:	[OUT]	Pointer of the record number
IsRead:	[IN]	true is read-only, otherwise is read/write

RETURN VALUE:

the file ID

NvRAM Library

- **NvRAM Interface For Module (For MT6575)**

- `bool NVM_CloseFileDesc(F_ID file_id)`

DESCRIPTION:

this function is called to close the file desc which is open by NVM_GetFileDesc.

PARAMETERS:

`file_id`: [IN] the file ID

RETURN VALUE:

true is success, otherwise is fail

NvRAM Library

- **NvRAM Interface For Module (For MT6575)**

- Example

```
#include "libnvram.h"
```

```
F_ID fid ;
```

```
int rec_size = 0;
```

```
int rec_num = 0;
```

```
Int your_file_lid = YOUR_FILE_LID;
```

```
bool isread = false;
```

```
YOUR_LID_STRUCT *your_lid_struct = NULL;
```

```
your_lid_struct =(YOUR_LID_STRUCT *) malloc(sizeof(YOUR_LID_STRUCT ));
```

```
If(your_lid_struct == NULL)
```

```
    return false;
```

```
fid = NVM_GetFileDesc(your_file_lid, &rec_size, &rec_num, isread );
```

```
If(fd<0)
```

```
    return false;
```

```
If(rec_size != read(fid.iFileDesc,your_lid_struct,rec_size)){
```

```
    free(your_lid_struct );
```

```
    return false;}
```

```
.....  
free(your_lid_struct);
```

```
if(!NVM_CloseFileDesc(fid))
```

```
    return false;
```

```
return true;
```



Appendix




Step by Step to **Reset** data(Example)

1. Update the version of file in which **data structure** you want to change

- In the path `mediatek\custom\[PROJECT]\cgen\inc\Custom_Nvram_LID.h`

```
#define AP_CFG_RDEB_WIFI_CUSTOM_LID_VERN0 "000"
```



```
#define AP_CFG_RDEB_WIFI_CUSTOM_LID_VERN0 "001"
```

2. **Change header file** which describes the definition of **data structure**

- In the path of
`mediatek\custom\[PROJECT]\cgen\cfgfileinc\CFG_wifi_File.h`

```
typedef struct _WIFI_CUSTOM_PARAM_STRUCT
{
    UINT_32      u4Resv;      /* Reserved */
} WIFI_CUSTOM_PARAM_STRUCT;
```



```
typedef struct _WIFI_CUSTOM_PARAM_STRUCT
{
    UINT_32      u4Resv;      /* Reserved */
    |  UINT_32      u4ResvAdd;
} WIFI_CUSTOM_PARAM_STRUCT;
```

3. Change header file which define its default value of NvRAM file

- In the path of

mediatek\custom\[PROJECT]\cgen\cfgdefault\CFG_WIFI_default.h



```

WIFI_CUSTOM_PARAM_STRUCT stWifiCustomDefault =
{
    0x0, // Reserved
};

WIFI_CUSTOM_PARAM_STRUCT stWifiCustomDefault =
{
    0x0, // Reserved
    0x5,
};

