Confidential B

MEDIATEK

MMI Platform Source Code Training

Agenda

- MMI Task Overview
- Source Code Structure Navigation
- Introduction of Writing an Application
- MMI Framework
- MMI/Protocol Interface
- Debugging Environment
- Configuration Management

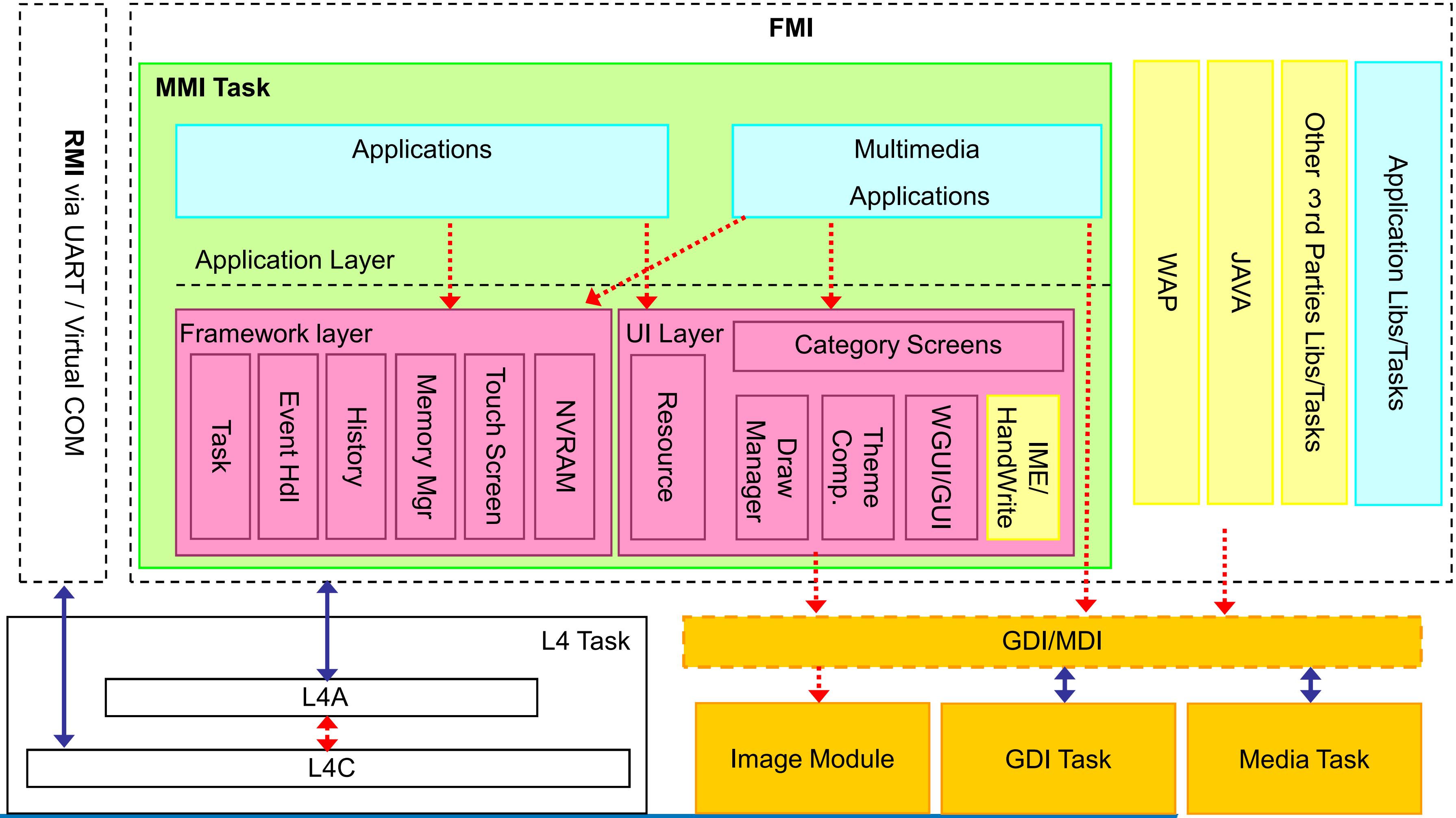
Confidential B

MEDIATEK

MMI Task Overview

MMI Architecture

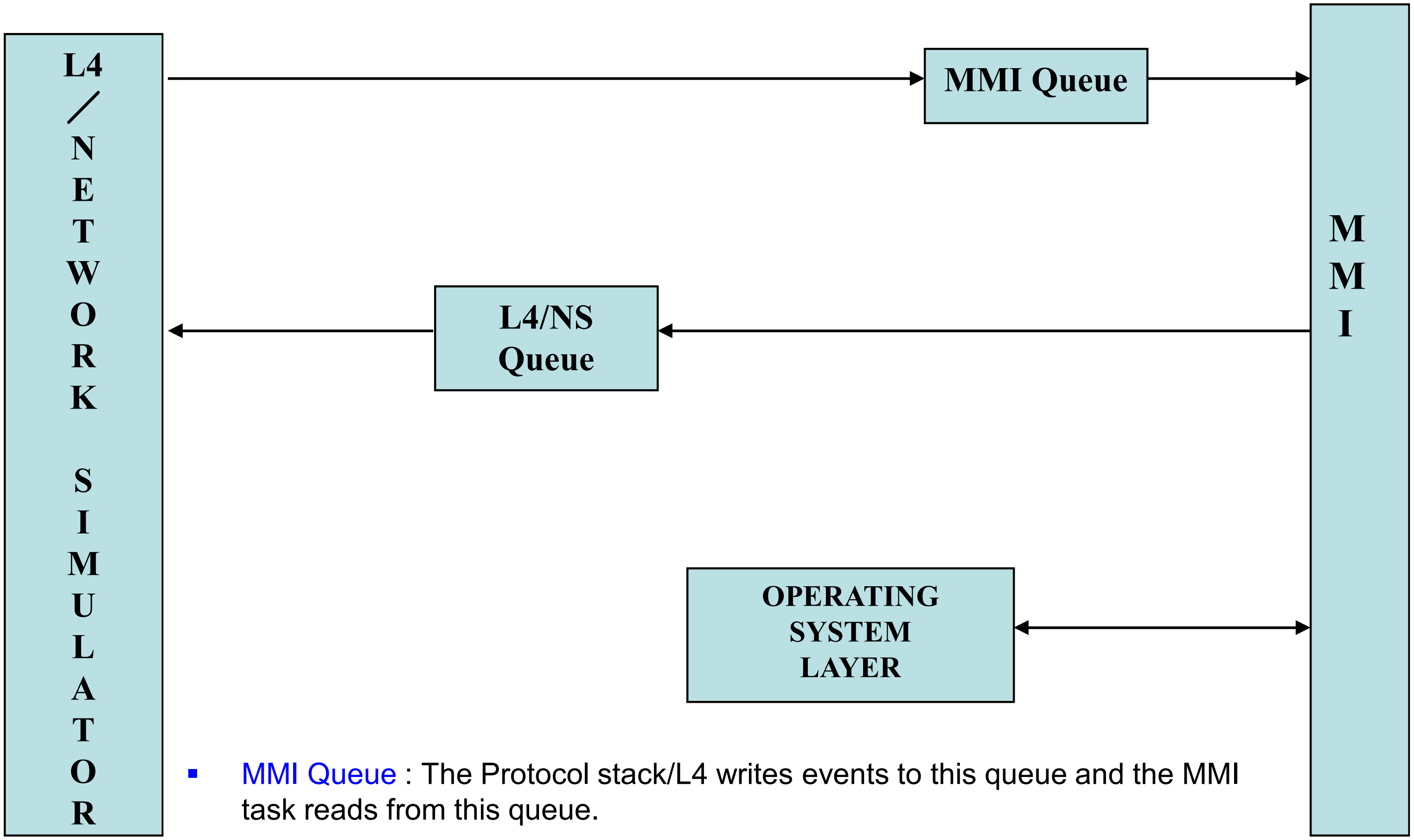
- 🟨 **3rd Party Components**
- 🟦 **Native Apps**
- ━ **Primitive Based Interface**
- ┅ **Function-Call Based Interface**



MMI Task Overview

- Application layer
  - Contains all application (core, logic).
- Framework layer
  - Contains wrappers for messages and events handling
  - Facilitate the application flow
  - Provides OS abstraction for portability
- UI layer
  - Contains wrappers for UI related functions.
  - Responses for UI display
- Handwriting adaptation
  - Contain wrappers for 3rd party solution
  - Responses for handwriting and virtual keypad
- GDI (Graphic Device Interface)
  - perform drawings about image, 2d graphics, etc.

Task Structure – Brief Explanation

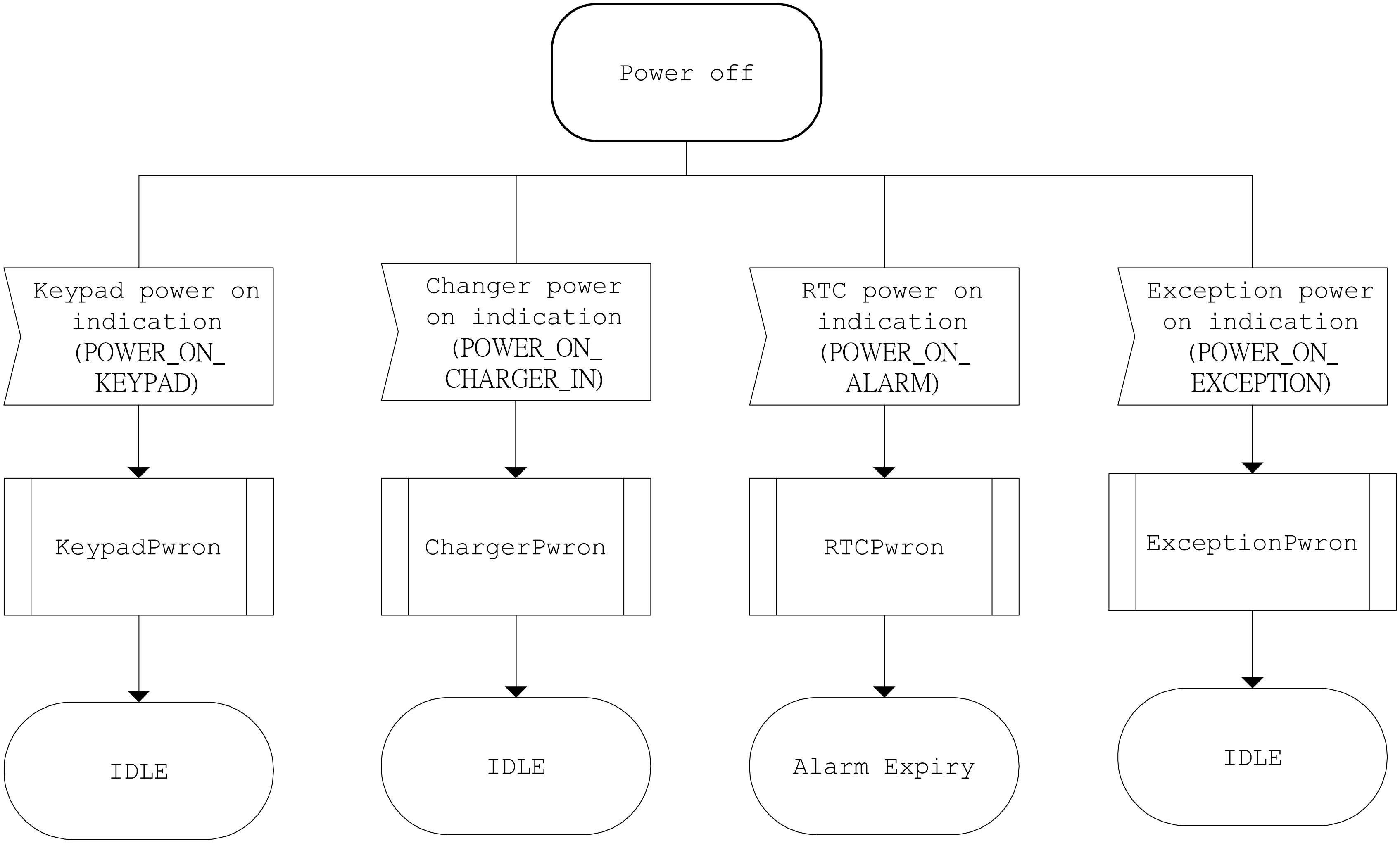


- MMI Queue : The Protocol stack/L4 writes events to this queue and the MMI task reads from this queue.
- L4 / NS Queue : The MMI task writes MMI events to this queue and L4 task / Network Simulator reads from this queue.

MMI Task Overview

- MMI Task Routines
  - Waits for messages sent to the MMI queue.
  - Messages in this queue are put by the L4.
  - Framework Layer processes the events.
  - Framework Layer triggers callbacks which is registered by application layer before.
  - Application layer uses UI Layer category functions for screen display.

