Project – Arbitrary Waveform Generator

AWG overview. Functional description. PC interface. Software development.

Introduction

An arbitrary waveform generator (AWG) has the ability to generate a variety of preset waveform types as well as output a user-defined waveform. The aim of this project is to implement a 2-channel AWG.

AWG Overview

The AWG is a real-time system. It needs to deliver waveform amplitude information (a "sample") to a digital-to-analog converter at precise intervals. It is a major task to develop hardware for the PC and write real-time drivers for the Windows® operating system. It is relatively easy to develop a stand-alone real-time embedded system. We will therefore exploit the GUI of Windows® and the real-time capability of a microcontroller to develop the AWG, as shown below:

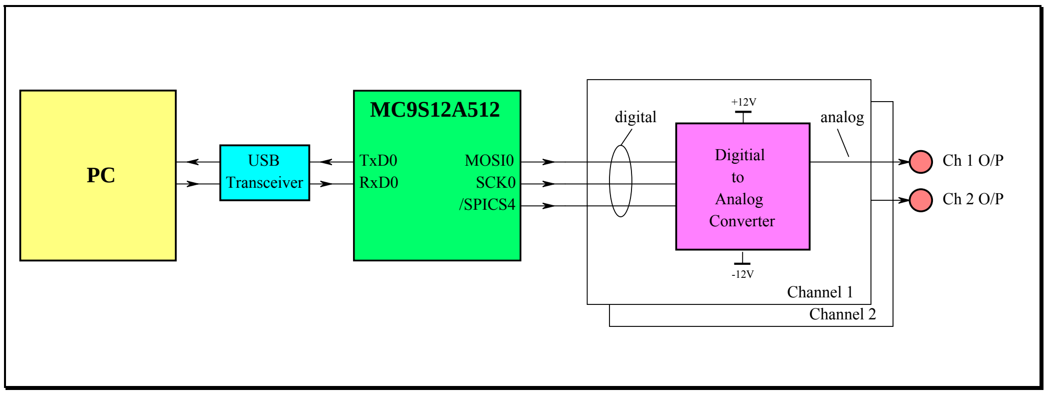


Figure P.1 – AWG Block Diagram

The PC is responsible for providing a user interface for the AWG.

A USB interface is the primary means of communication with the μC. One of the two available SCIs on the μC is used for this purpose. The operation of the AWG is controlled and monitored from a PC using the ModCon protocol.

The μC is responsible for communications with the PC and sending information about 2 independent waveforms to the digital-to-analog converter circuitry via a Serial Peripheral Interface (SPI).

The digital-to-analog converter circuitry converts the digital value into an analog signal, and outputs it to the external connector.

Functional Description

Waveform Timing

The time between samples is known as the sample period, and is usually denoted T_s. The AWG generates periodic waveforms, e.g. sine, square, triangle, etc. with a period denoted T_0. The AWG may store the sample values of a particular waveform over k periods of that waveform. An obvious requirement is that there must be an integral number of samples, denoted N, within these k periods, otherwise the waveform will not be periodic. That is, we must have:

$$\boxed{NT_s = kT_0} \tag{P.1}$$

It is generally not necessary (and it is sometimes undesirable) to restrict the value of k to one.

The value of the sample period, T_s, should be chosen so that there are at least 10 samples per period of waveform. The value of N is restricted by the available RAM in the μC.

You should think carefully how to implement precise waveform timing – e.g. an ISR.

Arbitrary Waveform

The arbitrary waveform, in CSV format, will be down loaded via the supplied PC interface. Arbitrary waveforms are to have 256 values, in the range 0 to 4095, corresponding to a full-scale output.

Functional Requirements

The AWG must meet the following requirements:

Specification	**Value**
Waveform	Sine, Square, Triangle, Sawtooth, Noise, Arbitrary
Frequency	0.1 Hz to 100 Hz in 0.1 Hz steps
Amplitude	0 to 10 V in 0.1 V steps
Offset	-10 V to 10 V in 0.1 V steps
Sample Period	1 ms

The μC needs to reliably deliver a sample value to the DAC every sample period. To do this, it needs to take into consideration any suspected interruptions such as PC communication and overhead such as SPI interfacing, etc. Some simple strategies to achieve this are to either:

- lookup a value held in a hard-coded (compile-time determined) array
- lookup a value held in a pre-calculated (runtime determined) array
- calculate a value using a simple formula or functions

Arithmetic Calculations

You may need floating-point calculations in your implementation. Be sure to commence a new project in CodeWarrior and specify the floating-point library during the New Project Wizard. You cannot build upon Lab 5, as it uses an integer arithmetic library.

