MTK平台软件架构

软件二部 王刚

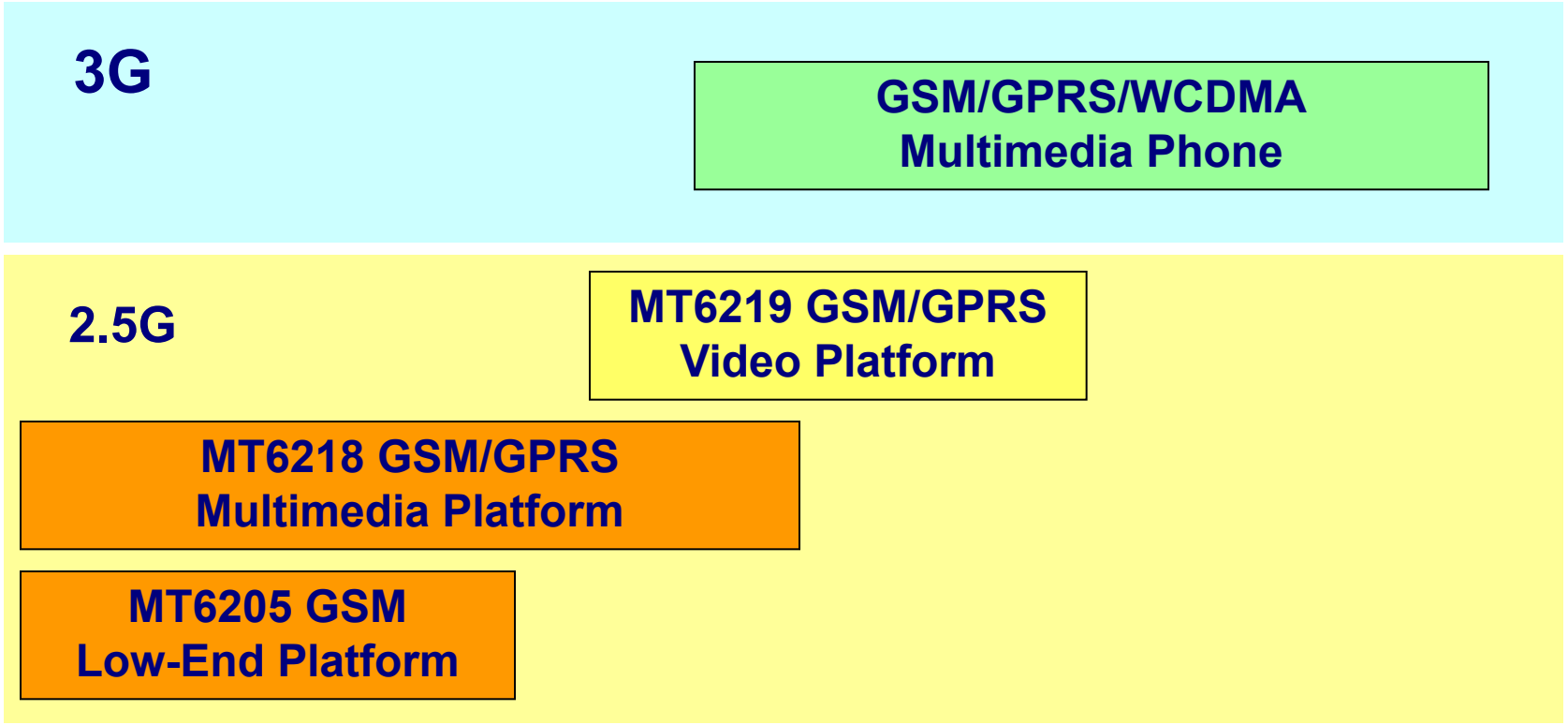
1. MTK方案简介
   - 联发科技介绍
   - MTK多媒体手机平台
   - GSM/GPRS手机软件方案
2. MTK软件分层介绍
   - 软件结构图表
   - OS
   - L1 protocol stack
   - Device driver
   - L2 L3 L4 protocol stack
   - MMI
3. MTK 方案Task架构
   - MMI task 消息处理过程
   - example