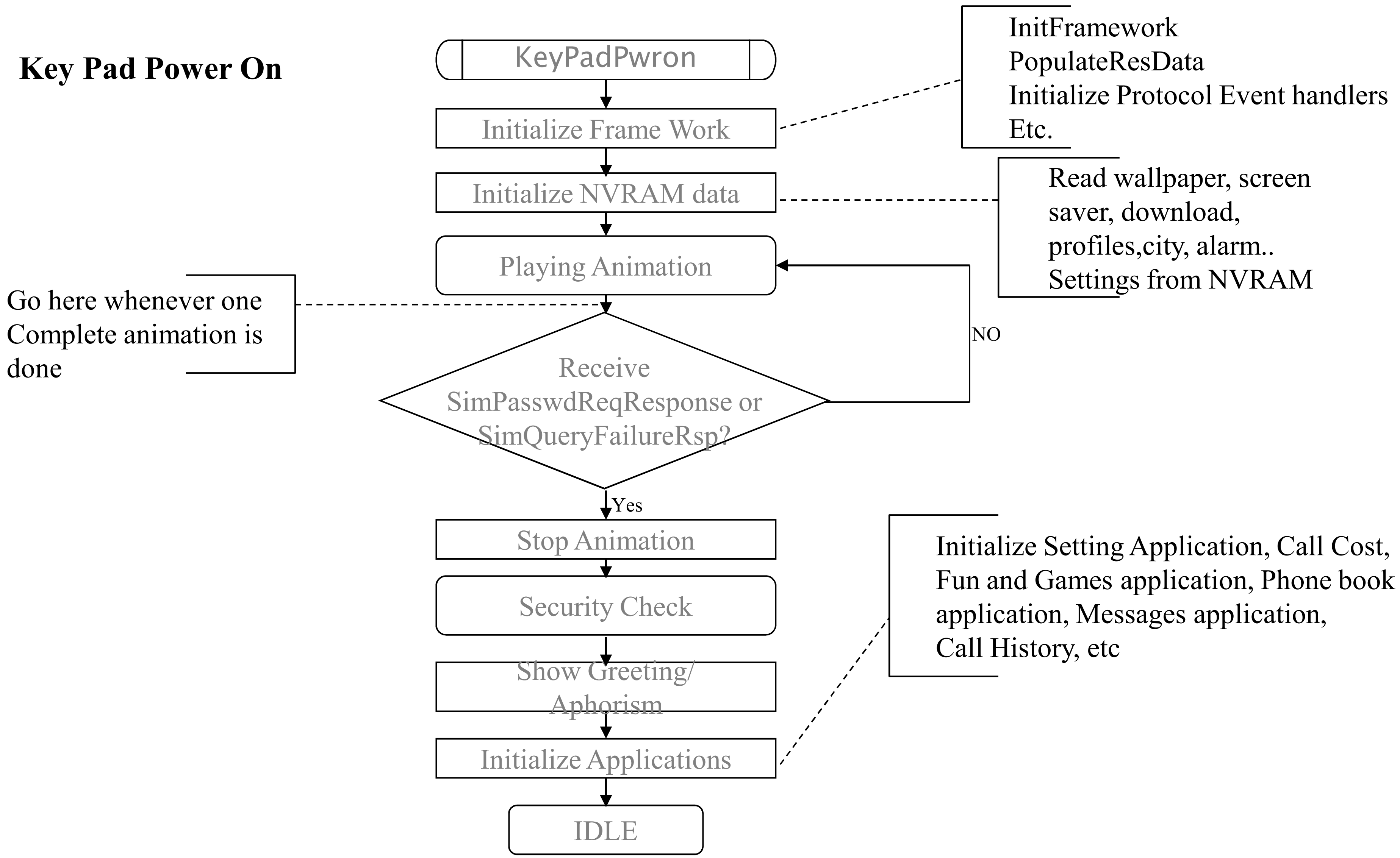
Power ON Sequence


Power off
Keypad power on indication (POWER_ON_ KEYPAD)
Changer power on indication (POWER_ON_ CHARGER_IN)
RTC power on indication (POWER_ON_ ALARM)
Exception power on indication (POWER_ON_ EXCEPTION)
KeypadPwron
ChargerPwron
RTCPwron
ExceptionPwron
IDLE
IDLE
Alarm Expiry
IDLE

Algorithm-Key Pad Power ON

Confidential B

Key Pad Power On



InitFramework
PopulateResData
Initialize Protocol Event handlers
Etc.

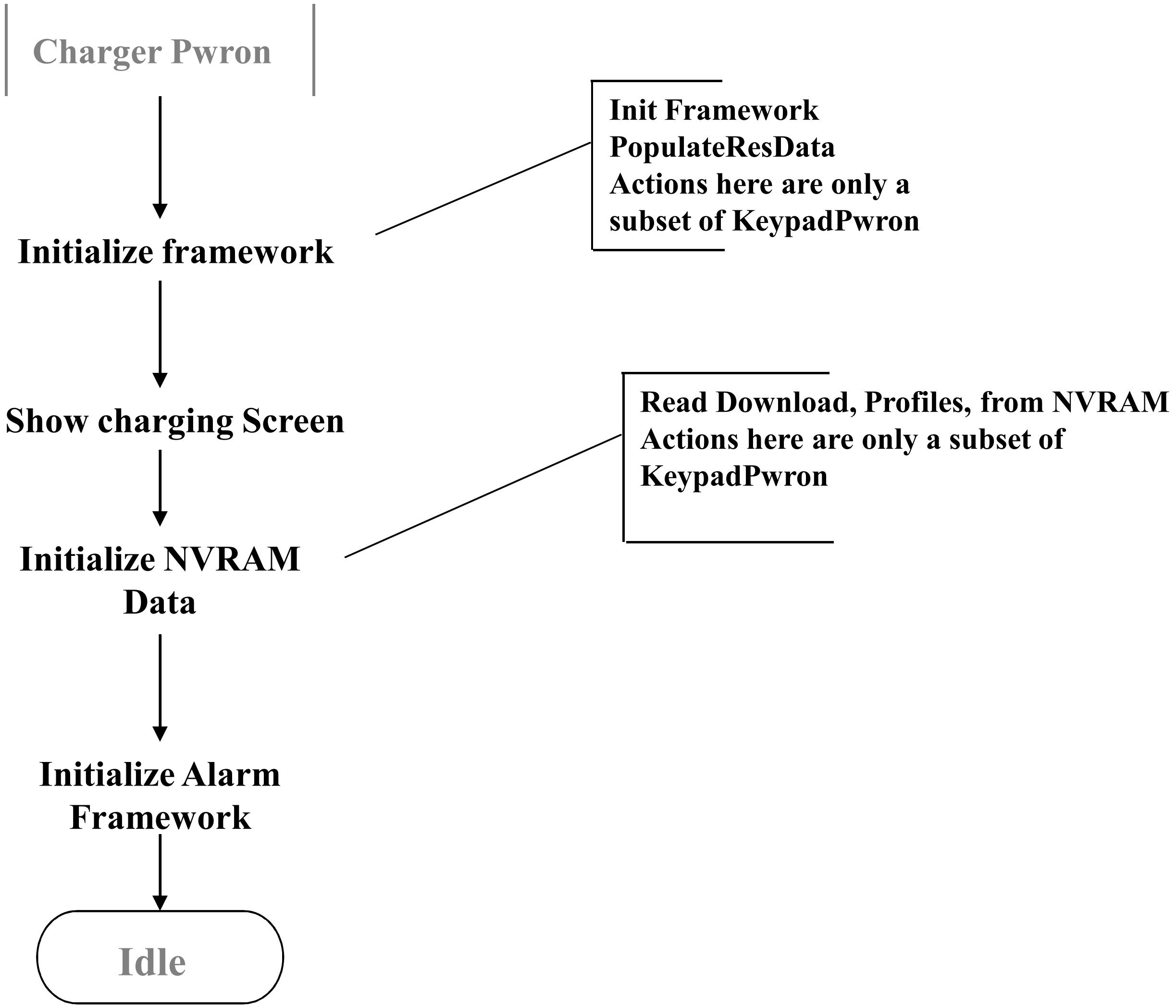
Read wallpaper, screen saver, download, profiles,city, alarm.. Settings from NVRAM

Go here whenever one Complete animation is done

Initialize Setting Application, Call Cost, Fun and Games application, Phone book application, Messages application, Call History, etc

Confidential B

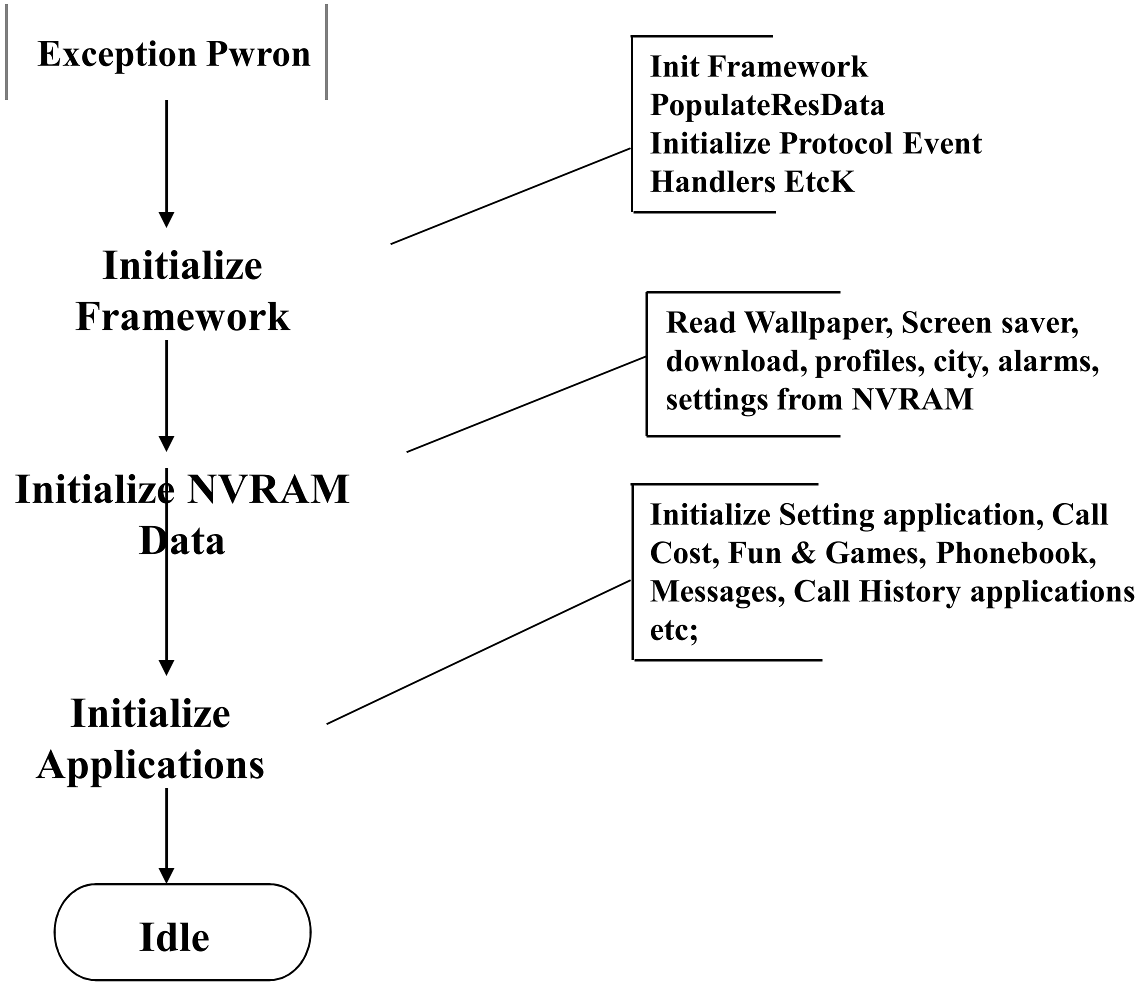
Algorithm-Charger Power ON



Init Framework
PopulateResData
Actions here are only a subset of KeypadPwron

Read Download, Profiles, from NVRAM
Actions here are only a subset of KeypadPwron

Algorithm-Exception Power ON



Exception Pwron

Initialize Framework

Init Framework
PopulateResData
Initialize Protocol Event Handlers EtcK

Initialize NVRAM Data

Read Wallpaper, Screen saver, download, profiles, city, alarms, settings from NVRAM

Initialize Applications

Initialize Setting application, Call Cost, Fun & Games, Phonebook, Messages, Call History applications etc;

Idle

Confidential B

MEDIATEK

Code Structure Navigation

MMI Related Libraries

- MMI contains several libraries
  - Mmiresource.lib : All resources (project-based)
  - conn_app.lib, inet_app.lib, media_app.lib, mmi_app.lib, mmi_framework.lib
- Source Code
  - plutommi\mmi : MMI framework and basic applications
  - Pltuommi\mtkapp : All other applications not in plutommi
- Benefits
  - Resource generating process only changes mmiresource library
  - Customers can change resources without changing other libraries.
  - Easy to maintain different resources for different projects

Code Structure and Navigation

- \plutommi
  - \Customer
    - mmiresource.lib
  - \MMI and Mtkapp
    - conn_app.lib, inet_app.lib, media_app.lib, mmi_app.lib, mmi_framework.lib

Code Structure and Navigation

- mcu/plutommi/Customer
  - CustResource
    - each project has it’s own folder
    - project-based resources (MMI_features.h, skins,…)
    - Res_MMI
  - Images
    - project-based images
  - ResGenerator
    - resource generator
  - Res_MMI

Code Structure and Navigation

- Plutommi\mmi
  - Inc
  - Framework
  - GUI
    - UI compoments
    - Category screens
    - Draw Manager
  - [App]
    - AppSrc
    - AppInc

- Plutommi\mtkapp
  - [App]
    - AppSrc
    - AppInc
  - GDI

Confidential B

MEDIATEK

Introduction of Writing an Application

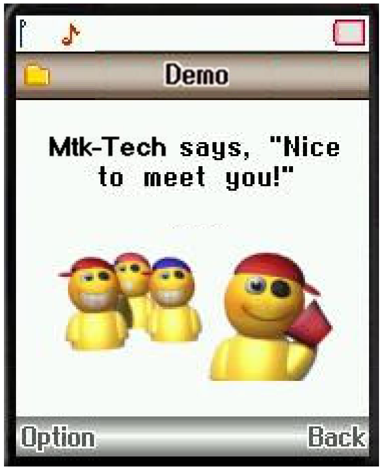
Procedure to Develop a New MMI Application

- Define the behavior
- Process the events from user and other tasks
- Control logic
- Decide the displayed content

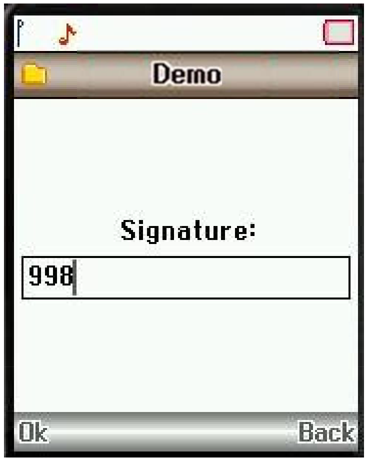
Procedure to Implement an MMI Application

- Declaration of IDs
  - Screens, Menu Items, Strings, Images, Audio, Media
- Populate Resources
- Initialization Routine
  - Routines to setup resource data for using in applications
    - Populate Strings, Images and Menu resources
    - Register Protocol Event and Highlight Handlers
- Highlight Handlers
    - Routines that execute user defined code corresponding to the high lighting menu item.
- Entry and Exit Functions
    - Functions to manage flow of screen for an application
    - Forward flow of screen is managed by the application
    - Backward flow of screen is managed by history

Example - Application Scenario

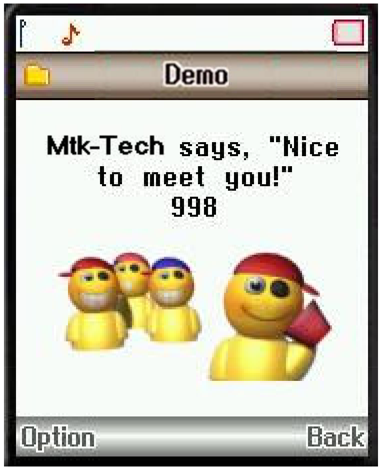


Mtk-Tech says, "Nice to meet you!"



