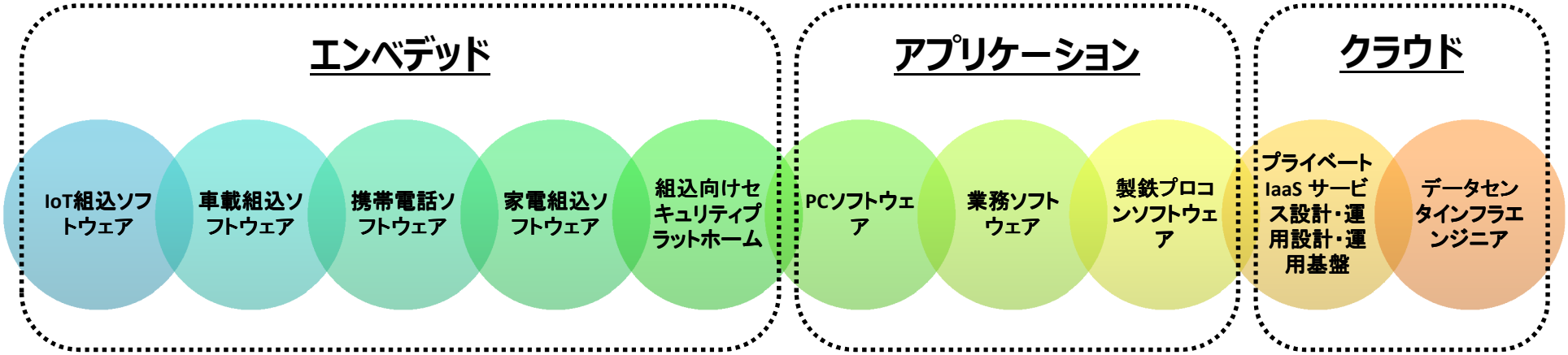
U-Boot 大研究

事業推進 第一エンベデッドグループ
Linux チーム 伊東 賢一

1人の満足から、社会の満足へ
株式会社大阪エヌデーエス

自己紹介

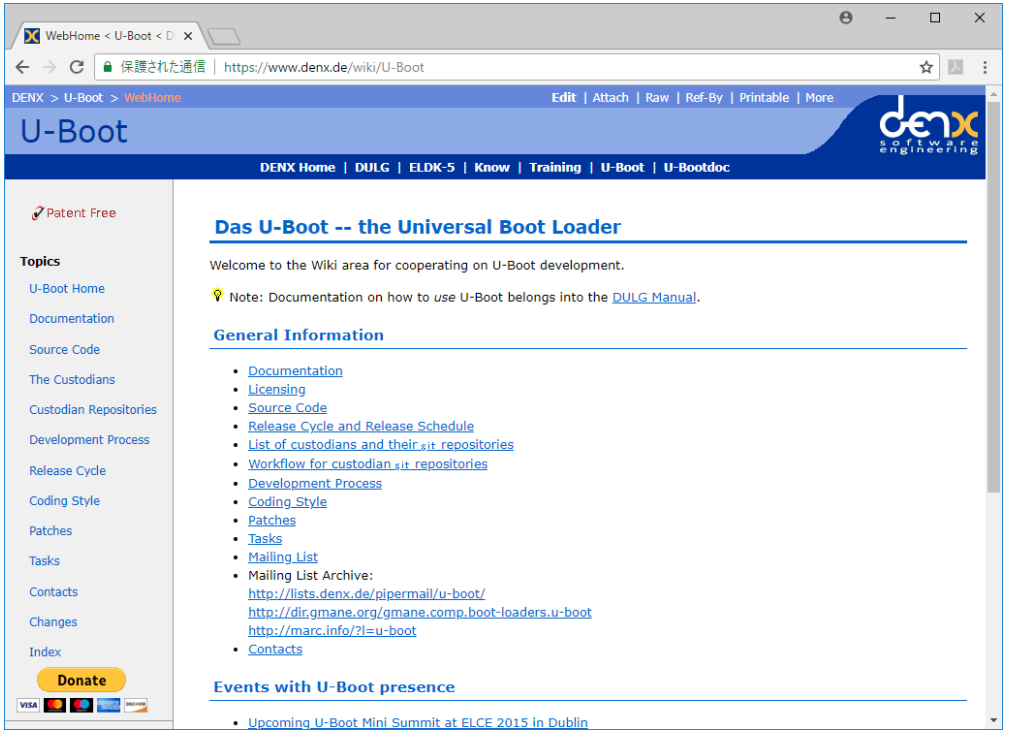
- 事業推進 第一エンベデッドG Linux T 伊東賢一
  - 組込Linuxを主業務とするグループに所属
  - 上から下まで幅広い経験を元にした柔軟なシステム提案



- その他(趣味など)
  - 熊本生まれ-新潟・東京・埼玉・栃木・千葉育ち-大阪在住43歳
  - ベーマガ世代／モノ作り（主に組込ソフト）が趣味
  - 子供5人（男男男女女（双子））

Das U-Boot とは

- 多様なプラットフォーム／プロセッサに対応したブートローダ
  - ARM、AVR32、Blackfin、x86、68k、MicroBlaze、MIPS、Altera NIOSx、PowerPC、SuperH・・・
- ライセンスはGPLv2
- 2か月に1回のリリース
- 1999 に初版（PPC用）
- 2002 にU-Bootに改名
  - Universal Boot Loader



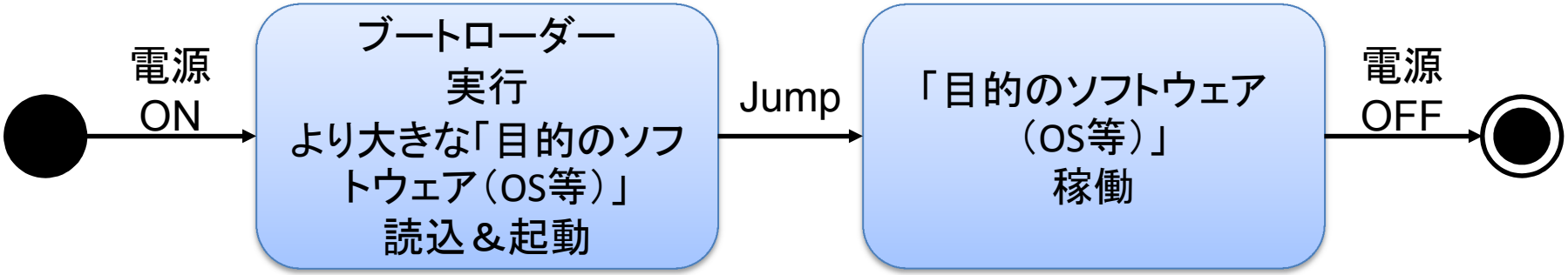
引用:https://www.denx.de/wiki/U-Boot

Boot とは

- 語源は bootstrap（ブーツのストラップ）
  - コレを自分自身で引っ張って泥沼から脱出したという故事から、自分自身で成し遂げるという意味になり、コンピュータ用語に転用。

