OSEK Assignment Report

Output size command

Text	Data	bss	dec	hex
7062	394	372	7828	1e94

Algorithmic

The task used are:

- **newSentenceTask**: to select the message to send each 180 s at the beginning and 180,5 s next.
- **codTask:** to code each letter in the corresponding symbol of morse code (. or -).
- **translateTask:** to translate each symbol of morse code in the right sequence of bit (0 or 1).
- **prinTask:** to display each bit translate.

The alarm used are:

- **alarmToPrint:** to activate prinTask each 100 ms
- **alarmToSentence:** to activate newSentenceTask, first time after 180s, then after 180,5 s.

The counter used is:

- **SystemCounter:** reach the max counter after about 180 s and increment by one each 1024 µs

newSentenceTask

It's activated the first time after 180 s and then after 180,5 s by mean of the alarm **alarmToSentence**. Inside this task first is stopped the **alarmToPrint**, that it is reprogrammed to start again after 0,5 s and repeat each 0,1 s. Then it selects the next message, if there is no more it choose it start again from first message. All messages are saved in the FLASH memory because they are only read.

codTask

Read the letter that must be coded and take the corresponding morse code from FLASH memory, where there are all coded letters saved in alphabetic order and the task can access directly through a difference of ASCII code. In case of a space or end string is assigned a particular code (/) that is translate properly next.

translateTask

Read the symbol that must be translated and take the corresponding string of bits direct from FLASH memory, where all symbols are saved in alphabetic order and by difference of ASCII is possible access directly. In case of codeword (check terminator string) it need assign manually the string of bits to send.

prinTask

Display LED ON if bit considering is 1 or LED OFF if bit is 0. If nothing is displayed could be either prinTask has been called but system is in pause to send message or there is an interference. It starts at the beginning. When it starts it assumes always to send a 1 because each message start with a letter and each letter is translate with a 1 at position zero.

Observations

FLASH memory is used just to store data that are only readable and for this reason we store the information about message, morse code and translate symbol. Is better use FLASH than SDRAM because FLASH memory is 32 kB while SDRAM memory is 2 kB and it can be used also for writeable data.
Activating the DEBUG mode, uncommented the *#define DEBUG* in assignment.cpp (line 15) and the *LIBRARY = serial* in assignment.oil (line 25), is possible to see on serial monitor the send messages and the timing for each bit.
Activating the DURATIONTASK mode, uncommented the *#define DURATIONTASK* in assignment.cpp (line 16) and the *LIBRARY = serial* in assignment.oil (line 25), is possible to see on serial monitor the duration of each task. For use this mode duplicate the STACKSIZE of newSentenceTask, codTask and translateTask.
The codTask and prinTask are pre-emptive because a prinTask, that have a higher priority, could need send a bit. The priority of the newSentenceTask is the higher because it can stop all system.

Timing

The counter performs a tick every 1024 µs so is impossible do operation with high precision. The real time obtain is:

- the prinTask is activated each 100.352 ms, with an error of 0.352%
- the newSentenceTask starts after 179,999744 s, with an error of 0.000256 s
- each newSentenceTask starts after 180,499456 s, with an error of 0.000544 s
- message is sender again after 0,4999712 s, with an error of 0,0000288 s

Each task duration is:

- **newSentenceTask**: 96 µs always
- **codTask:** the worst case is 24 µs, the best case is 20 µs
- **translateTask:** 16 µs, sometimes 8 µs
- **prinTask:** the worst case is 44 µs, the best case is 8 µs

For the time error we do not respect the real time scheduling and happen that the newSentenceTask does not arrive at the same time of prinTask but before and, for this reason, is not required a control in prinTask to see if there is a pause but for safe reason is better do it.

This is a representation of meaning part of scheduling where we assume the right precision time.