Signature:

998



Mtk-Tech says, "Nice to meet you!"
998

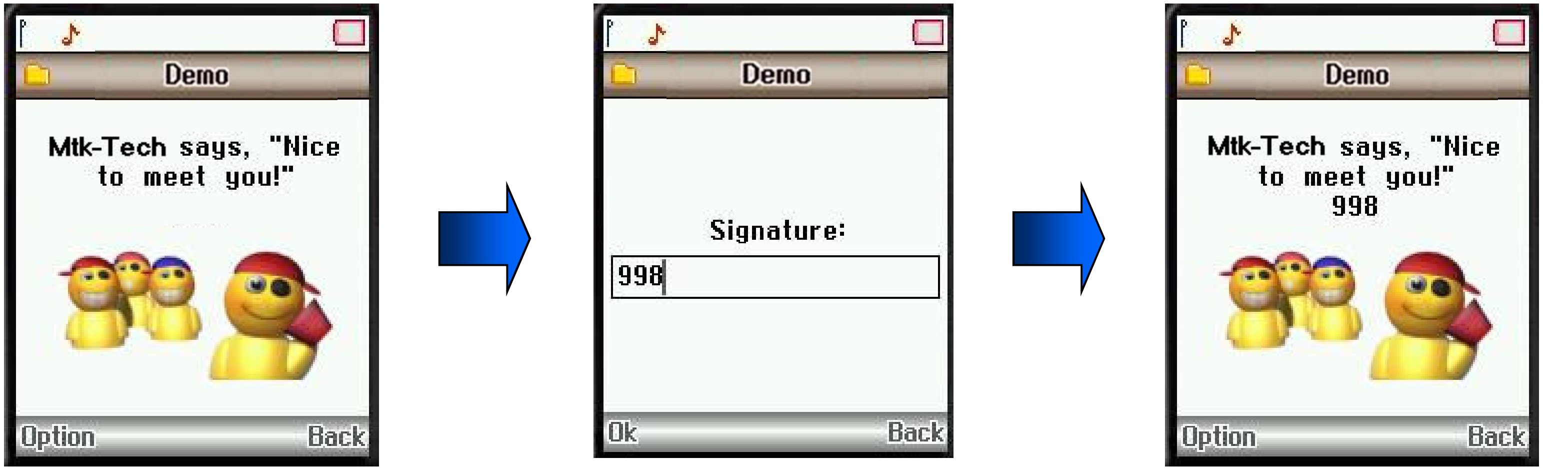
Screen1:

Hello message

Screen2:

Input the signature

Example - Select Category Screens



Demo

Mtk-Tech says, "Nice to meet you!"

Option Back

Demo

Signature:

998

Ok Back

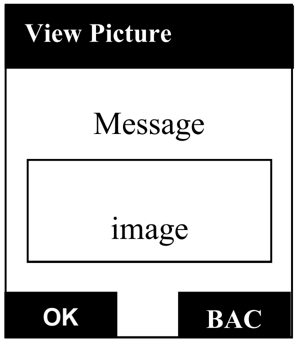
Demo

Mtk-Tech says, "Nice to meet you!" 998

Option Back

Category69Screen

Category66Screen

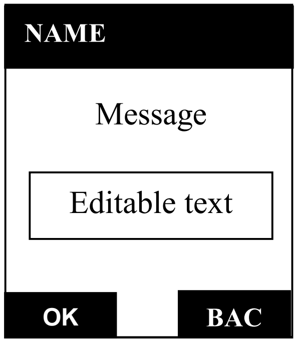


View Picture

Message

image

OK BAC



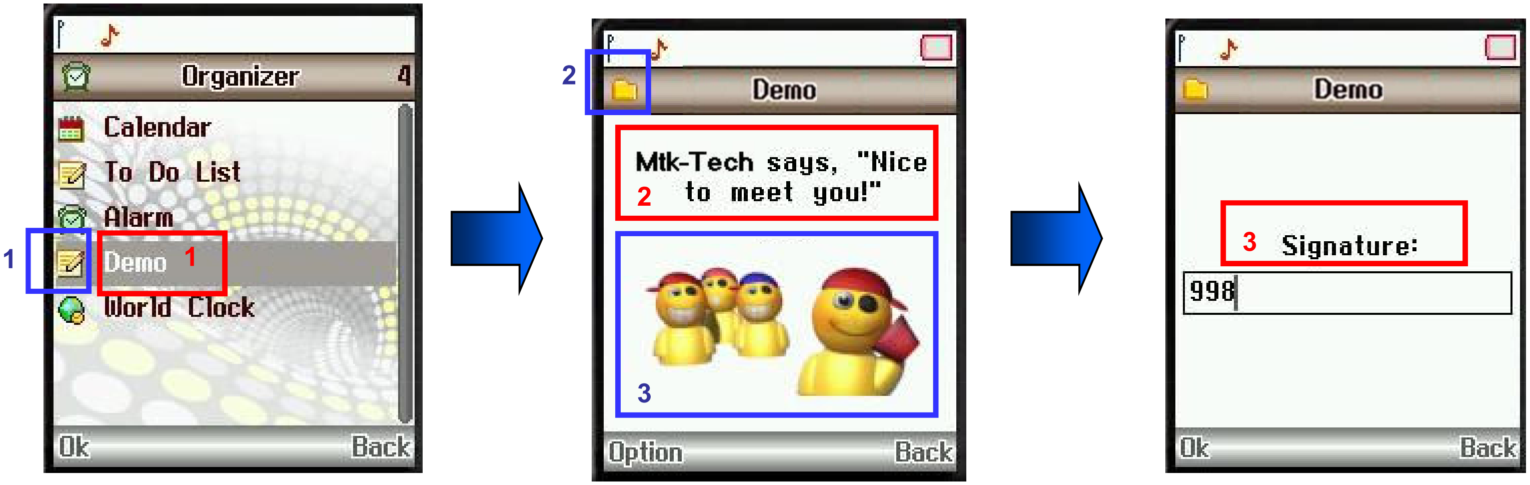
NAME

Message

Editable text

OK BAC

Example - Define Resources



Mtk-Tech says, "Nice to meet you!"

Signature: 998

String : 3

Image : 3

Audio : None

Example - Define Resources

- \Plutommi\MMI\inc\MMIDataType.h
  - RESOURCE_BASE_RANGE(**DEMO_APP**, 500),

- \Plutommi\MMI\[App]\[App]Inc**[App]Def.h**
  -

```
typedef enum          /* 2 category screens */
{
   SCR_ID_DEMO_APP_SCR1 = DEMO_APP+1,
   ……
} SCR_ID_DEMO_ENUM;
```

```
typedef enum          /* 3 string resource */
{
   STR_ID_DEMO_APP_TITLE = DEMO_APP+1,
   ……
} STR_ID_DEMO_ENUM;
```

```
typedef enum          /* 3 image resource*/
{
   IMG_ID_DEMO_APP_ICON = DEMO_APP+1,
   ……
} IMG_ID_DEMO_ENUM;
```

Example - Populate Resources

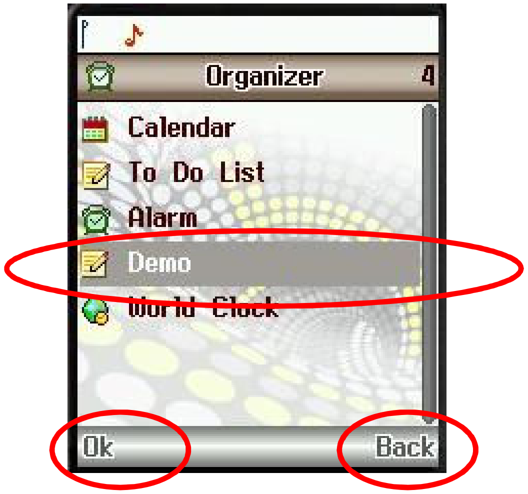
- \Plutommi\Customer\CustResource\[Prj]_MMI\Res_MMI\Res_[App].c
  -

```
void PopulateDemoAppRes(void)
{
   ADD_APPLICATION_STRING2(STR_ID,string,comment);
   ……
   ADD_APPLICATION_IMAGE2(IMG_ID, file, comment);
   ……
   ADD_APPLICATION_MENUITEM(…);
   ADD_APPLICATION_MENUITEM2(…);
   ……
   ADD_APPLICATION_MENUITEM_HILITE_HANDLER(
                              ORGANIZER_DEMOAPP_MENU,
                              HighlightDemoAppMenu);
}
```

Example – Scenario Flow

- Highlight Handler

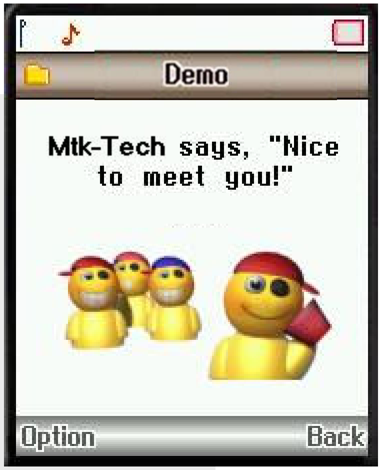
```
void HighlightDemoAppMenu(void)
{
    ChangeLeftSoftkey(STR_GLOBAL_OK, IMG_GLOBAL_OK);
    SetRightSoftkeyFunction(GoBackHistory, KEY_EVENT_UP);
    SetLeftSoftkeyFunction(EntryDemoAppMenu, KEY_EVENT_UP);
    SetKeyHandler(EntryDemoAppMenu, KEY_RIGHT_ARROW,
                                                    KEY_EVENT_DOWN);
}
```



Example – Scenario Flow

- Screen Entry Function
  –

```
void EntryDemoAppMenu(void)
{
   EntryNewScreen(SCR_ID_DEMO_APP_SCR1, NULL,
                           EntryDemoAppMenu, NULL);
   …… /* program logic */
   /* construct displayed string and icon */
   ShowCategory66Screen(STR_ID_DEMO_APP_TITLE,
                                    IMG_ID_DEMO_APP_ICON,
                                    STR_GLOBAL_OPTIONS,
                                    IMG_GLOBAL_OPTIONS,
                                    STR_GLOBAL_BACK,
                                    IMG_GLOBAL_BACK,
                                    g_demo_app_strWelcome,
                                    IMG_ID_DEMO_APP_WELCOME, NULL);
   …… /* program logic */
   /* Set softkey functions or key handlers */
}
```



Mtk-Tech says, "Nice to meet you!"

