TCC893X UART USER GUIDE

TCC893x_UART_USER_GUIDE

Rev. 0.30

Feb. 27. 2014

Telechips

DISCLAIMER

All information and data contained in this material are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability. Any new issue of this material invalidates previous issues. Product availability and delivery are exclusively subject to our respective order confirmation form; the same applies to orders based on development samples delivered. By this publication, Telechips, Inc. does not assume responsibility for patent infringements or other rights of third parties that may result from its use.

Further, Telechips, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.
No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of Telechips, Inc.
This product is designed for general purpose, and accordingly customer be responsible for all or any of intellectual property licenses required for actual application. Telechips, Inc. does not provide any indemnification for any intellectual properties owned by third party.
Telechips, Inc. can not ensure that this application is the proper and sufficient one for any other purposes but the one explicitly expressed herein. Telechips, Inc. is not responsible for any special, indirect, incidental or consequential damage or loss whatsoever resulting from the use of this application for other purposes.

COPYRIGHT STATEMENT

Copyright in the material provided by Telechips, Inc. is owned by Telechips unless otherwise noted.
For reproduction or use of Telechips' copyright material, permission should be sought from Telechips. That permission, if given, will be subject to conditions that Telechips' name should be included and interest in the material should be acknowledged when the material is reproduced or quoted, either in whole or in part. You must not copy, adapt, publish, distribute or commercialize any contents contained in the material in any manner without the written permission of Telechips. Trade marks used in Telechips' copyright material are the property of Telechips.

Important Notice

For customers who use licensed Codec ICs and/or licensed codec firmware of mp3:
"Supply of this product does not convey a license nor imply any right to distribute content created with this product in revenue-generating broadcast systems (terrestrial. Satellite, cable and/or other distribution channels), streaming applications(via internet, intranets and/or other networks), other content distribution systems(pay-audio or audio-on-demand applications and the like) or on physical media(compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use other firmware of mp3:
"Supply of this product does not convey a license under the relevant intellectual property of Thomson and/or Fraunhofer Gesellschaft nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use Digital Wave DRA solution:
"Supply of this implementation of DRA technology does not convey a license nor imply any right to this implementation in any finished end-user or ready-to-use terminal product. An independent license for such use is required."

For customers who use DTS technology:
"This product made under license to certain U.S. patents and/or foreign counterparts."
"© 1996 – 2011 DTS, Inc. All rights reserved."

For customers who use Dolby technology:
"Supply of this Implementation of Dolby technology does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Dolby Laboratories, 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 Dolby Laboratories."

For customers who use MS technology:
"This product is subject to certain intellectual property rights of Microsoft and cannot be used or disctributed further without the appropriate license(s) from Microsoft."

Revision History

Date	Version	Description
2013-02-13	0.10	This document is a guide to theUART. Initial release
2013-07-01	0.20	Revise the contents about setting DMA. Add the description of DMA switching.
2014-02-27	0.30	Add the description of UART baud rate.

TABLE OF CONTENTS

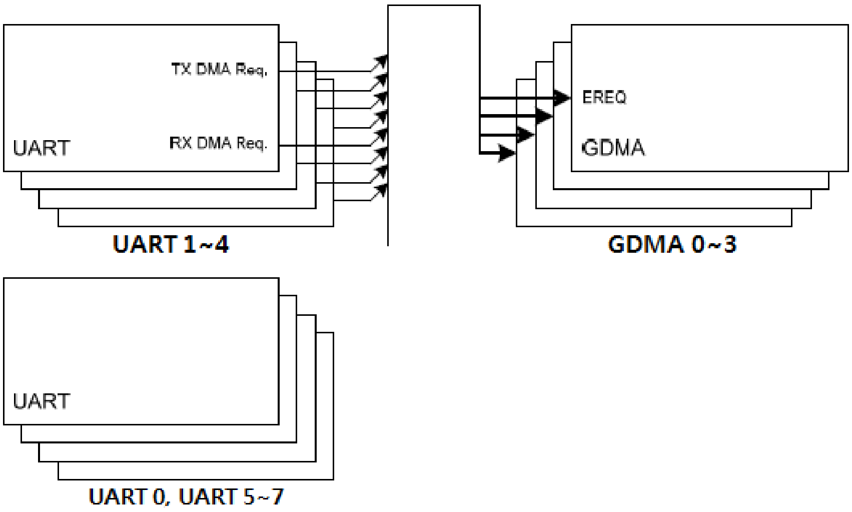
Contents

1 Introduction

This document is to describe method which make user to use UART for TCC893x.

2 UART

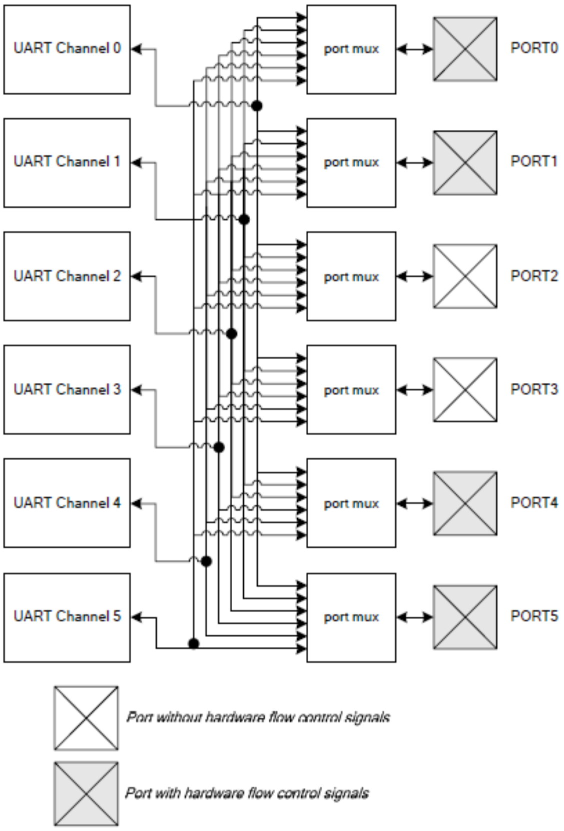
2.1 TCC893X UART



In case of tcc893x platform, it has eight uart channels and 1,2,3,4 channels can be operated with DMA as default. Uart 0,5,6,7 can also be operated with DMA but they need DMA switching.

