



GoDroid

Programming Guide

Ext-Uboot

v1.0



Copyright Statement

Copyright in this document is owned by GoWarrior Community. Any person is hereby authorized to view, copy, print and distribute this document subject to the following conditions:

- The document may be used for informational purposes only
- The document may be used for non-commercial purposes only
- Any copy of this document or portion thereof must include this copyright notice

This document is provided "as is" without any warranty of any kind, either express or implied, statutory or otherwise; without limiting the foregoing, the warranties of satisfactory quality, fitness for a particular purpose or non-infringement are expressly excluded and under no circumstances will GoWarrior Community be liable for direct or indirect loss or damage of any kind, including loss of profit, revenue, goodwill or anticipated savings.

This document is intended only to assist the reader in the use of the product. GoWarrior Community makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, which is used at your own risk and should not be relied upon. The information could include technical inaccuracies or typographical errors. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property right is granted by this document.

The product described in this document is subject to continuous development and improvements. GoWarrior Community also reserves the right to make changes to the specifications and product description at any time without notice.

Third-party brands and names mentioned in this publication are for identification purpose only and may be the property of their respective owners.

Android™ is a registered trademark of Google Inc. Linux® is a registered trademark of Linus Torvalds. Microsoft® and Windows® are registered trademarks of Microsoft Corporation. Supply of this product does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Google Inc., Linus Torvalds, and Microsoft Corporation to use this implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Google Inc., Linus Torvalds and Microsoft Corporation.

For the latest version of this document refer to:

www.gowarriorosh.com

Copyright © 2015 GoWarrior Community All Rights Reserved.

Table of Contents

| | |
|---|---------------|
| Preface | 1 |
| Overview | 1 |
| Audience | 1 |
| Applicable Products..... | 1 |
| Reference Documents..... | 1 |
| Conventions..... | 2 |
| How to Contact Us..... | 3 |
| 1 Introduction | 4 |
| 1.1 Overview..... | 4 |
| 1.2 Background | 4 |
| 2 U-Boot Architecture | 5 |
| 2.1 U-Boot Stage 1 | 5 |
| 2.2 U-Boot Stage 2 | 7 |
| 2.3 U-Boot Start Kernel | 10 |
| 2.4 U-Boot Global Variables..... | 13 |
| 3 Transplant on TIGER Board..... | 15 |
| 4 Recovery System Entering Rules | 16 |
| Appendix: Glossary | 17 |
| Revision History | 18 |
| Document Change | 18 |
| Software Change..... | 18 |

List of Tables

| | |
|---|----|
| Table 1. Typographical Conventions..... | 2 |
| Table 2. Symbol Conventions | 2 |
| Table 3. List of Abbreviations..... | 17 |
| Table 4. Document Change History | 18 |
| Table 5. Software Change History..... | 18 |

List of Figures

| | |
|--|----|
| Figure 1. U-Boot Memory Distribution | 6 |
| Figure 2. Env_relocate Regular Process..... | 8 |
| Figure 3. Register Device List..... | 9 |
| Figure 4. U-Boot Stage 2 | 10 |
| Figure 5. U-Boot Start Process | 13 |
| Figure 6. Recovery System Entering Process | 16 |

Preface

Overview

This manual is to provide detailed guidance of software development for developers using GoDroid U-Boot software development kit. The guide includes basic introduction to U-Boot and some areas requiring attention when transplanting on the GoWarrior TIGER Board. The main chapters are as follows:

- **Chapter 1: Introduction**

This chapter gives compact description on Bootloader.

- **Chapter 2: U-Boot Architecture**

This chapter provides details on the U-Boot design steps.

- **Chapter 3: Transplant on TIGER Board**

This chapter summarizes how to transplant on GoWarrior TIGER Board.

- **Chapter 4: Recovery System Entering Rules**

This chapter introduces rules to enter Recovery System.

Audience

This manual is primarily written to provide complete guidance for those who want to exploit GoWarrior TIGER Board, such as makers, tinkers, innovators, students, etc.

Applicable Products

This manual is applicable for the GoWarrior TIGER Board.