Confidential B

MEDIATEK

MMI Framework

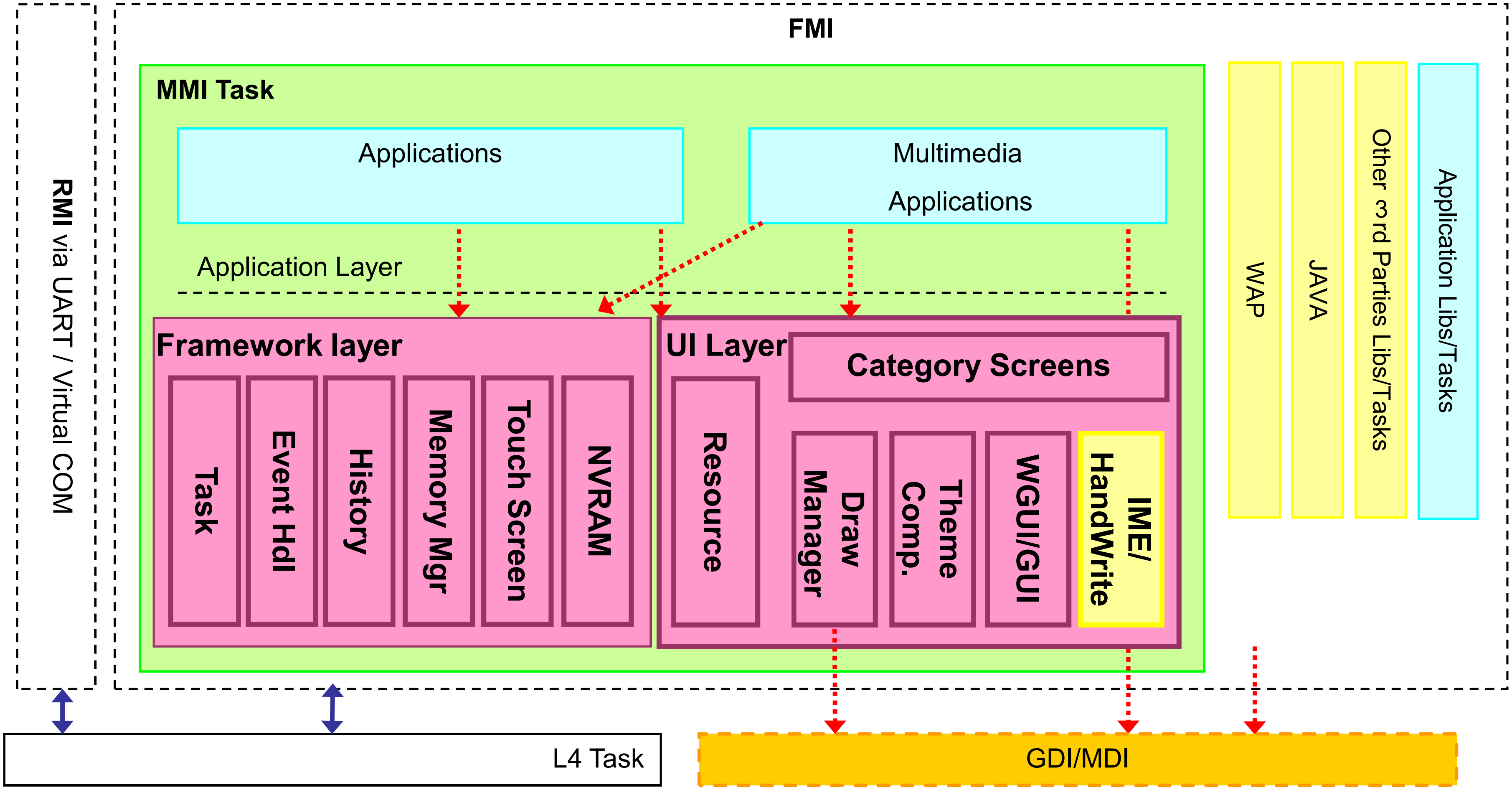
MMI Architecture

🟨 3rd Party Components

🟦 Native Apps

— Primitive Based Interface

⋯ Function-Call Based Interface



FMI

MMI Task

Applications

Multimedia Applications

Application Layer

Framework layer

Task

Event Hdl

History

Memory Mgr

Touch Screen

NVRAM

UI Layer

Category Screens

Resource

Draw Manager

Theme Comp.

WGUI/GUI

IME/ HandWrite

WAP

JAVA

Other 3rd Parties Libs/Tasks

Application Libs/Tasks

RMI via UART / Virtual COM

L4 Task

GDI/MDI

Elements of MMI Framework

- Event Mechanism – Registers and executes application callbacks for various events
- History – Helps application maintain screen flow and store intermediate data
- Memory Manager – Provides the different memory mechanism for MMI applications
- Touch Screen
- NVRAM – Provides wrappers for data storage and retrieval of data from NVRAM.

Event Mechanism

- For application to manage event handlers at run time.

- Types of Events
  - Key Event
  - Protocol Event
  - Entry/Exit Screen
  - Timer

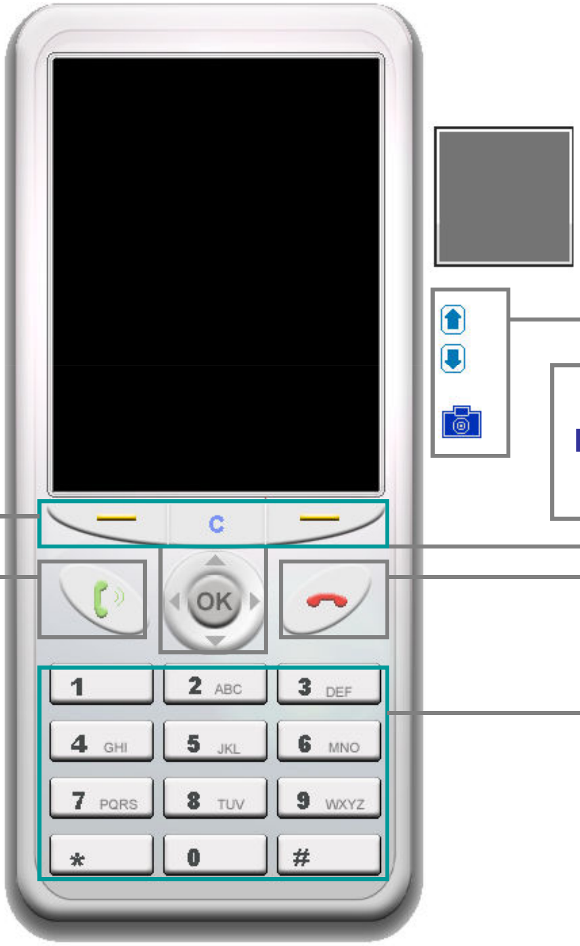
Key Event Handling

- Typically used by applications and category functions
  - Set key handler for particular key
  - Set Key handler for group of keys
  - Execute current key handler for key press event
  - Clear key handlers for particular key
  - Clear key handlers for all keys
  - Special handling for Power and End Key

Key Code

Keys defined in mmi_keypads_enum:

{KEY_0, KEY_1, KEY_2, …, KEY_LSK, …}



KEY_VOL_UP
KEY_VOL_DOWN
KEY_CAMERA

KEY_LSK
KEY_RSK
KEY_CLEAR

KEY_UP_ARROW,
KEY_LEFT_ARROW KEY_ENTER KEY_RIGHT_ARROW
KEY_DOWN_ARROW

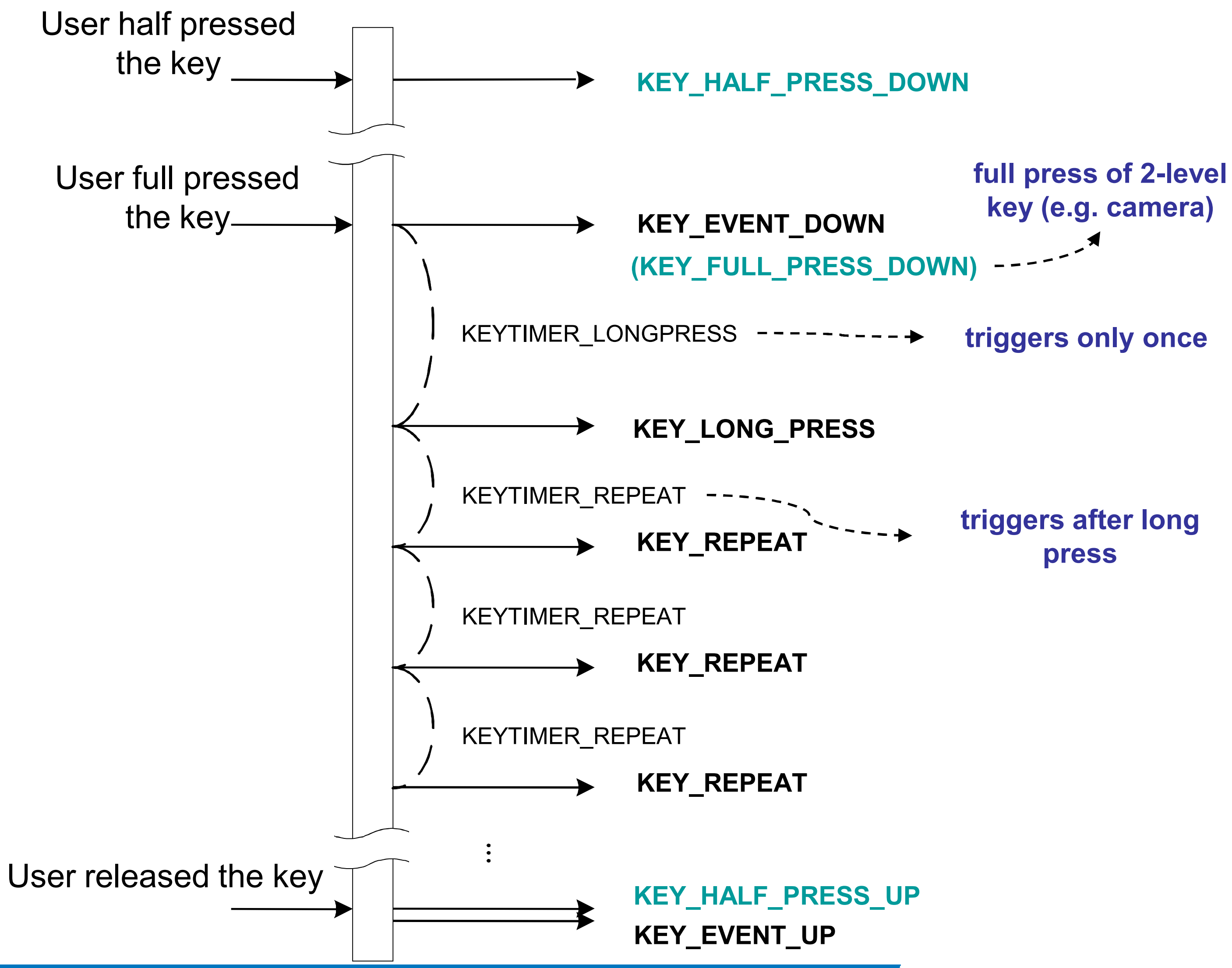
KEY_SEND

KEY_END

KEY_1	KEY_2	KEY_3
KEY_4	KEY_5	KEY_6
KEY_7	KEY_8	KEY_9
KEY_STAR	KEY_0	KEY_POUND

Orange color key codes: mandatory

Key Type



User half pressed the key

KEY_HALF_PRESS_DOWN

User full pressed the key

KEY_EVENT_DOWN

(KEY_FULL_PRESS_DOWN)

full press of 2-level key (e.g. camera)

KEYTIMER_LONGPRESS

triggers only once

KEY_LONG_PRESS

KEYTIMER_REPEAT

KEY_REPEAT

triggers after long press

KEYTIMER_REPEAT

KEY_REPEAT

KEYTIMER_REPEAT

KEY_REPEAT

⋮

User released the key

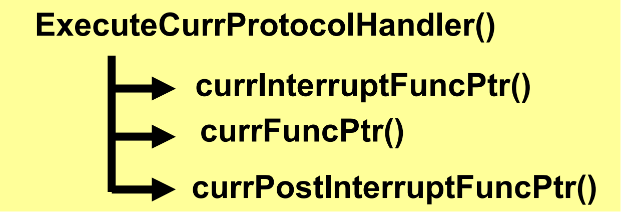
KEY_HALF_PRESS_UP

KEY_EVENT_UP

Time

MMI Task – Detailed Description (contd.)

- Protocol Event Handlers API – Typically used by applications
  - Set protocol and interrupt event handler.
  - Execute Current protocol event handler.
  - Clear handler for specific protocol event.
  - Clear all protocol event handler.



ExecuteCurrProtocolHandler()

currInterruptFuncPtr()

currFuncPtr()

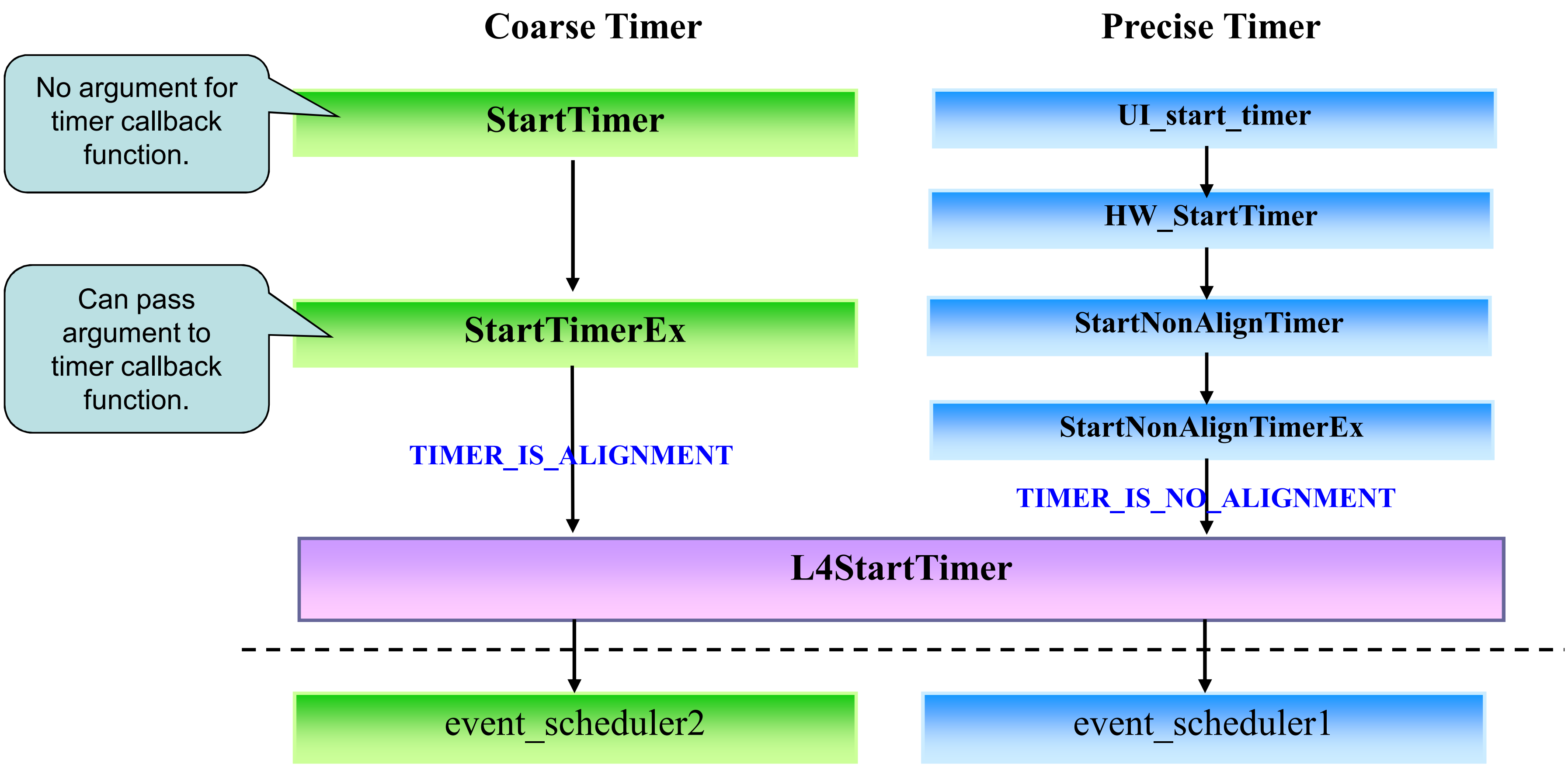
currPostInterruptFuncPtr()

- Entry/Exit Screen Handlers
  - Entry the new screen; register the entry and exit handler
  - Highlight Handlers
  - Hint Handlers

MMI Task – Detailed Description (contd.)

