Rekenaarstelsels 245 - Prakties 9
Computer Systems 245 - Practical 9

2015-10-09

1 Kodegenerator

In hierdie prakties gebruik ons die *Peripheral Code Generator* om kode te genereer vir die randapparatuur op die mikroverwerker.

Begin deur 'n projek in e2 studio te skep vir die demobord, maar hierdie keer kies die **Peripheral code Generator** opsie. Dit voeg die *Code Generator* opsie by die projek in die *Project Explorer* nadat die projek geskep is.

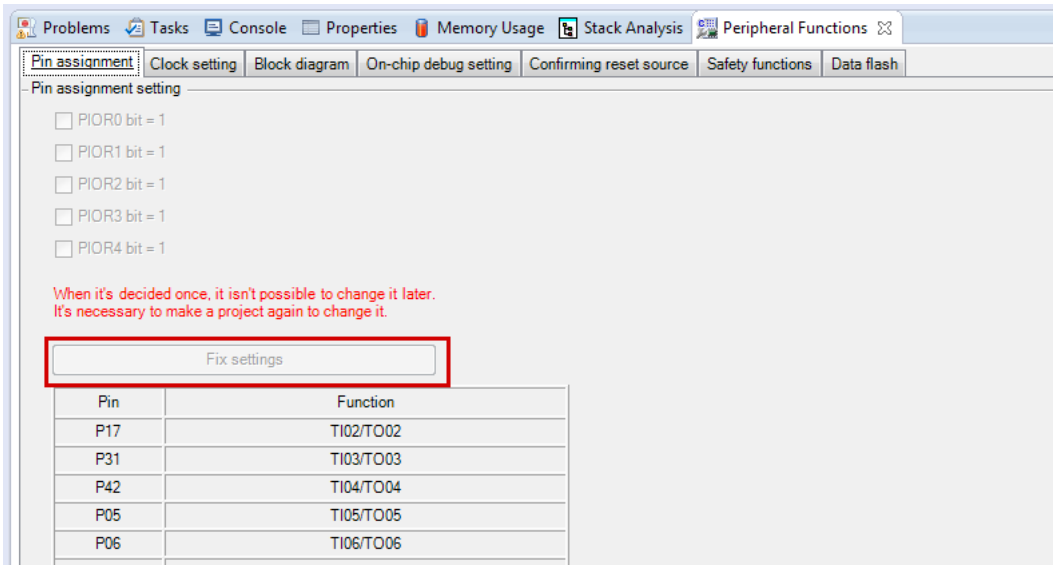
In die *Project Explorer*, maak *Code Generator* ⇒ *Peripheral functions* oop en dubbelkliek *Clock generator*. 'n Nuwe venster sal oopmaak in een van e2 studio se panele. Op die *Pin assignment*-oortjie, kliek die **Fix settings** knoppie. Die mikroverwerker kan sommige penne se gebruike verander, maar die kodegenerator kan dit net een keer per projek doen.

1 Code Generator

In this practical you will be using the Peripheral Code Generator to generate code for the peripheral functions of the microcontroller.

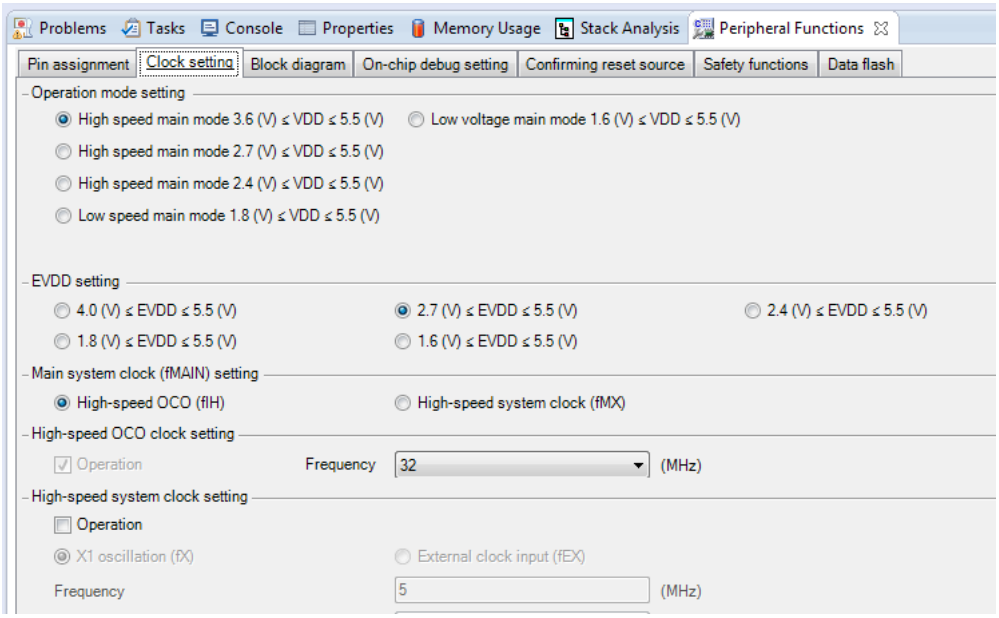
Start by creating a normal project in e2 studio for the demo board, but this time, select the **Peripheral code Generator** in the project creation wizard. This will add the Code Generator option in the Project Explorer once the project has been created.