一、MTK方案简介

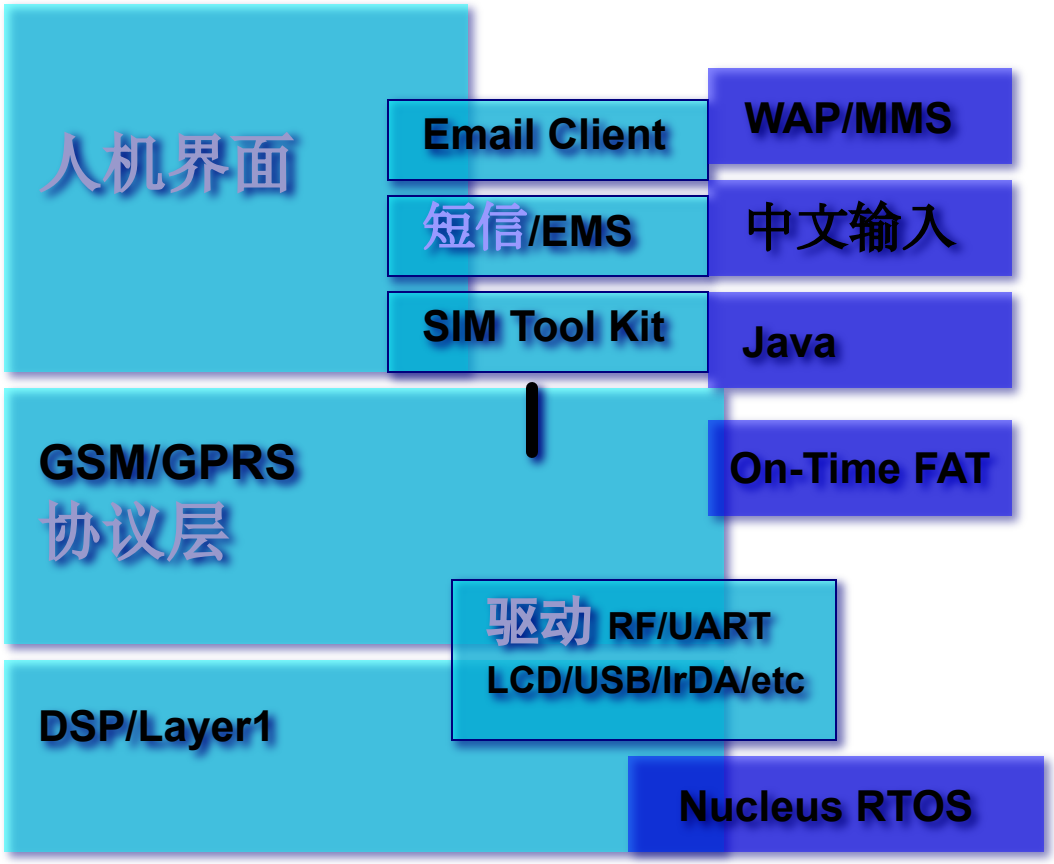
联发科技简介

- 1997年成立,全球第五大集成电路设计公司 (台湾第一大), 年营业额为10亿美金
- 全球最大之光盘及DVD播放机芯片组制造商
- 研发及技术中心: 台北 新竹 深圳 安徽合肥
- 无线通信部门于2000年成立,研发手机基带/RF芯片组及软件及发展平台,部门约200工程师

联发多媒体手机平台


3G
GSM/GPRS/WCDMA
Multimedia Phone
2.5G
MT6219 GSM/GPRS
Video Platform
MT6218 GSM/GPRS
Multimedia Platform
MT6205 GSM
Low-End Platform

