Pulse Width Modulation Controller Software

Kerim Kilic
De Haagse Hogeschool

Dennis van den Berg
De Haagse Hogeschool

Folkert Kevelam
De Haagse Hogeschool

***Abstract*—We present the high level design of the software which is being used to generate sinusoidal waves from the Universal Four Leg. The Universal Four Leg is a sophisticated piece of hardware which can be configured as a inverter. Using this software, the different parameters of the sinusoidal wave can be manipulated.**

I. Introduction

DC voltage have shown great promise in transfering power with less losses than AC voltage. AC voltage is being used in a lot of home appliances. In order to use the voltage of a DC grid for AC applications the Universal Four Leg inverter is being used. With the Universal Four Leg and the software which has been developed in the past months, it is now possible to power various AC Loads from DC grids. With the software which is being described in this paper, the various parameters of the AC output voltages can be manipulated, thus allowing to drive various loads.

Delft
December 27, 2019

II. Communication protocol

The communication protocol for both the User Interface as for the terminal version of the software is made up of different commands. Some commands require a channel, where the channel is the specific leg the user wants to manipulate it's parameters. The microcontroller always sends an acknowledgement message to the computer, which indicates if the command is accepted or rejected. If the command is accepted the microcontroller will send 'OK' as the acknowledgement message. If the command is rejected the microcontroller will send 'REJ' as the acknowledgement message.

A. User Interface

The communication between the microcontroller and user interface are made up of messages of 6 bytes each. The first byte represents the start byte which in our case is 0x24, which is also in ASCII equal to the Dollar sign ($). The second byte represents the command that is being sent over (e.g. the frequency). The third, fourth and fifth byte represent the value that is being sent over the USART communication. The last byte represents the CRC checksum. If the value is N/A, for instance in the case of the command start, then the bytes representing the value will have the value of 0xFF FF FF.

B. Terminal

In order for the software to also work with terminals like, the Arduino serial monitor or the serial plotter, the terminal communication uses string messages. The syntax for this version of the protocol is:
{Command}_{Channel} {Value}\r \n. Where between the brackets come the string representation of the command, channel and value (e.g. frequency_1 10).

III. Algorithms

Hier komt iets over het algoritme

IV. Embedded software architecture

A large part of the project pertains the control and generation of a PWM waveform described by user input. To that end, the embedded software must be able to perform its generation duties whilst recieving or outputting paramters/data. There are multiple ways of achieving said multitasking such as polling or interrupt-driven execution. Since the required hardware does not have the ability to prioritise specific interrupts, a pure interrup-driven approach was ruled out, instead, it was chosen to focus on a dual apporach. The routines with a deadline such as the waveform generation would work with interrupt while the routines without a deadline such as user interaction would work with a polling system.

A. Polling state

The interactivity between the user and device is implemented using serial communication. This means that beside controlling the board itselfs, the microcontroller needs to reserve time to get user input. As described above, the genereal approach used in this project is the hybrid interrupt-driven/polling approach. The waveform is generated and output using interrupts, whilst the user interaction is done using polling.

Polling consists of checking if a change has been made every once in a while. This is usually done in the main loop of the program. Every time the system loops a routine checks a value again and if nothing has changed, it continues.

There are 2 ways to poll changes without interrupts, using blocking or non-blocking routines. When a polling routine is the only routine that needs to poll, blocking routines are very easy to use and reliable, however, as soon as multiple routines need to poll, a single routine can't block the microcontroller. This means that more complex projects need to have non-blocking routines. A computer implements a sort of hybrid between blocking and non-blocking. It is possible to get a