In the Project Explorer, open the Code Generator ⇒ Peripheral functions, and double click the Clock generator. A window will open in one of e2 studio's panes. In the first tab (Pin assignment) click the button **Fix settings**. The microcontroller (MCU) has support to remap some of the pins, but the Code Generator only supports this in a new project.



Gaan na die **Clock Settings**-oortjie en verander die opsies soorgelyk aan die beeld onder. Die hoëspoed aanboord klok, OCO, moet gekies word en opgestel word om teen 32MHz te ossilleer.

At the **Clock Setting** tab, set the options to match the screenshot below. The High-speed on-chip oscillator, OCO, clock needs to be selected and enabled, and set to 32MHz.

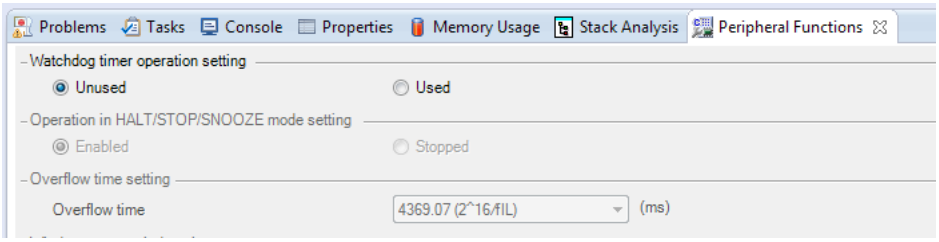


In *Project Explorer*, gaan na *Code Generator* ⇒ *Peripheral functions* ⇒ *Watchdog timer* en skakel die waghond funksie af.

'n Waghondteller is 'n hardeware eenheid wat die mikroverwerker herstel indien dit nie binne 'n vasgestelde periode rustend gamaak word nie. Dit is 'n handige funksie om die mikroverwerker te laat herbegin indien die sagteware vasloop.

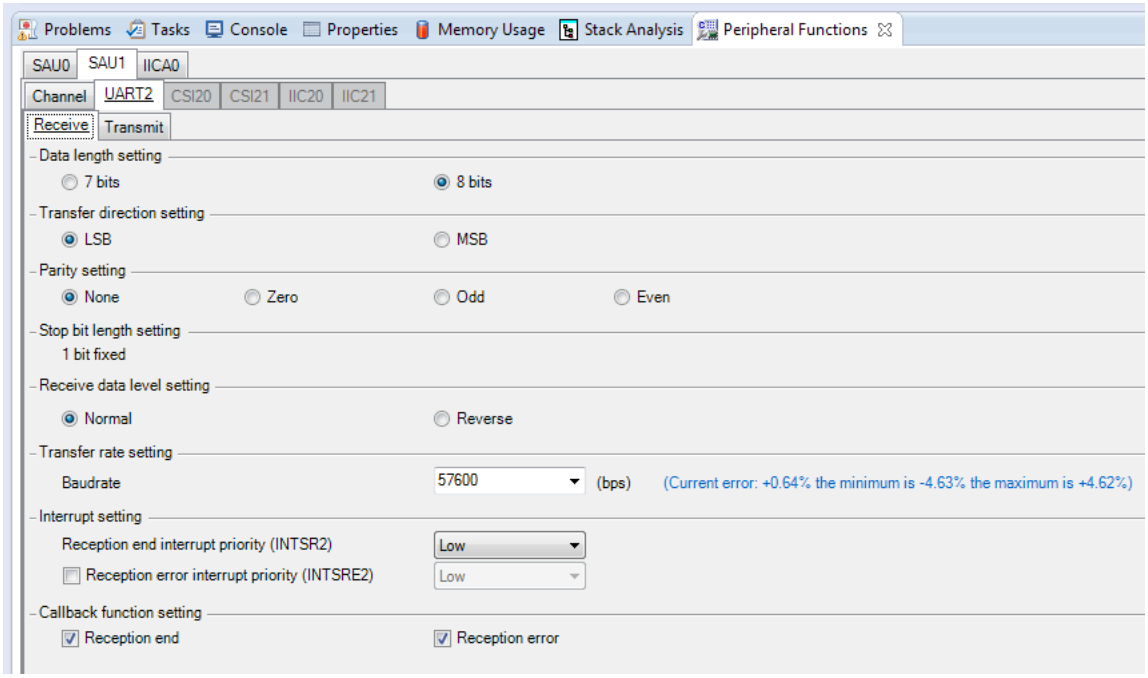
In the *Project Explorer*, go to *Code Generator* ⇒ *Peripheral functions* ⇒ *Watchdog timer* and disable the watchdog timer.

