Rekenaarstelsels 245 - Prakties 8
Computer Systems 245 - Practical 8

2015-10-02

1 Inleiding tot UART

In hierdie prakties ondersoek ons die UART (universal asynchronous receiver/transmitter) van die RL78 mikroverwerker. 'n UART is 'n hardeware-eenheid wat in die mikroverwerker ingebou is en die doel het om parallelle data om te skakel na 'n seriële stroom van bisse om oor 'n kabel te stuur. Dit gebruik 'n eenvoudige, maar bekende en konfigureerbare protokol hiervoor.

'n Enkele UART boodskap het een beginbis, 5 tot 9 databisse (gewoonlik 8), 'n opsionele pariteitsbis, en 'n stopbis. Tydens ons toetse sal die boodskappe 8 databisse en geen pariteitsbis hê nie.

Moet nie deurmekaar raak tussen UART en RS232 nie. RS232 is 'n volledige kommunikasiestandaard. 'n Volledige RS232 toepassing het 8 datalyne (9 insluitend grond) en gebruik -12 en +12 volt om ene en nulle voor te stel. RS232 het ook ekstra lyne om die vloei van data te beheer. Soms het UARTs ook 'n paar van die vloeibeheerlyne.

2 Implementasie on the RL87

Op die demobordjie word die UART na USB vertaling hanteer deur 'n udp78F073MC, wat ook 'n Renesas mikroverwerker is. Dit is opgestel om beide USB-UART vertaling te doen, sowel as vir die programmering van die RL78. Die UART-lyne is aan die mikroverwerker se pen 42 (RxD) en 43 (TxD)[1] verbind. Intern aan die RL78 word dit as poort P14/RxD2 en P13/TxD2 gesien. Dit beteken dus ons USB-UART is op UART2 van die RL78 gekoppel (daar is meerdere UARTs).

In die RL78 word dieselfde hardeware-eenhede gebruik om UART, SPI, I^2C en ander seriële kommunikasie te hanteer. Ons moet dus in sagteware vir hierdie eenhede sê watter tipe kommunikasie ons wil doen.

[1] Extract - YRPBRL78G13 schematic.pdf

1 Introduction to UART

In this practical you will be testing the capabilities of the UART (universal asynchronous receiver/transmitter) on the RL78 microcontroller. A UART is a piece of hardware built into the microcontroller that converts parallel data to a serial stream for transmission over a cable. It uses a simple but standard configurable protocol.

A single UART transmission has a start bit, 5-9 databits (usually 8), an optional parity bit, and a stop bit. In all of our applications the transmission will have 8 bits, and no parity bit.

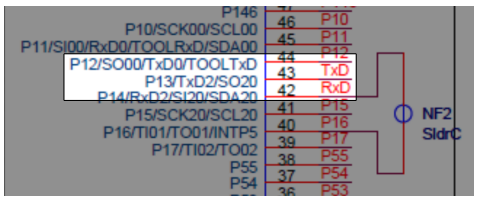
Do not confuse UART with RS232. RS232 is a complete communication standard. A full RS232 implementation has 8 lines (9 including ground) and uses -12v and 12v for voltage levels. RS232 also has additional lines for flow control, such as RTS, CTS. You will also come across UART implementations that have a few of the flow control lines.

2 Implementation on the RL87

On the development board the USB to UART conversion is handled by a udp78F073MC, which is another Renesas microcontroller. It is programmed to do both USB-UART conversion, as well as programming the RL78. The UART lines are connected to pins 42 (RxD) and 43 (TxD)[1]. Internally these are represented as pins P14/RxD2 and P13/TxD2. This means that UART2 is used to communicate via the USB-UART with the PC (The microcontroller has multiple UARTs).

In the RL78 the same hardware blocks are used for UART, SPI, I^2C and other serial communication. In our program we need to tell the hardware which type of communication we want to do.

[1] Extract - YRPBRL78G13 schematic.pdf



3 Programmeer die bord

Laai die projek van learn.sun.ac.za af en maak dit oop in e2 studio, deur 'n nuwe projek in te voer.