- Timer API
  - Coarse Timer
    - StartTimer(U16 timerid, U32 delay, FuncPtr funcPtr)
    - StartTimerEx(U16 timerid, U32 delay, oslTimerFuncPtr funcPtr, void* arg)
  - Precise Timer: usually used in UI part
    - StartNonAlignTimer(U16 timerid, U32 delay, FuncPtr funcPtr)
    - StartNonAlignTimerEx(U16 timerid, U32 delay, oslTimerFuncPtr funcPtr, void* arg)
  - StopTimer(U16 timerid)

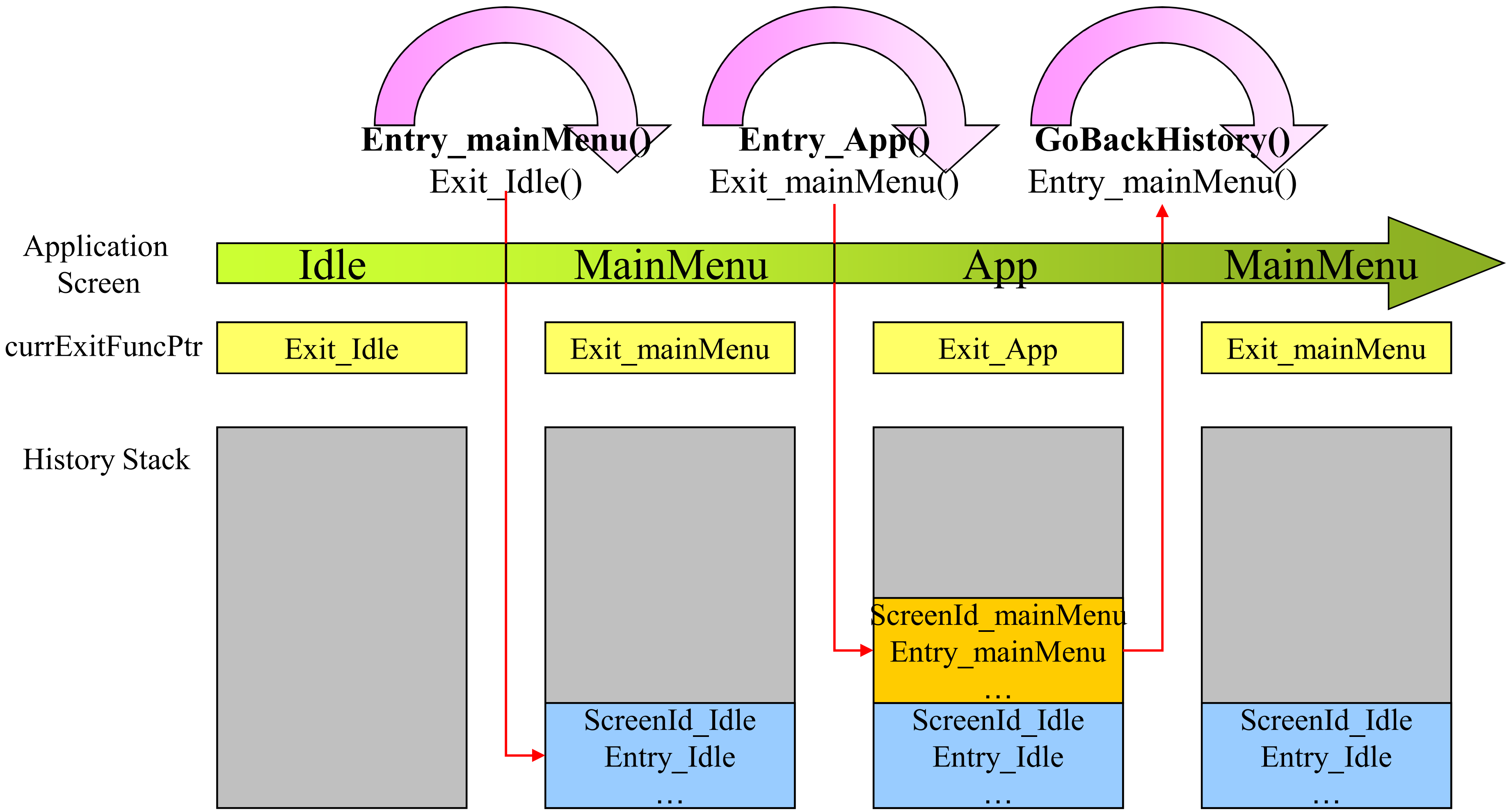
MMI Start Timer API



History

- Collection of nodes
- Each node contains the context of a screen
- Implemented as a LIFO stack
- Structure of history node
  - Screen ID: associated with a screen to be saved
  - Entry Function Pointer: to redraw the screen
  - Input Buffer: to save running text data for this screen
  - GUI Buffer: to save UI related information for this screen

Example of History Operation



History API

- – Add Node to History
- – Delete ‘N’ nodes from history
- – Go back ‘N’ nodes in history
- – Retrieve history for a screen
- – Retrieve input buffer for screen
- – Retrieve UI buffer for screen
- – Dump History for debugging
- – Initialize history

Memory Management

- Control buffer can only allocate fixed-size small buckets (<2KB).
  - get_ctrl_buffer
  - free_ctrl_buffer

- ASM (Application Shared Memory)
  - Screen-based ASM
    - Allocated/Release when entering/exiting a screen
  - Application-based ASM
    - Allocated/Release when entering/leaving an application

Screen-based ASM

- It is typically allocated inside a screen-entry function, and released inside the corresponding screen-exit function.
- If memory isn’t released in screen exit function, the handset will ASSERT
- Because the memory is used for the active screen, we could control the memory fragmentation

App-based ASM

- The application using app-based ASM have the following property
  - When its screens are overlapped by other MMI screens (e.g. incoming call), it usually holds its memory.
  - The application could provide the destroy mechanism, and allow the other application to kill it.
- When an application fails to allocate app-based ASM, it can choose to enter “Out of Memory” screen and prompt user to stop other application.

App-based ASM Config

mcu\applib\mem\include\app_mem_config.h

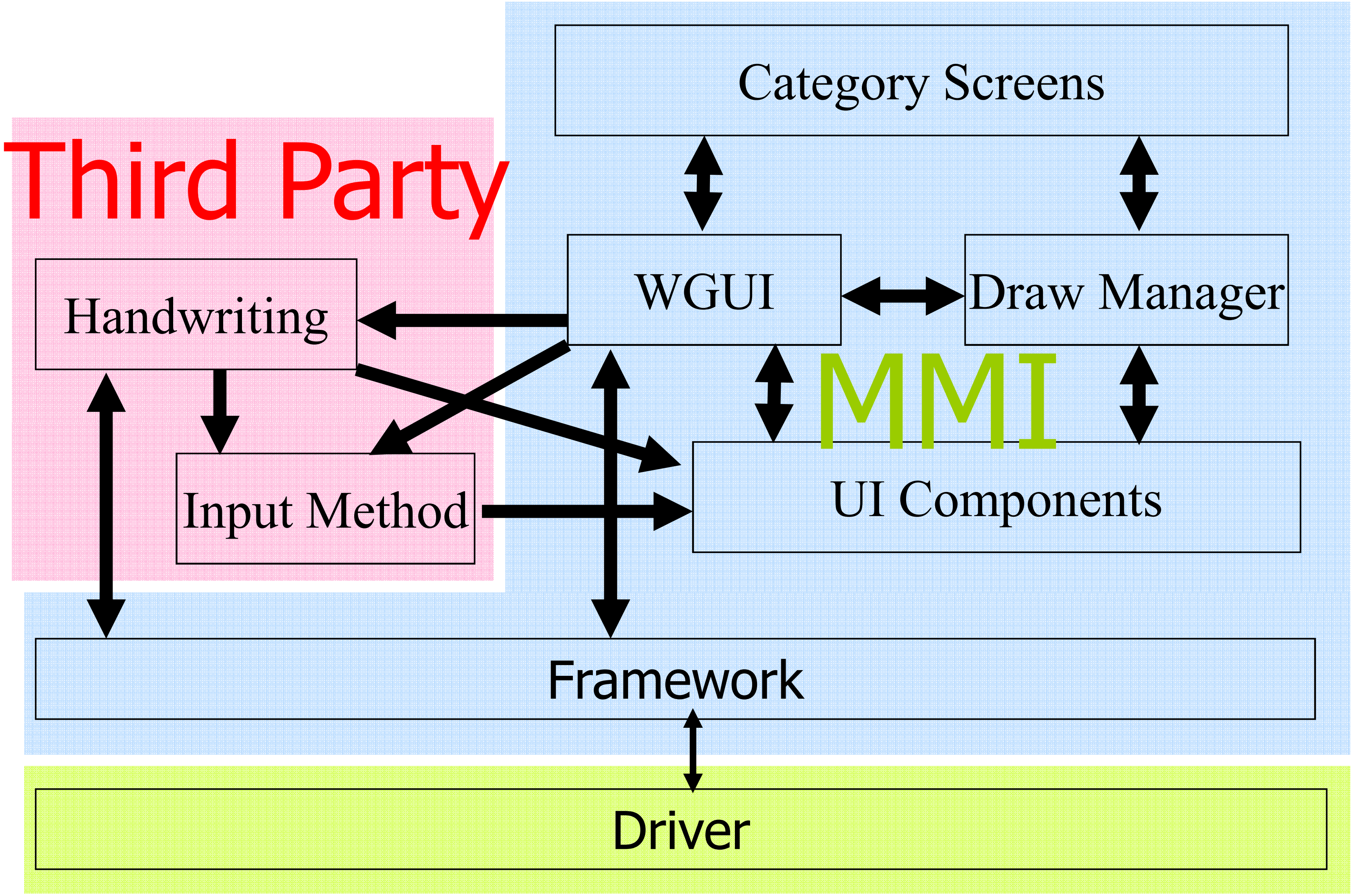
```
#ifdef __MMI_BARCODEREADER__
    #include "lcd_sw_rnd.h"
    #include "PixtelDataTypes.h"
    #include "gdi_include.h"
    #include "barcodereaderGprot.h"
    #include "custom_mmi_default_value.h"

    #define APPMEM_BR_APP_TOTAL_SIZE BR_APP_TOTAL_SIZE
#endif
…
#if !defined(APPMEM_BR_APP_TOTAL_SIZE)
    #define APPMEM_BR_APP_TOTAL_SIZE (0)
#endif
…
typedef union
{
    …
    kal_uint8 APP_BR[APPLIB_MEM_AP_POOL_SIZE_CONFIG(APPMEM_BR_APP_TOTAL_SIZE)];
    …
}
```

Screen-based ASM Config

Mcu\plutommi\mmi\inc\ScrMemMgr.h

```
/* Video Player */
#ifdef __MMI_VIDEO_PLAYER__
    #include "lcd_sw_rnd.h"
    #include "gdi_include.h"
    #include "VdoPlyGProt.h"

    #define SCRMEM_VDOPLY_POOL_SIZE     (VDOPLY_OSD_BUFFER_SIZE)
    S32 vdoply_mem = (SCRMEM_VDOPLY_POOL_SIZE);
#endif /* __MMI_VIDEO_PLAYER__ */
…
#if !defined(SCRMEM_VDOPLY_POOL_SIZE)
    #define SCRMEM_VDOPLY_POOL_SIZE (0)
#endif
…
typedef union
{
    …
    U8 SCR_VDOPLY[APPLIB_MEM_SCREEN_POOL_SIZE_CONFIG(SCRMEM_VDOPLY_POOL_SIZE)]; /*
    Video Player */
   …
} screen_asm_pool_union;
```

Touch Screen Overview

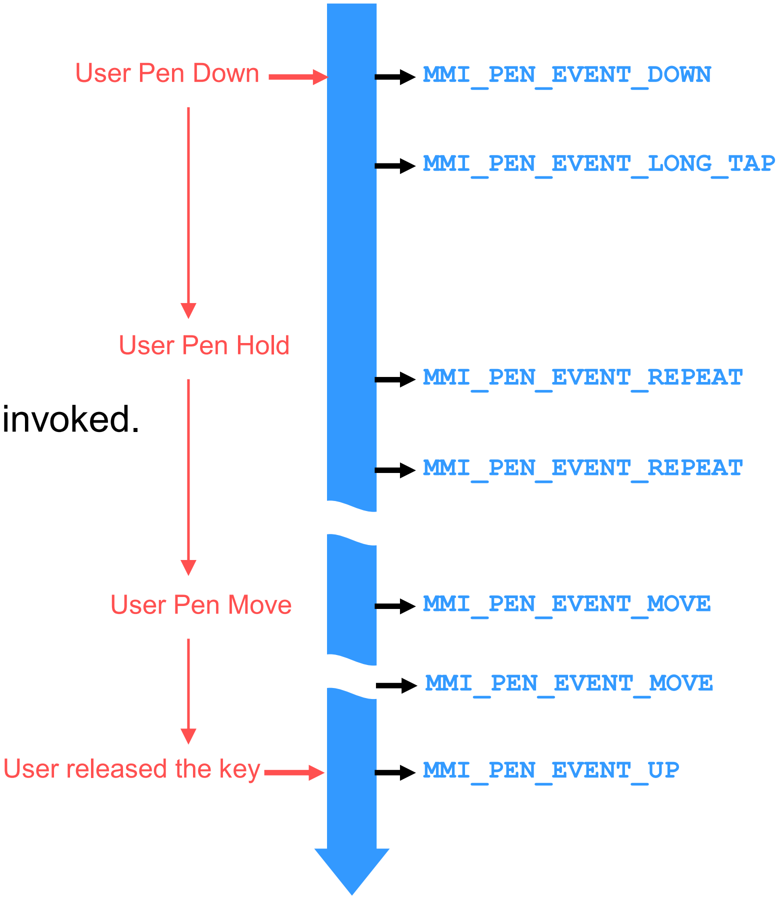


Touch Screen Dual Mode

- Event-based mode
  - For normal user interface such as menus.
  - Low sampling rate.
  - Power saving.
- Stroke-based mode
  - For handwriting.
  - High sampling rate.
- How to decide mode?
  - Driver decide whether the pen event position is inside the hand-writing region.