A watchdog timer is a timer that resets the MCU if it has not received input in a certain amount of time. It is useful to let the MCU automatically recover from software crashes.



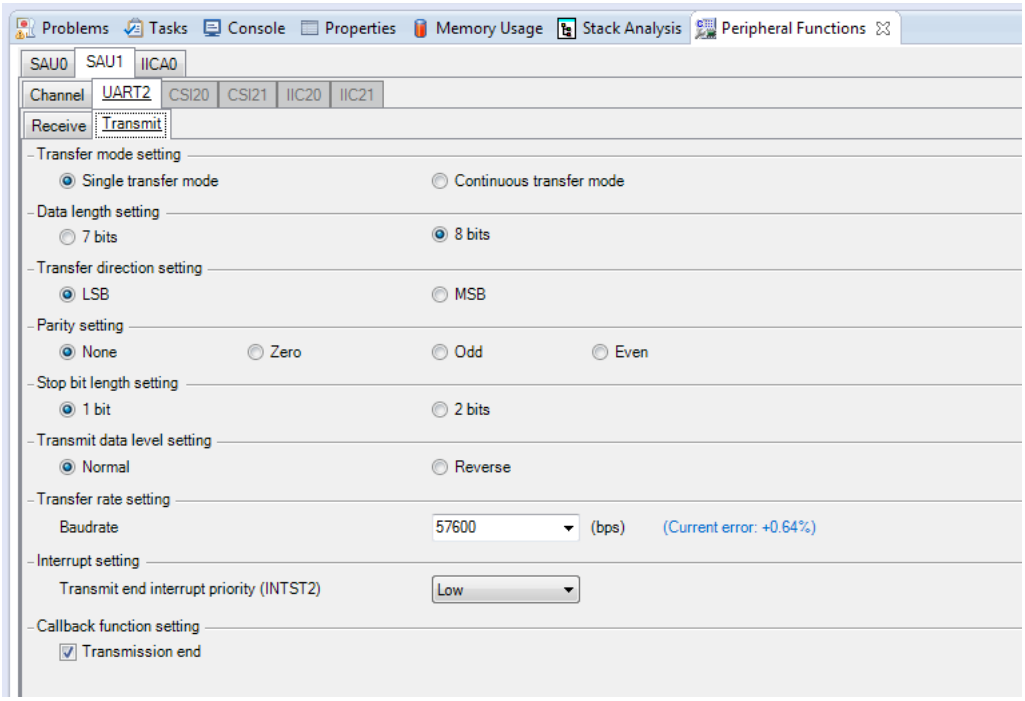
Gaan na *Code Generator* ⇒ *Peripheral functions* ⇒ *Serial*. Kies die oortjies *SAU1*⇒*Channel*, en stel kanaal 0 op vir UART2, stuur/ontvang funksie. Gaan ook die opstelling onder die *UART2*- en *Receive*-oortjies na volgens die skermbeelde hieronder. Indien jy nie die korrekte simbooltempo kan kies nie, tik dit in.

Open the *Serial* options from *Code Generator* ⇒ *Peripheral functions*. Go to the *SAU1* (serial array unit 1) tab, and further to the *Channel* tab, and select UART2 for channel 0, with Transmit/Receive function. Also verify at the *UART2* tab and *Receive* tab that the settings match the screenshot. If you cannot select the baud rate, you can type it in.