Reference Documents

- GoWarrior_GoDroid_Release Notes
- GoWarrior_GoDroid_Compiling Server Installation and Configuration Guide
- GoWarrior_GoDroid_Developer Guide

Conventions

Typographical Conventions

| Item | Format |
|---|---|
| codes, keyboard input commands, file names, equations, and math | Courier New, Size 10.5 |
| Variables, code variables, and code comments | <i>Courier New, Size, Italic</i> |
| Menu item, buttons, tool names | Ebrima, Size 10.5, Bold e.g. Select USB Debugging |
| Screens, windows, dialog boxes, and tabs | Ebrima, Size 10.5, Bold Enclosed in double quotation marks e.g. Open the "Debug Configuration" dialog box |

Table 1. Typographical Conventions

Symbol Conventions




| Item | Description |
|--|--|
|  Caution | Indicates a potential hazard or unsafe practice that, if not avoided, could result in data loss, device performance degradation, or other unpredictable results. |
|  Note | Indicates additional and supplemental information for the main contents. |
|  Tip | Indicates a suggestion that may help you solve a problem or save your time. |

Table 2. Symbol Conventions

How to Contact Us

Submit all comments and error reports to the author at

info@gowarriorosh.com

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

Questions regarding GoWarrior, contact our support team at the email listed below:

support@gowarriorosh.com

1 Introduction

1.1 Overview

Compared to Google's official little kernel(lk) as the bootloader, GoDroid adopts U-Boot to boot kernel and load the rootfs. This document describes details about how to customize and transplant U-Boot on TIGER Board.

1.2 Background

- Boot Loader

Boot Loader is a small program that runs before running the operating system kernel. Through this little program, the hardware device and map memory space are initialized, thus setting up a suitable hardware and software environment for the system, in preparation for the final call to the operating system kernel.

- U-Boot

U-Boot is short for Universal Bootloader, designed by Wolfgang Denk in German DENX Software Engineering Center in 1999. There are features such as:

- ✓ Supports for various hardware architectures including ARM, x86, PPC, MIPS, m68k, NIOS, Blackfin.
- ✓ Supports for several operation systems such as Linux, VxWorks, NETBSD, QNX, RTEMS, ARTOS, LynxOS.
- ✓ Supports up to 216 kinds of development boards
- ✓ Open source code, following the GPL term
- ✓ Easy to transplant and debug

2 U-Boot Architecture

This chapter focuses on the internal details about U-Boot design.

2.1 U-Boot Stage 1

Most bootloaders are the two-stage bootloader, that is stage 1 and stage 2, U-Boot is no exception. Stage 1 generally contains the following processes:

1. Initialize Cache

Cache, between CPU and memory, is a kind of storage with small memory capacity but high access rate. Cache can save part of data just used or frequently used by CPU, thus CPU can access those data immediately the second time it calls and reduce the waiting time. So using cache will greatly improve the CPU read and write speed.

What Cache initialization does is clearing all the Cache flag bits to zero and the possible Cache dirty data. This action avoids writing the dirty data back to RAM and protect the current runtime environment from damage.

2. Initialize Stack

3. Run initialization functions

These functions are in the `init_sequence` array. U-Boot uses this array to store function points of initialization functions that most development boards will execute. The `init_sequence` array is roughly as follows apart from many compile options:

```
init_fnc_t *init_sequence[] = {
    timer_init,      /* time initialization --cpu/arm/cpu/armv7/omap-common/timer.c */
    env_init,        /* environment variable initialization --common/env_flash.c or common/env_nand.c */
    init_baudrate,   /* baud rate initialization --lib_arm/board.c */
    serial_init,     /* serial initialization --drivers/serial/serial.c */
    console_init_f,  /* communication console initialization --common/console.c */
    display_banner,  /* print the place of U-boot in memory --arch/arm/lib/board.c */
    dram_init,       /* configure the available RAM size --cpu/arm/armv7/omap-common/hwinit-common.c */
    NULL,
};
```

4. Reallocate memory space

The standard method to allocate U-Boot memory space is allocating memory from top down as shown below:

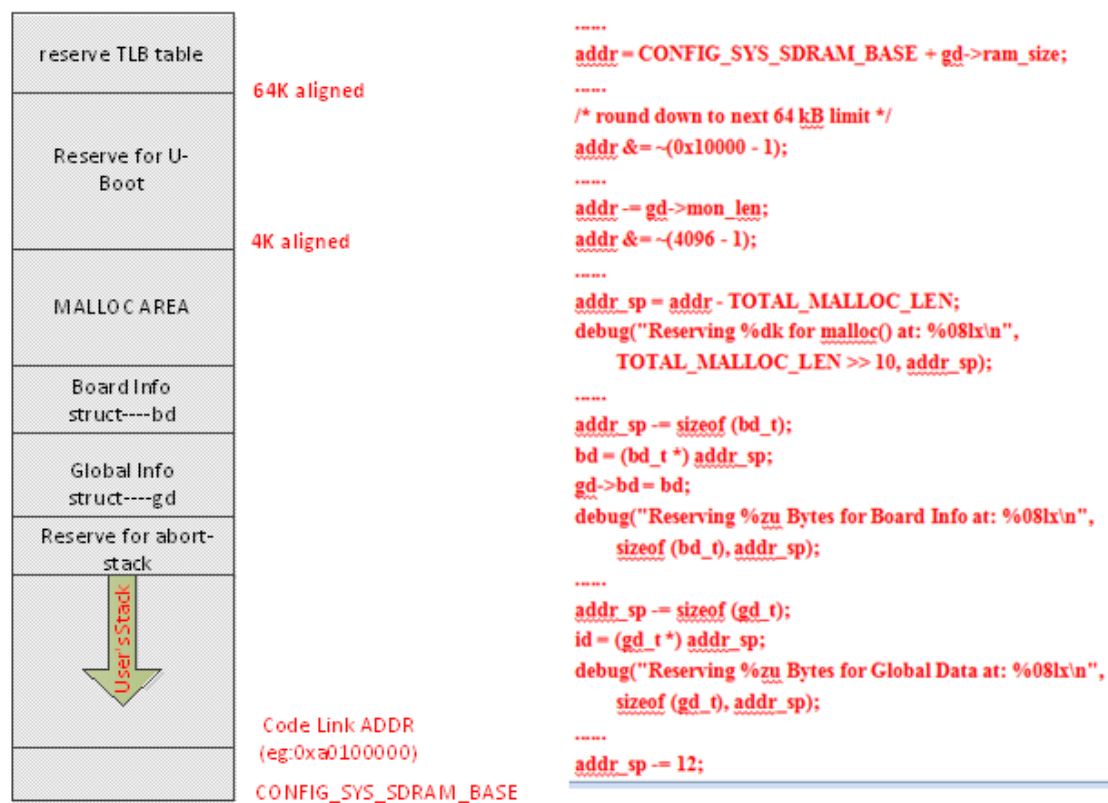


Figure 1. U-Boot Memory Distribution

5. Run the `relocate_code` function to copy U-Boot from the link address to run address.
6. Clear BSS section.
7. Jump to **U-Boot Stage 2**, please see [2.2 U-Boot Stage 2](#).

2.2 U-Boot Stage 2

The main steps of this stage are as follows:

1. Initialize NOR Flash.
2. Initialize NAND Flash.
3. Run `env_relocate` function to initialize the environment variable.

We need to know what environment variable means first. Actually, it represents the parameters of the system operation environment. The table below lists some of the environment variables:

| Environment Variable | Description |
|----------------------|--|
| Bootdelay | Waiting seconds before automatically system starts |
| Baudrate | Baud rate of serial console |
| Etmask | Mask of Ethernet interface |
| Ethaddr | Address of Ethernet card |
| Bootfile | Default download file |
| Bootargs | Start arguments to be passed to kernel |
| Bootcmd | System start commands |
| Serverip | Server IP address |
| Ipaddr | Local IP address |
| Stdin | Standard input device |
| Stdout | Standard output device |

| Environment Variable | Description |
|----------------------|----------------|
| Stderr | Standard error |

Table 3. List of Environment Variables

The figure below illustrates the regular process of `env_relocate`:

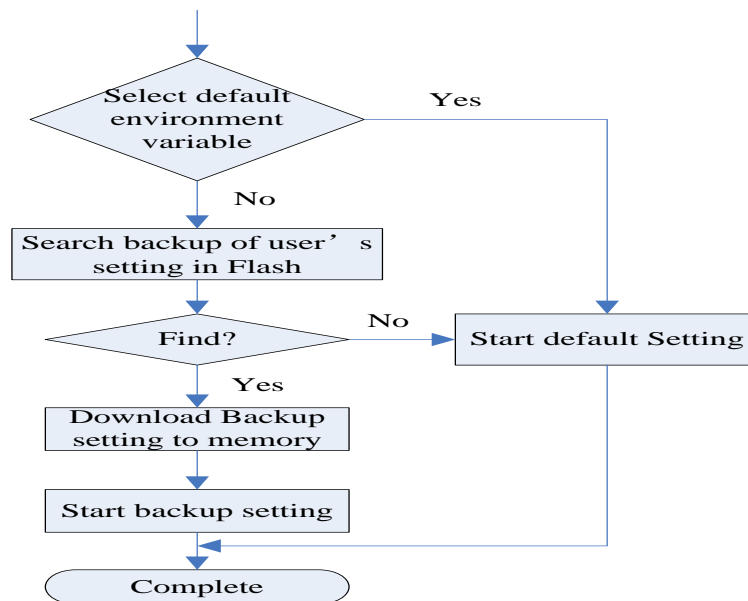
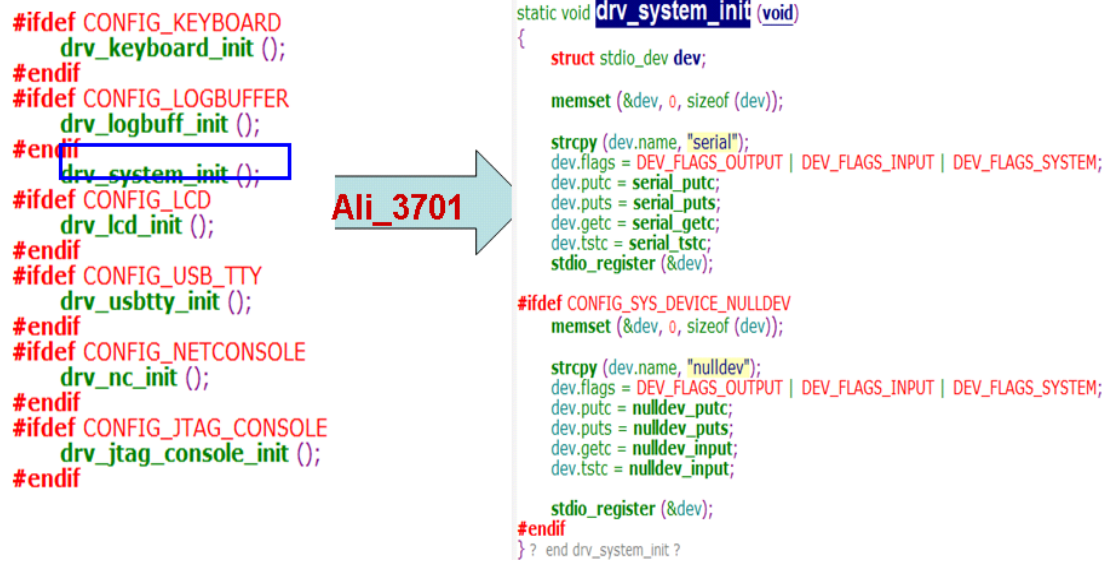


Figure 2. Env_relocate Regular Process

1. Retrieve IP/MAC address.
2. Package the serials, keyboard, display and other basic input/output device as the `stdio_dev` structure by `stdio_init`, and register these devices to device list to make them available to console. Source code as follows describes the process:



```

#ifdef CONFIG_KEYBOARD
    drv_keyboard_init ();
#endif
#ifdef CONFIG_LOGBUFFER
    drv_logbuff_init ();
#endif
drv_system_init ();
#ifdef CONFIG_LCD
    drv_lcd_init ();
#endif
#ifdef CONFIG_USB_TTY
    drv_usbttty_init ();
#endif
#ifdef CONFIG_NETCONSOLE
    drv_nc_init ();
#endif
#ifdef CONFIG_JTAG_CONSOLE
    drv_jtag_console_init ();
#endif
    
```

Ali_3701

```

static void drv_system_init(void)
{
    struct stdio_dev dev;

    memset (&dev, 0, sizeof (dev));

    strcpy (dev.name, "serial");
    dev.flags = DEV_FLAGS_OUTPUT | DEV_FLAGS_INPUT | DEV_FLAGS_SYSTEM;
    dev.putc = serial_putc;
    dev.puts = serial_puts;
    dev.getc = serial_getc;
    dev.tstc = serial_tstc;
    stdio_register (&dev);

#ifdef CONFIG_SYS_DEVICE_NULLDEV
    memset (&dev, 0, sizeof (dev));

    strcpy (dev.name, "nulldrv");
    dev.flags = DEV_FLAGS_OUTPUT | DEV_FLAGS_INPUT | DEV_FLAGS_SYSTEM;
    dev.putc = nulldrv_putc;
    dev.puts = nulldrv_puts;
    dev.getc = nulldrv_getc;
    dev.tstc = nulldrv_tstc;

    stdio_register (&dev);
#endif
}
    
```

Figure 3. Register Device List

3. Load boot media/logo and kernel/see/ae code and so on.
4. Enter `main_loop` and wait for users' command input, if a user does not input any command within the specified time, U-Boot will execute the default command to start Linux Kernel.