Touch Screen Events

- Event-based Mode
  - MMI_PEN_EVENT_DOWN
  - MMI_PEN_EVENT _LONGTAP
    - At pen down position only. At-most once.
  - MMI_PEN_EVENT _REPEAT
    - At any position. Any number of times.
  - MMI_PEN_EVENT_MOVE
  - MMI_PEN_EVENT _UP
  - MMI_PEN_EVENT_ABORT
    - Mmi_pen_reset() or mmi_pen_disable() is invoked.
    - Ex. MMI screen is switched.
- Stroke-based Mode
  - MMI_PEN_STROKE_DOWN
  - MMI_PEN_STROKE_MOVE
  - MMI_PEN_STROKE_UP



Touch Screen

- Driver
  - Events sampling and translates ADC value to coordinate.
  - Touch panel calibration
  - Provide interface to configure touch panel parameters
- Framework
  - Specify rules for pen events translation
  - Pen state management
  - Provide interfaces of driver services for MMI
- UI components
  - Display and manipulate UI objects
  - Translate component-specified events

Touch Screen

- WGUI and Draw Manager
  - Decide the size and position of each UI object
  - Route events
- Category screens
  - Compose the screen using Draw Manager
  - Provide interface for applications
- Hand-Writing Module
  - Collects user input and pass to recognizing engine
  - Provides virtual keypad interface

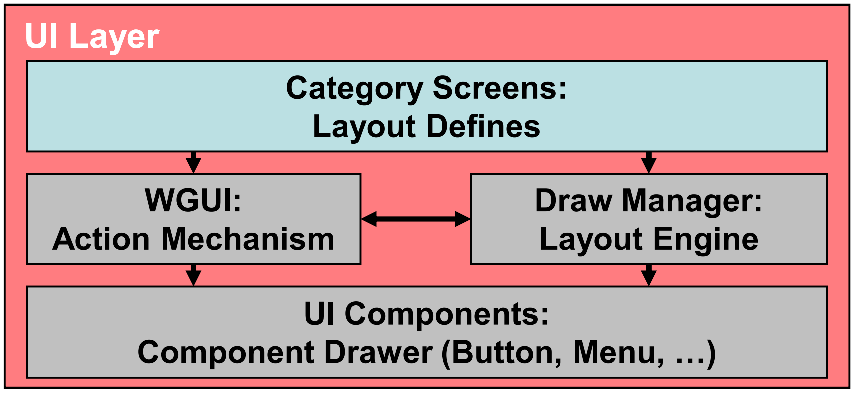
Confidential B

MEDIATEK

UI Layer

MMI Task – UI Layer

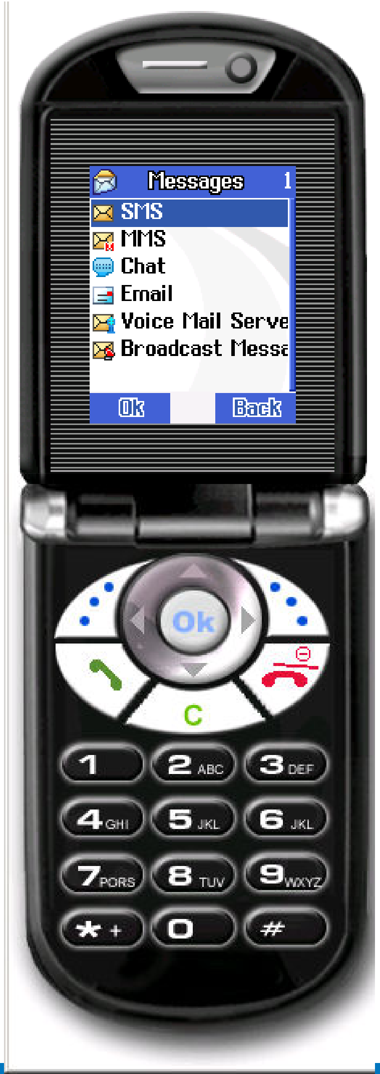
- Provides UI display functions to applications
- Components of UI Layer
  - Category Screens
    - Intelligent wrappers to draw the screens of applications
    - Accept resources such as String IDs and Image IDs from applications.
    - Keep the application independent of the layout and the look-and-feel of screens
    - Provide interfaces of history

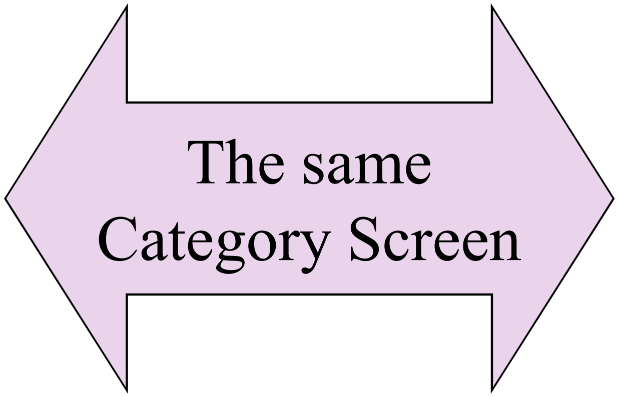


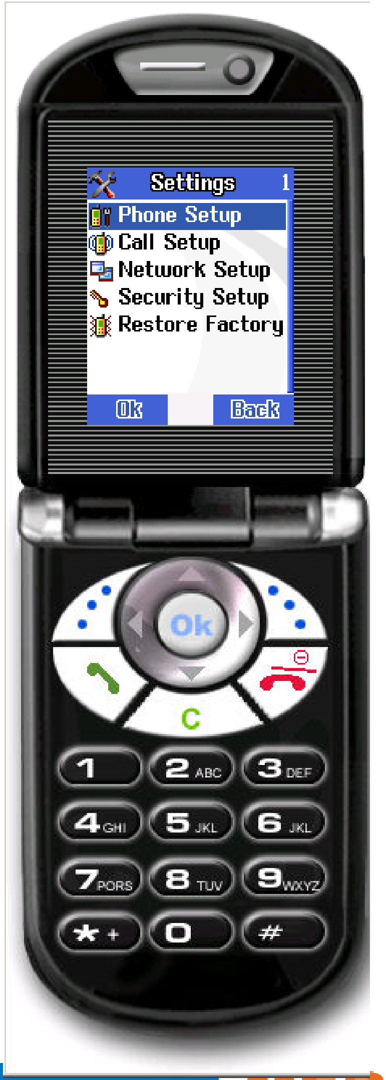
MMI Task – UI Layer

- Fonts
  - This is the data that is used by the graphics library to render characters on the display
- Images
  - Set of device independent images used as Icons, splash screens and wallpapers
- Graphics Library (GDI)
  - Provides the support for graphics primitives
  - Contains functions to display Fonts and Images

Concept of Category Screen




The same Category Screen



Confidential B

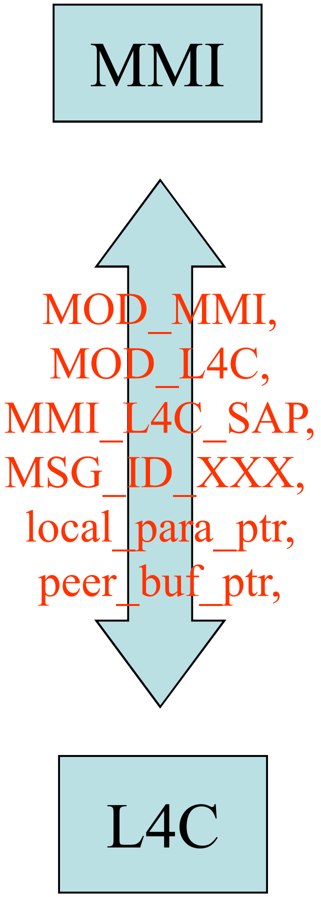
MEDIATEK

MMI/Protocol Interface

MMI/Protocol Interface

- How To Communicate
  - Send/Receive messages thru the message Queue.
- Communication Data

```
typedef struct ilm_struct {
     oslModuleType    oslSrcId; // Source module ID.
     oslModuleType    oslDestId; // Destination module ID.
     oslMsgType       oslSapId; // service access point.
     oslMsgType       oslMsgId; // message name ID.
     oslParaType       *oslDataPtr; //local parameter buffer
     oslPeerParaPtr     *oslPeerBuffPtr; //peer buffer pointer
} ilm_struct;
```



MMI/Protocol Interface

- How to listen a message from MMI Queue:
  - From task create and entry a message loop.
    - OslReceiveMsgExtQ(mmi_qid, &mmi_message);
- How to send a message to L4C:
  - Step1: Construct a local parameter buffer.
  - Step2: Assign required values into local parameter buffer.
  - Step3: Fill out information in ilm_struct
  - Step4: Send out the message to the L4C module.
- How to receive a message from L4C:
  - Register a response message callback.
    - SetProtocolEventHandler(FuncCB, msg_id)

MMI/Protocol Interface
Local Paramenter

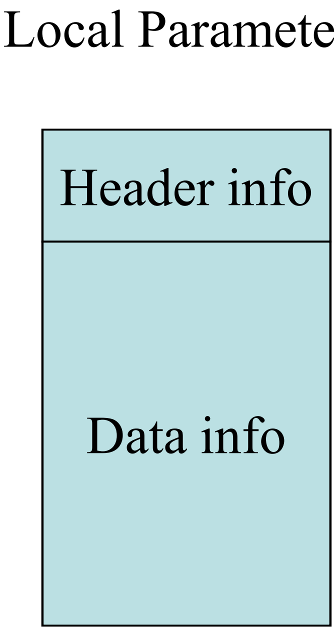
- Message Information:

  Header info + Data info:

```
Ex: typedef struct {
   LOCAL_PARA_HDR
   kal_uint8  volume_type;
   kal_uint8  volume_level;
} mmi_eq_set_volume_req_struct;
```

- How To Create Local Parameter:
  - Dynamic to allocate memory buffer:

    By OslConstructDataPtr function.
- When to Free Local Parameter:
  - While L4 receive the information, after finishing to process the message, L4 task will automatically free this buffer.



MMI/Protocol Interface
Peer Buffer

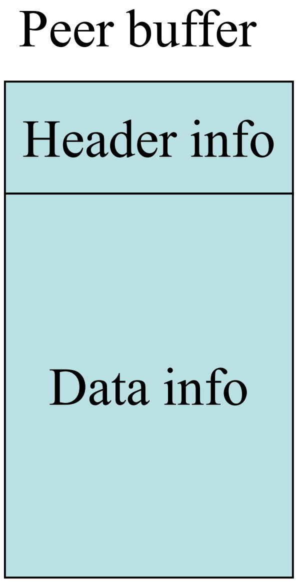
- Message Information:
  - Header info + Data info

    Ex: typedef struct {

    PEER_BUFF_HDR

    void *ptr;

    } mmi_example;



- How To Create Peer Buffer Parameter:
  - Dynamic to allocate memory buffer:
    - MMI did not use this buffer to communicate with L4
- When to Free Peer Buffer:
  - While other task receive the information, after finishing to process the message, other task will automatically free this buffer.