GSM/GPRS 手机软件方案

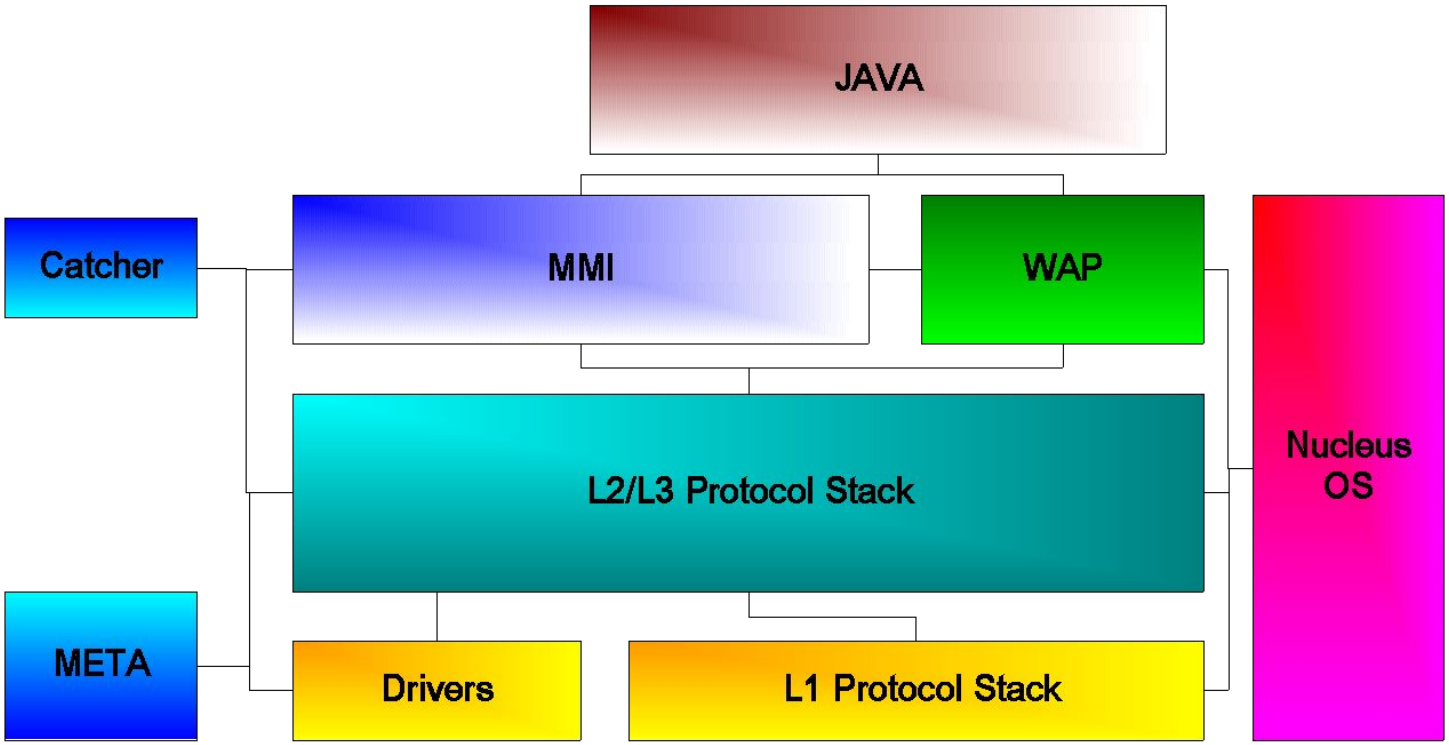


联发科技提供给客户

联发科技已集成
客户须取得原开发
商之受权

二、MTK软件分层介绍

MTK软件架构图表


JAVA
Catcher
MMI
WAP
L2/L3 Protocol Stack
Nucleus OS
META
Drivers
L1 Protocol Stack