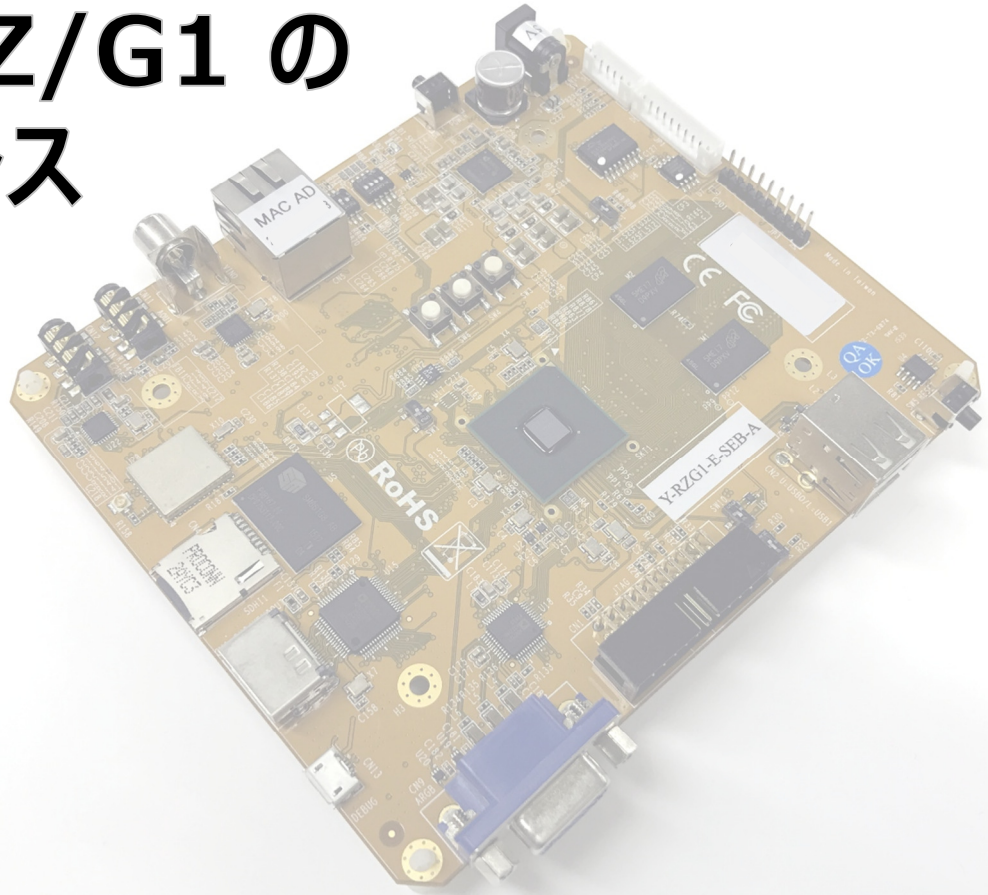
ブートローダーとは

- コンピュータシステムの電源投入時、あるいはシステムのリセット後、モニタやOSなどなんらかの基本的なシステムソフトウェアを主記憶に展開し、ユーザプログラムを実行できるようにするまでの処理の流れをいう。
- ブートローダ（英: boot loader）は、以上のプロセスで使われるローダ、すなわち、不揮発性の補助記憶にある目的のプログラムを読出し、揮発性の主記憶に書込むプログラムのことである。



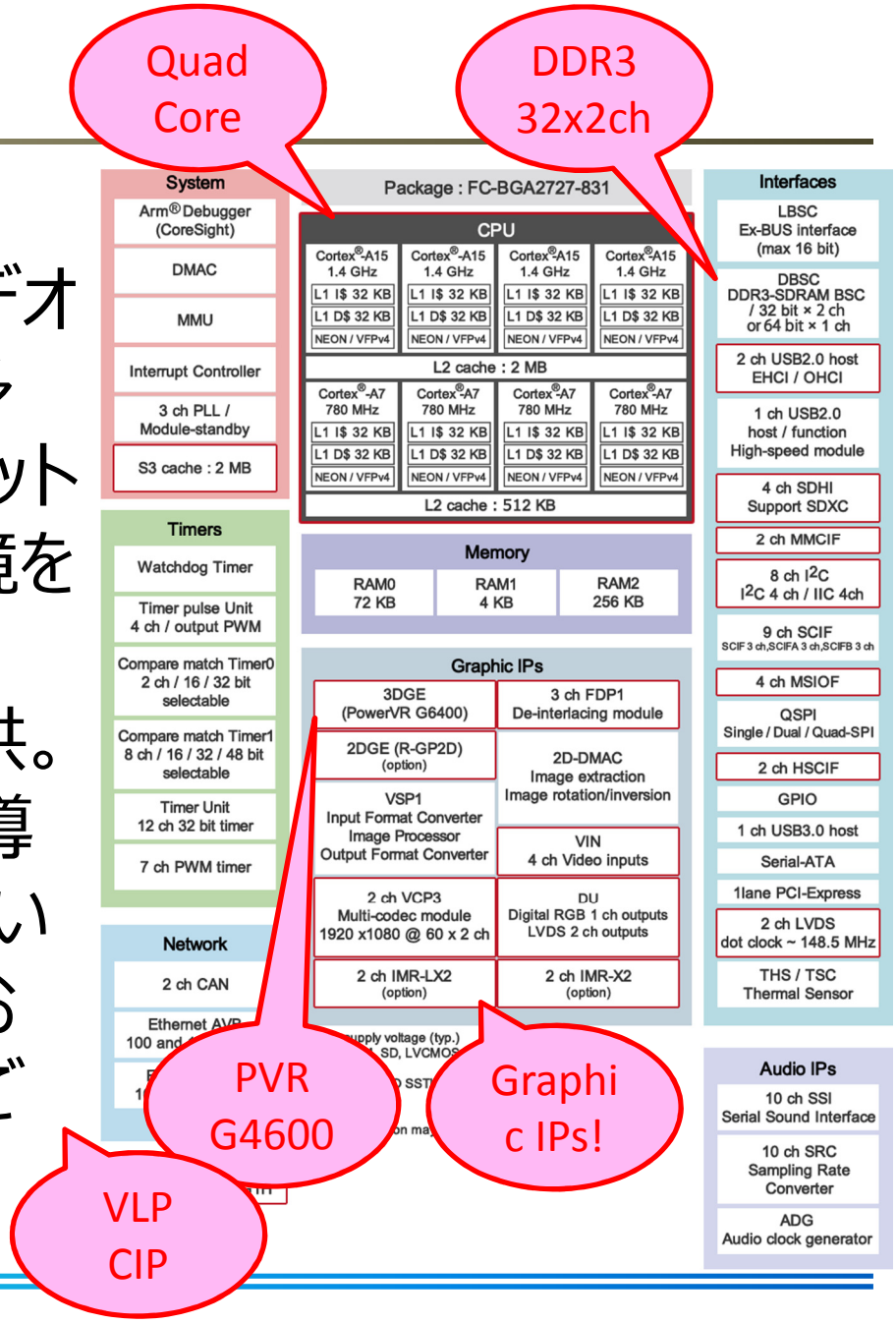
引用:https://ja.wikipedia.org/wiki/%E3%83%96%E3%83%BC%E3%83%88

Renesas RZ/G1 の ブートシーケンス



RZ/G1シリーズ

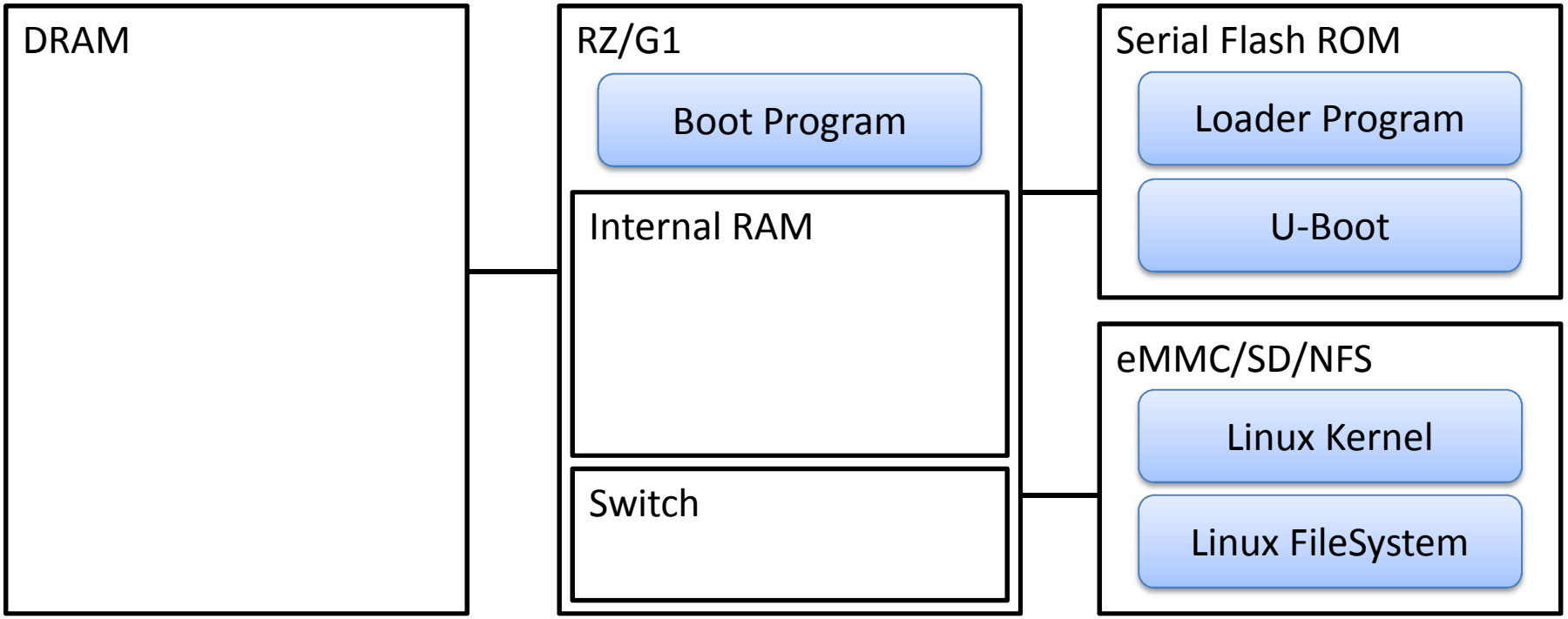
- RZ/Gシリーズはカメラ入力、3Dグラフィックス、FullHDビデオコーデック、GbEを備えたな多機能プロセッサ。システムプラットフォームとしてLinux OS環境を体系的に整備した「RZ/G Linuxプラットフォーム」を提供。このプラットフォームはLinux導入・運用管理コストを下げたいお客様、Linuxに不慣れなお客様にフレンドリーな環境をご提供するスキームです。