2.2 UART channel mux

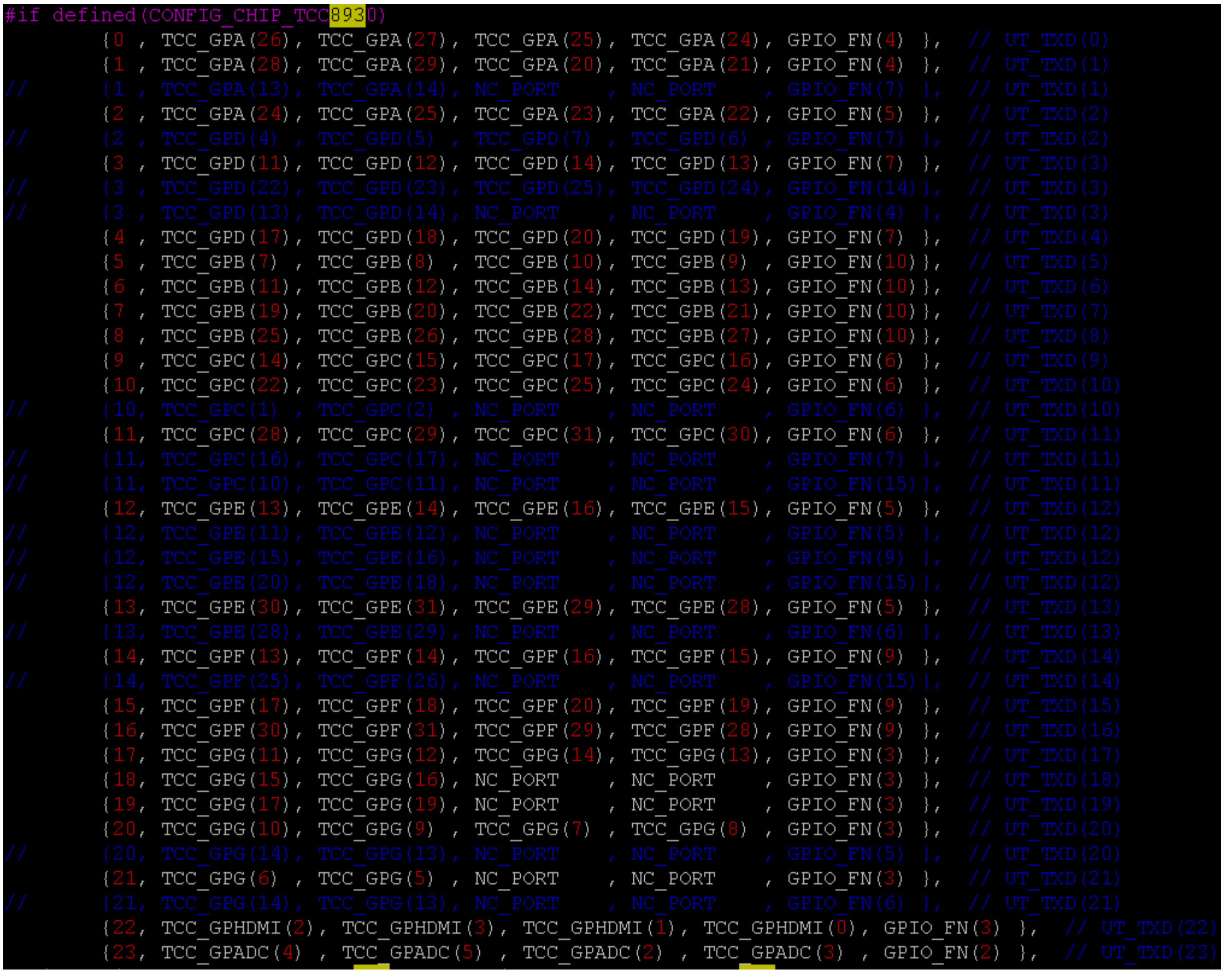
Each uart channel has port mux. So, it is possible to change from itself to other port of uart



2.2.1 How to change port

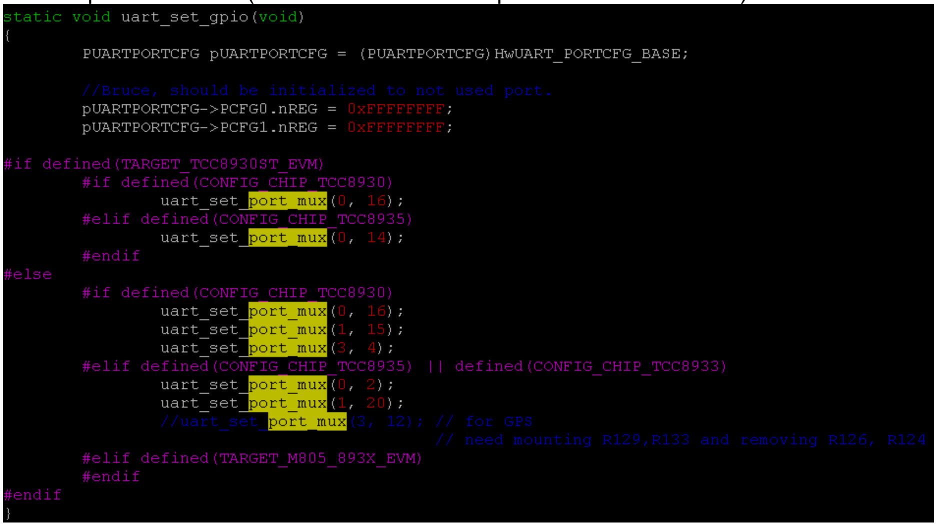
TCC893X has so many uart ports and duplicated ports in some ports.
For example, such as TCC8930 platform, there are following uart ports.

1. bootloader



```
#if defined(CONFIG_CHIP_TCC8930)
        {0 , TCC_GPA(26), TCC_GPA(27), TCC_GPA(25), TCC_GPA(24), GPIO_FN(4) },   // UT_TXD(0)
        {1 , TCC_GPA(28), TCC_GPA(29), TCC_GPA(20), TCC_GPA(21), GPIO_FN(4) },   // UT_TXD(1)
//      {1 , TCC_GPA(13), TCC_GPA(14), NC_PORT    , NC_PORT    , GPIO_FN(7) },   // UT_TXD(1)
        {2 , TCC_GPA(24), TCC_GPA(25), TCC_GPA(23), TCC_GPA(22), GPIO_FN(5) },   // UT_TXD(2)
//      {2 , TCC_GPD(4) , TCC_GPD(5) , TCC_GPD(7) , TCC_GPD(6) , GPIO_FN(7) },   // UT_TXD(2)
        {3 , TCC_GPD(11), TCC_GPD(12), TCC_GPD(14), TCC_GPD(13), GPIO_FN(7) },   // UT_TXD(3)
//      {3 , TCC_GPD(22), TCC_GPD(23), TCC_GPD(25), TCC_GPD(24), GPIO_FN(14)},   // UT_TXD(3)
//      {3 , TCC_GPD(13), TCC_GPD(14), NC_PORT    , NC_PORT    , GPIO_FN(4) },   // UT_TXD(3)
        {4 , TCC_GPD(17), TCC_GPD(18), TCC_GPD(20), TCC_GPD(19), GPIO_FN(7) },   // UT_TXD(4)
        {5 , TCC_GPB(7) , TCC_GPB(8) , TCC_GPB(10), TCC_GPB(9) , GPIO_FN(10)},   // UT_TXD(5)
        {6 , TCC_GPB(11), TCC_GPB(12), TCC_GPB(14), TCC_GPB(13), GPIO_FN(10)},   // UT_TXD(6)
        {7 , TCC_GPB(19), TCC_GPB(20), TCC_GPB(22), TCC_GPB(21), GPIO_FN(10)},   // UT_TXD(7)
        {8 , TCC_GPB(25), TCC_GPB(26), TCC_GPB(28), TCC_GPB(27), GPIO_FN(10)},   // UT_TXD(8)
        {9 , TCC_GPC(14), TCC_GPC(15), TCC_GPC(17), TCC_GPC(16), GPIO_FN(6) },   // UT_TXD(9)
        {10, TCC_GPC(22), TCC_GPC(23), TCC_GPC(25), TCC_GPC(24), GPIO_FN(6) },   // UT_TXD(10)
//      {10, TCC_GPC(1) , TCC_GPC(2) , NC_PORT    , NC_PORT    , GPIO_FN(6) },   // UT_TXD(10)
        {11, TCC_GPC(28), TCC_GPC(29), TCC_GPC(31), TCC_GPC(30), GPIO_FN(6) },   // UT_TXD(11)
//      {11, TCC_GPC(16), TCC_GPC(17), NC_PORT    , NC_PORT    , GPIO_FN(7) },   // UT_TXD(11)
//      {11, TCC_GPC(10), TCC_GPC(11), NC_PORT    , NC_PORT    , GPIO_FN(15)},   // UT_TXD(11)
        {12, TCC_GPE(13), TCC_GPE(14), TCC_GPE(16), TCC_GPE(15), GPIO_FN(5) },   // UT_TXD(12)
//      {12, TCC_GPE(11), TCC_GPE(12), NC_PORT    , NC_PORT    , GPIO_FN(5) },   // UT_TXD(12)
//      {12, TCC_GPE(15), TCC_GPE(16), NC_PORT    , NC_PORT    , GPIO_FN(9) },   // UT_TXD(12)
//      {12, TCC_GPE(20), TCC_GPE(18), NC_PORT    , NC_PORT    , GPIO_FN(15)},   // UT_TXD(12)
        {13, TCC_GPE(30), TCC_GPE(31), TCC_GPE(29), TCC_GPE(28), GPIO_FN(5) },   // UT_TXD(13)
//      {13, TCC_GPE(28), TCC_GPE(29), NC_PORT    , NC_PORT    , GPIO_FN(6) },   // UT_TXD(13)
        {14, TCC_GPF(13), TCC_GPF(14), TCC_GPF(16), TCC_GPF(15), GPIO_FN(9) },   // UT_TXD(14)
//      {14, TCC_GPF(25), TCC_GPF(26), NC_PORT    , NC_PORT    , GPIO_FN(15)},   // UT_TXD(14)
        {15, TCC_GPF(17), TCC_GPF(18), TCC_GPF(20), TCC_GPF(19), GPIO_FN(9) },   // UT_TXD(15)
        {16, TCC_GPF(30), TCC_GPF(31), TCC_GPF(29), TCC_GPF(28), GPIO_FN(9) },   // UT_TXD(16)
        {17, TCC_GPG(11), TCC_GPG(12), TCC_GPG(14), TCC_GPG(13), GPIO_FN(3) },   // UT_TXD(17)
        {18, TCC_GPG(15), TCC_GPG(16), NC_PORT    , NC_PORT    , GPIO_FN(3) },   // UT_TXD(18)
        {19, TCC_GPG(17), TCC_GPG(19), NC_PORT    , NC_PORT    , GPIO_FN(3) },   // UT_TXD(19)
        {20, TCC_GPG(10), TCC_GPG(9) , TCC_GPG(7) , TCC_GPG(8) , GPIO_FN(3) },   // UT_TXD(20)
//      {20, TCC_GPG(14), TCC_GPG(13), NC_PORT    , NC_PORT    , GPIO_FN(5) },   // UT_TXD(20)
        {21, TCC_GPG(6) , TCC_GPG(5) , NC_PORT    , NC_PORT    , GPIO_FN(3) },   // UT_TXD(21)
//      {21, TCC_GPG(14), TCC_GPG(13), NC_PORT    , NC_PORT    , GPIO_FN(6) },   // UT_TXD(21)
        {22, TCC_GPHDMI(2), TCC_GPHDMI(3), TCC_GPHDMI(1), TCC_GPHDMI(0), GPIO_FN(3) },   // UT_TXD(22)
        {23, TCC_GPADC(4) , TCC_GPADC(5) , TCC_GPADC(2) , TCC_GPADC(3) , GPIO_FN(2) },   // UT_TXD(23)
```