三大组成单元

- MS (Mobile Station) 执行软件: 运行于MS上

1. 操作系统 Nucleus
2. 物理层协议栈
3. 驱动程序
4. gsm协议栈
5. MMI

- META :The Mobile Engineering Testing Architecture
- Catcher

操作系统

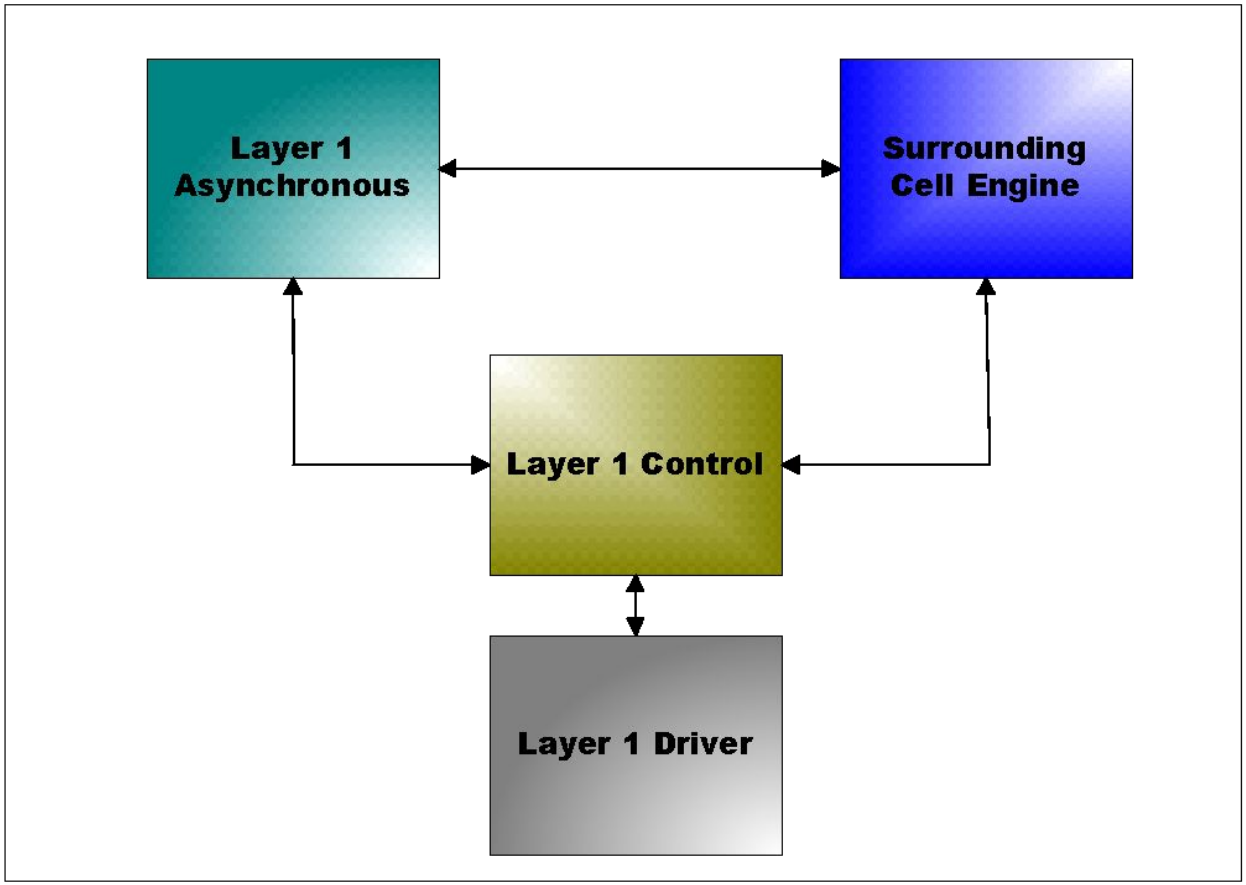
- Nucleus

  实时操作系统，MTK封装了适配层，将OS封装了一些API，这些API为其他软件提供服务，如:队列，消息，timer，内存管理等

L1层协议栈(GSM 物理层)

- L1或者叫物理层，提供物理介质上的bit流传输，遵循 gsm 技术05系列规范
- 为上层软件提供服务，且控制逻辑信道到物理信道的映射和安排
- 无线控制以及TDMA桢

L1层逻辑图


Layer 1
Asynchronous
Surrounding
Cell Engine
Layer 1 Control
Layer 1 Driver

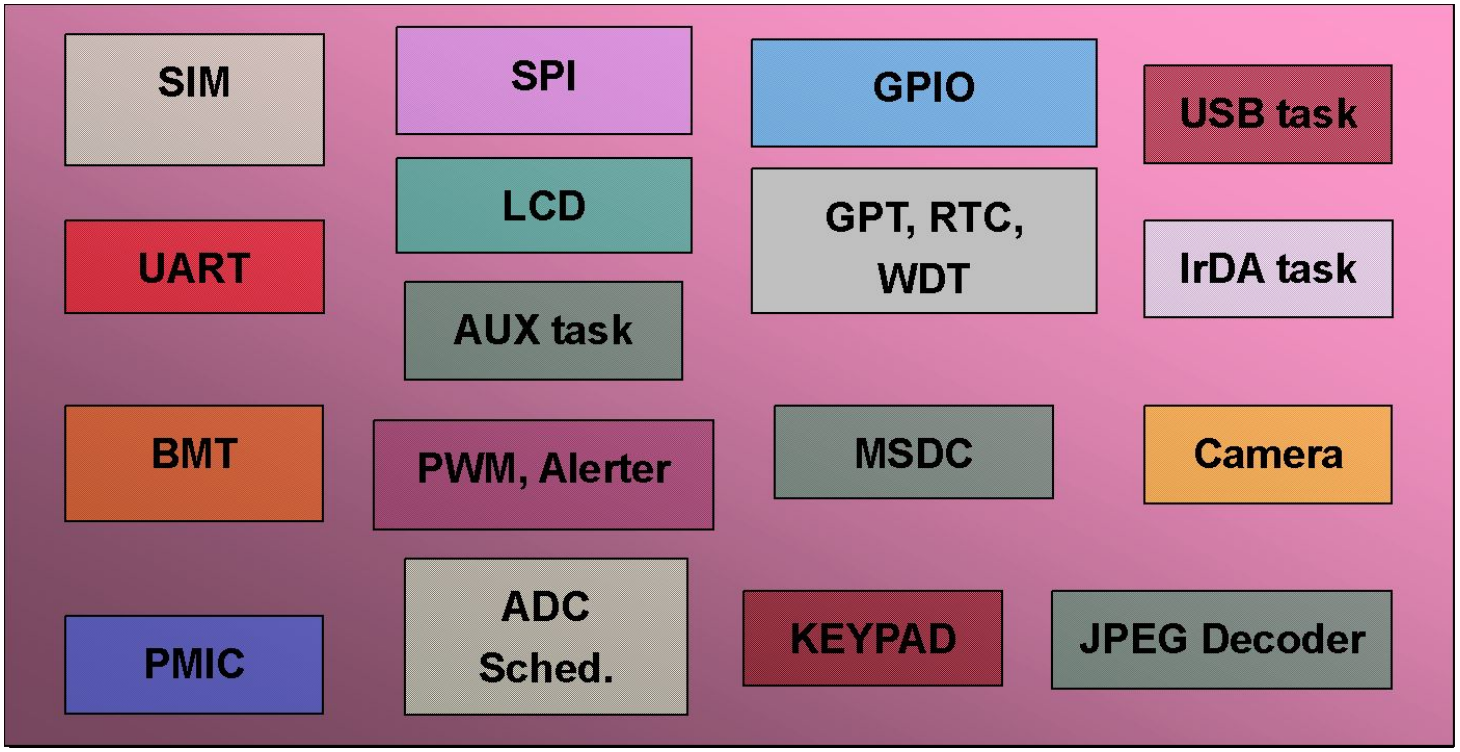
L1各部分功能

- L1异步逻辑:处理上层软件的消息请求，发送L1处理后的结果给上层软件
- Surrounding Cell Engine ：处理相邻小区的功率测量以及同步信息获取
- L1层控制:处理无线环境中的TDMA时序安排，包括定时提前以及来自基站的功率控制
- L1驱动:DSP 以及无线控制