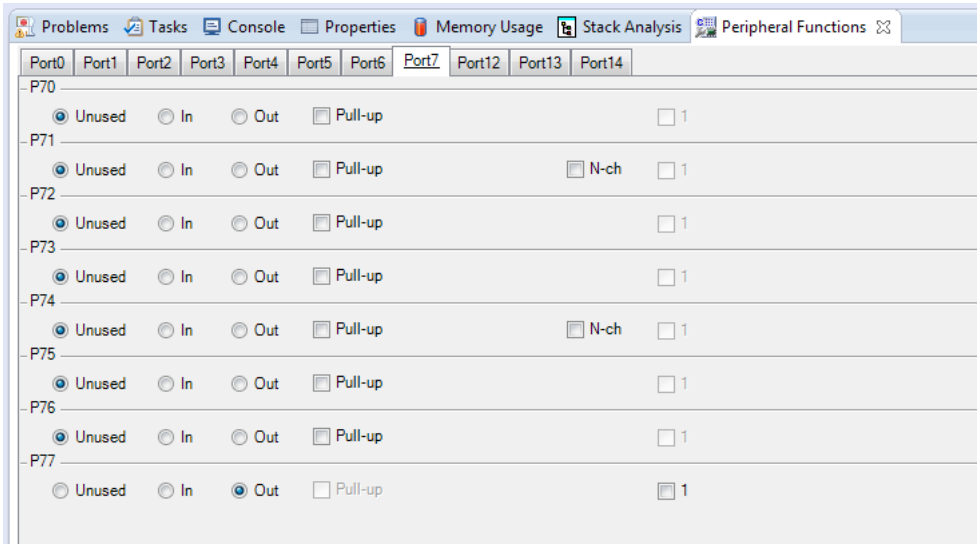
Gaan ook na die *Transmit*-oortjie en stel op soos onder.

Go to the *Transmit* tab and set up as screenshot below.



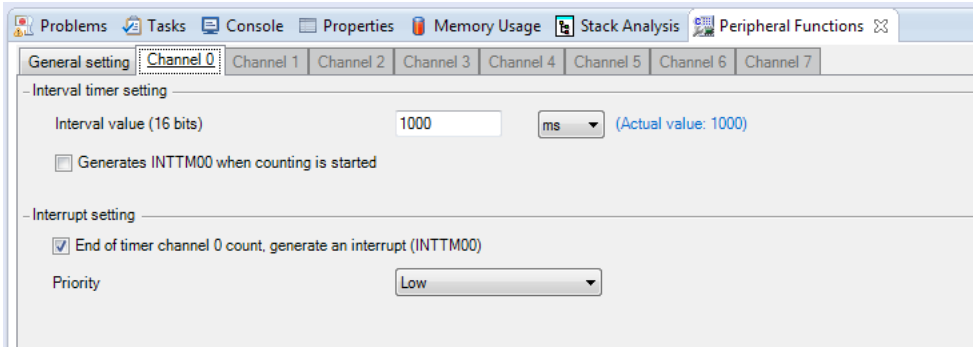
Maak die poortopsies oop by *Code Generator* ⇒ *Peripheral functions* ⇒ *Ports* en stel pen 7 van poort 7 as 'n uittree. Hierdie pen word vir die LED gebruik.

Open the port options at *Code Generator* ⇒ *Peripheral functions* ⇒ *Ports* and set pin 7 of Port 7 as an output. This pin is used for the LED.



Open die telleropsies by *Code Generator* ⇒ *Peripheral functions* ⇒ *Timers*. Stel kanaal 0 om 'n intervalteller te wees. Gaan na kanaal 0 se oortjie en stel die interval op 1000 millisekondes.

Open the Timer options at *Code Generator* ⇒ *Peripheral functions* ⇒ *Timers*. Set Channel 0 to be an interval timer. Go to the Channel 0 tab, set the interval value to 1000 milliseconds (one second).



Kliek nou op die **Generate Code** knop regs bo in die kodegenerator se *Peripheral functions*-venster. 'n Stel nuwe bronkodelêers word geskep wat jou oorspronklike `jou_projek_naam.c`-lêer vervang. Maak die `r_main.c`-lêer oop en let op die volgende kommentaarstellings:

Now click the **Generate Code** button in the top-right corner of the *Peripheral functions* tab of the code generator page. A set of new source files are generated in your `src` directory, replacing your original `project_name.c` file. In the `r_main.c` file you will notice many comments such as the following:

```
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Orals waar hierdie kommentaarstellings voorkom kan jy jou eie kode daartussen skryf, sonder dat die kodegenerator dit in die toekoms sal oorskryf.

You can place your code between these comments. This way your code will be preserved if you decide to re-generate the code with different settings.

2 Kode Opstelling

Die kodegenerator skep 'n hele aantal handige funksies. Maak r_cg_serial.h oop en gaan na die einde van die lêer. Funkies handig vir ons is:

```
void R_SAU1_Create(void);
void R_UART2_Start(void);
void R_UART2_Stop(void);
MD_STATUS R_UART2_Send(uint8_t * const tx_buf, uint16_t tx_num);
MD_STATUS R_UART2_Receive(uint8_t * const rx_buf, uint16_t rx_num);
static void r_uart2_callback_receiveend(void);
static void r_uart2_callback_sendend(void);
```

Die eerste 5 van hulle is in r_cg_serial.c gedefiniëer en kan nie verander word nie. Die twee *callback*-funksies is in r_cg_serial_user.c, en kan deur jou verander word. Die funksies R_UART2_Send en R_UART2_Receive neem beide twee parameters: 'n buffer vir data en 'n getal vir bufferlengte.

Beide hierdie funksies neem die parameters wat vir hulle oorgestuur word, stoor die waardes en stel die UART op om met onderbrekings te werk. Daarna word daar teruggekeer na jou program. Wanneer die UART klaar is met die stuur van 'n enkele karakter word 'n onderbreking veroorsaak en die volgende karakter word gestuur. As die hele buffer gestuur is (tot die lengte soos gegee deur die tweede parameter), word die *callback*-funksie geroep. Dit beteken stuur en ontvangs word in die agtergrond deur onderbrekings hanteer en jou kode hoef nie te wag vir dit om klaar te maak nie. **Om te weet of die stuur en ontvangs klaar is moet jy jou eie vlaggies definieer en in die *callback*-funksie stel.**

In r_cg_timer.h sal jy die volgende funksies sien:

```
void R_TAU0_Create(void);
void R_TAU0_Channel0_Start(void);
void R_TAU0_Channel0_Stop(void);
```

Daar is nog 'n *callback*-funksie in r_cg_timer_user.c. Dit word geroep elke keer as die teller 'n onderbreking veroorsaak, en jy kan jou eie kode daarin skryf.

2 Code Setup

The generated code produces various useful functions. Open the r_cg_serial.h and scroll to the bottom of the file. Functions useful to us are:

The top 5 functions are all in r_cg_serial.c, and cannot be edited. The two callback functions (called by the interrupt) are in r_cg_serial_user.c, and can be edited by you, the user. The functions R_UART2_Send and R_UART2_Receive both take two parameters: a buffer for data and a number for buffer length.

Both these functions take the parameters sent to them, store their values and set up the UART to work with interrupts. They then return control to the user program. When the UART completes a single transmission an interrupt function is called and the next character is sent. When the entire buffer is transmitted (as specified by the second parameter) the callback function is called. This means that when the transmit or receive function is called, it is quickly done, and the code can continue. There is no built-in indication of when a transmission or receive is completed. **You need to implement you own flag to check when the transmission or reception is done, by adding code to the callback function.**

In the r_cg_timer.h file you will find these functions:

There is another function in r_cg_timer_user.c, which is the callback function for the timer, and can be edited. This callback is called when the timer causes an interrupt.

3 Opdrag

Skryf 'n program wat luister vir instruksies op die UART. Elke keer as 'n karakter ontvang word moet dit gestoor word, behalwe as een van die volgende karakters ontvang word:

- As die karakter 'l' ontvang word moet die toestand van die LED verander word.
- As die karakter 't' ontvang word moet, óf die vorige gestoorde karakter elke sekonde op die UART uitgestuur word, óf dit moet ophou om die karakter elke sekonde te stuur.

Onthou om jou onderbrekingsfunksies so kort as moontlik te hou. Stel vlaggies en veranderlikes in die onderbrekings en reageer dan daarop in jou hooflus.

Inhandiging 1 Teen vanmiddag 17:00 moet jy ten minste die stappe gevolg het om die kodegenerator te gebruik soos beskryf word in afdeling 1. Gee jou hele projek as 'n zip-lêer in op `learn.sun.ac.za`.

Inhandiging 2 Handig jou projek, met die kode om die bogenoemde aksies te doen, in op `learn.sun.ac.za` voor 23:55 op Donderdag 15 Oktober. Weereens moet dit 'n zip-lêer wees met die hele projek daarin.

3 Instructions

Write a program that listens for instructions on the UART. Every time a character is received the program should store it in a variable, except when one of the following characters are received:

- If an 'l' is received, the board should toggle the LED.
- If a 't' is received, the UART should either start to transmit the stored character back to the computer every second, or stop transmitting the character back every second.

Remember to keep your interrupt/callback functions short. Only set flags and store values in the interrupt, then do all the processing in the main loop.

Submission 1 Today you should at least generate code as described in section 1. Hand in the project with the generated code on `learn.sun.ac.za` before today at 17:00. Your submission should be a zip-file of the entire project.

Submission 2 Hand in the project containing your code to do the above described actions on `learn.sun.ac.za` before 23:55 on Thursday 15 October. Your submission should again be a zip-file of the entire project.