Port map in bootloader(bootable/bootloader/lk/platform/tcc893x/uart.c)

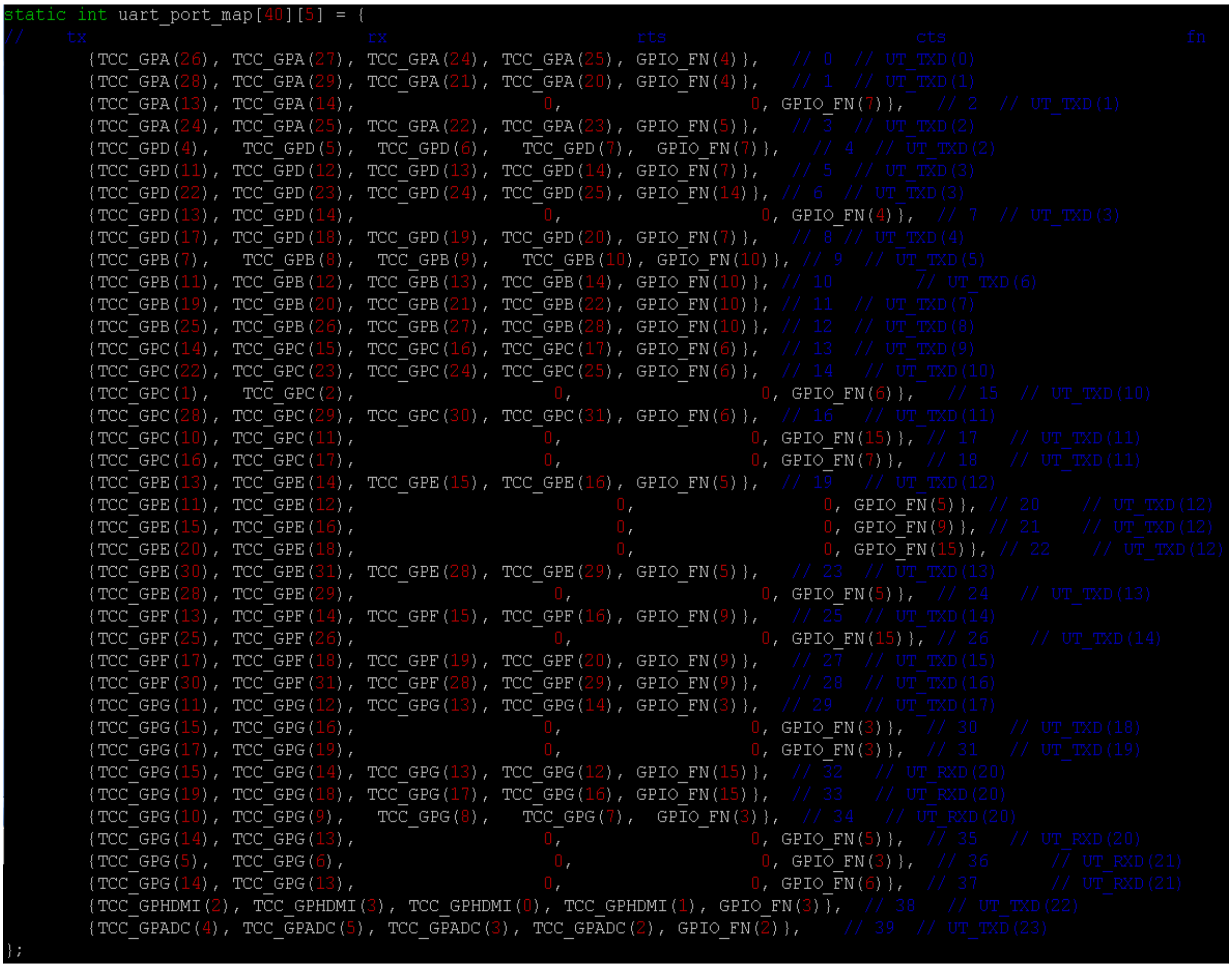


```
static void uart_set_gpio(void)
{
        PUARTPORTCFG pUARTPORTCFG = (PUARTPORTCFG)HwUART_PORTCFG_BASE;

        //Bruce, should be initialized to not used port.
        pUARTPORTCFG->PCFG0.nREG = 0xFFFFFFFF;
        pUARTPORTCFG->PCFG1.nREG = 0xFFFFFFFF;

#if defined(TARGET_TCC8930ST_EVM)
        #if defined(CONFIG_CHIP_TCC8930)
                uart_set_port_mux(0, 16);
        #elif defined(CONFIG_CHIP_TCC8935)
                uart_set_port_mux(0, 14);
        #endif
#else
        #if defined(CONFIG_CHIP_TCC8930)
                uart_set_port_mux(0, 16);
                uart_set_port_mux(1, 15);
                uart_set_port_mux(3, 4);
        #elif defined(CONFIG_CHIP_TCC8935) || defined(CONFIG_CHIP_TCC8933)
                uart_set_port_mux(0, 2);
                uart_set_port_mux(1, 20);
                //uart_set_port_mux(3, 12); // for GPS
                                            // need mounting R129,R133 and removing R126, R124
        #elif defined(TARGET_M805_893X_EVM)
        #endif
#endif
}
```

If you open uart.c file in lk bootloader(bootable/bootloader/lk/platform/tcc893x/), you can find This is channel selection.
If you want to change port, you should fix this code.