引用:https://www.renesas.com/ja-jp/products/microcontrollers-microprocessors/rz.html

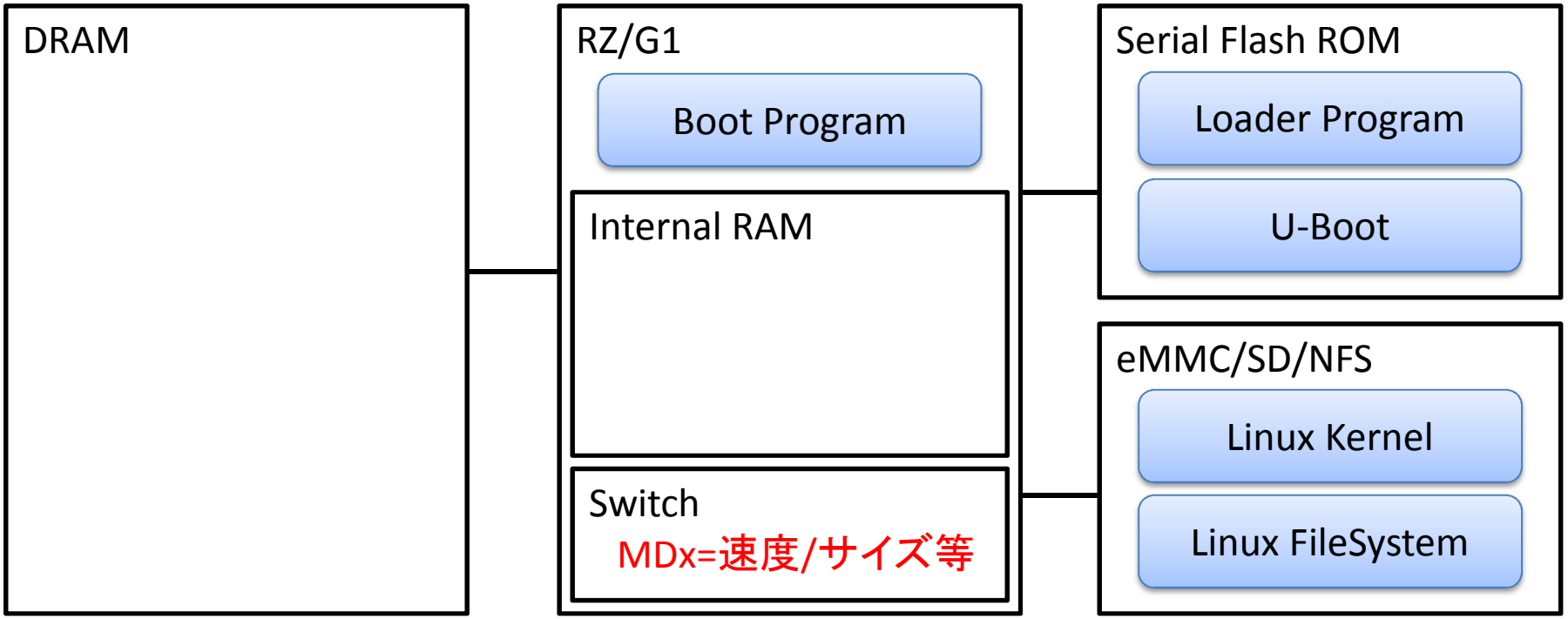
RZ/G1 の場合（１）登場するソフトウェア

- ブートローダーは３段階
  - Boot Program：SoC内のROMに配置（不可視）
  - Loader Program：Serial Flash ROM に配置
  - U-boot：Serial Flash ROM に配置



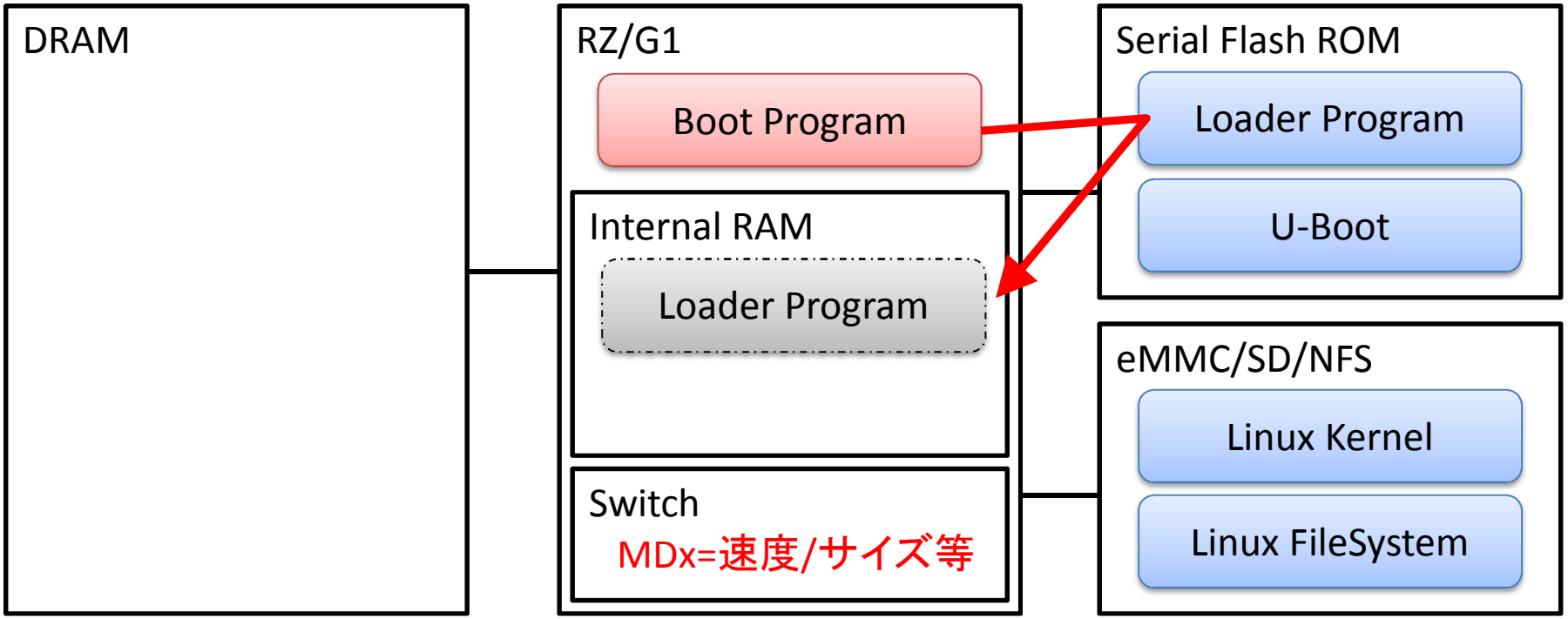
RZ/G1 の場合（２）Serial Flash 設定

- Boot Program が動作
  - Switch の状態によって Loader Program の読み出し方法をカスタマイズ可能。Serial Flash のデバイス多様性に配慮。
  - 主に読み込み速度、サイズ、使用コアの指定。



