Assignment OSEK

Trampoline & Arduino

Giovanni Pollo

290136

1 Structure & Algorithm

The structure chosen is based on an extended task. There is an event, that is triggered every $100ms$, used to guarantee the timing of the system.

The conversion is done online, thanks to the global variable *LED*. The external loop is used to read all sentences, while the internal one is used to analyze every single letter. Every character is compared to 'A', and the value of *pos* is computed.

After having obtained *pos*, we can get the morse code of the considered letter and then convert it into a sequence of 0 and 1, that it is saved in the variable *LED* thanks to the **populateLED** function.

The $180s$ pause is implemented using a counter (variable *cnt*) that counts up to 1800. In fact:

$$max\_cnt\_value = \frac{pause\_time}{event\_time} = \frac{180\ s}{0.1\ s} = 1800 \tag{1}$$

The $0.5s$ pause is done in the same way, with the only difference that the value of the counter variable is 5.

2 Timing & Errors

2.1 Timing

As explained in the first paragraph, the code is based on a periodic alarm (every $100ms$) that activates an event. The only problem is that the Trampoline *SystemCounter* is the same as the *Systick* used in Arduino, that counts a tick every $1024\mu s$. To obtain $100ms$ period, the value assigned to **CYCLETIME** must be:

$$CYCLETIME = \frac{event\_time}{tick\_time} = \frac{100ms}{1024\mu s} = \frac{100 \cdot 10^{-3}s}{1024 \cdot 10^{-6}s}$$
$$= 97.65625 \approx 98$$

The choice for CYCLETIME is 98. We can now compute what is the real value for $100ms$:

$$real\_100ms = 98 \cdot 1024\mu s = 100.352ms$$

From this it is easy to evaluate the default error:

$$Default\_Error = \frac{real\_100ms - 100ms}{100ms} \tag{2}$$

$$Default\_Error = \frac{100.352ms - 100ms}{100ms} = 0.352\% \tag{3}$$

2.2 Errors

To analyze the errors of the program, I used the Arduino function **micros()**. We can identify 3 errors:

- $100ms$: For this error, I obtained 0.352%

$$Error = \frac{value\_with\_micros - ideal\_value}{ideal\_value}$$
$$Error = \frac{100352\mu s - 100000\mu s}{100000\mu s} = 0.352\%$$

- $500ms$: For this error, I obtained:

$$Error = \frac{value\_with\_micros - ideal\_value}{ideal\_value}$$
$$Error = \frac{501760\mu s - 500000\mu s}{500000\mu s} = 0.352\%$$

- $180s$: For this error, I obtained:

$$Error = \frac{value\_with\_micros - ideal\_value}{ideal\_value}$$
$$Error = \frac{180633600\mu s - 180000000\mu s}{180000000\mu s} = 0.352\%$$

It's easy to notice that all the errors are the same, and they are all coeherent with the *Default_Error* evalueted in section 2.1, with the equations 2 and 3

3 Memory Occupation

In order to analyze the memory occupation, I compared my solution with a blank code (an empty PeriodicTask triggered every $100ms$).

Text	**Data**	**Bss**	**Dec**
5730 Bytes	278 Bytes	382 Bytes	6390 Bytes

Table 1: Blank code memory occupation

Text	**Data**	**Bss**	**Dec**
5730 Bytes	278 Bytes	382 Bytes	6390 Bytes

Table 2: My solution memory occupation

By comparing Table 1 and table Table 2, it's easy to see the benefit of the online translation. If fact, because we translate letter by letter, the data occupation doesn't increase too much.