设备驱动

- 设备驱动支持所有MCU(微处理器单元)外设的控制功能
- 本模块处理一些用户可见操作的一些设备，如键盘，LCD等
- 通过L4层接口，访问寄存器来控制外设

MTK平台外设

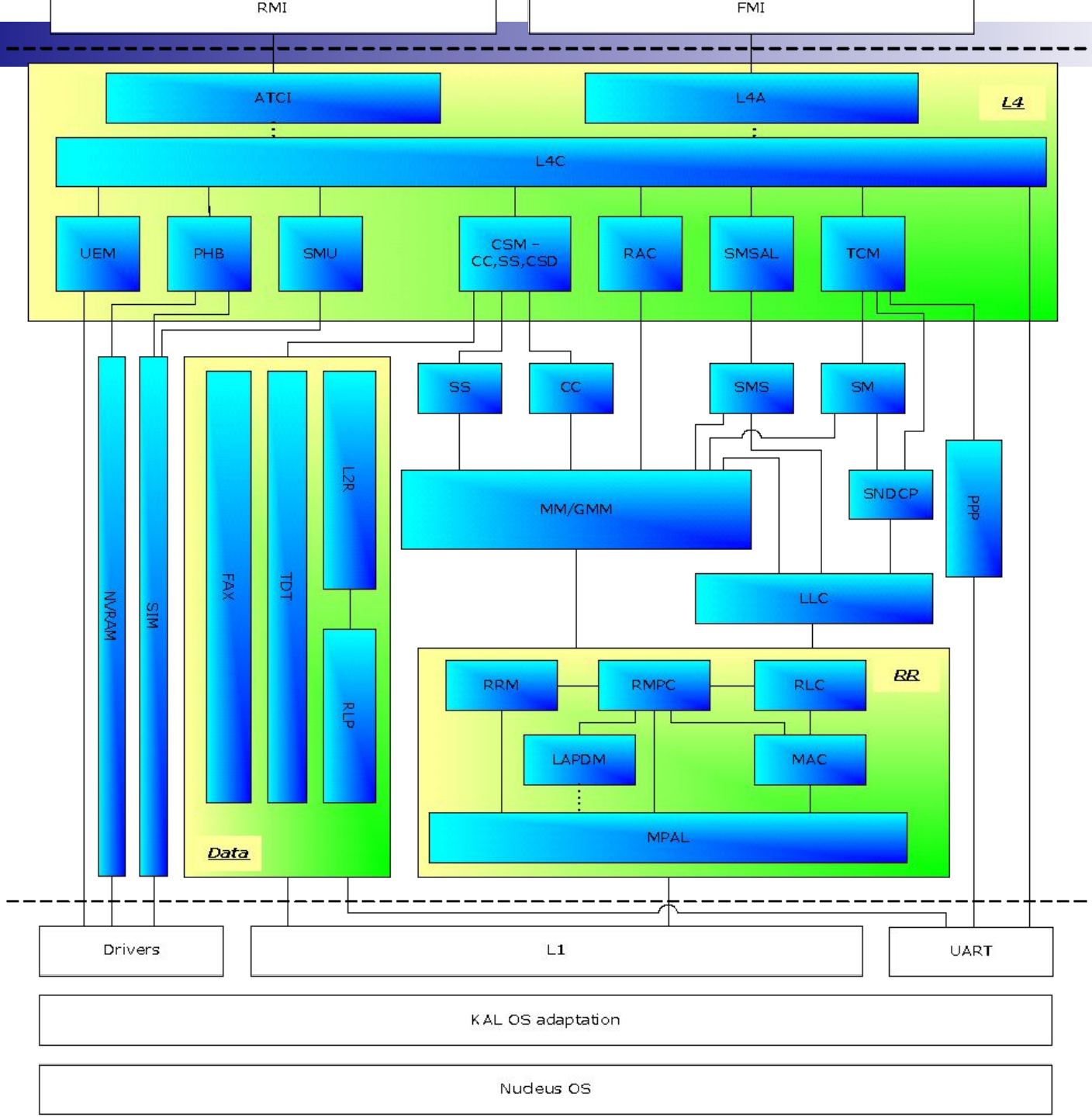

SIM
SPI
GPIO
USB task
LCD
GPT, RTC, WDT
UART
IrDA task
AUX task
BMT
PWM, Alerter
MSDC
Camera
ADC Sched.
PMIC
KEYPAD
JPEG Decoder