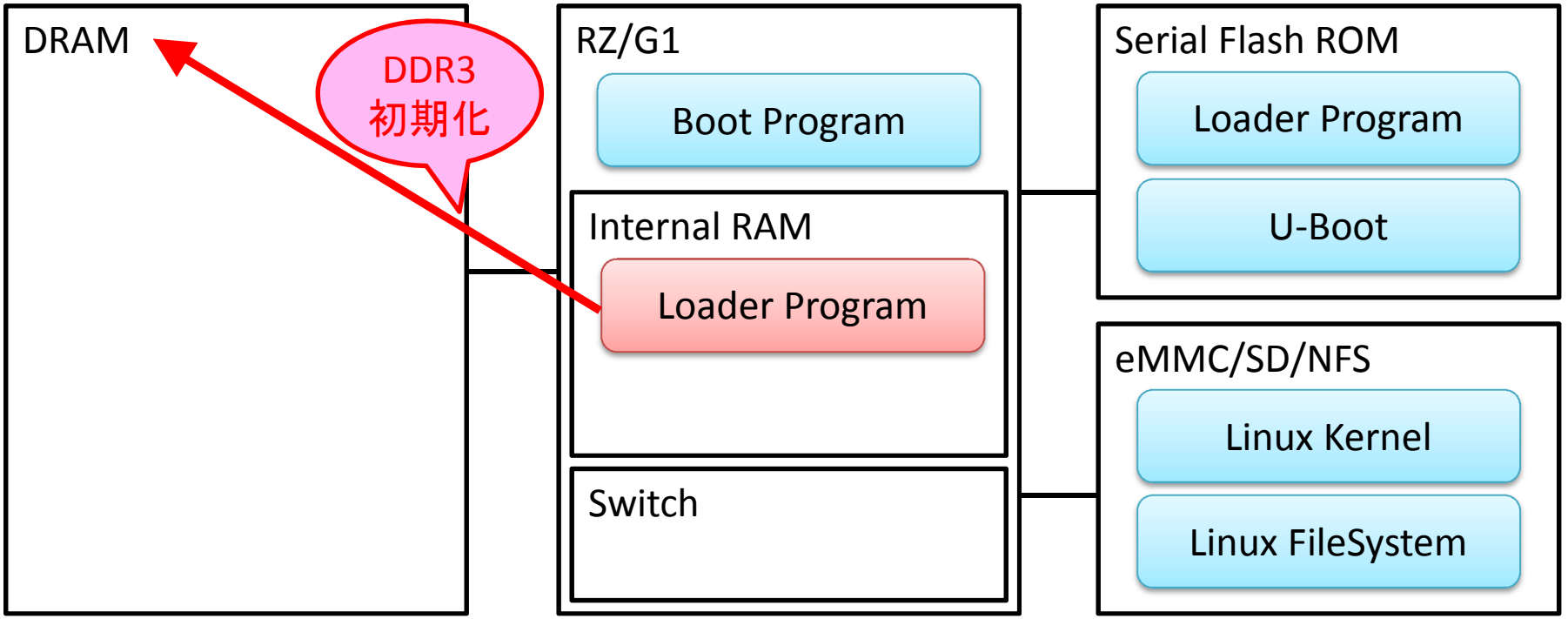
RZ/G1 の場合（３）Loader Prg 読出

- Boot Program 動作
  - Loader ProgramをInternal RAM に読込
  - 4KB or 16KBは Switch MDx にて選択
  - U-Boot/Linux kernel などはサイズが大きすぎる



RZ/G1 の場合（４）DRAM 初期化

- Loader Program 動作
  - DRAMコントローラーを初期化してDRAMを利用可能にする

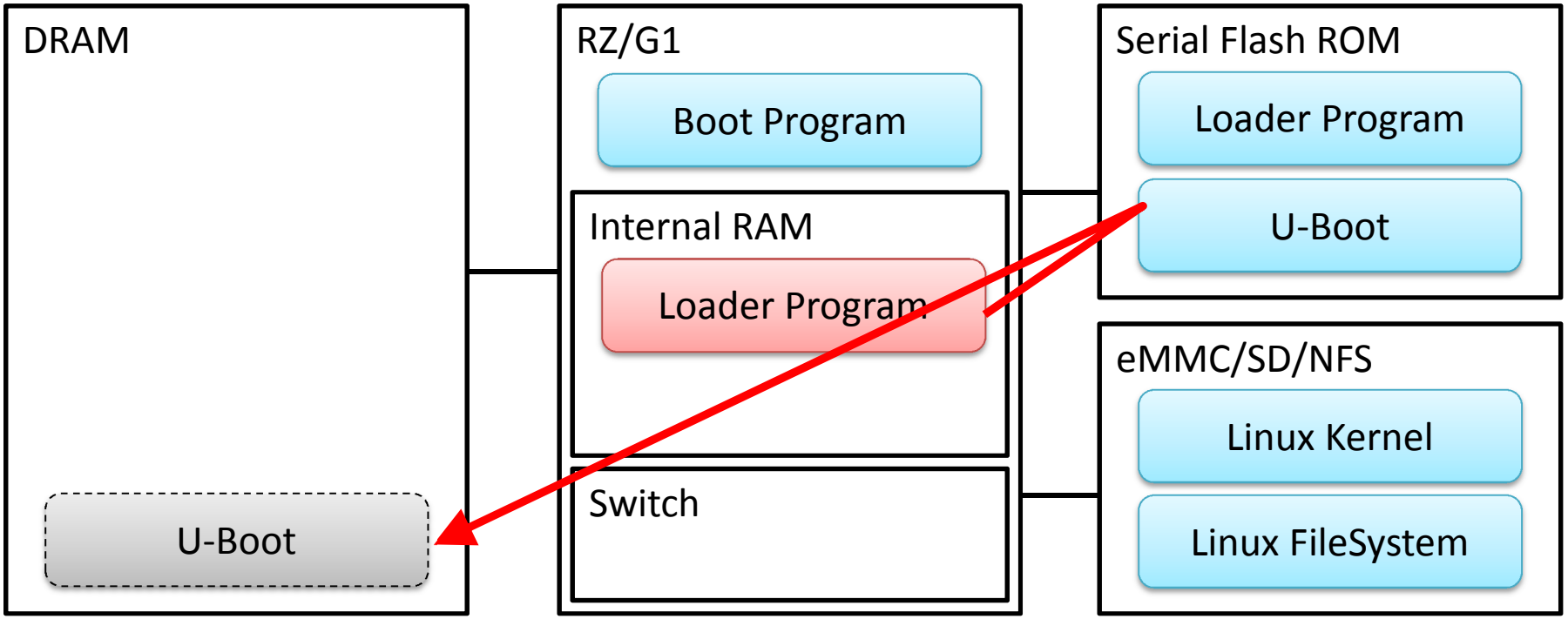


RZ/G1 の場合（５）U-Boot 読出

❖Loader Program 動作

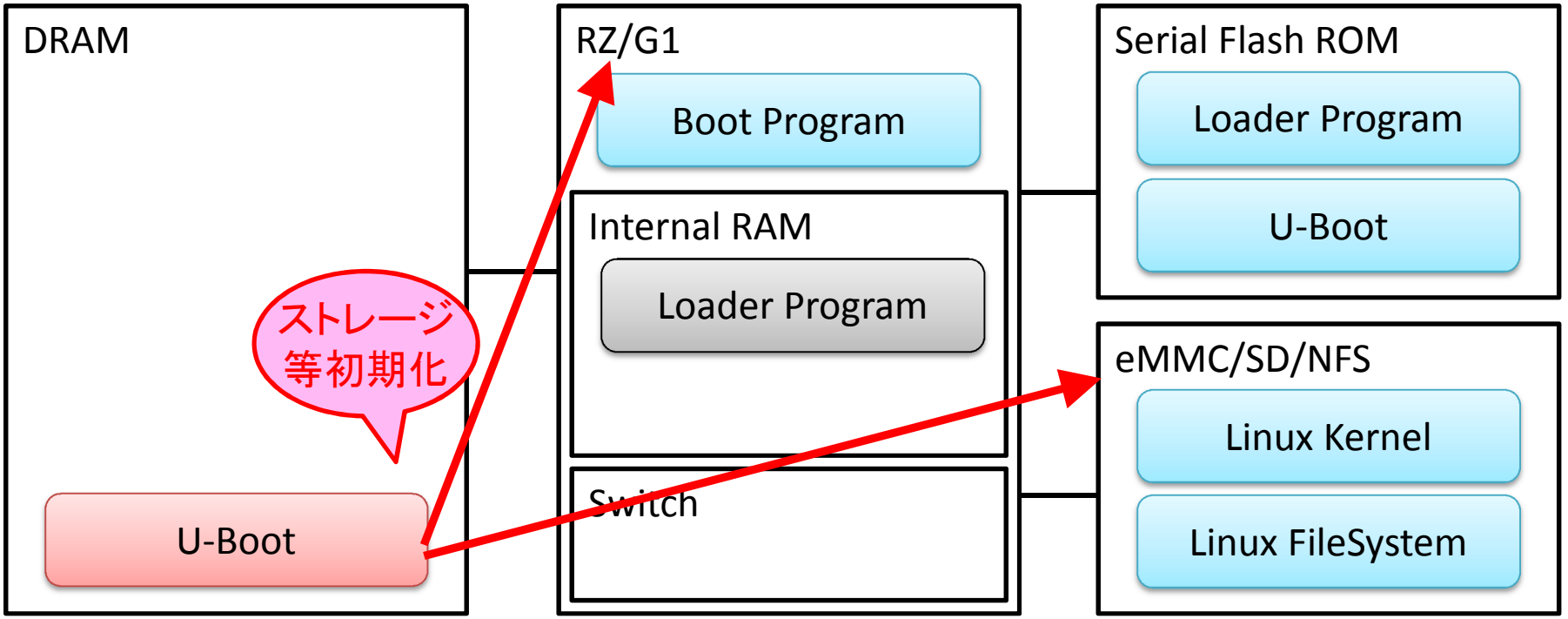
❑U-Boot を DRAM に読出＆起動

- NOR Flash（直接実行できる）の時代は直接 U-Boot を起動していたが、NAND Flash（直接実行できない）の普及によってローダーから起動されるようになった。