Example (Set Volume)

- Set a volume request:

```
void SetVolumeLevelReq(volume_type_enum volume_type,U8 volume_level)
{
    MYQUEUE Message;
    mmi_eq_set_volume_req_struct *setVolumeLevelReq;
    Message.oslMsgId = MSG_ID_MMI_EQ_SET_VOLUME_REQ; //Message ID, reference the l4a.h file
    setVolumeLevelReq = OslConstructDataPtr(sizeof(mmi_eq_set_volume_req_struct)); //Create local parameter buffer
    setVolumeLevelReq->volume_type = volume_type;
    setVolumeLevelReq->volume_level = volume_level;
    Message.oslDataPtr = (oslParaType *)setVolumeLevelReq; //Local parameter buffer
    Message.oslPeerBuffPtr= NULL; //Peer parameter buffer
    Message.oslSrcId=MOD_MMI; //Send from Source module
    Message.oslDestId=MOD_L4C; //Send to destination  module
    OslMsgSendExtQueue(&Message); //Send to L4 task
}
```

Example (Play Pattern)

- Play a Pattern(LED/LCD/VIB) request:

```
void SendPlayPatternReqToHW(U8 pattern, U8 action)
{
        MYQUEUE Message;
         mmi_eq_play_pattern_req_struct *displayLedPattern;
        Message.oslMsgId = MSG_ID_MMI_EQ_PLAY_PATTERN_REQ;
        displayLedPattern = OslConstructDataPtr(sizeof(mmi_eq_play_pattern_req_struct));
        displayLedPattern->pattern = pattern;
        displayLedPattern->action = action;
        Message.oslDataPtr = (oslParaType *)displayLedPattern;
        Message.oslPeerBuffPtr= NULL;
        Message.oslSrcId=MOD_MMI;
        Message.oslDestId=MOD_L4C;
        OslMsgSendExtQueue(&Message);
}
```

Example(GPIO detect)

- Register GPIO Detect indication:
  - SetProtocolEventHandler(GpioDetectInd, MSG_ID_MMI_EQ_GPIO_DETECT_IND);
- Receive GPIO Detect indication:

```
void GpioDetectInd(void * info)
{
    mmi_eq_gpio_detect_ind_struct *gpioDetectInd;
    gpioDetectInd =     (mmi_eq_gpio_detect_ind_struct *) info;
    switch(gpioDetectInd->gpio_device)
    {
            case EXT_DEV_EARPHONE:
                        EarphonePlugInPopup();
            break;
            …..
    }
}
```

Confidential B

MEDIATEK

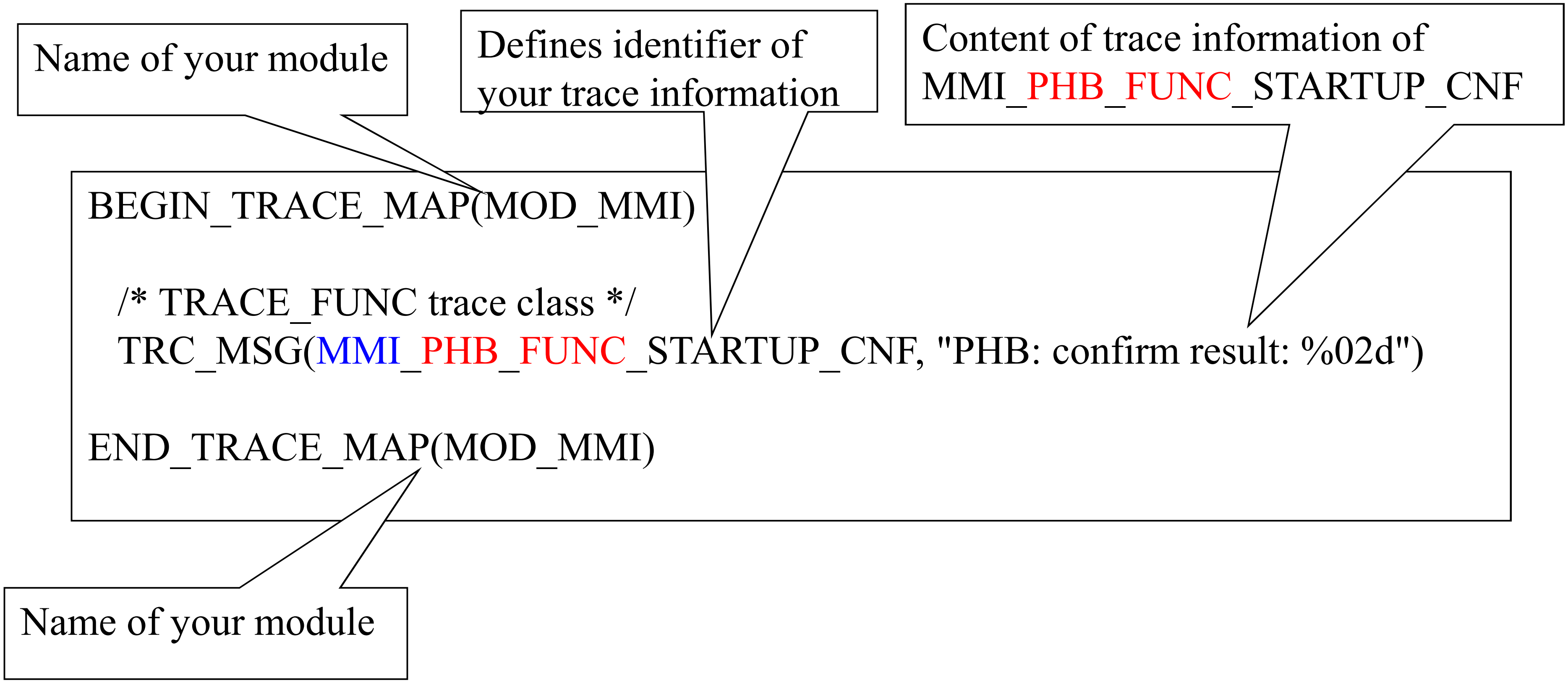
Debug Support

Debugging Support

- MMI_PRINT
  - Transfer raw data via data cable
  - Support to print out the %s string type for dynamic trace
- MMI_TRACE (Recommended)
  - Only transfer trace message id via data cable
  - Save target ROM size and bandwidth of data cable

Debugging Support (contd.)

- Describe the trace map (at ***MMI_<mod>_trc.h***)



Name of your module

Defines identifier of your trace information

Content of trace information of MMI_PHB_FUNC_STARTUP_CNF

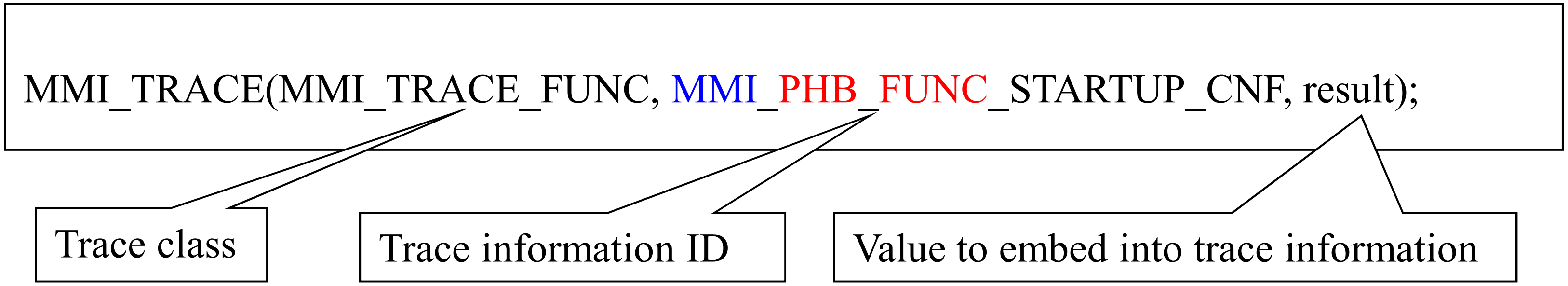
```
BEGIN_TRACE_MAP(MOD_MMI)

   /* TRACE_FUNC trace class */
   TRC_MSG(MMI_PHB_FUNC_STARTUP_CNF, "PHB: confirm result: %02d")

END_TRACE_MAP(MOD_MMI)
```

Name of your module

Debugging Support (contd.)

- Inside your code, call MMI_TRACE(…) function.

Code:



```
MMI_TRACE(MMI_TRACE_FUNC, MMI_PHB_FUNC_STARTUP_CNF, result);
```

Trace class

Trace information ID

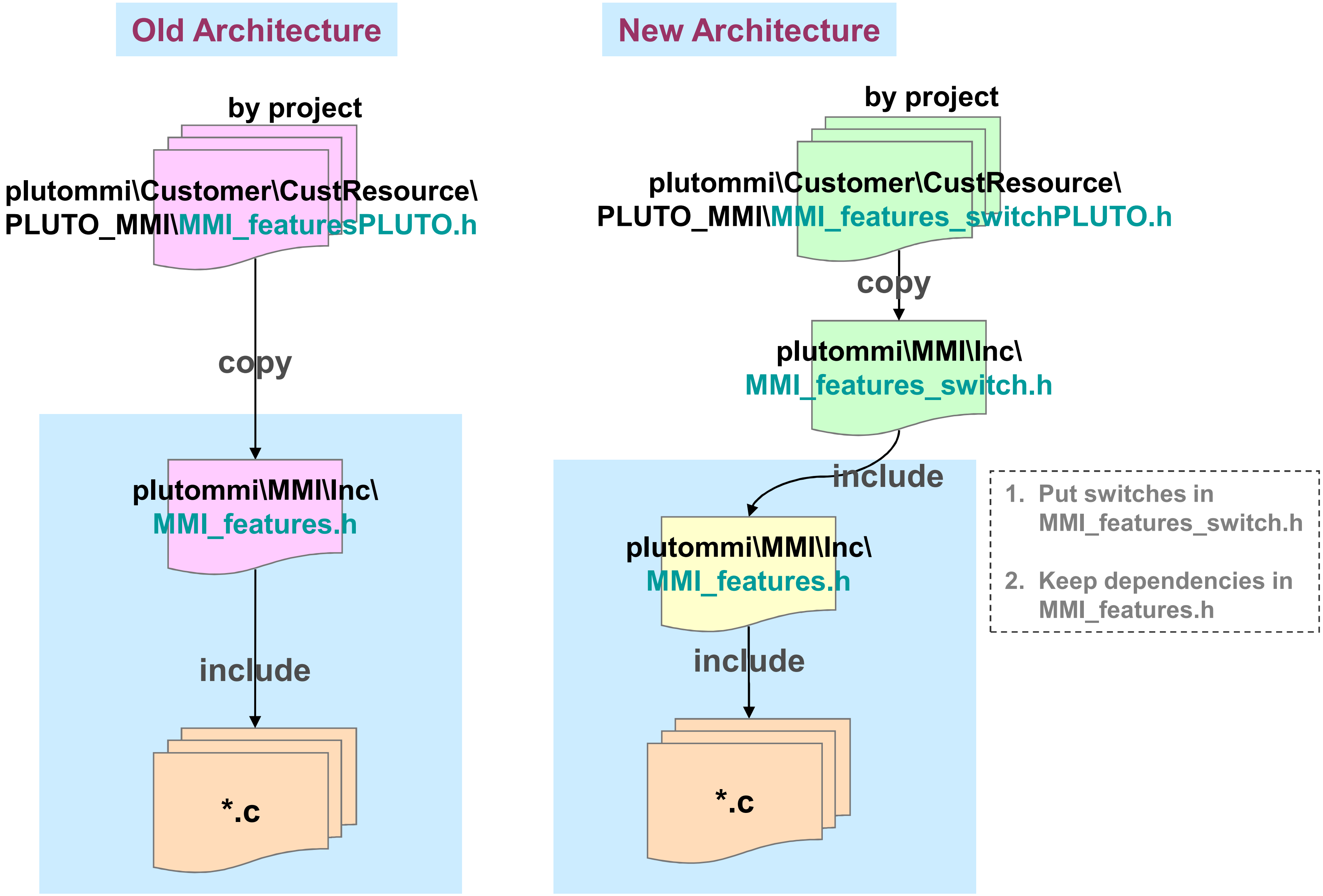
Value to embed into trace information

Log in Catcher:

“PHB: confirm result: 08”