外设功能解释

- SIM: Subscriber Identity Module
- UART: Universal Asynchronous Receiver/Transmitter
- SPI: Serial Port Interface
- LCD: Liquid Crystal Display
- GPIO: General Purpose Input/Output
- GPT: General Purpose Timer
- RTC: Real Time Clock
- WDT: Watch Dog Timer
- PWM: Pulse Width Modulation
- Alerter
- Keypad
- PMIC: Power Management IC
- BMT: Battery Charging Management task
- AUX task: Auxiliary task
- ADC Sched.: Analog to Digital Converter Scheduler
- USB task: The USB 1.1 protocol and driver
- IrDA task: The IrDA and driver
- Camera: Camera driver for integrating with 3rd party camera module
- MSDC: Memory card driver, supporting SD, MMC cards and Memory Stick
- JPEG Decoder: Software for controlling hardware JPEG decoder

L2/L3/L4层协议栈

- 本部分覆盖了许多gsm/gprs协议需求点
- 为上层应用程序提供卓越的gsm/gprs平台
- 软件平台非常适合手机上面的操作以及通过AT命令进行PC操作


RMI
FMI
ATCI
L4A
L4
L4C
UEM
PHB
SMU
CSM – CC,SS,CSD
RAC
SMSAL
TCM
SS
CC
SMS
SM
NVRAM
SIM
FAX
TDT
L2R
RLP
Data
MM/GMM
SNDCP
PPP
LLC
RR
RRM
RMPC
RLC
LAPDM
MAC
MPAL
Drivers
L1
UART
KAL OS adaptation
Nucleus OS

模块介绍

- RMI Remote MMI，PC端通过UART口与协议栈进行通讯
- FMI Feature rich MM
- L4 MMI通过L4与gsm/gprs协议栈进行通讯，包括以下子模块
  1. ATCI: AT Command Interpreter, 解释来自PC端的命令并命令L4做相应的动作
  2. L4A: L4 adaptation Layer, MMI与L4A通过消息通信
  3. L4C: L4 Control entity, 处理所有的应用程序请求和响应
  4. UEM: User equipments adaptation, 驱动相关的适配层

5. PHB: Phone book management, 电话簿相关的处理，如分类等
6. SMU: SIM management Unit, 安全性管理以及STK
7. CSM: Circuit switching protocol stack management电路交换协议栈管理
8. RAC: Registration access control
9. SMSAL: Short message service application layer
10. TCM: Terminal context management

- NVRAM Nor-volatile RAM, 是MMI到Flash的一个适配层，保存一些默认设置
- SIM Subscriber identity module. Handle SIM behavior as ETSI 11.11 description
- DATA 电路交换数据服务，包括以下子模块

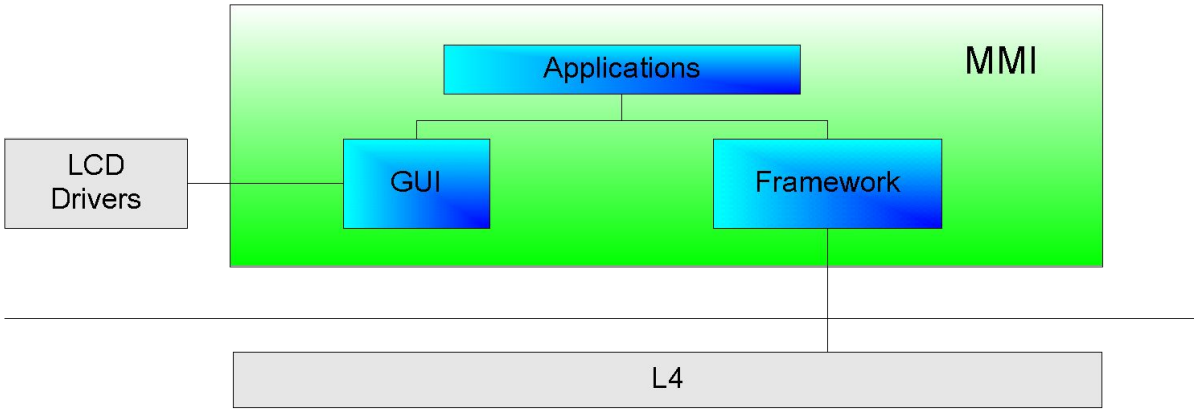
1. FAX: Group 3 Facsimile
2. TDT: Transparent circuit switching data
3. L2R: Layer 2 relay protocol for non-transparent circuit switching data
4. RLP: Radio link protocol for non-transparent circuit switching data

- CC Circuit-switched call control 电路交换呼叫控制
- SS supplementary service 附加服务
- SMS short message service 短消息服务
- SM session management 会话管理
- MM/GMM mobility management 移动性能管理
- SNDCP sub-network dependent convergence protocol