In conclusion, U-Boot stage 2 process as shown below:

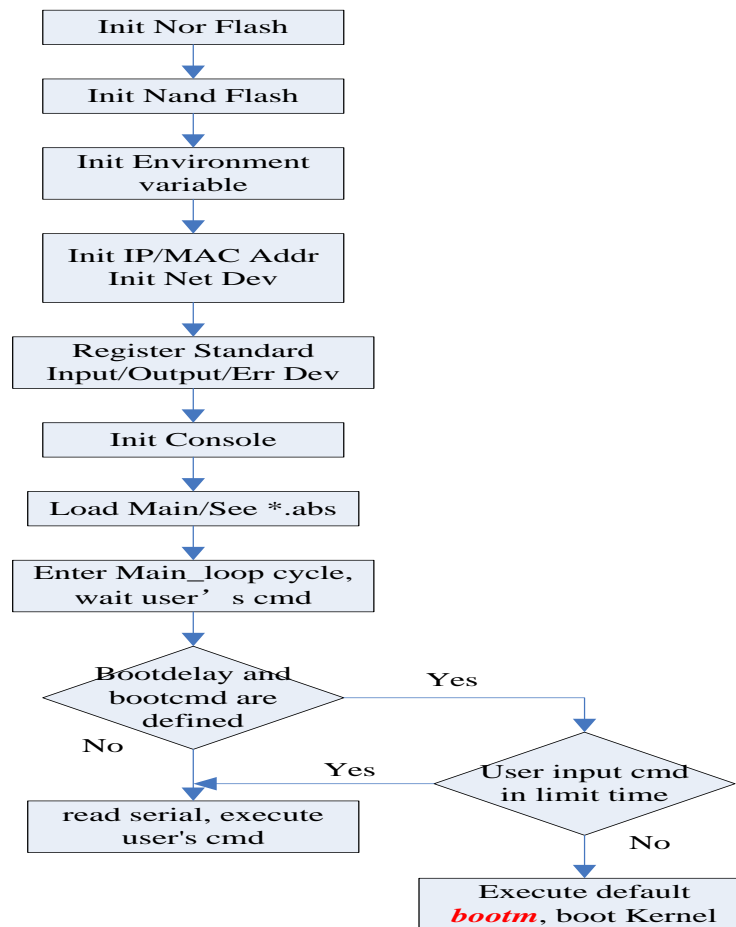


Figure 4. U-Boot Stage 2

2.3 U-Boot Start Kernel

U-Boot will enter the `main_loop` and wait for a user to input a command for certain time. It will run the received command instead of the default `bootm` command to start linux kernel.

U-Boot starts kernel in the following steps:

- Resolve the address entered by a user in `bootm`
- Get U-Boot image head information, and verify header CRC and data CRC.
- Call the `do_bootm_linux` function to start kernel and enter the Linux world.

`Kernel_entry` is the Kernel function finally called. Its prototype is:

```
(*kernel_entry)(int zero, int arch, uint params);
```

Arg zerp: set R0 register to 0.

Arg arch: machine code stored in R1 register.

Arg params: address passed to Linux kernel by U-Boot, stored in R2 register.

```
static void boot_jump_linux(bootm_headers_t *images)
{
    unsigned long machid = gd->bd->bi_arch_number;
    char *s;
    void (*kernel_entry)(int zero, int arch, uint params);
    unsigned long r2;

    kernel_entry = (void (*)(int, int, uint))images->ep;

    s = getenv("machid");
    if (s) {
        strict_strtoul(s, 16, &machid);
        printf("Using machid 0x%lx from environment\n", machid);
    }

    debug("## Transferring control to Linux (at address %08lx)" \
        "...\\n", (ulong) kernel_entry);
    bootstage_mark(BOOTSTAGE_ID_RUN_OS);
    announce_and_cleanup();

    r2 = gd->bd->bi_boot_params;

    kernel_entry(0, machid, r2);
} ? end boot_jump_linux ?
```