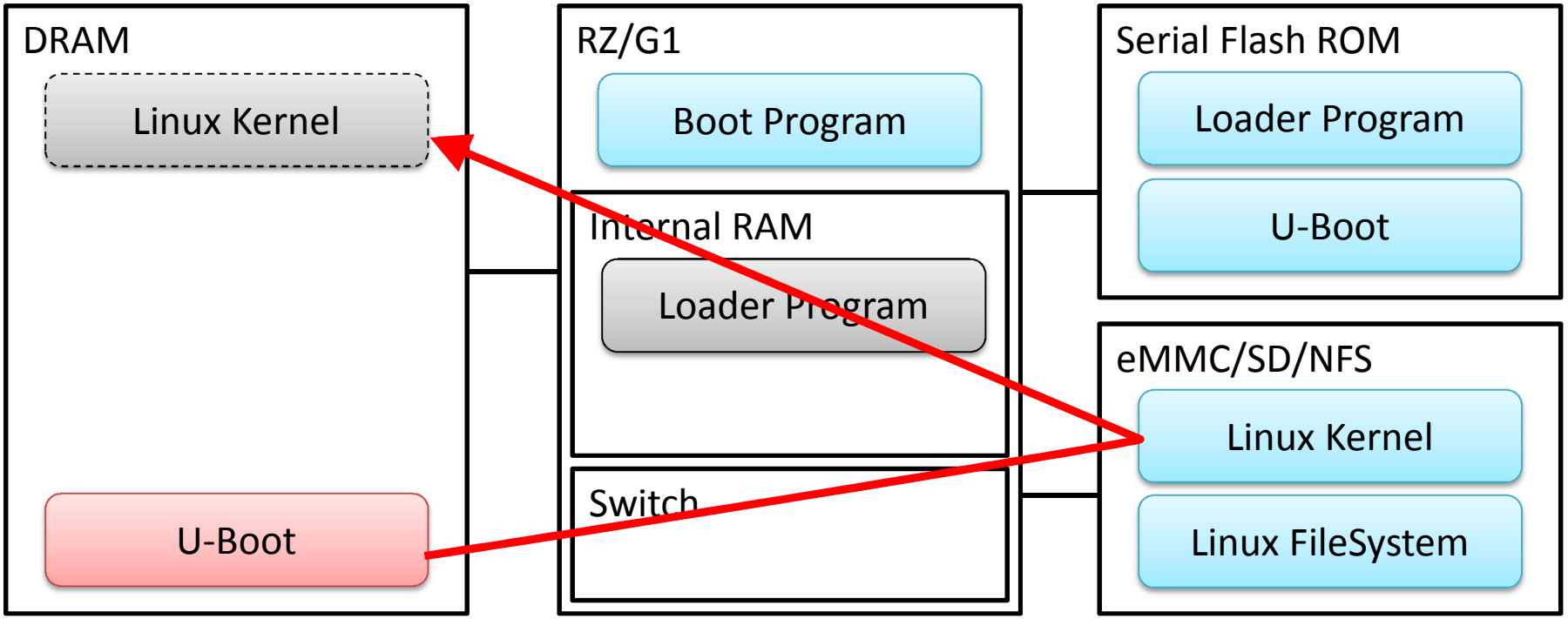
RZ/G1 の場合（６）起動デバイス等初期化

- U-Boot 動作
  - 起動対象となりうるデバイス（eMMC/SD/Ether等）の初期化
  - タイマー、シリアルポート、ピンマルチなどなど初期化



RZ/G1 の場合（７）Linuxカーネルの読出

- U-Boot 動作
  - 予め設定したパラメータに従い Linux Kernel を読出

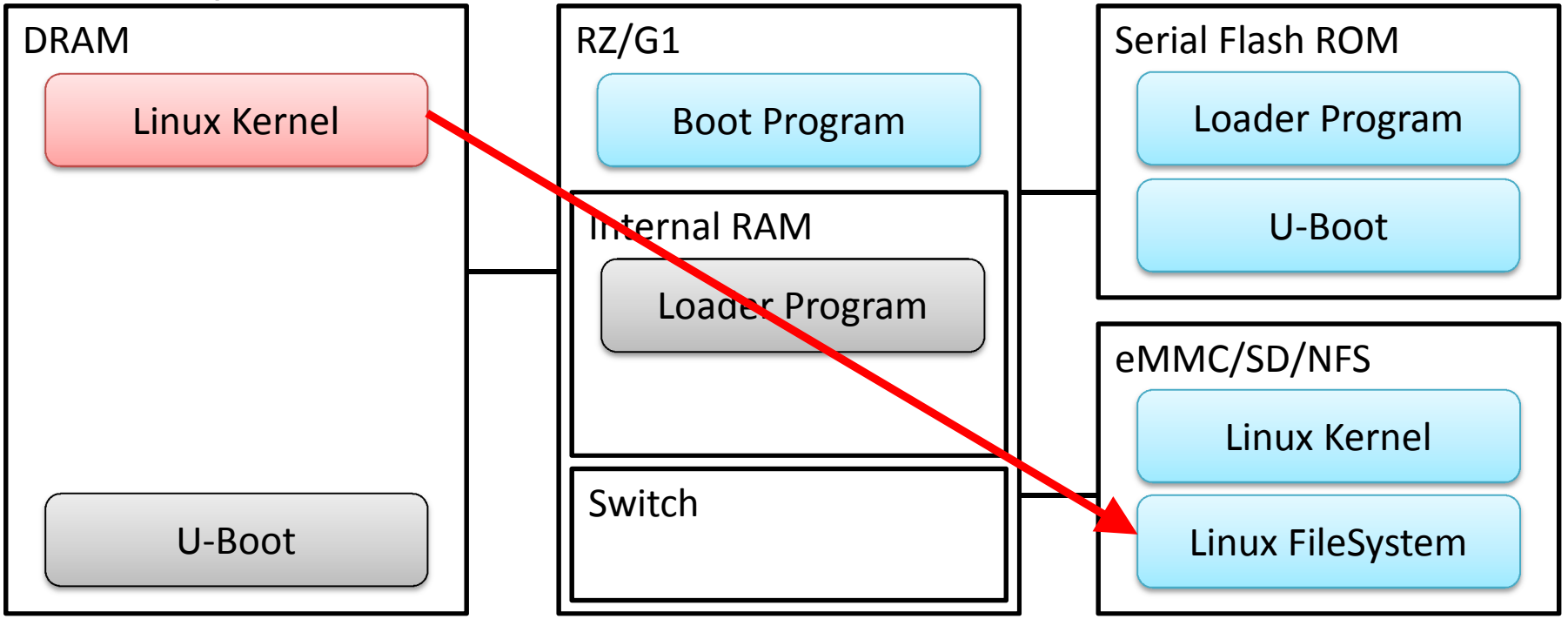


RZ/G1 の場合（８）Linuxカーネルの起動

❖U-Boot 動作

❑予め設定したパラメータに従い Linux Kernel を起動

- Root Filesystem の場所
- Ether MAC Address の設定
- 各種起動オプション

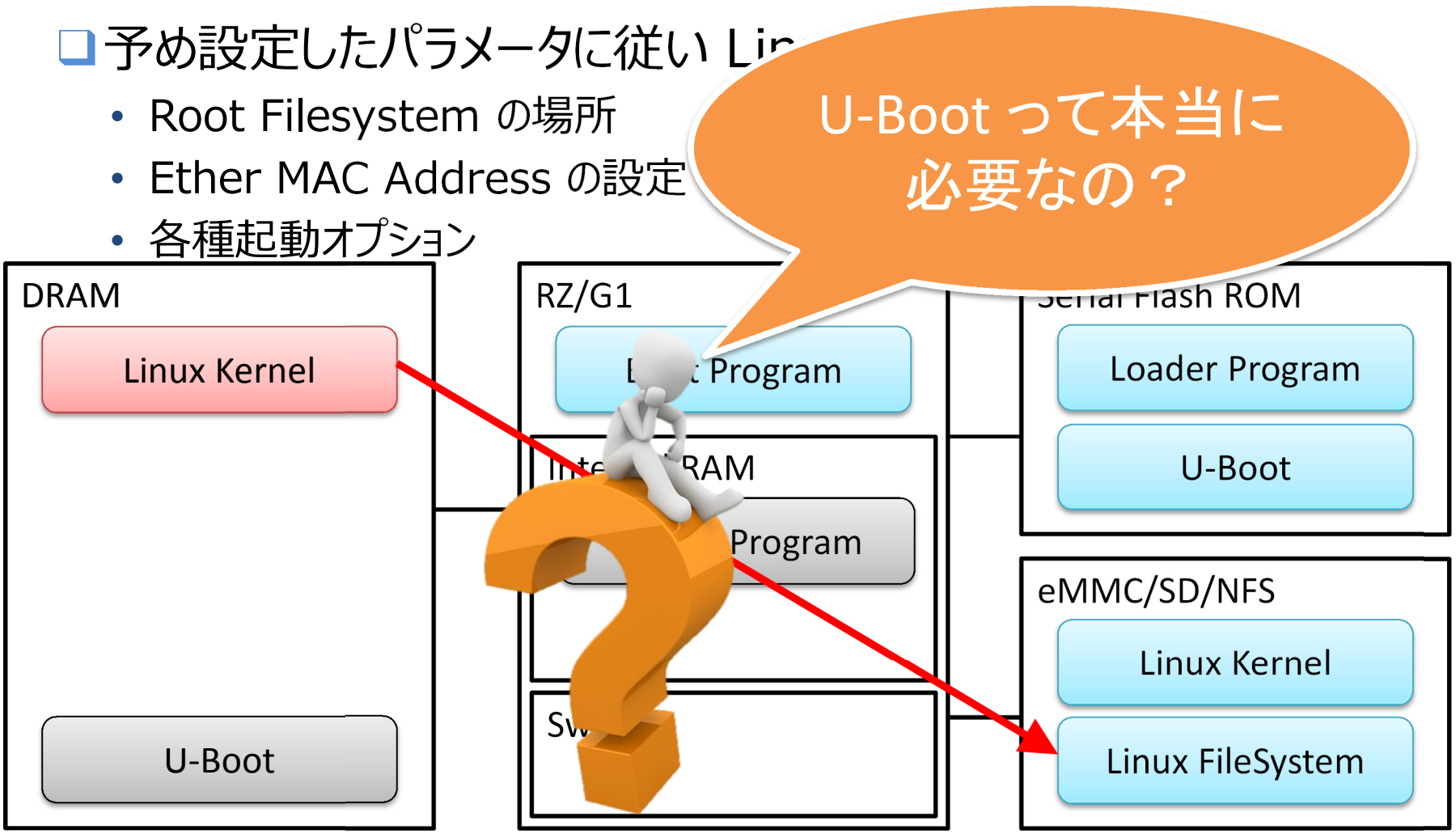


RZ/G1 の場合（８）Linuxカーネルの起動

❖U-Boot 動作

❑予め設定したパラメータに従い Lin

- Root Filesystem の場所
- Ether MAC Address の設定
- 各種起動オプション



引用:https://pixabay.com/da/spørgsmålstegn-spørgsmål-svar-1019922/

U-Boot の意義

❖開発環境

- ❑起動対象の変更差分を設定によって吸収できる。
- ❑起動オプションを設定によって変更できる。
- ❑起動対象が暴走する場合でも前段で止めることができる。
- ❑ベアメタルでなければ実行できないテストなどの実現が比較的簡単。

❖製品環境

- ❑多重起動などの機能を比較的簡単に実現できる。
- ❑イメージ書き換えなどの保守機能を比較的簡単に実現できる。

RZ/G1 の場合（１）カスタムボードで使う

- カスタムボード向けカスタマイズ
  - SoCメーカーがリファレンスボード用の U-Boot を提供していることが多い。（通常コミュニティコードの体裁を取っている）
  - ほとんどのケースでは 同SoCのリファレンスがあるため、ボード差分のカスタマイズを行うため、ボードの差異（搭載デバイスや接続方法）の検討から始める。

RZ/G1 の場合（２）カスタムボードで使う

- 最初に確認するドキュメント
  - ルネサス RZ/G シリーズ Linux BSP ポーティングガイド
  - U-Boot README 内 Building the Software:
    - クロスコンパイル環境の指定方法
    - コンフィギュレーションの方法
    - コンパイル方法
    - コンパイル結果（ターゲット）の説明

RZ/G1 の場合（３）カスタムボードで使う

- U-Boot README 内 U-Boot Porting Guide:

```
int main(int argc, char *argv[])
{
    sighandler_t no_more_time;

    signal(SIGALRM, no_more_time);
    alarm(PROJECT_DEADLINE - toSec (3 * WEEK));

    if (available_money > available_manpower) {
            Pay consultant to port U-Boot;
            return 0;
    }

    Download latest U-Boot source;

    Subscribe to u-boot mailing list;
    if (clueless)
            email("Hi, I am new to U-Boot, how do I get started?");
```

RZ/G1 の場合（４）カスタムボードで使う

```
while (learning) {
        Read the README file in the top level directory;
        Read http://www.denx.de/twiki/bin/view/DULG/Manual;
        Read applicable doc/*.README;
        Read the source, Luke;
        /* find . -name "*.[chS]" | xargs grep -i <keyword> */
}

if (available_money > toLocalCurrency ($2500)) Buy a BDI3000;
else Add a lot of aggravation and time;

if (a similar board exists) {       /* hopefully... */
        cp -a board/<similar> board/<myboard>
        cp include/configs/<similar>.h include/configs/<myboard>.h
} else {
        Create your own board support subdirectory;
        Create your own board include/configs/<myboard>.h file;
}
Edit new board/<myboard> files
Edit new include/configs/<myboard>.h
```

RZ/G1 の場合（５）カスタムボードで使う

```
while (!accepted) {
        while (!running) {
                do {
                        Add / modify source code;
                } until (compiles);

                Debug;
                if (clueless)
                        email("Hi, I am having problems...");
        }

        Send patch file to the U-Boot email list;
        if (reasonable critiques)
                Incorporate improvements from email list code review;
        else
                Defend code as written;
    }
    return 0;
}
```

RZ/G1 の場合（６）カスタムボードで使う

```
int main(int argc, char *argv[])
{
    sighandler_t no_more_time;

    signal(SIGALRM, no_more_time);
    alarm(PROJECT_DEADLINE - toSec (3 * WEEK));

[…]
}

void no_more_time (int sig)
{
     hire_a_guru();
}
```

RZ/G1 の場合（７）カスタムボードで使う

- ほとんどのケースではリファレンスボードから開発

  cp -a board/<similar> board/<myboard>