- LLC Logical link control 逻辑连接控制
- RR Radio resource management, 包括以下子模块
  1. RRM: Handles cell selection and PLMN selection
  2. RMPC: Handles the procedures in Idle/Dedicated state including the surrounding cell scheme and measurement reporting
  3. LAPDM: Handles the procedure defined in GSM layer 2
  4. RLC: Radio link control protocol
  5. MAC: Medium access control protocol
  6. MPAL: Adaptation layer for RR and L1A
- PPP Point to Point protocol layer,客户端点对点协议

MMI介绍

- UI架构
- 应用程序
- 与ps的通信



MMI Framework

- OLS 操作系统适配置层, 对操作系统进行封状
- Task MMI任务, 与L4 task 进行通信
- File system 与存储设备通信,进行文件存取

MMI GUI介绍

- Theme 主题风格, 主要使菜单等颜色设置, 以及背景图片等
- UI component
- Category Screen 每一个界面都是一个screen
- Font 字体风格
- Editor 编辑筐
- 输入法
- 访问LCD 驱动
- MMI定制工具

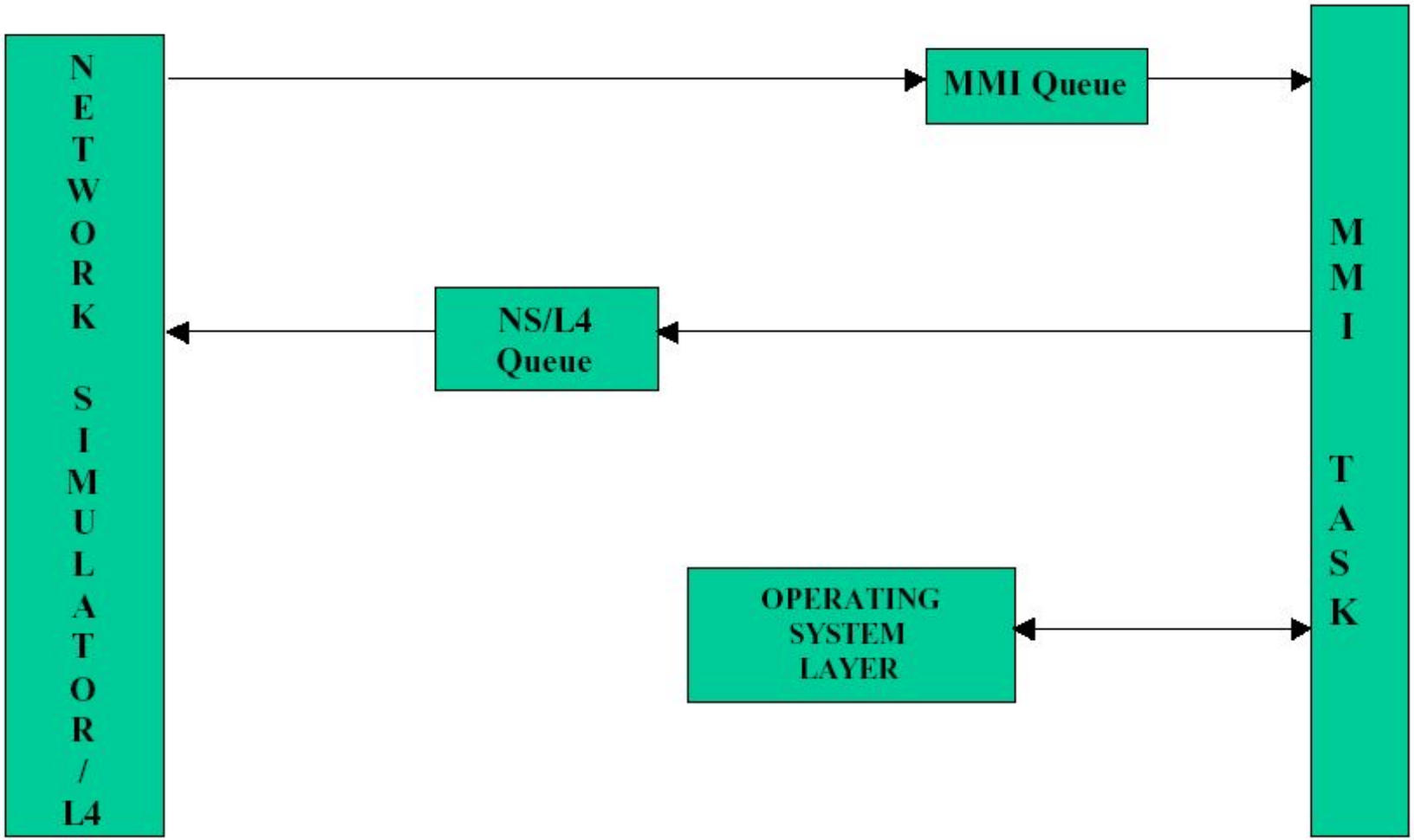
MMI应用程序

- Phonebook Message Call History
- Setting User profile Fun and game
- Organizer Service Shortcut Camera
- ……