File ⇒ Import ⇒ General ⇒ Existing project into workspace

Onder *project explorer*, regskliek die projek, gaan na **Build configuration** en maak seker **Hardware Debug** is gekies.

Onthou dat brugskakelaar J6 tot J9 in die 1-2 (OCD) posisie moet wees om die demobord te programmeer, en in die 2-3 posisie om die UART te kan gebruik.

Bou die program en programeer dit na die demobord. Om dit te programeer moet jy *debug* modus begin, en weer stop. Skuif die brugskakelaars na die *Virtual UART* posisie, en maak Realterm oop. Kies die korrekte poort en 'n simbooltempo van 19200, en kliek 'Change' in Realterm.

Tik in die boonste teksveld. Die demobord behoort die karakter te ontvang, 1 by die waarde te tel en dan terug te stuur. "A" raak dus "B", en so aan. Let op dat die karakters wat jy tik nie in die teksveld sal verskyn nie, maar word direk na die bord gestuur. Die teksveld wys net die karakters wat vanaf die UART ontvang is.

Gebruik 'n stukkie draad en plaas dit deur gaatjie 6 (TxD) van J1[2], en draai dit dat dit vas sit. Doen dieselfde vir gaatjie 1 (grond). Verbind nou die ossilloskoop aan hierdie twee draadjies. Maak seker dat die ossilloskoop op "DC" gestel is.

Stel die ossilloskoop om te sneller op die UART boodskap. Gebruik die ossilloskoop om die spanningsvlakke en die simbooltempo van die UART te meet. Neem 'n skermkiekie van die golfvorm op die ossilloskoop. Plaas dit in 'n PDF dokument, saam met die waardes van die spanningsvlakke en simbooltempo wat jy gemeet het. Handig die PDF in op `learn.sun.ac.za` voor **Maandag 5 Oktober om 23:55**. **Antwoord ook die vrae** op `learn.sun.ac.za` voor die selfde tyd.

[2]Extract - YRPBRL78G13 schematic.pdf

3 Program the board

Download the project from learn.sun.ac.za and open it in e2 studio by importing a new project.

In the project explorer, right click on the project, go to **Build configuration**, and make sure that **Hardware Debug** is selected.

Remeber that the jumpers J6 to J9 need to be in position 1-2 (OCD) to program the demoboard, and in position 2-3 to use the UART.

Build the program and program it on to the demo board. To program it, you need to enter debug mode, and the quit again. Move the jumpers to the Virtual UART position and open realterm. Select the correct port and a baudrate of 19200, and click 'Change' in Realterm.

Type in the text box. The demo board should receive the character, add one to the value, and transmit it back to you. "A" becomes "B", etc. Note that input you type in the text box will not appear there, but will be transmitted directly to the board. The text box only shows characters received from the UART.

Use a piece of wire and connect it through hole 6 (TxD) of J1[2]. Do the same for hole 1 (Ground). Connect the oscilloscope between these two wires. Make sure the oscilloscope is set to DC.

Set the oscilloscope to trigger on the UART transmission. Use the oscilloscope to measure the voltage levels and the baud rate of the UART. Take a screenshot of the wave form. Put the screenshot, along with the values you measured for the voltage levels and baudrate into a PDF document. Hand it in on `learn.sun.ac.za` before **Monday 5 October at 23:55**. Also answer the questions on `learn.sun.ac.za` before the same deadline.

[2]Extract - YRPBRL78G13 schematic.pdf

4 Verander die simbooltempo

Die bronkode lêer r_main.c is die hoof lêer waarin die program geskryf word. r_cg_serial.h bevat die definisies van 'n aantal konstantes waarmee die UART opgestel word. En r_cg_serial.c bevat die kode wat die UART opstel en die funksies waarmee ons lees en skryf na die UART.