Before starting Linux Kernel, U-Boot must provide a complete Linux Kernel startup arguments saved in struct tage and struct tag_head. The data structure is located in include/asm-arm/setup.h, and a corresponding data structure in the path linux/arch/arm/include/asm/setup.h can be found in Linux Kernel.

```
struct tag {
    struct tag_header hdr;

    union {
        struct tag_core core;        //pass information related to root
        struct tag_system system;
        struct tag_mem32 mem;        //pass Memory start address and size
        struct tag_videotext videotext;    //pass information related to
        struct tag_ramdisk ramdisk;    //pass information related to
    };
};
```

```
        struct tag_initrd    initrd;

        struct tag_serialnr  serialnr;    //pass information related to
serial
        port

        struct tag_revision  revision;    //pass information related to
revision

        struct tag_videolfb  videolfb;    //pass information related to
video

        struct tag_cmdline   cmdline;    //pass information related to
cmdline

        struct tag_acorn     acorn;    //pass information related to ARM
CPU

        struct tag_memclk    memclk;    //pass information related to
Memory clock

    } u;

};
```


The complete process is as follow:

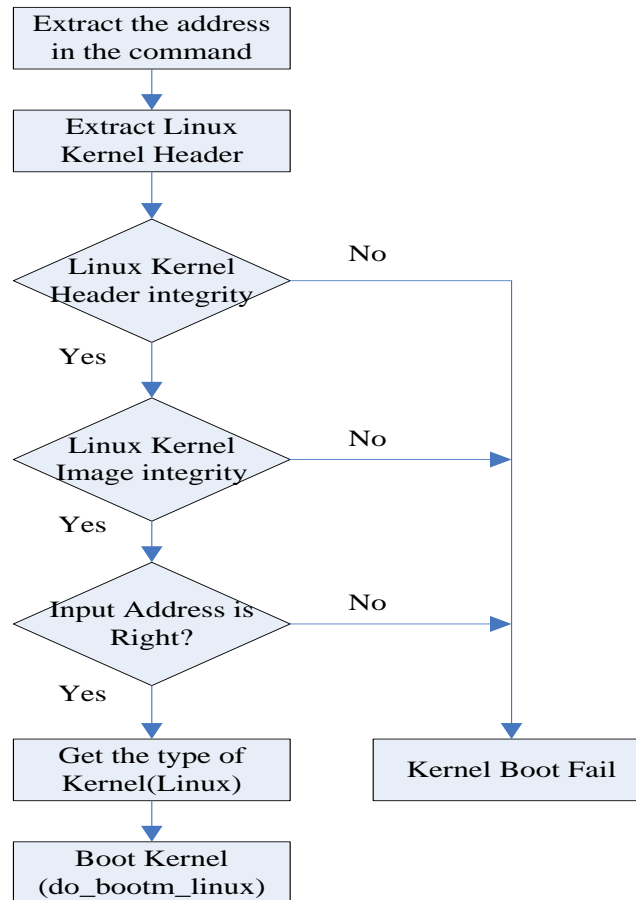


Figure 5. U-Boot Start Process

2.4 U-Boot Global Variables

gd_t and *bd_t*, defined in `./include/asm-arm/global_data.h` and `./include/asm-arm/u-boot.h` separately, are the most important data structure in U-Boot. They save or pass values in initialization operation.

```

typedef struct global_data {

    bd_t      *bd;          //board_info struct point, saves board information

    unsigned long    flags;    //indicator flag ldevice initialized or
                               something else

    unsigned long    baudrate;    //serial port baud rate
    
```

```
    unsigned long    have_console; //serial port initialized

    unsigned long    env_add;      //address of environment params

    unsigned long    env_valid; //environment params available CRC
    verify flag

    unsigned long    fb_base;     //frame buffer base address

#ifdef CONFIG_VFD
    unsigned char    vfd_type;    //display type
#endif

    unsigned long    cpu_clk;     //CPU clock in Hz

    unsigned long    bus_clk;

    unsigned long    ram_size;

    unsigned long    reset_status; //reset status register at boot

    void             **jt;        //jump table

} gd_t;
```

```
Typedef struct bd_info {

    int              bi_baudrate    //serial port baud rate

    unsigned long    bi_ip_addr;    //IP address

    struct environment_s*bi_env;

    ulong            bi_arch_number; //board ID number

    ulong            bi_boot_params; //init parameters

    struct

    {                //DRAM BANKS configuration, start address and size

        ulong start;

        ulong size;

    }bi_dram[CONFIG_NR_DRAM_BANKS]
```

```
} bd_t;
```

3 Transplant on TIGER Board

The directory is added under the board directory when U-Boot is transplanted to TIGER Board platform. For example, `board/ALi/Ali_3921` directory is added for GoDroid.