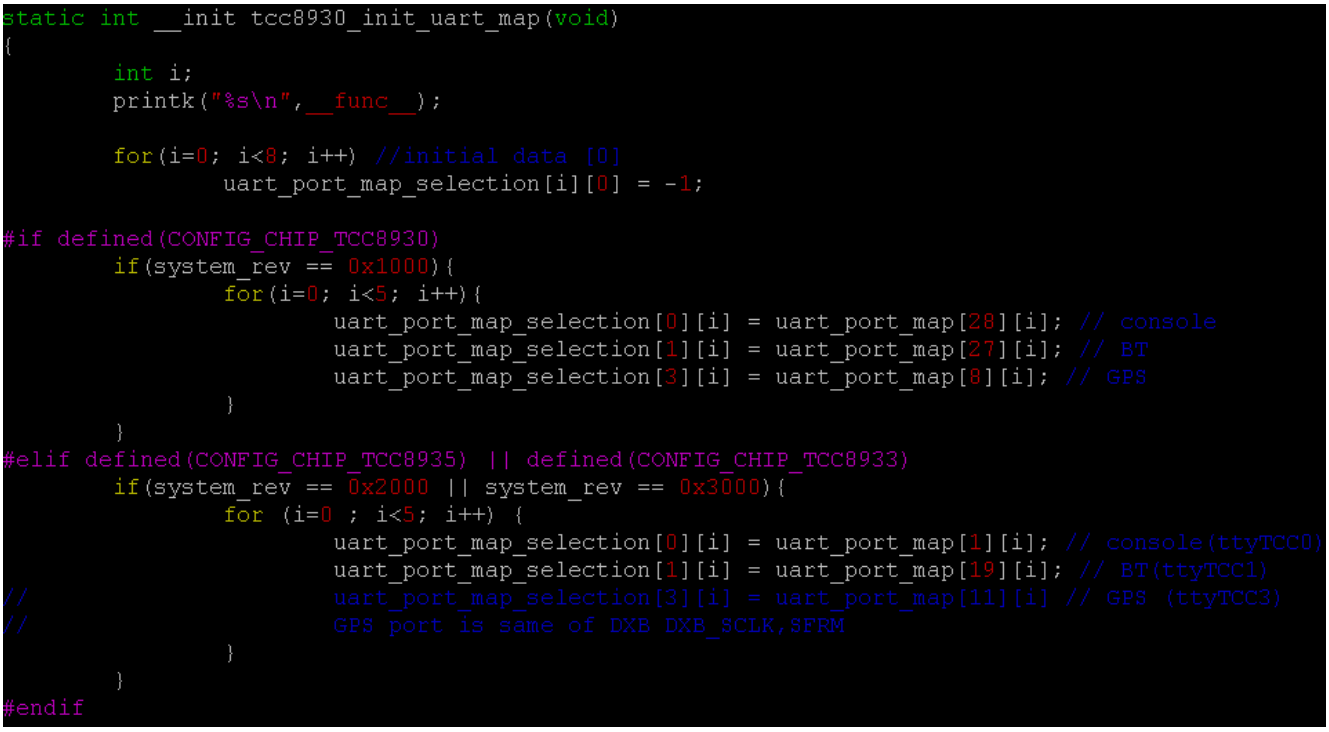
You can map port of uart.
Above feature show uart port mappings.
Uart_set_port_mux(0, 16); -> this means uart channel 0 use uart port 16.
There are port numbers in feature .

2. kernel



```
static int uart_port_map[40][5] = {
//    tx                        rx                      rts                         cts                         fn
        {TCC_GPA(26), TCC_GPA(27), TCC_GPA(24), TCC_GPA(25), GPIO_FN(4)},   // 0  // UT_TXD(0)
        {TCC_GPA(28), TCC_GPA(29), TCC_GPA(21), TCC_GPA(20), GPIO_FN(4)},   // 1  // UT_TXD(1)
        {TCC_GPA(13), TCC_GPA(14),                0,                0, GPIO_FN(7)},   // 2  // UT_TXD(1)
        {TCC_GPA(24), TCC_GPA(25), TCC_GPA(22), TCC_GPA(23), GPIO_FN(5)},   // 3  // UT_TXD(2)
        {TCC_GPD(4),   TCC_GPD(5),   TCC_GPD(6),    TCC_GPD(7),   GPIO_FN(7)},   // 4  // UT_TXD(2)
        {TCC_GPD(11), TCC_GPD(12), TCC_GPD(13), TCC_GPD(14), GPIO_FN(7)},   // 5  // UT_TXD(3)
        {TCC_GPD(22), TCC_GPD(23), TCC_GPD(24), TCC_GPD(25), GPIO_FN(14)}, // 6  // UT_TXD(3)
        {TCC_GPD(13), TCC_GPD(14),                0,                 0, GPIO_FN(4)},   // 7  // UT_TXD(3)
        {TCC_GPD(17), TCC_GPD(18), TCC_GPD(19), TCC_GPD(20), GPIO_FN(7)},   // 8 // UT_TXD(4)
        {TCC_GPB(7),   TCC_GPB(8),   TCC_GPB(9),    TCC_GPB(10), GPIO_FN(10)}, // 9  // UT_TXD(5)
        {TCC_GPB(11), TCC_GPB(12), TCC_GPB(13), TCC_GPB(14), GPIO_FN(10)}, // 10        // UT_TXD(6)
        {TCC_GPB(19), TCC_GPB(20), TCC_GPB(21), TCC_GPB(22), GPIO_FN(10)}, // 11  // UT_TXD(7)
        {TCC_GPB(25), TCC_GPB(26), TCC_GPB(27), TCC_GPB(28), GPIO_FN(10)}, // 12  // UT_TXD(8)
        {TCC_GPC(14), TCC_GPC(15), TCC_GPC(16), TCC_GPC(17), GPIO_FN(6)},  // 13  // UT_TXD(9)
        {TCC_GPC(22), TCC_GPC(23), TCC_GPC(24), TCC_GPC(25), GPIO_FN(6)},  // 14   // UT_TXD(10)
        {TCC_GPC(1),   TCC_GPC(2),                 0,                0, GPIO_FN(6)},   // 15  // UT_TXD(10)
        {TCC_GPC(28), TCC_GPC(29), TCC_GPC(30), TCC_GPC(31), GPIO_FN(6)},  // 16   // UT_TXD(11)
        {TCC_GPC(10), TCC_GPC(11),                0,                0, GPIO_FN(15)}, // 17   // UT_TXD(11)
        {TCC_GPC(16), TCC_GPC(17),                0,                0, GPIO_FN(7)},  // 18   // UT_TXD(11)
        {TCC_GPE(13), TCC_GPE(14), TCC_GPE(15), TCC_GPE(16), GPIO_FN(5)},  // 19   // UT_TXD(12)
        {TCC_GPE(11), TCC_GPE(12),                      0,                    0, GPIO_FN(5)}, // 20    // UT_TXD(12)
        {TCC_GPE(15), TCC_GPE(16),                      0,                    0, GPIO_FN(9)}, // 21    // UT_TXD(12)
        {TCC_GPE(20), TCC_GPE(18),                      0,                    0, GPIO_FN(15)}, // 22    // UT_TXD(12)
        {TCC_GPE(30), TCC_GPE(31), TCC_GPE(28), TCC_GPE(29), GPIO_FN(5)},   // 23  // UT_TXD(13)
        {TCC_GPE(28), TCC_GPE(29),                 0,                0, GPIO_FN(5)},   // 24   // UT_TXD(13)
        {TCC_GPF(13), TCC_GPF(14), TCC_GPF(15), TCC_GPF(16), GPIO_FN(9)},   // 25  // UT_TXD(14)
        {TCC_GPF(25), TCC_GPF(26),                 0,                0, GPIO_FN(15)}, // 26    // UT_TXD(14)
        {TCC_GPF(17), TCC_GPF(18), TCC_GPF(19), TCC_GPF(20), GPIO_FN(9)},   // 27  // UT_TXD(15)
        {TCC_GPF(30), TCC_GPF(31), TCC_GPF(28), TCC_GPF(29), GPIO_FN(9)},   // 28  // UT_TXD(16)
        {TCC_GPG(11), TCC_GPG(12), TCC_GPG(13), TCC_GPG(14), GPIO_FN(3)},  // 29   // UT_TXD(17)
        {TCC_GPG(15), TCC_GPG(16),                0,                0, GPIO_FN(3)},  // 30   // UT_TXD(18)
        {TCC_GPG(17), TCC_GPG(19),                0,                0, GPIO_FN(3)},  // 31   // UT_TXD(19)
        {TCC_GPG(15), TCC_GPG(14), TCC_GPG(13), TCC_GPG(12), GPIO_FN(15)},  // 32   // UT_RXD(20)
        {TCC_GPG(19), TCC_GPG(18), TCC_GPG(17), TCC_GPG(16), GPIO_FN(15)},  // 33   // UT_RXD(20)
        {TCC_GPG(10), TCC_GPG(9),   TCC_GPG(8),    TCC_GPG(7),   GPIO_FN(3)},  // 34   // UT_RXD(20)
        {TCC_GPG(14), TCC_GPG(13),                0,                0, GPIO_FN(5)},  // 35   // UT_RXD(20)
        {TCC_GPG(5),   TCC_GPG(6),                 0,                0, GPIO_FN(3)},  // 36      // UT_RXD(21)
        {TCC_GPG(14), TCC_GPG(13),                0,                0, GPIO_FN(6)},  // 37      // UT_RXD(21)
        {TCC_GPHDMI(2), TCC_GPHDMI(3), TCC_GPHDMI(0), TCC_GPHDMI(1), GPIO_FN(3)},  // 38   // UT_TXD(22)
        {TCC_GPADC(4), TCC_GPADC(5), TCC_GPADC(3), TCC_GPADC(2), GPIO_FN(2)},    // 39  // UT_TXD(23)
};
```