PC Interface

The interface to the AWG is via a PC running an interface program under the Microsoft Windows® operating system.

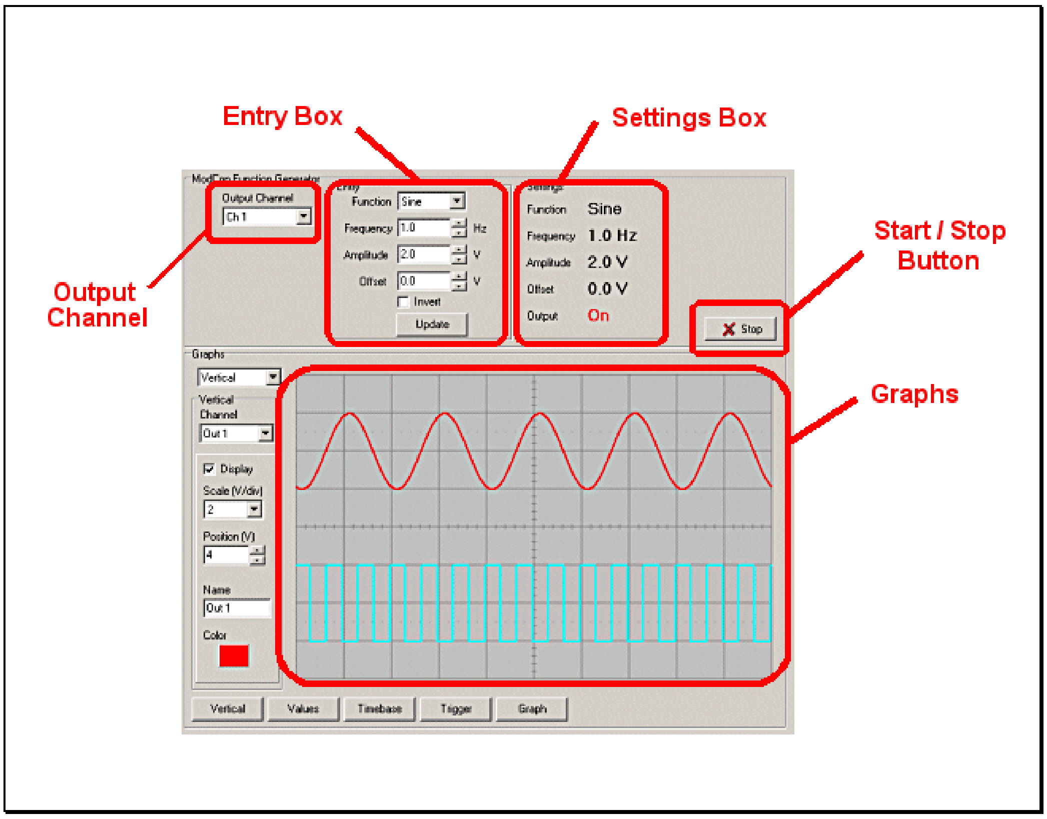
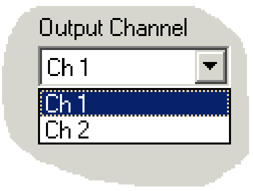


Figure P.2 – PC Interface

You may use the supplied ModCon PC Interface program to communicate with your AWG. In the µC you will need to implement the ModCon Protocol fully.

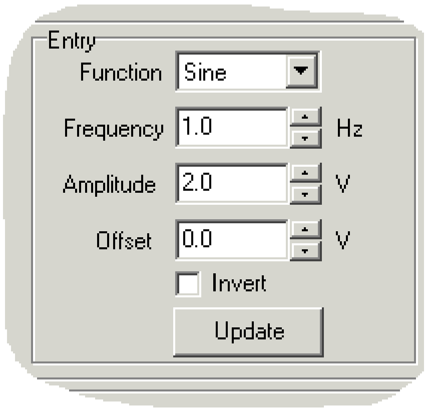
To implement the functionality of downloading an arbitrary waveform, you will need to expand the ModCon protocol accordingly, and use the "Manual packet" feature of the PC interface.

Output Channel



Use the output channel pull-down list to choose the currently active output channel of the function generator. The parameters of the selected channel will be updated in the Entry and Settings group boxes.