  cp include/configs/<similar>.h include/configs/<myboard>.h
  Edit new board/<myboard> files
  Edit new include/configs/<myboard>.h

- シングルコア・シングルスレッドのベアメタルプログラミング
- ICEは必須

RZ/G1 の場合（８）カスタムボードで使う

❖ファイル数と有効ファイル

```
me@mypc:~/silk/u-boot/u-boot-sh$ cloc .
     7950 text files.
     7138 unique files.
     1232 files ignored.
http://cloc.sourceforge.net v 1.60  T=34.51 s (181.7 files/s, 51979.4 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                             2852         132033         197359         714447
C/C++ Header                  2719          72710         164126         412022
Assembly                       275           8957          16793          41060
make                           371           4167           9449           7653
[…]
-------------------------------------------------------------------------------
SUM:                          6273         219671         389774        1184601
-------------------------------------------------------------------------------
me@mypc:~/silk/u-boot/u-boot-sh$ find -name *.o | wc -l
223
me@mypc:~/silk/u-boot/u-boot-sh$
```

RZ/G1 の場合（９）カスタムボードで使う

❖ディレクトリ構成

/arch	Architecture specific files
[...]	
/api	Machine/arch independent API for external apps
/board	Board dependent files
[...]	
/common	Misc architecture independent functions
/disk	Code for disk drive partition handling
/doc	Documentation (don't expect too much)
/drivers	Commonly used device drivers
/dts	Contains Makefile for / internal U-Boot fdt.
/examples	Example code for standalone applications, etc.
/fs	Filesystem code (cramfs, ext2, jffs2, etc.)
/include	Header Files
[...]	
/lib	Files generic to all architectures(hash, archiver, etc.)
/net	Networking code
/post	Power On Self Test
/spl	Secondary Program Loader framework
/tools	Tools to build S-Record or U-Boot images, etc.

RZ/G1 の場合（１０）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/include/configs/skrzg1e.h
#define CONFIG_SYS_TEXT_BASE　0xE6304000

$(u-boot)/arch/arm/cpu/armv7/start.S
.globl _start
_start: b　reset
　ldr pc, _undefined_instruction
　ldr pc, _software_interrupt
[…]
/*
 * the actual reset code
 */
reset:
[…]
　bl　_main

RZ/G1 の場合（１１）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/arch/arm/lib/crt0.S

```
/*
 * entry point of crt0 sequence
 */
.global _main
_main:

[...]
    bl  board_init_f
[...]
    ldr pc, =board_init_r
```

RZ/G1 の場合（１２）カスタムボードで使う

❖ボード初期化処理までの道のり

```
$(u-boot)/arch/arm/lib/board.c
init_fnc_t *init_sequence[] = {
    arch_cpu_init,      /* basic arch cpu dependent setup */
[...]
    board_early_init_f,
[...]
};

void board_init_f(ulong bootflag)
{
[...]
    for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
        if ((*init_fnc_ptr)() != 0) {
            hang ();
        }
    }
[...]
};
```

RZ/G1 の場合（１３）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/board/renesas/skrzg1e/skrzg1e.c

```
int board_early_init_f(void)
{
    u32 val;
    /* TMU0 */
    val = readl(MSTPSR1);
    val &= ~TMU0_MSTP125;
    writel(val, SMSTPCR1);

    /* IIC1 */
[...]
    /* SCIF2 */
    /* ETHER */
    /* MMC/SD */
    writel(SD1_97500KHZ, SD1CKCR);
    return 0;
}
```

RZ/G1 の場合（１４）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/arch/arm/lib/crt0.S

```
/*
 * entry point of crt0 sequence
 */
.global _main
_main:

[...]
    bl  board_init_f
[...]
    ldr pc, =board_init_r
```