Port map in kernel



```
static int __init tcc8930_init_uart_map(void)
{
        int i;
        printk("%s\n",__func__);

        for(i=0; i<8; i++) //initial data [0]
                uart_port_map_selection[i][0] = -1;

#if defined(CONFIG_CHIP_TCC8930)
        if(system_rev == 0x1000){
                for(i=0; i<5; i++){
                        uart_port_map_selection[0][i] = uart_port_map[28][i]; // console
                        uart_port_map_selection[1][i] = uart_port_map[27][i]; // BT
                        uart_port_map_selection[3][i] = uart_port_map[8][i]; // GPS
                }
        }
#elif defined(CONFIG_CHIP_TCC8935) || defined(CONFIG_CHIP_TCC8933)
        if(system_rev == 0x2000 || system_rev == 0x3000){
                for (i=0 ; i<5; i++) {
                        uart_port_map_selection[0][i] = uart_port_map[1][i]; // console(ttyTCC0)
                        uart_port_map_selection[1][i] = uart_port_map[19][i]; // BT(ttyTCC1)
//                      uart_port_map_selection[3][i] = uart_port_map[11][i] // GPS (ttyTCC3)
//                      GPS port is same of DXB DXB_SCLK,SFRM
                }
        }
#endif
```

If you open board-(platform name)-uart.c file in kernel(arch/arm/mach-tcc893x/), you can find This is channel selection.
If you want to change port, you should also fix this code.
Uart_port_map_selection[uart channel number][i] = uart_port_map[uart port number][i]
For example, uart_port_map_selection[0][i] = uuart_port_map[28][i]   -> uart 0 channel use uart 28 port(gpio f[30],f[31])

2.3 Default UART on TCC893X

On TCC893X, there are default setting about uart.

- uart0 -> console
- uart1 -> Bluetooth
- uart3 -> GPS

If you want to change these, you can. But we recommend these setting.

2.4 UART baud rate

2.4.1 Divisor Latch Register

Divisor Latch Register (DLL) **UART_BASE + 0x00 (DLAB=1)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0								Divisor Latch LSB							

Field	Name	RW	Reset	Description
7-0	Divisor Latch LSB	R/W	0x00	This is for generation of the desired baud rate clock.

Divisor Latch Register (DLM) **UART_BASE + 0x04 (DLAB=1)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0								Divisor Latch MSB							

Field	Name	RW	Reset	Description
7-0	Divisor Latch MSB	R/W	0x00	This is for generation of the desired baud rate clock.

The value can be calculated as follows.

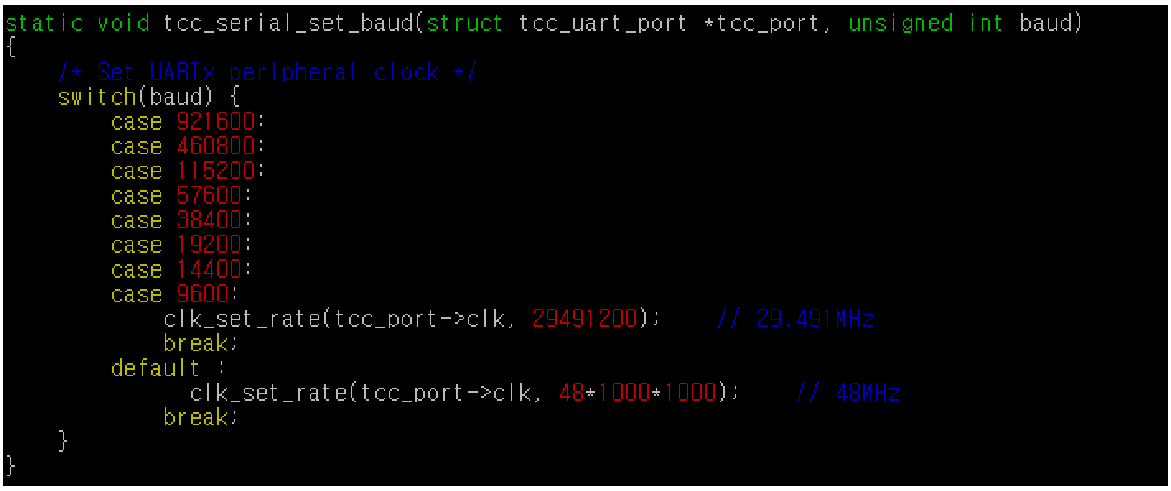
{DLM, DLL} = fUART / (16 * desired baud rate)

For example,

If UART clock frequency is 48MHz (from CKC) and the baud-rate you want is 115,200 bps, the divisor value should be 26(48M/ (115200 x16)) in decimal.

2.4.2 How to set the fUART

If you need to change the fUART, revise the the function “tcc_serial_set_baud”. The function can be found in “kernel/drivers/tty/serial/tcc_serial.c”.



```
static void tcc_serial_set_baud(struct tcc_uart_port *tcc_port, unsigned int baud)
{
    /* Set UARTx peripheral clock */
    switch(baud) {
        case 921600:
        case 460800:
        case 115200:
        case 57600:
        case 38400:
        case 19200:
        case 14400:
        case 9600:
            clk_set_rate(tcc_port->clk, 29491200);    // 29.491MHz
            break;
        default :
              clk_set_rate(tcc_port->clk, 48*1000*1000);    // 48MHz
            break;
    }
}
```

In case of TCC893x, the values of fUART in this code(29.491MHz and 48MHz) are from pll_2(1GHz) and pll_5(720MHz) so the real frequency of 48MHz is same as the value in the code(48,000,000) but the real value of 29.491MHz should be 29411700.