This chapter introduces the functions of some important files.

- `Ali_3921.c`: Provides API to retrieve information about Board/Chip.
- `Ali_3921_part.c`: Provides API to read the partition data, especially for `kernel/see/ae/ramdisk`, `bootlogo/bootmedia` and recovery partition, in NAND Flash.
- `Ali_see_boot.c`: Provides API to boot see and CPU.
- `Baseparams.c`: Provides API to get baseparameters information.
- `Bcb.c`: Provides API to get misc partition information.
- `Board.c`: Revised board init process.
- `Bootargs.c`: Provides API to get bootargs partition information.
- `Bootkey.c`: Provides API to detect the Front Panel keys and Remote Controller keys when U-Boot starts.
- `Bootm.c`: Revised boot Linux process.
- `Deviceinfo.c`: Provides API to get deviceinfo partition data.

4 Recovery System Entering Rules

In the `board.c` file, in addition to the basic boot process, we also take some particular cases into consideration. When the partition is corrupted and causing boot process failure, you need to start the Recovery System to restore or upgrade the system.

The detailed processes are showed as follow:

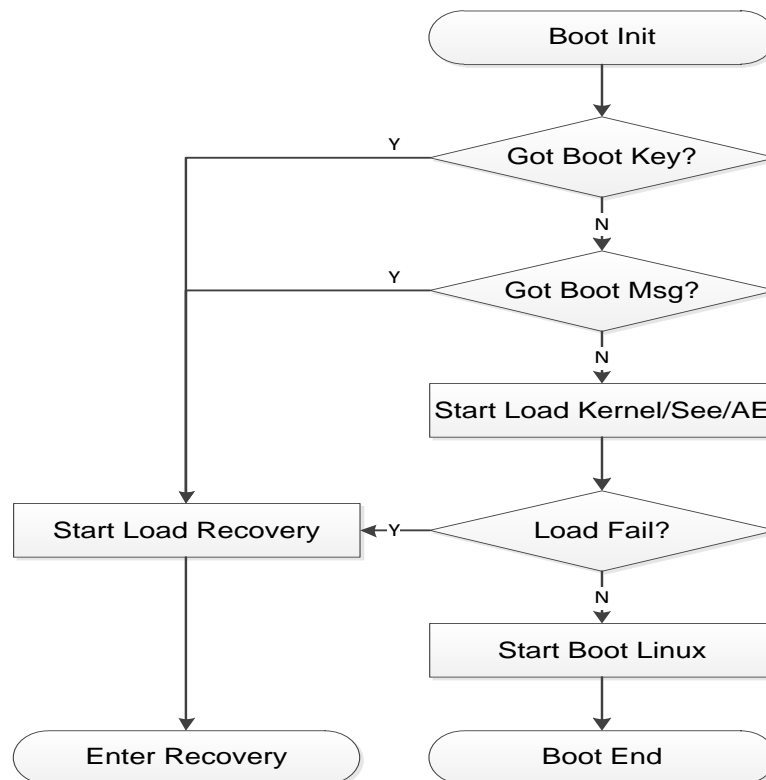


Figure 6. Recovery System Entering Process

Appendix: Glossary

| Abbr. | Full Name |
|--------|------------------------|
| GPL | General Public License |
| U-Boot | Universal BootLoader |

Table 3. List of Abbreviations

Revision History

Document Change

| Revision | Changes | Date |
|----------|-----------------|--------------------|
| v1.0 | Initial Release | September 07, 2015 |

Table 4. Document Change History

Software Change

| Revision | Changes | Date |
|----------|-----------------|--------------------|
| v1.0 | Initial Release | September 07, 2015 |

Table 5. Software Change History



www.gowarriorosh.com

Headquarters

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