```

4. Change the related information of NvRAM file into the definition of “g_akCFG_File_Custom”

- In the path of

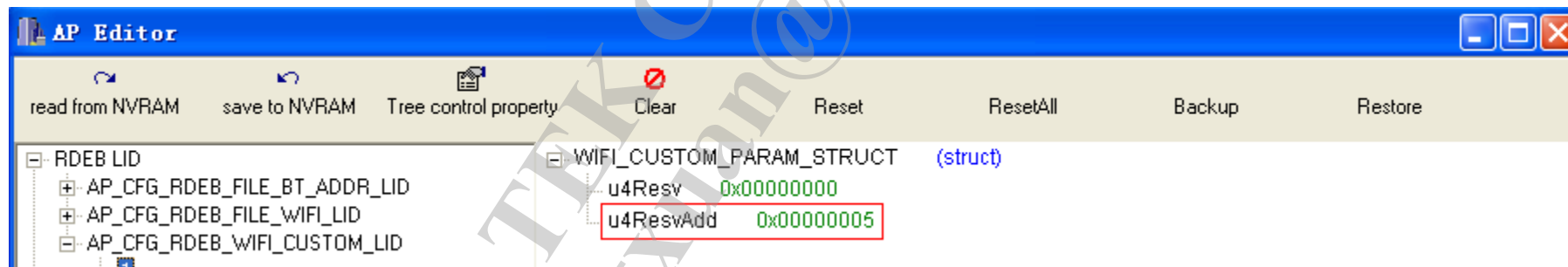
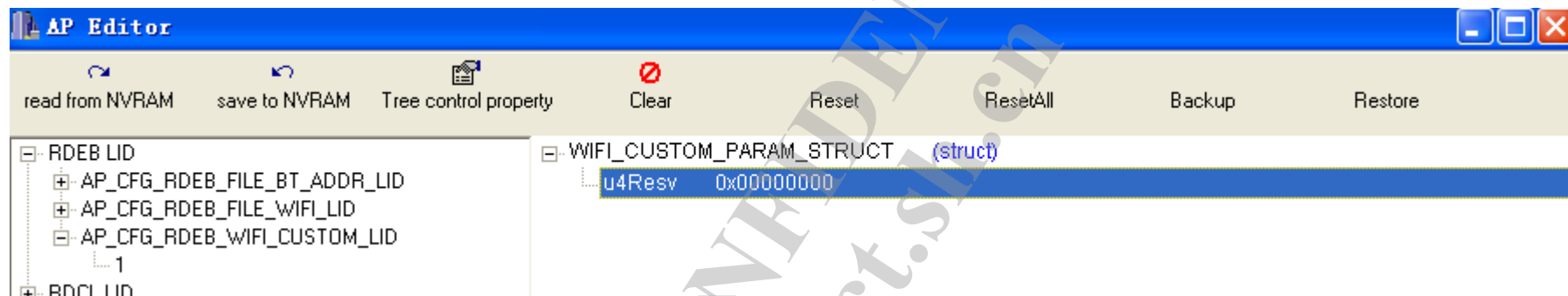
mediatek\custom\[PROJECT]\cgen\inc\CFG_file_info_custom.h

```

{
    "/data/nvram/APCFG/APRDEB/WIFI_CUSTOM", VER(AP_CFG_RDEB_WIFI_CUSTOM_LID), CFG_FILE_WIFI_CUSTOM_REC_SIZE,
    CFG_FILE_WIFI_CUSTOM_REC_TOTAL, SIGNLE_DEFAULT_REC, (char *)&stWifiCustomDefault, DataReset, NULL
},

```

Result of Reset




Step by Step to **Convert** data(Example)

1. Update the version of file in which **data structure** you want to change

- In the path mediatek\custom\[PROJECT]\cgen\inc\Custom_Nvram_LID.h

```
#define AP_CFG_RDEB_WIFI_CUSTOM_LID_VERNO 000
```




```
#define AP_CFG_RDEB_WIFI_CUSTOM_LID_VERNO 001
```

2. Change header file which describes the definition of data structure and declaration of convert function

- In the path of
mediatek\custom\[PROJECT]\cgen\cfgfileinc\CFG_wifi_File.h

```
typedef struct _WIFI_CUSTOM_PARAM_STRUCT
{
    UINT_32      u4Resv;          /* Reserved */
} WIFI_CUSTOM_PARAM_STRUCT;
```



```
typedef struct _WIFI_CUSTOM_PARAM_STRUCT
{
    UINT_32      u4Resv;          /* Reserved */
    |  UINT_32      u4ResvAdd;
} WIFI_CUSTOM_PARAM_STRUCT;
```

```

/*****
 *          FUNCTION DECLARATIONS
 *****/
*/
extern int WifiCustom_ConvertFunction(int, int, char*, char*);
/*****

```

3. Change header file which define its default value of NvRAM file

- In the path of

mediatek\custom\[PROJECT]\cgen\cfgdefault\CFG_WIFI_default.h



```
WIFI_CUSTOM_PARAM_STRUCT stWifiCustomDefault =
{
    0x0, // Reserved
};

WIFI_CUSTOM_PARAM_STRUCT stWifiCustomDefault =
{
    0x0, // Reserved
    0x5,
};
```