2.4.3 Error rate

The real value of baud rate is not exactly same as the ideal one because baud rate is made from the fUART devided by an interger. If you do not change the fUART, the error rates of baud are like the table below.

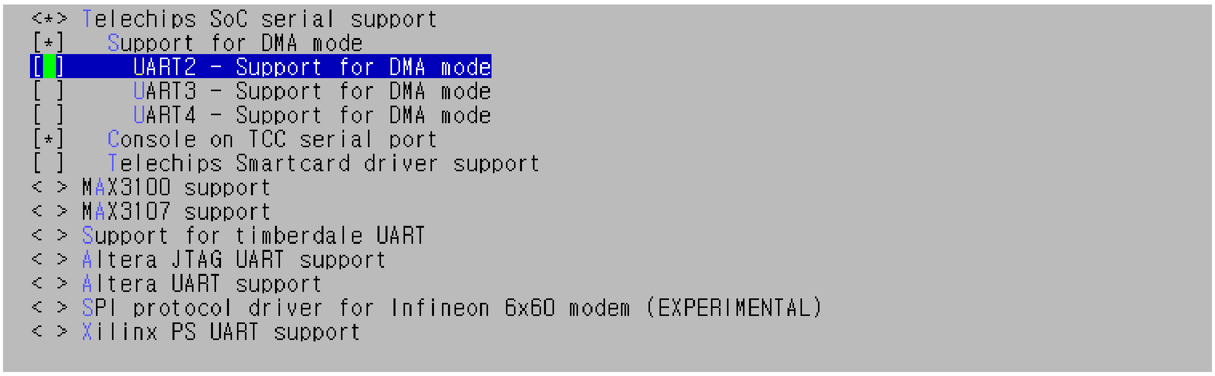
Baud rate	fUART	Ideal DIV	Real DIV	DIV * 16	Real baud rate	Err rate (%)
2400	48000000	1,250.0000	1,250	20000	2400.00000	0.00000
4800	48000000	625.0000	625	10000	4800.00000	0.00000
9600	29411700	191.4824	191	3056	9624.24738	0.25258
14400	29411700	127.6549	128	2048	14361.18164	-0.26957
19200	29411700	95.7412	96	1536	19148.24219	-0.26957
38400	29411700	47.8706	48	768	38296.48438	-0.26957
57600	29411700	31.9137	32	512	57444.72656	-0.26957
115200	29411700	15.9569	16	256	114889.45313	-0.26957
230400	29411700	7.9784	8	128	229778.90625	-0.26957
230400	48000000	13.0208	13	208	230769.23077	0.16026
460800	29411700	3.9892	4	64	459557.81250	-0.26957
921600	29411700	1.9946	2	32	919115.62500	-0.26957

3 Setting the configuration of DMA

3.1 How to set DMA of UART

Uart 1,2,3,4 can use DMA but uart 1 for Bluetooth uses DMA basically. If you want to use DMA of uart 2,3,4, you should check option of DMA.
You can find option in kernel menuconfig.

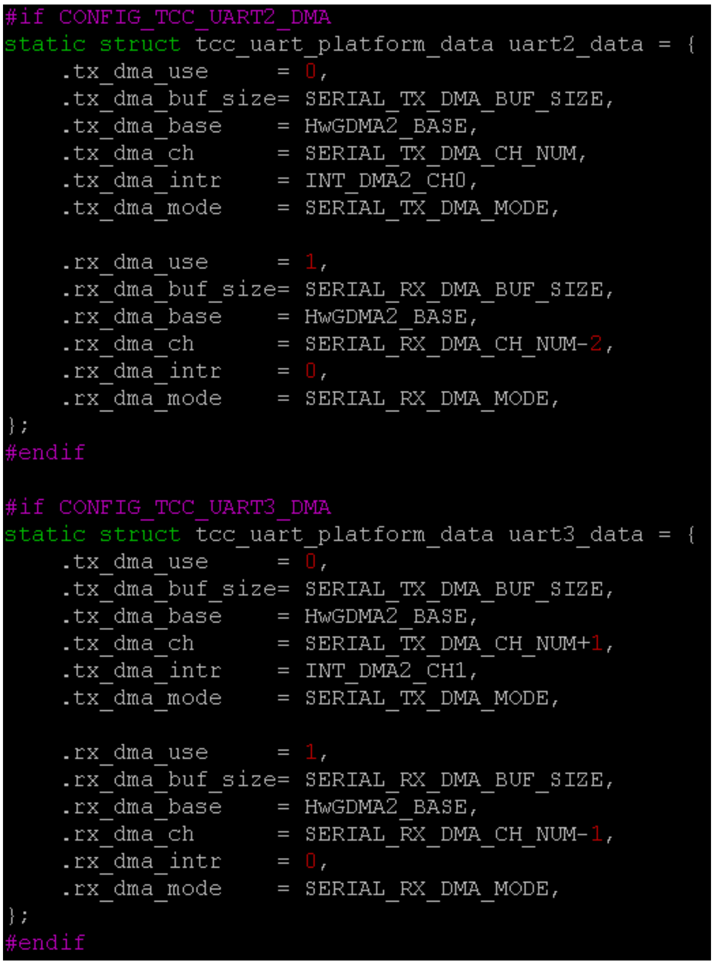
- (in kernel folder) make menuconfig --> device drivers --> Character devices --> Serial drivers



```
<*> Telechips SoC serial support
[*]   Support for DMA mode
[ ]     UART2 - Support for DMA mode
[ ]     UART3 - Support for DMA mode
[ ]     UART4 - Support for DMA mode
[*]   Console on TCC serial port
[ ]   Telechips Smartcard driver support
< > MAX3100 support
< > MAX3107 support
< > Support for timberdale UART
< > Altera JTAG UART support
< > Altera UART support
< > SPI protocol driver for Infineon 6x60 modem (EXPERIMENTAL)
< > Xilinx PS UART support
```

"UART2 – Support for DMA mode" is DMA option of UART 2. There are DMA options of UART 2,3,4.