Confidential B

Configuration Management



Old Architecture

by project

plutommi\Customer\CustResource\
PLUTO_MMI\MMI_featuresPLUTO.h

copy

plutommi\MMI\Inc\
MMI_features.h

include

*.c

New Architecture

by project

plutommi\Customer\CustResource\
PLUTO_MMI\MMI_features_switchPLUTO.h

copy

plutommi\MMI\Inc\
MMI_features_switch.h

include

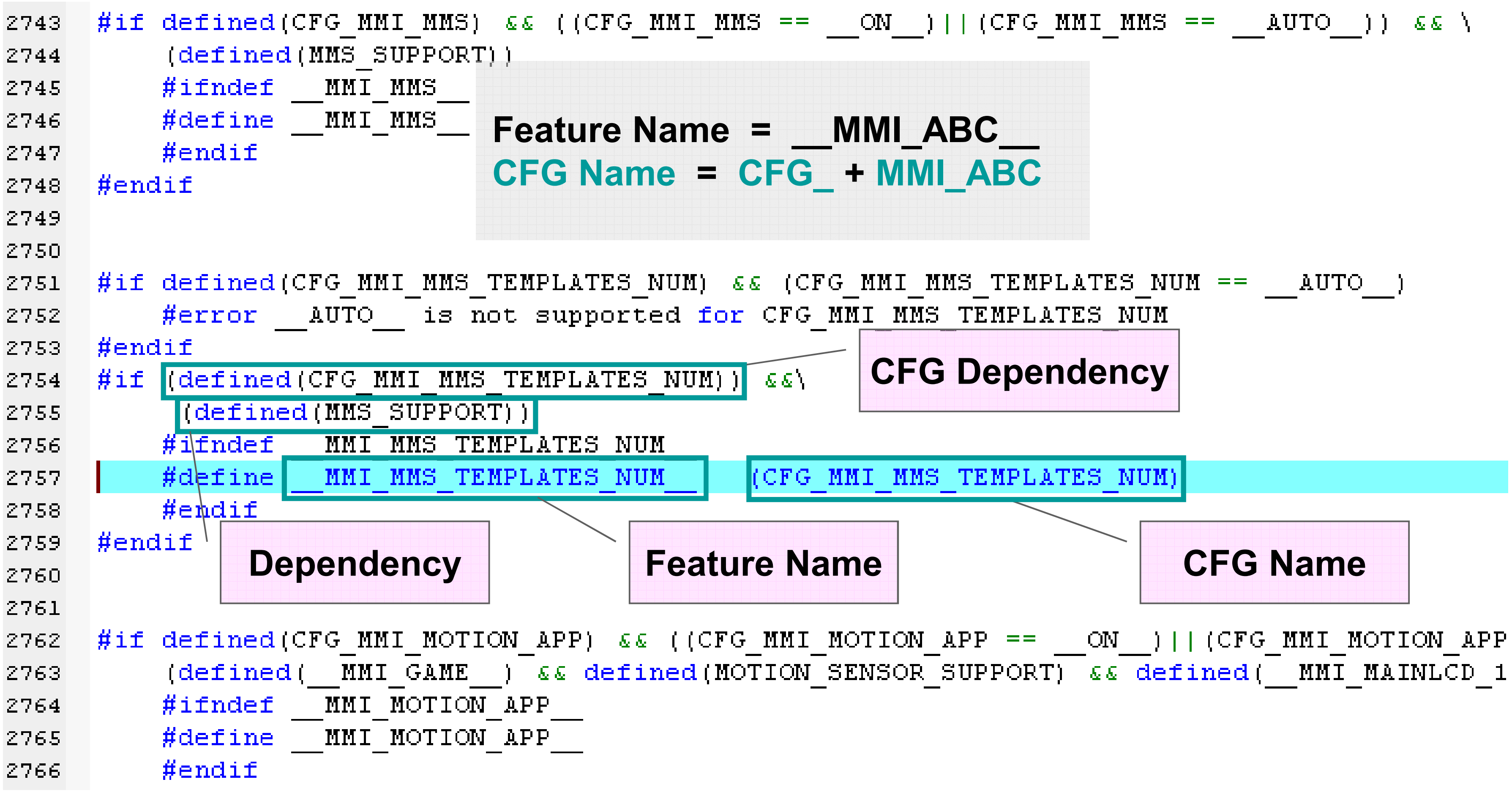
plutommi\MMI\Inc\
MMI_features.h

include

*.c

1. Put switches in MMI_features_switch.h
2. Keep dependencies in MMI_features.h

MMI_features.h



```
#if defined(CFG_MMI_MMS) && ((CFG_MMI_MMS == __ON__)||(CFG_MMI_MMS == __AUTO__)) && \
    (defined(MMS_SUPPORT))
    #ifndef __MMI_MMS__
    #define __MMI_MMS__
    #endif
#endif


#if defined(CFG_MMI_MMS_TEMPLATES_NUM) && (CFG_MMI_MMS_TEMPLATES_NUM == __AUTO__)
    #error __AUTO__ is not supported for CFG_MMI_MMS_TEMPLATES_NUM
#endif
#if (defined(CFG_MMI_MMS_TEMPLATES_NUM)) &&\
     (defined(MMS_SUPPORT))
    #ifndef __MMI_MMS_TEMPLATES_NUM__
    #define __MMI_MMS_TEMPLATES_NUM__     (CFG_MMI_MMS_TEMPLATES_NUM)
    #endif
#endif


#if defined(CFG_MMI_MOTION_APP) && ((CFG_MMI_MOTION_APP == __ON__)||(CFG_MMI_MOTION_APP
    (defined(__MMI_GAME__) && defined(MOTION_SENSOR_SUPPORT) && defined(__MMI_MAINLCD_1
    #ifndef __MMI_MOTION_APP__
    #define __MMI_MOTION_APP__
    #endif
```

Feature Name = __MMI_ABC__

CFG Name = CFG_ + MMI_ABC

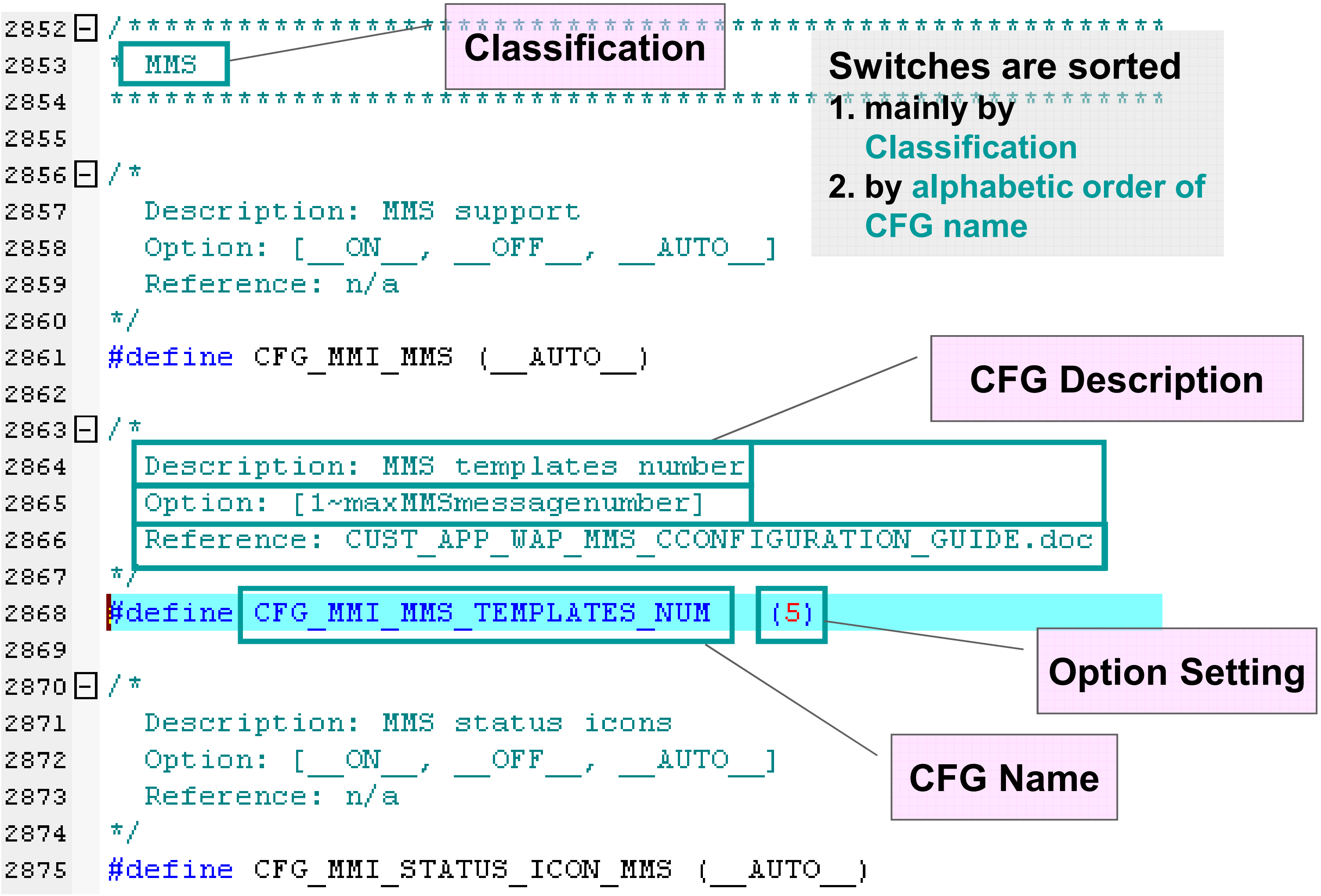
CFG Dependency

Dependency

Feature Name

CFG Name

MMI_features_switch.h



```
/*********************************************************************
 * MMS
 *********************************************************************

/*
  Description: MMS support
  Option: [__ON__, __OFF__, __AUTO__]
  Reference: n/a
*/
#define CFG_MMI_MMS (__AUTO__)

/*
  Description: MMS templates number
  Option: [1~maxMMSmessagenumber]
  Reference: CUST_APP_WAP_MMS_CCONFIGURATION_GUIDE.doc
*/
#define CFG_MMI_MMS_TEMPLATES_NUM  (5)

/*
  Description: MMS status icons
  Option: [__ON__, __OFF__, __AUTO__]
  Reference: n/a
*/
#define CFG_MMI_STATUS_ICON_MMS (__AUTO__)
```

Classification

Switches are sorted

1. mainly by Classification
2. by alphabetic order of CFG name

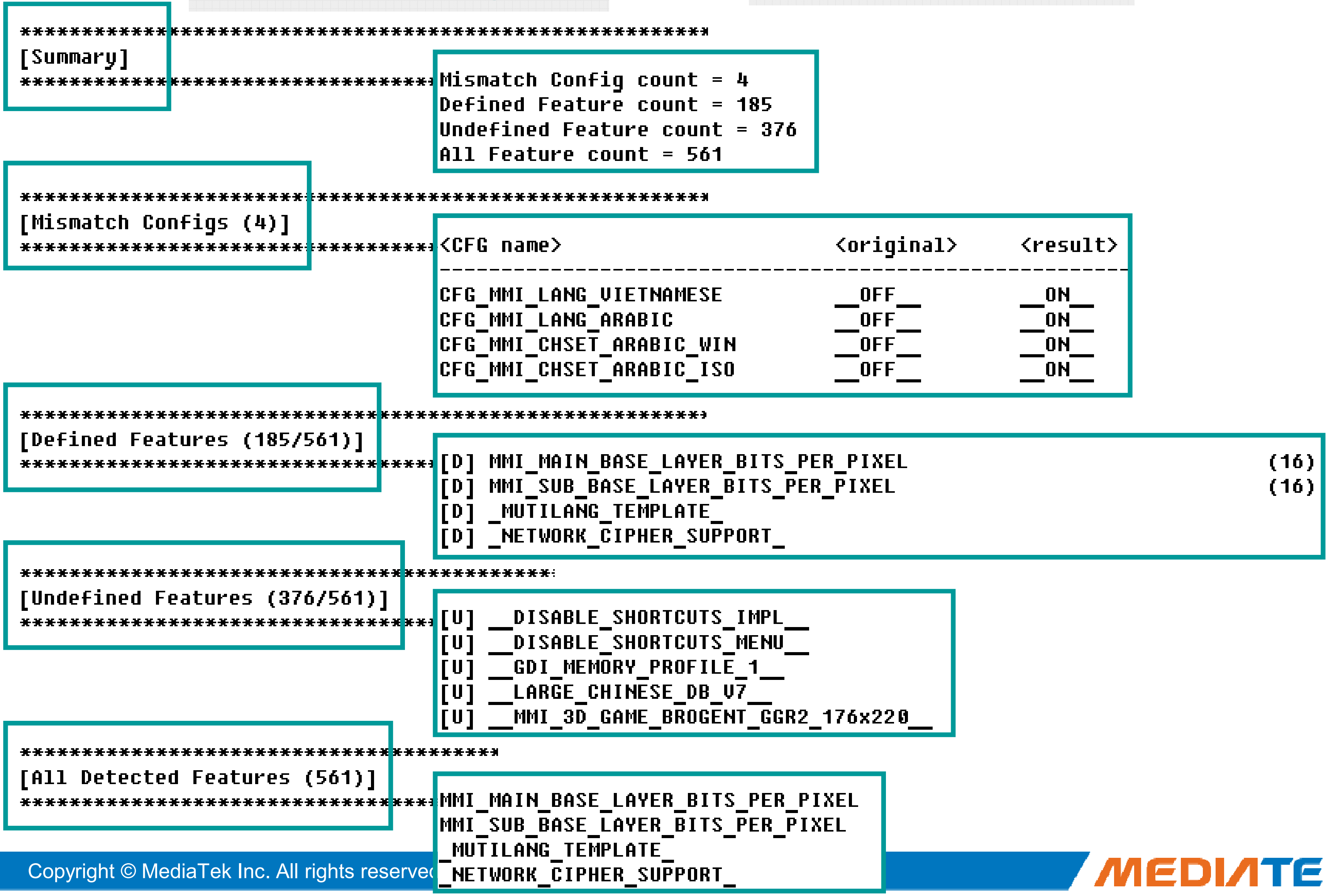
CFG Description

Option Setting

CFG Name

mcu\build\PrjName\log\
MMI_features.log

make <PrjName> gprs
mmi_feature_check



```
*********************************************************
[Summary]
**********************************Mismatch Config count = 4
                                  Defined Feature count = 185
                                  Undefined Feature count = 376
                                  All Feature count = 561

*********************************************************
[Mismatch Configs (4)]
**********************************<CFG name>                       <original>     <result>
                                  --------------------------------------------------------
                                  CFG_MMI_LANG_VIETNAMESE          __OFF__        __ON__
                                  CFG_MMI_LANG_ARABIC              __OFF__        __ON__
                                  CFG_MMI_CHSET_ARABIC_WIN         __OFF__        __ON__
                                  CFG_MMI_CHSET_ARABIC_ISO         __OFF__        __ON__

*********************************************************
[Defined Features (185/561)]
**********************************[D] MMI_MAIN_BASE_LAYER_BITS_PER_PIXEL                               (16)
                                  [D] MMI_SUB_BASE_LAYER_BITS_PER_PIXEL                                (16)
                                  [D] _MUTILANG_TEMPLATE_
                                  [D] _NETWORK_CIPHER_SUPPORT_

*********************************************
[Undefined Features (376/561)]
**********************************[U] __DISABLE_SHORTCUTS_IMPL__
                                  [U] __DISABLE_SHORTCUTS_MENU__
                                  [U] __GDI_MEMORY_PROFILE_1__
                                  [U] __LARGE_CHINESE_DB_V7__
                                  [U] __MMI_3D_GAME_BROGENT_GGR2_176x220__

***************************************
[All Detected Features (561)]
**********************************MMI_MAIN_BASE_LAYER_BITS_PER_PIXEL
                                  MMI_SUB_BASE_LAYER_BITS_PER_PIXEL
                                  _MUTILANG_TEMPLATE_
                                  _NETWORK_CIPHER_SUPPORT_
```

Confidential B

MEDIATEK

Q & A

MEDIATEK

www.mediatek.com