In die Hardeware Handleiding word daar in afdeling 12.6.4, bladsy 614, 'n formule verskaf waarmee die simbooltempo (baud rate) van die UART bepaal kan word. Dit is as volg:

(Baud rate) = Operation clock (F_{MCK}) frequency of target channel ÷ (SDRmn[15:9] + 1) ÷ 2 [bps]

Die simbooltempo van die UART word dus bepaal deur die waarde van die hoogste 7 bisse in die SDR-registers, so wel as die frekwensie van die F_{MCK} klok. Ons kan in r_cg_serial.c, lyn 96 en lyn 103 sien die SDR-registers word gelyk gestel aan die `_9A00_UART2_RECEIVE_DIVISOR` en die `_9A00_UART2_TRANSMIT_DIVISOR` konstantes.

In r_cg_serial.h sien ons die volgende op lyn 381 en 382:

```
#define _9A00_UART2_RECEIVE_DIVISOR      (0x9A00U)
#define _9A00_UART2_TRANSMIT_DIVISOR     (0x9A00U)
```

Die hoogste 7 bisse van die getal word gebruik om die simbooltempo vir die UART te bepaal. `0x9A00U` beteken `9A00` in hex, geïnterpreteer as 'n ongetekende integer. Dit vertaal na `10011010 00000000` in binêr. Die hoogste 7 bisse is `1001101`, en is 77 in desimaal.

Die UART gebruik nie die hoof verwerkerklok nie, maar meng (deel) eerder die klok af om 'n laer frekwensie klok (F_{MCK}) op te wek. Dit is omdat die UART se snelheid baie stadiger is as die CPU se klokfrekwensie. Die deler word opgestel deur 'n waarde te skryf na die SPS register. Die moontlike waardes wat in die regsiter gestoor kan word, word in table 12-4 op bladsy 615 van die hardeware handleiding gewys. Die skryf van die gepaste waarde na die SPS register gebeur in ons program op lyn 67 van r_cg_serial.c:

```
SPS1 = _0002_SAU_CK00_FCLK_2 | _0020_SAU_CK01_FCLK_2;
```

Die konstantes wat hier gebruik word, word in serial.h gedefiniëer op lyn 46 en lyn 63.

```
#define _0002_SAU_CK00_FCLK_2      (0x0002U)
#define _0020_SAU_CK01_FCLK_2      (0x0020U)
```

`0x0002U` vertaal na `0000 0010` in binêr. En `0x0020U` is gelyk aan `0010 0000` in binêr. Volgens tabel 12-4 op bladsy 615 van die hardeware handleiding deel ons dus met 2^2, of 4.

4 Change the baud rate

The file r_main.c is the main source file in which the program is written. r_cg_serial.h contains some definitions of constants used for configuring the UART. And r_cg_serial.c contains the code that does the setup of the UART, as well as the funcktions used to read and write to the UART.

In the Hardware Manual, section 12.6.4, page 614, a formula is given to calculate the baudrate at which the UART operates. The formula is as follows:

The UART's baud rate is thus set by the value of the highest 7 bits in the SDR registers, as well as the frequency of the F_{MCK} clock. On line 96 and line 102 of r_cg_serial.c we can see the SDR registers are set to the values of the `_9A00_UART2_RECEIVE_DIVISOR` and `_9A00_UART2_TRANSMIT_DIVISOR` constants.

We see the following on line 381 and 382 of r_cg_serial.h:

The highest 7 bits of these numbers are used to determine the baudrate for the UART. `0x9A00U` means `9A00` in hex, interpreted as an unsigned integer. In binary this is `10011010 00000000`. The highest 7 bits are `1001101`, and is 77 in decimal.

The UART does not use the main CPU clock directly, but instead mixes it down (divides) to generate a clock of a lower frequency (F_{MCK}). This is because the UART's speed is much slower than the CPU's clock frequency. The possible values we can divide with is given in table 12-4 on page 615 of the Hardware Manual. Our program writes the chosen value to the SPS register on line 67 of r_cg_serial.c:

The constants that are used here are defined in serial.h on line 46 and line 63.