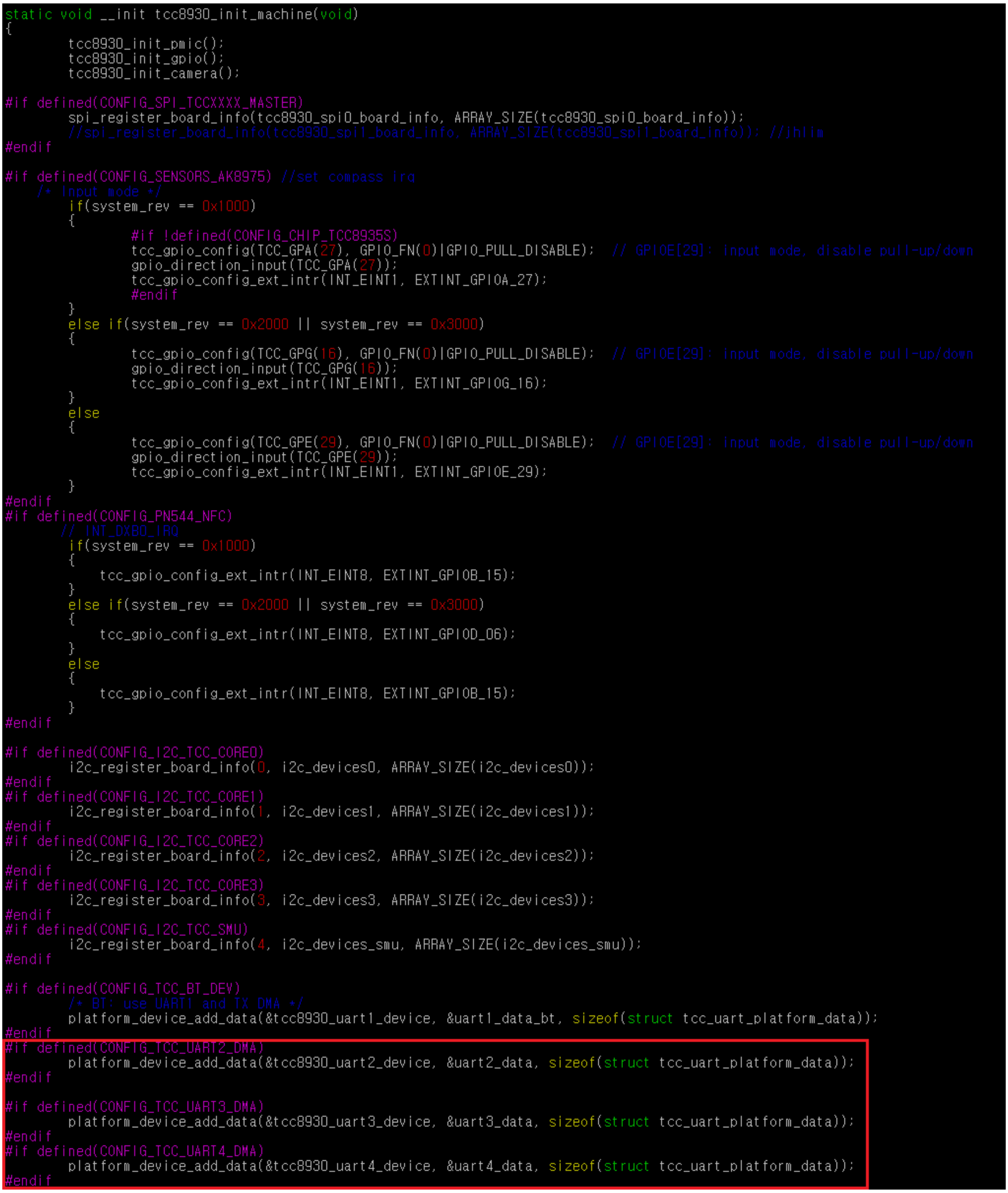
3.2 Setting platform data for DMA



```
#if CONFIG_TCC_UART2_DMA
static struct tcc_uart_platform_data uart2_data = {
    .tx_dma_use      = 0,
    .tx_dma_buf_size= SERIAL_TX_DMA_BUF_SIZE,
    .tx_dma_base     = HwGDMA2_BASE,
    .tx_dma_ch       = SERIAL_TX_DMA_CH_NUM,
    .tx_dma_intr     = INT_DMA2_CH0,
    .tx_dma_mode     = SERIAL_TX_DMA_MODE,

    .rx_dma_use      = 1,
    .rx_dma_buf_size= SERIAL_RX_DMA_BUF_SIZE,
    .rx_dma_base     = HwGDMA2_BASE,
    .rx_dma_ch       = SERIAL_RX_DMA_CH_NUM-2,
    .rx_dma_intr     = 0,
    .rx_dma_mode     = SERIAL_RX_DMA_MODE,
};
#endif

#if CONFIG_TCC_UART3_DMA
static struct tcc_uart_platform_data uart3_data = {
    .tx_dma_use      = 0,
    .tx_dma_buf_size= SERIAL_TX_DMA_BUF_SIZE,
    .tx_dma_base     = HwGDMA2_BASE,
    .tx_dma_ch       = SERIAL_TX_DMA_CH_NUM+1,
    .tx_dma_intr     = INT_DMA2_CH1,
    .tx_dma_mode     = SERIAL_TX_DMA_MODE,

    .rx_dma_use      = 1,
    .rx_dma_buf_size= SERIAL_RX_DMA_BUF_SIZE,
    .rx_dma_base     = HwGDMA2_BASE,
    .rx_dma_ch       = SERIAL_RX_DMA_CH_NUM-1,
    .rx_dma_intr     = 0,
    .rx_dma_mode     = SERIAL_RX_DMA_MODE,
};
#endif
```

If you open board-(platform ex,tcc8920).c file(in arch/arm/mach-tcc89xx/), you can find above codes. These are platform data for DMA about uart2,3,4. If you can't find these codes in that file, that platform doesn't be set yet. In case this, you can add these codes in that file(ex, board-tcc9300.c in arch/arm/tcc93xx)

You should also add following codes.



```
static void __init tcc8930_init_machine(void)
{
        tcc8930_init_pmic();
        tcc8930_init_gpio();
        tcc8930_init_camera();

#if defined(CONFIG_SPI_TCCXXXX_MASTER)
        spi_register_board_info(tcc8930_spi0_board_info, ARRAY_SIZE(tcc8930_spi0_board_info));
        //spi_register_board_info(tcc8930_spi1_board_info, ARRAY_SIZE(tcc8930_spi1_board_info)); //jhlim
#endif

#if defined(CONFIG_SENSORS_AK8975) //set compass irq
    /* Input mode */
        if(system_rev == 0x1000)
        {
                #if !defined(CONFIG_CHIP_TCC8935S)
                tcc_gpio_config(TCC_GPA(27), GPIO_FN(0)|GPIO_PULL_DISABLE);  // GPIOE[29]: input mode, disable pull-up/down
                gpio_direction_input(TCC_GPA(27));
                tcc_gpio_config_ext_intr(INT_EINT1, EXTINT_GPIOA_27);
                #endif
        }
        else if(system_rev == 0x2000 || system_rev == 0x3000)
        {
                tcc_gpio_config(TCC_GPG(16), GPIO_FN(0)|GPIO_PULL_DISABLE);  // GPIOE[29]: input mode, disable pull-up/down
                gpio_direction_input(TCC_GPG(16));
                tcc_gpio_config_ext_intr(INT_EINT1, EXTINT_GPIOG_16);
        }
        else
        {
                tcc_gpio_config(TCC_GPE(29), GPIO_FN(0)|GPIO_PULL_DISABLE);  // GPIOE[29]: input mode, disable pull-up/down
                gpio_direction_input(TCC_GPE(29));
                tcc_gpio_config_ext_intr(INT_EINT1, EXTINT_GPIOE_29);
        }
#endif
#if defined(CONFIG_PN544_NFC)
       // INT_DXB0_IRQ
        if(system_rev == 0x1000)
        {
            tcc_gpio_config_ext_intr(INT_EINT8, EXTINT_GPIOB_15);
        }
        else if(system_rev == 0x2000 || system_rev == 0x3000)
        {
            tcc_gpio_config_ext_intr(INT_EINT8, EXTINT_GPIOD_06);
        }
        else
        {
            tcc_gpio_config_ext_intr(INT_EINT8, EXTINT_GPIOB_15);
        }
#endif

#if defined(CONFIG_I2C_TCC_CORE0)
        i2c_register_board_info(0, i2c_devices0, ARRAY_SIZE(i2c_devices0));
#endif
#if defined(CONFIG_I2C_TCC_CORE1)
        i2c_register_board_info(1, i2c_devices1, ARRAY_SIZE(i2c_devices1));
#endif
#if defined(CONFIG_I2C_TCC_CORE2)
        i2c_register_board_info(2, i2c_devices2, ARRAY_SIZE(i2c_devices2));
#endif
#if defined(CONFIG_I2C_TCC_CORE3)
        i2c_register_board_info(3, i2c_devices3, ARRAY_SIZE(i2c_devices3));
#endif
#if defined(CONFIG_I2C_TCC_SMU)
        i2c_register_board_info(4, i2c_devices_smu, ARRAY_SIZE(i2c_devices_smu));
#endif

#if defined(CONFIG_TCC_BT_DEV)
        /* BT: use UART1 and TX DMA */
        platform_device_add_data(&tcc8930_uart1_device, &uart1_data_bt, sizeof(struct tcc_uart_platform_data));
#endif
#if defined(CONFIG_TCC_UART2_DMA)
        platform_device_add_data(&tcc8930_uart2_device, &uart2_data, sizeof(struct tcc_uart_platform_data));
#endif

#if defined(CONFIG_TCC_UART3_DMA)
        platform_device_add_data(&tcc8930_uart3_device, &uart3_data, sizeof(struct tcc_uart_platform_data));
#endif
#if defined(CONFIG_TCC_UART4_DMA)
        platform_device_add_data(&tcc8930_uart4_device, &uart4_data, sizeof(struct tcc_uart_platform_data));
#endif
```

Codes in red square are needed for setting DMA. These code also are in that file. And if you can’t find these codes, you should add these.