RZ/G1 の場合（１５）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/board/renesas/skrzg1e/skrzg1e.c

```
/************************************************************************
* This is the next part if the initialization sequence: we are now
 * running from RAM and have a "normal" C environment, i. e. global
 * data can be written, BSS has been cleared, the stack size in not
 * that critical any more, etc.
************************************************************************
*/
void board_init_r(gd_t *id, ulong dest_addr)
{
[...]
    board_init();   /* Setup chipselects */
[...]
    board_late_init();

[...]
    for (;;) {
        main_loop();
    }
}
```

RZ/G1 の場合（１６）カスタムボードで使う

❖ボード初期化処理までの道のり

$(u-boot)/board/renesas/skrzg1e/skrzg1e.c

```
int board_init(void)
{
    u32 val;
    /* adress of boot parameters */
    gd->bd->bi_boot_params = SKRZG1E_SDRAM_BASE + 0x100;
    /* Init PFC controller */
    r8a7745_pinmux_init();
    /* ETHER Enable */
    gpio_request(GPIO_FN_ETH_CRS_DV, NULL);
    gpio_request(GPIO_FN_ETH_RX_ER, NULL);
    gpio_request(GPIO_FN_ETH_RXD0, NULL);
[...]
    gpio_request(GPIO_FN_IRQ8, NULL);

    val = readl(PUPR3);
    val &= ~(PUPR3_ETH & ~PUPR3_ETH_MAGIC);
    writel(val, PUPR3);
```

RZ/G1 の場合（１７）カスタムボードで使う

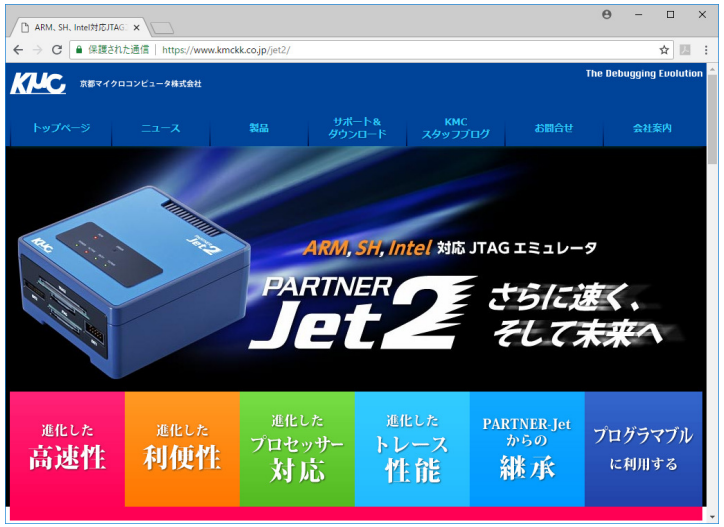
❖ボード初期化処理までの道のり

```
[...]
    sh_timer_init();
    gpio_request(GPIO_GP_1_24, NULL);   /* PHY_RST */
    val = readl(PUPR1);
    val &= ~PUPR1_DREQ0_N;
    writel(val, PUPR1);
    gpio_direction_output(GPIO_GP_1_24, 0);
    mdelay(20);
    gpio_set_value(GPIO_GP_1_24, 1);
    udelay(1);
    /* mmc0 */
    gpio_request(GPIO_GP_4_31, NULL);
    gpio_direction_output(GPIO_GP_4_31, 1);
    /* sdhi1 */
    gpio_request(GPIO_GP_4_26, NULL);
[...]
    return 0;
}
```

RZ/G1 の場合（１８）カスタムボードで使う

❖Serial Flash ROM への書込

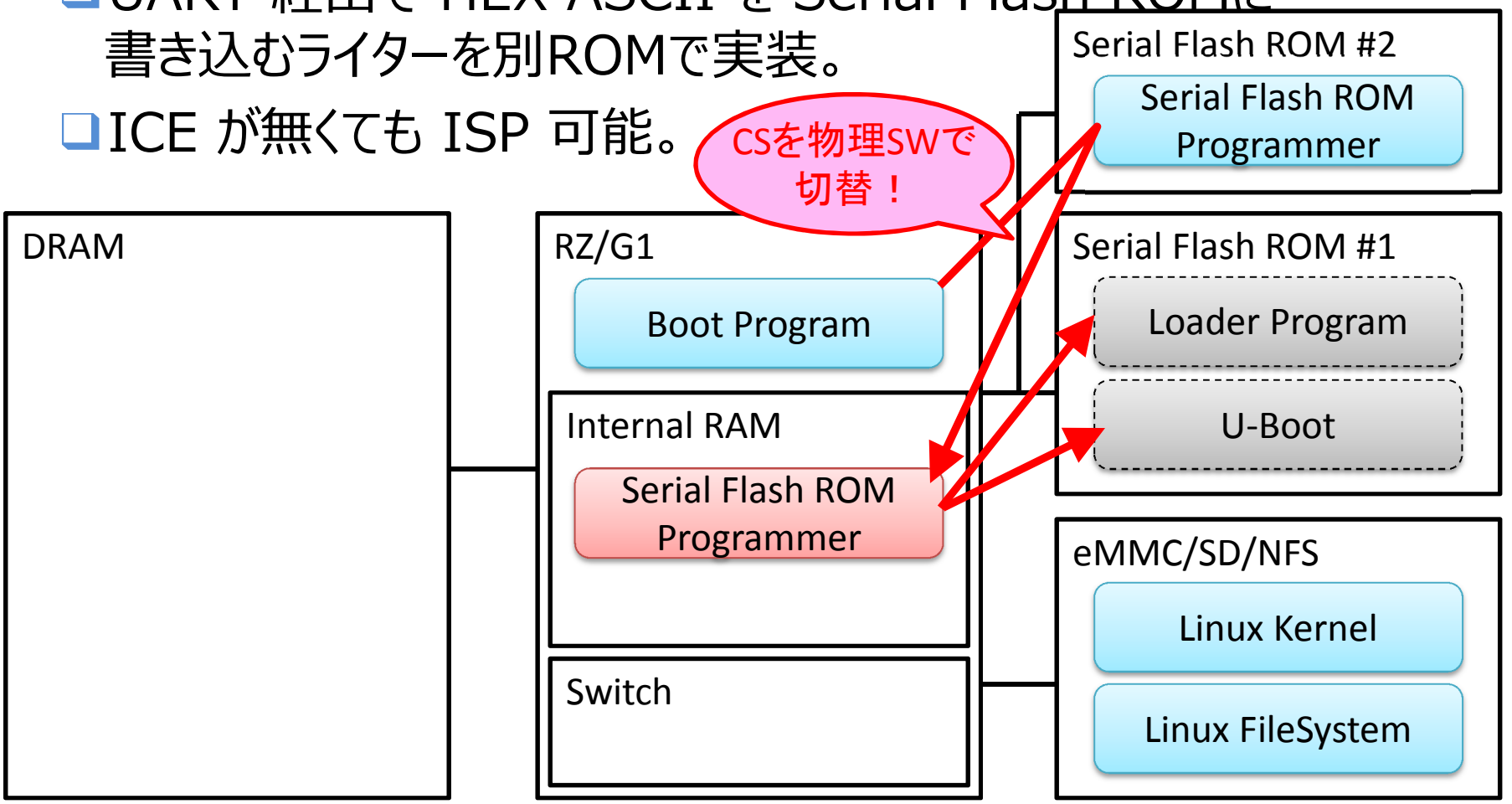
- ❑RZ/G1 と標準的な I/F を持つSerial Flash ROMの場合、ICE から ISP が可能なものもある。
- ❑ICE を使い Internal RAM から起動し U-Boot/OS から書き込む方法も考えられる。
- ❑製品は予め Serial Flash ROM に書き込んでから SMD する。



引用:https://www.kmckk.co.jp/jet2/

RZ/G1 の場合（１９）カスタムボードで使う

- ❖リファレンスボードの秘密
  - ❑UART 経由で HEX ASCII を Serial Flash ROMに書き込むライターを別ROMで実装。
  - ❑ICE が無くても ISP 可能。