三、MTK 方案Task架构

系统初始化

- Hardware boot and setup system stack etc
- Nucleus Plus RTOS initialization
- Hardware Initialization
- Tasks/Modules initialization/configuration
- Tasks Creation
- TCT_schedule() for scheduler to context switch


NETWORK SIMULATOR / L4
MMI Queue
NS/L4 Queue
OPERATING SYSTEM LAYER
MMI TASK

MMI task

- 主要管理应用程序，task从与之相关的队列中读取event
  - MMI Queue 协议栈/L4 将events写入到队列中，MMI task 从队列中读取event
  - L4 Queue MMI task 将MMI events 写入队列，L4 task 从队列中读取event

MMI task

- MMI 注册消息事件
- 在MMI队列上面等待消息
- 协议栈将消息放入MMI消息队列
- Framework Layer 处理events
- Framework layer 调用应用程序注册的回调函数
- 回调函数中应用程序用UI category 函数和风格进行screen 显示

MTK Customer创建task

```
typedef struct {
   kal_char            *comp_name_ptr;
   kal_char            *comp_qname_ptr;
   kal_uint32           comp_priority;
   kal_uint16           comp_stack_size;
   kal_uint8           comp_ext_qsize;
   kal_uint8           comp_int_qsize;
   kal_create_func_ptr  comp_create_func;
   kal_bool            comp_internal_ram_stack;
} comptask_info_struct;

const comptask_info_struct custom_comp_config_tbl[ MAX_CUSTOM_TASKS ] =
{
   /* INDX_CUSTOM1 */
   {"CUST1", "CUST1 Q", 210, 1024, 10, 0,
customMMI_create, KAL_FALSE},
NULL, KAL_FALSE},

}
```

Tast create

```
kal_bool
customMMI_create(comptask_handler_struct **handle)
{
  static const comptask_handler_struct customcms_handler_info =
  {
    MMI_task,  /* task entry function */
    NULL,  /* task initialization function */
    NULL,  /* task configuration function */
    NULL,  /* task reset handler */
    NULL,  /* task termination handler */
  };

  *handle = (comptask_handler_struct *)&customcms_handler_info;
  return KAL_TRUE;
}
```

注册消息事件

- SetProtocolEventHandler (mmi_msg_handle_new_msg_ind, MSG_ID_SMS_NEW_MSG_INDEX_IND);

```
for(count = 0; count < maxProtocolEvent; count++)
{
    if(protocolEventHandler[count].eventID == eventID)
    {
        isNewEvent = FALSE;
        break;
    }
}
protocolEventHandler[count].eventID = eventID;
protocolEventHandler[count].entryFuncPtr = funcPtr;
```

发送消息

```
typedef struct ilm_struct {
   module_type       src_mod_id;
   module_type       dest_mod_id;
   sap_type          sap_id;
   msg_type          msg_id;
   local_para_struct *local_para_ptr;
   peer_buff_struct  *peer_buff_ptr;
} ilm_struct;
```

```
msg_send_ext_queue(send_ilm);
```

MMI task 等候消息

```
if(!OslReadCircularQ(&Message))
{
OslReceiveMsgExtQ(qid, &Message);
OslGetMyTaskIndex( &my_index );
                OslStackSetActiveModuleID(
my_index, MOD_MMI );

}
```

处理消息

```
void ProtocolEventHandler(U16 eventID,void* MsgStruct,int mod_src, void *peerBuf)
{
        ExecuteCurrProtocolHandler((U16)eventID,MsgStruct,mod_src, peerBuf);
}

        for(count = 0; count < maxProtocolEvent; count++)
        {
                if(protocolEventHandler[count].eventID == eventID)
                {
                        currFuncPtr =
    (PsExtPeerFuncPtr)protocolEventHandler[count].entryFuncPtr;
                        break;
                }
        }

        if ( (currFuncPtr) && (!interrup_result) )
        {
                MMI_TRACE( (MMI_TRACE_G1_FRM,
    MMI_FRM_INFO_EVENT_EXECURPTO_HDLR, eventID));
                //(*currFuncPtr)(MsgStruct,mod_src);
                (*currFuncPtr)(MsgStruct, mod_src, peerBuf);
        }
```

Example: 按键处理过程

- MMI task注册某个按键的处理函数
- 硬件扫描键盘，发现按键，产生中断，且修改相应的寄存器（记录哪个键被按）
- 低级中断激活高级中断，高级中断产生event
- Keypad task 等待事件，收到event，读取寄存器，解析按键，知道哪个键被按，发送message 到 UEM task
- UEM task 发送按键事件到MMI task
- MMI Framework 找到MMI注册的该按键的处理函数，执行相应的动作

THANKS!