`0x0002U` is `0000 0010` in binary. And `0x0020U` is `0010 0000` in binary. According to table 12-4 on page 615 of the hardware manual, we divide by 2^2, or 4.

F_{CLK} is vir hierdie projek in E2studio as 12MHz opgestel. F_{MCK} word dus $12MHz \div 4 = 3MHz$.

As ons die simbooltempo formule nou toepas kom ons uit op:

F_{CLK} is set up to be 12MHz for this project in E2Studio. F_{MCK} thus becomes $12MHz \div 4 = 3MHz$.

Solving the baud rate formula, we get:

Baud rate = Operation clock (F_{MCK}) frequency of target channel ÷ (SDRmn[15:9] + 1) ÷ 2 [bps]
Baud rate = $F_{MCK} \div (77 + 1) \div 2$
Baud rate = 19230 bps

Gewoonlik sal 'n mens die formule toepas om die delers te bereken, en dus die waarde wat in die SPS en SDR registers gestoor moet word. Vir 'n simbooltempo van 19200bps sal die SDR registers 'n waarde van 77.125 moet hê.

Aangesien ons nie 'n desimale getal in 'n 8-bis register kan stoor nie moet ons die naaste heelgetal gebruik. Daarom kom daar 'n fout voor tussen die teoretiese simbooltempo en die werklike waarmee die mikroverwerker funksioneer. In ons geval 19230 in plaas van 19200.

Normally you would apply the formula in the reverse direction to calculate the values of the dividers. Thus the values to store in the SPS and SDR registers. For a baud rate of 19200 we calculate a value of 77.125 for the SDR registers.

Because we can not store a decimal value in a 8-bit register we need to use the closest integer. This causes an error between the theoretical and real baud rate at which the microcontroller functions. In our case 19230 rather than 19200.

5 Opdrag

1. Gaan die waarde van `_9A00_UART2_RECEIVE_DIVISOR` en `_9A00_UART2_TRANSMIT_DIVISOR` na en bereken die persentasie fout wat in die simbooltempo voorkom. Gebruik die formules wat in die Hardeware Handleiding verskaf word in afdeling 12.6.4, bladsy 614.

2. Bereken die SDR-registerwaarde en persentasie fout vir die volgende simbooltempo's. Dui dit duidelik aan as jy 'n ander CPU klok deler (SPS) gebruik.

   a) 19200

   b) 38600

   c) 115200

3. Bereken die maksimum en minimum moontlike SDR-registerwaardes waarmee die bordjie steeds met 'n ander toestel wat teen 'n simbooltempo van 19200 praat, kan kommunikeer.

4. Voeg kode by in die main.c lêer om die LED te skakel (toggle) elke keer as die 'l'-karakter ontvang word.

5. Voeg kode by in die main.c lêer wat die boodskap "Hello" stuur elke keer as die 'h'-karakter ontvang word.

Sit jou berekeninge en kode in 'n PDF dokument en handing dit in op `learn.sun.ac.za` voor 23:55 op **Donderdag 9 Oktober.**

5 Assignment

1. Check the value of `_9A00_UART2_RECEIVE_DIVISOR` and `_9A00_UART2_TRANSMIT_DIVISOR` and calculate the percentage error in baud rate. Use the formulas given in the hardware manual in section 12.6.4, page 614.

2. For the following baud rates, calculate the SDR register value, as well as the percentage error at that value. If you use a different CPU clock divisor (SPS), indicate it clearly.

   a) 19200

   b) 38600

   c) 115200

3. Calculate the maximum and minimum possible SDR register values at which the board would still successfully communicate to another device, that is set up to communicate at a baud rate of 19200.

4. Add code to the main file to toggle the LED every time the 'l' character is received.

5. Add code to the main file that sends the word "Hello" every time the 'h' character is received.

Put your calculations and code in a PDF document, and hand it in on learn.sun.ac.za before 23:55 on **Thursday 8 October.**