4. Change the related information of NvRAM file into the definition of “*g_akCFG_File_Custom*”

- In the path of

mediatek\custom\[PROJECT]\cgen\inc\CFG_file_info_custom.h



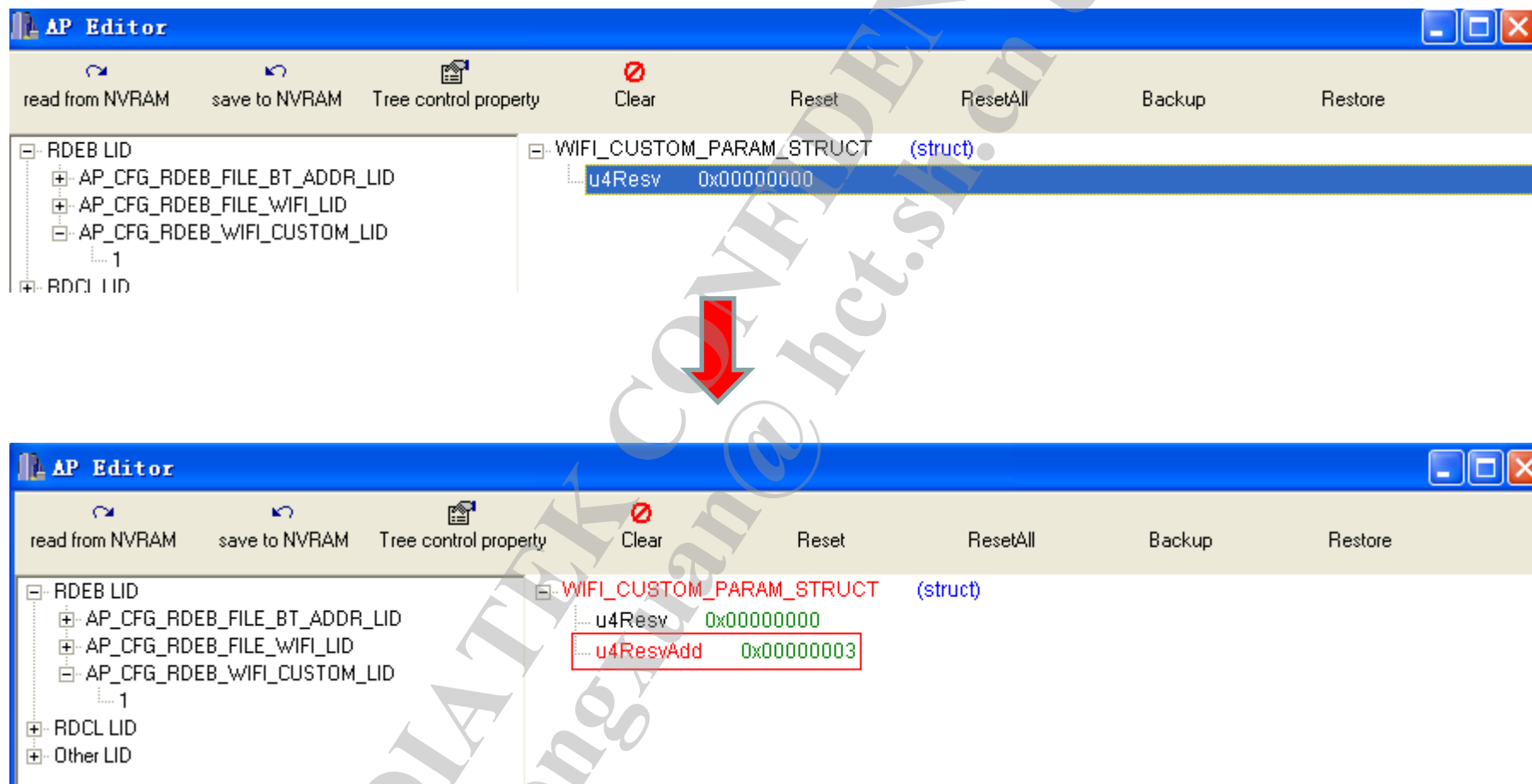
```
{
    "/data/nvram/APCFG/APRDEB/WIFI_CUSTOM",
    CFG_FILE_WIFI_CUSTOM_REC_TOTAL,
    VER(AP_CFG_RDEB_WIFI_CUSTOM_LID),
    SINGLE_DEFAULT_REC,
    CFG_FILE_WIFI_CUSTOM_REC_SIZE,
    ((char *)&stWifiCustomDefault, DataConvert, WifiCustom_ConvertFunction
}
```