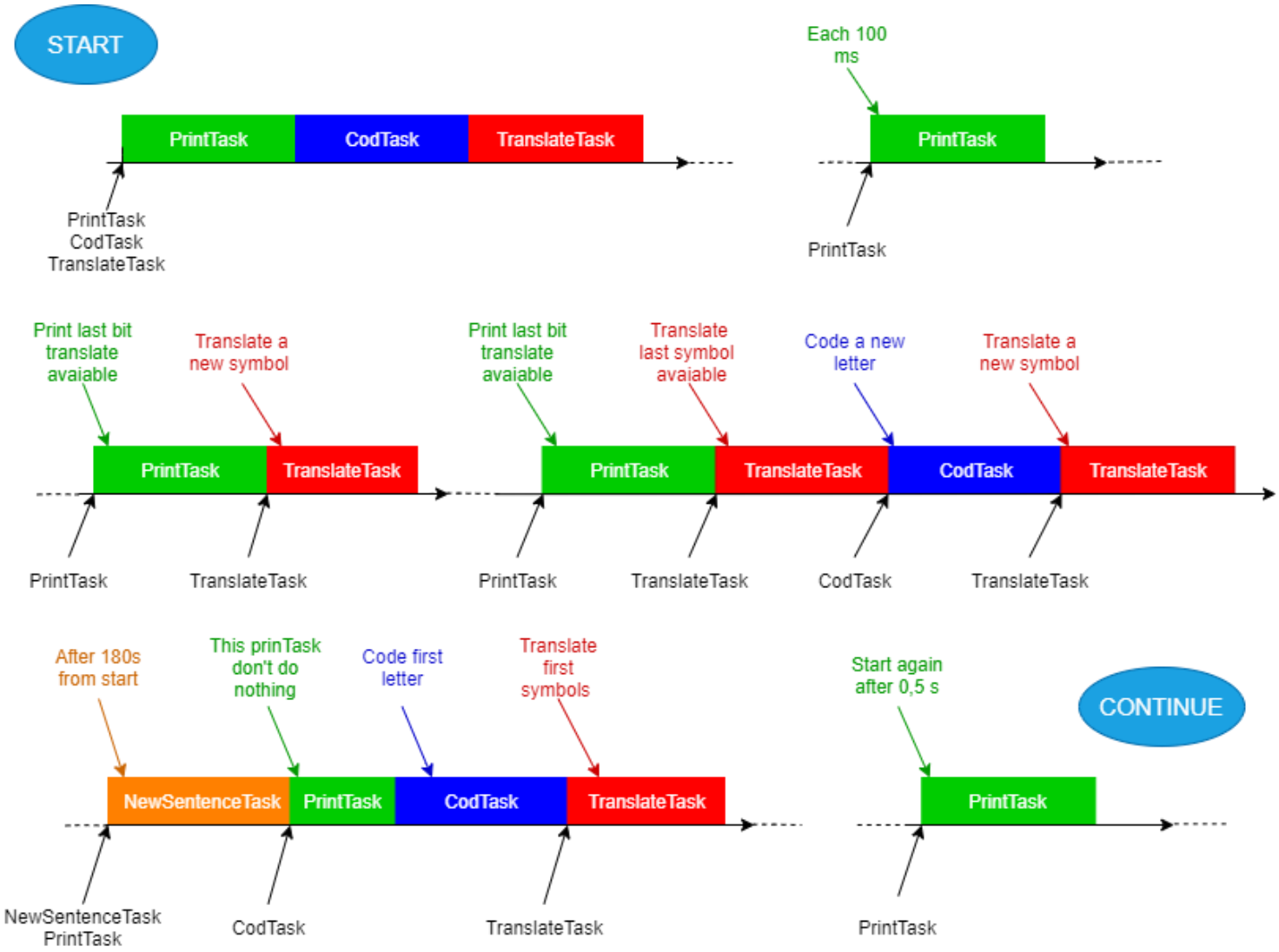

Figure 1: all possible scenario of scheduling