3.3 Switching DMA of UART

Uart 1,2,3,4 use DMA as default. Switch DMA of uart1, 2, 3, 4 if you want to use DMA of uart 0, 5, 6, 7.
You can switch DMA mode between UART0 and 4, UART1 and 5, UART2 and 6, UART3 and 7.

IOBUS Configuration Register Map (Base Address = 0x76066000)

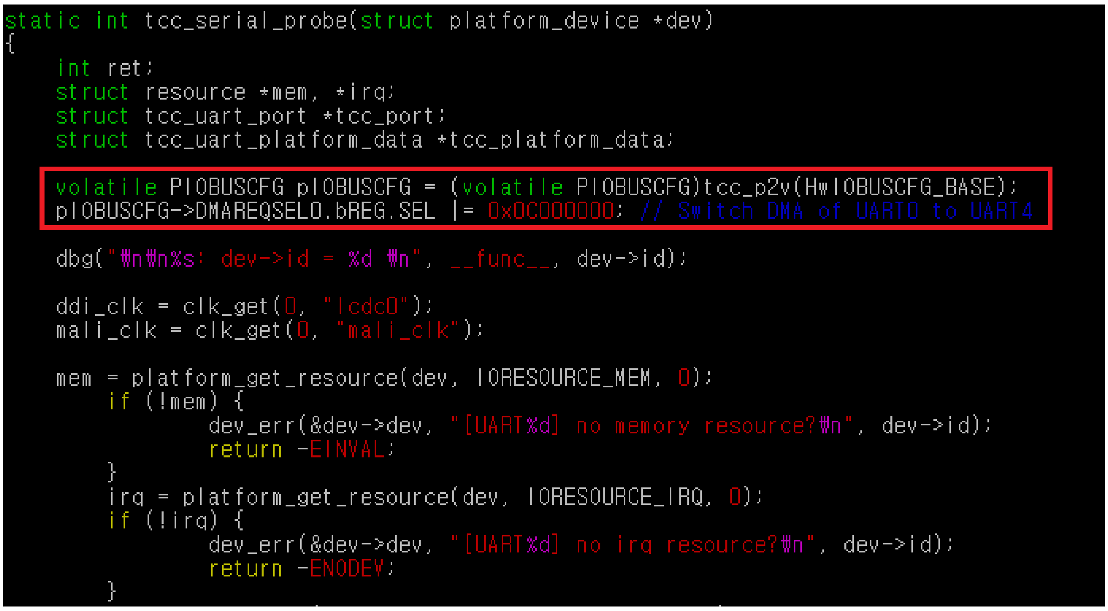
Name	Address	R/W	Reset	Description
-	-	-	-	-
DMAREQSEL0	0x04	R/W	0x00000000	DMA Request Selector for DMA0
DMAREQSEL1	0x08	R/W	0x00000000	DMA Request Selector for DMA1
DMAREQSEL2	0x0C	R/W	0x00000000	DMA Request Selector for DMA2
HCLKEN0	0x10	R/W	0xFFFFFFFF	IOBUS AHB Clock Enable Register 0
HCLKEN1	0x14	R/W	0xFFFFFFFF	IOBUS AHB Clock Enable Register 1
HRSTEN0	0x18	R/W	0xFFFFFFFF	IOBUS AHB HRESET Control Register 0
HRSTEN1	0x1C	R/W	0xFFFFFFFF	IOBUS AHB HRESET Control Register 1
MEMPWR	0x20	R/W	0x00000FFF	Memory Power Controll
RTCWAIT	0x24	R/W	0x00000000	RTC Wait Count
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
IO_A2X	0x38	R/W	0x00000EEE	IOBUS AHB2AXI Control Register
-	-	-	-	-

DMAREQSEL0, 1, and 2 select external DMA request sources of DMA controller 0, 1, and 2 respectively.

DMAREQSEL0,1,2 **0x04, 0x08, 0x0C**

Field	Name	RW	Reset	Description
31	-	R/W	0	
30	-	R/W	0	0 = UART #1 RX 1 = UART #5 RX
29	-	R/W	0	0 = UART #1 TX 1 = UART #5 TX
28	-	R/W	0	
27	-	R/W	0	0 = UART #0 RX 1 = UART #4 RX
26	-	R/W	0	0 = UART #0 TX 1 = UART #4 TX
25	-	R/W	0	0 = I2C Slave #0 TX 1 = I2C Slave #1 TX
24	-	R/W	0	
23	-	R/W	0	
22	-	R/W	0	
21	-	R/W	0	
20	-	R/W	0	
19	-	R/W	0	0 = I2C Slave #0 RX 1 = I2C Slave #1 RX
18	-	R/W	0	
17	-	R/W	0	
16	-	R/W	0	
15	-	R/W	0	
14	-	R/W	0	
13	-	R/W	0	
12	-	R/W	0	
11	-	R/W	0	0 = UART #3 RX 1 = UART #7 RX
10	-	R/W	0	0 = UART #3 TX 1 = UART #7 TX
9	-	R/W	0	0 = UART #2 RX 1 = UART #6 RX
8	-	R/W	0	0 = UART #2 TX 1 = UART #6 TX
7	-	R/W	0	
6	-	R/W	0	0 = GPSB #5 RX 1 = GPSB #2 RX
5	-	R/W	0	0 = GPSB #4 RX 1 = GPSB #1 RX
4	-	R/W	0	0 = GPSB #3 RX 1 = GPSB #0 RX
3	-	R/W	0	

If you open tcc_serial.c file in kernel(drivers/tty/serial/), you can find tcc_serial_probe function.



```
static int tcc_serial_probe(struct platform_device *dev)
{
    int ret;
    struct resource *mem, *irq;
    struct tcc_uart_port *tcc_port;
    struct tcc_uart_platform_data *tcc_platform_data;

    volatile PIOBUSCFG pIOBUSCFG = (volatile PIOBUSCFG)tcc_p2v(HwIOBUSCFG_BASE);
    pIOBUSCFG->DMAREQSEL0.bREG.SEL |= 0x0C000000; // Switch DMA of UART0 to UART4

    dbg("₩n₩n%s: dev->id = %d ₩n", __func__, dev->id);

    ddi_clk = clk_get(0, "lcdc0");
    mali_clk = clk_get(0, "mali_clk");

    mem = platform_get_resource(dev, IORESOURCE_MEM, 0);
        if (!mem) {
                dev_err(&dev->dev, "[UART%d] no memory resource?₩n", dev->id);
                return -EINVAL;
        }
        irq = platform_get_resource(dev, IORESOURCE_IRQ, 0);
        if (!irq) {
                dev_err(&dev->dev, "[UART%d] no irq resource?₩n", dev->id);
                return -ENODEV;
        }
```

Codes in red square switch DMA of UART0 to UART4.