5. Add definition of convert function in C source file

- In the path of *mediatek\custom\ common\ cgen\CFG_file_info.c*

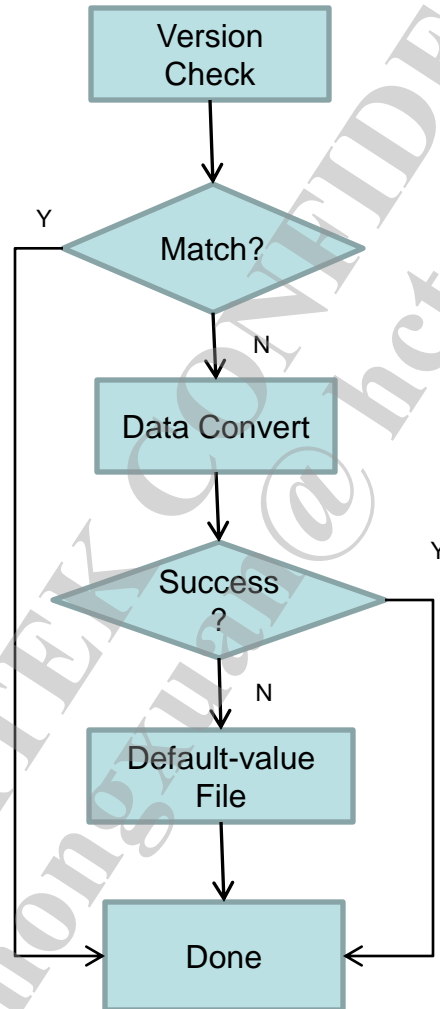
```
int WifiCustom_ConvertFunction(int CurrentVerID, int NewVerID, char* pSrcMem, char*pDstMem)
{
    |
    if(NULL == pSrcMem || NULL == pDstMem) {
        return 0;
    }
    else if (0 == CurrentVerID && 1 == NewVerID) {
        memcpy(pDstMem, pSrcMem, sizeof(unsigned int));
        *((unsigned int*)(pDstMem + 4)) = 0x3;
        return 1;
    }
    else {
        return 0;
    }
}
```

Result of Convert



Introduction of Convert function(1/n)

- WorkFlow**



Introduction of Convert function(2/n)

■ **Prototype**

- Int XXXX_ConvertFunction(int CurrentVerID,int NewVerID,char *pSrcMem, char* pDstMem)
 - *CurrentVerID: The file version which restore from BinRegion in first boot up.*
 - *NewVerID: The file version which you have updated*
 - *SrcMem: the memory saved date in file which restore from Binregion when first boot up.*
 - *pDstMem: the memory saved data which you want the file to be after convert.(must match the structure in new version)*

```
WIFI_CUSTOM_PARAM_STRUCT stWifiCustomDefault =  
{  
    0x0, // Reserved  
};
```

```
typedef struct _WIFI_CUSTOM_PARAM_STRUCT  
{  
    UINT_32 u4Resv; /* Reserved */  
    UINT_32 u4ResvAdd;  
} WIFI_CUSTOM_PARAM_STRUCT;
```



Appendix II



Sync your thread with Nvram Daemon

- ***Why your private thread should sync with nvram daemon***
 - After bring up, Nvram daemon will check /data/nvram folder and do some initialization. After that, other threads can access /data/nvram correctly. If other thread access /data/nvram before nvram_daemon ready, there may bring about unpredictable results. So your thread must sync with nvram daemon.
- ***How your private thread should sync with nvram daemon***
 - When nvram daemon is ready, it will set system variable “nvram_init” to “Ready” by property_set(“nvram_init”, “Ready”).
 - You can get status of “nvram_init” by property_get, then check if nvram daemon is “Ready”.
 - When nvram daemon is “Ready”, you can access /data/nvram safely. Otherwise your private thread should wait.

Sync your thread with Nvram Daemon

- **Sample Code**

```
#define MAX_RETRY_COUNT 20
int read_nvram_ready_retry = 0;

while(read_nvram_ready_retry < MAX_RETRY_COUNT)
{
    read_nvram_ready_retry++;
    property_get("nvram_init", nvram_init_val, NULL);
    if(strcmp(nvram_init_val, "Ready") == 0)
    {
        break;
    }
    else
    {
        usleep(500*1000);
    }
}
NVRAM_LOG("Get nvram restore ready retry cc=%d\n", read_nvram_ready_retry);
if(read_nvram_ready_retry >= MAX_RETRY_COUNT)
{
    printf("Get nvram restore ready faild\n");
    NVRAM_LOG("Get nvram restore ready faild!!!\n");
}
```

MEDIATEK

www.mediatek.com