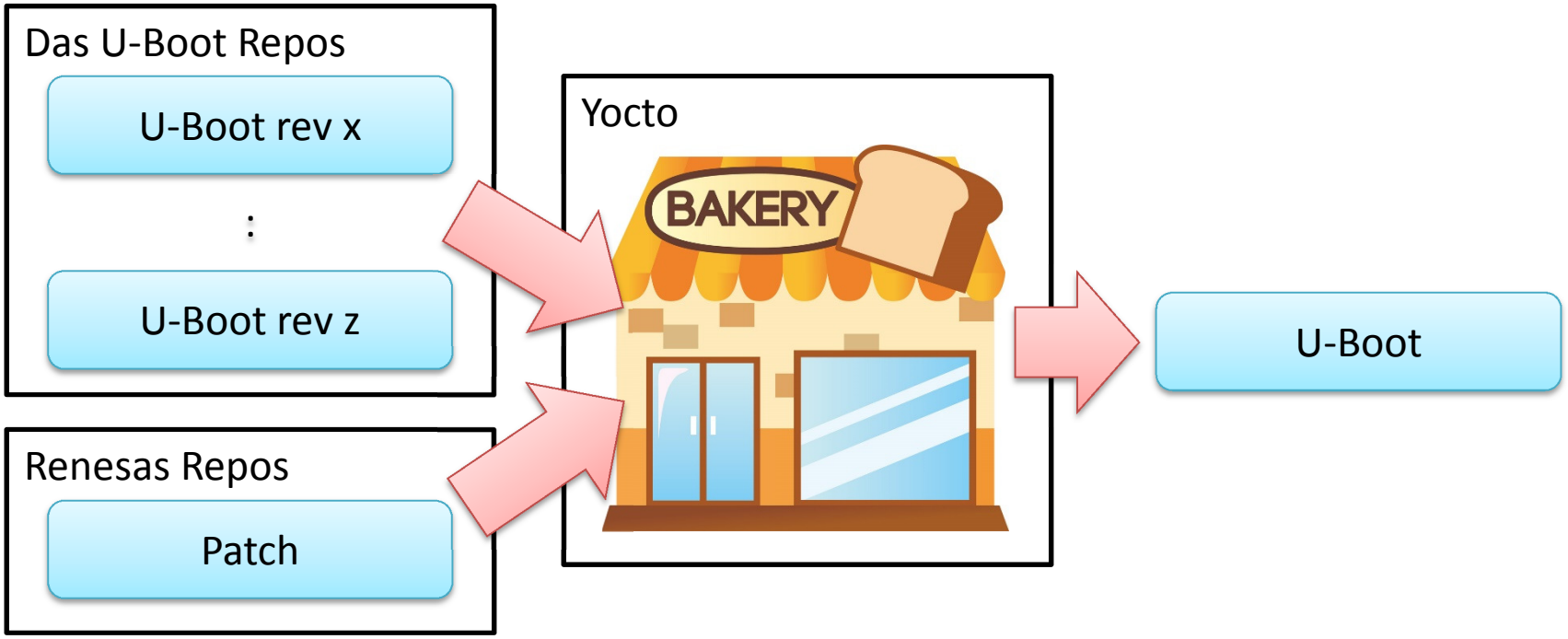


Yocto 環境における U-Boot カスタマイズ

画像:https://www.pakutaso.com/

U-Boot を make する（）

- 普通のケースでは Yocto で bitbake している（はず）
  - Yocto でどんな U-Boot が bake されているのか確認しなければならない・・・



画像:http://01.gatag.net/0009073-free-illustraition/

U-Boot を make する（１）

- カスタマイズするために同じソースを準備
  - U-Boot レシピの環境変数から使用しているソースを抽出

$ bitbake virtual/bootloader -e | grep ^SRC_URI=

SRC_URI="
git://git.denx.de/u-boot-sh.git;branch=renesas/bsp/rcar-gen2-1.9.4;protocol=git
file://0001-add-r8a7743-and-r8a7745-support.patch
file://0002-add-skrzg1e-and-skrzg1m-support.patch
file://0004-ARM-cpu-Add-ARMv7-barrier-operations-support.patch
file://0007-gpio-sh-pfc-fix-gpio-input-read.patch
file://0008-serial-sh-interface-for-rzg1.patch "

U-Boot を make する（２）

- SRC_URI= git: のリビジョンを確認
  - 同レシピファイルを確認

$ bitbake virtual/bootloader -e | grep ^FILE=

FILE="/home/bspdev/rzg1_rzg1esk_elinux/build/../meta-renesas/meta-rzg1/recipes-bsp/u-boot/u-boot_2013.01.01.bb“

- SRC_URI= git: のリビジョンを確認
  - レシピからリビジョンを確認

$cat /home/bspdev/rzg1_rzg1esk_elinux/build/../meta-renesas/meta-rzg1/recipes-bsp/u-boot/u-boot_2013.01.01.bb

[…]

SRCREV = "cb82c56b5342908120e0cad4679dca7d6058d728"

SRC_URI = "git://git.denx.de/u-boot-sh.git;branch=renesas/bsp/rcar-gen2-1.9.4;protocol=git“

[…]

U-Boot を make する（３）

- ソース一式を集める
  - Git から clone ＆ patch をコピー

$ git clone git://git.denx.de/u-boot-sh.git;branch=renesas/bsp/rcar-gen2-1.9.4;protocol=git

4Cloning into 'u-boot-sh'...

remote: Counting objects: 561027, done.

[…]

Resolving deltas: 100% (463779/463779), done.

Checking connectivity... done.

$ git checkout -b tmp cb82c56b5342908120e0cad4679dca7d6058d728

Switched to a new branch 'tmp‘

$ cp /home/bspdev/rzg1_rzg1esk_elinux/build/../meta-renesas/meta-rzg1/recipes-bsp/u-boot/u-boot/0001-add-r8a7743-and-r8a7745-support.patch .

[…more 4files…]

U-Boot を make する（４）

- Yoctoと同じパッチを適用する

$ cd u-boot-sh

$ patch -p1 < ../0001-add-r8a7743-and-r8a7745-support.patch

patching file arch/arm/cpu/armv7/rmobile/Makefile

[…]

patching file drivers/usb/host/ehci-r8a779x.c

[…more 4 files…]

- Makeを実行する

$ make -j 2 CROSS_COMPILE=arm-linux-gnueabi- CC=arm-linux-gnueabi-gcc skrzg1e_vin_config

Configuring for skrzg1e_vin - Board: skrzg1e, Options: QOS_PRI_VIN

$ make -j 2 CROSS_COMPILE=arm-linux-gnueabi- CC=arm-linux-gnueabi-gcc all

カスタマイズ終了後にソースから patch を作成し、新しい append レシピに盛り込めばOK

みんな大好き
U-Boot TIPS!

画像:https://www.pakutaso.com/

U-Boot を使う（１）コマンドの追加

- 特殊な機能をコマンドとしてお手軽に実装
  - ベアメタルでのテストプログラム
    - Linuxからではやりづらい物理メモリやデバイスへの直接制御
    - スケジューラーに邪魔されないアクセススループット計測
  - 特別なローダーやライター
    - 特殊仕様のリプログラマやアップデーター（OTAなど）
  - 起動要因によるブートシーケンス分岐
    - システム更新のロールバック
    - 縮退運用や代替運用への切り替え
- バイナリとして別プログラムという方法もあり

U-Boot を使う（２）コマンドの追加

❖新規コマンドのプログラム追加

- $(u-boot)/common/cmd_nds.c

```
#include <common.h>
#include <command.h>
static int do_nds(cmd_tbl_t *cmdtp, int flag, int argc, char * const argv[])
{
    int i;
    for( i = 0 ; i < argc ; i++ ){
        printf("argv[%d]=%s¥n",i,argv[i]);
    }
    return 0; // Return 0 on success.
}
U_BOOT_CMD(
    nds,                    // Command Name
    CONFIG_SYS_MAXARGS,  // maximum number of arguments
    1,                      // autorepeat allowed
    do_nds,                 // Implementation function
    "Help message.",        // Help  message(Long)
    "[args..]"              // Usage message(Short)
);
```

U-Boot を使う（３）コマンドの追加

- Makefileにターゲットとして追加
  - $(u-boot)/common/Makefile
    - COBJS-$(CONFIG_CMD_NDS) += cmd_nds.o
    - もしくは
    - COBJS-y += cmd_nds.o

U-Boot を使う（４）コマンドの追加

❖実行結果

```
U-Boot 2013.01.01-gcb82c56-dirty (Jun 28 2018 - 19:26:01)
[...]
=> nds 1 2 3
argv[0]=nds
argv[1]=1
argv[2]=2
argv[3]=3
=> help nds
nds - Help message.
Usage:
nds [args..]
=> help
base    - print or set address offset
bootm   - boot application image from memory
[...]
mtest   - simple RAM read/write test
mw      - memory write (fill)
nds     - Help message.
[...]
```

お問い合わせ

大阪本社：

〒540-0001 大阪府大阪市中央区城見１丁目４番７０号 住友生命ＯＢＰプラザビル　８Ｆ

TEL：06-6945-6802

FAX：06-6945-6801

URL：http://www.nds-osk.co.jp

東京オフィス：

〒154-0015 東京都世田谷区桜新町２丁目22番3号 NDSビル

TEL：03-3439-2360

FAX：03-3493-3509

新大阪開発センター：

〒532-0011 大阪府大阪市淀川区西中島7丁目1番5号 辰野新大阪ビル９Ｆ

TEL：06-6195-9237

FAX：06-6195-9238

事業推進 第一エンベデッドグループ Linuxチーム 伊東　賢一

MAIL ： ito@nds-osk.co.jp

【2018Japan IT Week関西 IoT/M2M展 出展】

2018年2月21～23日にインテックス大阪で開催された

‘2018Japan IT Week関西 IoT/M2M展’に出展いたしました。

総称 2018 Japan IT Week 関西 内 第2回 関西 IoT/M2M展

画像:https://www.pakutaso.com/