Entry


Entry
Function Sine
Frequency 1.0 Hz
Amplitude 2.0 V
Offset 0.0 V
Invert

The Entry group box reflects the current settings that will be applied to the currently selected output channel when you push the Update button. The parameters you can set are:

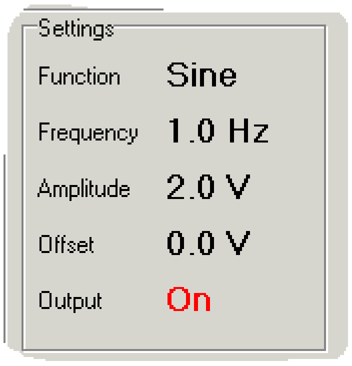
Function – choose between Sine, Square, Triangle, Sawtooth, Noise and Arbitrary.

Frequency - choose between 0.1 Hz and 100 Hz in 0.1 Hz steps.

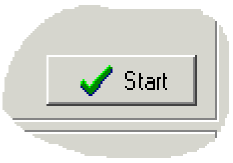
Amplitude – choose between 0 V and 10 V in 0.1 V steps.

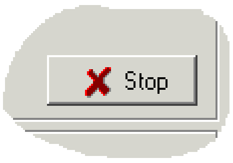
Offset – choose between -10 V and 10 V in 0.1 V steps.

Settings


Settings
Function Sine
Frequency 1.0 Hz
Amplitude 2.0 V
Offset 0.0 V
Output On

The Settings group box reflects the actual settings that have been applied to the currently selected output channel. The Settings also tell you whether the function generator output is on or off. The Start / Stop button in the bottom right-hand corner allows you to turn the function generator on or off.





Software Development

There are two approaches to take for this simple embedded system – use a foreground / background approach, or use a real-time operating system (RTOS). The advantage of the foreground / background approach, for simple systems, is that it is easy to implement. For more complex systems, a real-time operating system simplifies the software design.

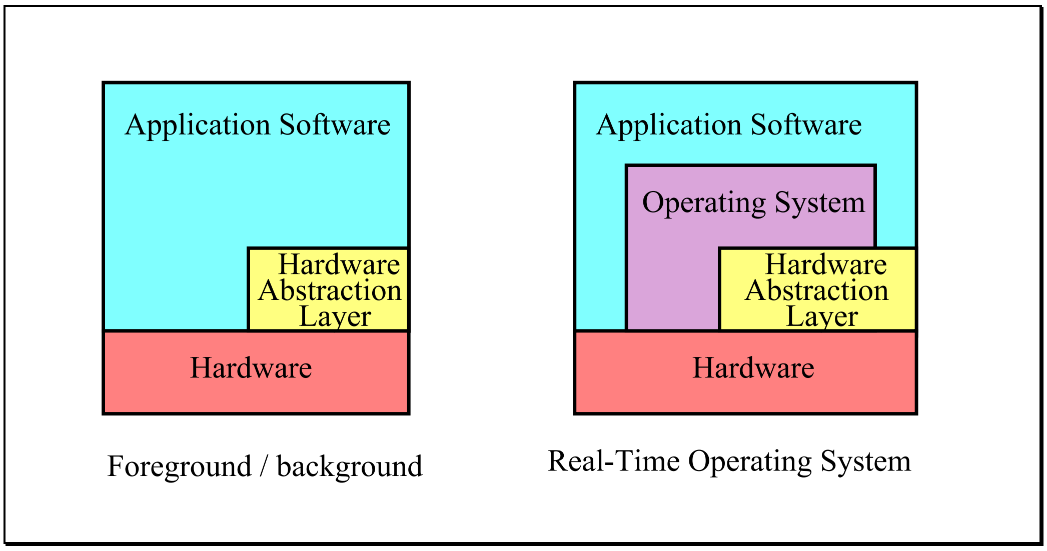


Figure P.3 – Two software approaches for an embedded system

For example, there are two channels (two outputs) from the AWG. With a real-time operating system, only one piece of code (a program) needs to be written and tested – it is then used by two independent "threads". Management of time also becomes a lot easier with an RTOS. You will have the opportunity to use a simple RTOS.

RTOS

The RTOS is supplied as a library of object code with header files (no C source code is supplied – a typical scenario in industry). The documentation for the RTOS is supplied in a separate document.

Application

The AWG application code should exploit the RTOS capabilities – a thread should be created for each of the output channels, and a thread should be created for communication with the PC interface program.

Strategy

The software should be developed in a modular fashion.

It is perfectly acceptable, and even advisable, to first get the AWG going using a foreground / background approach, perhaps with just one output channel. It is important in many projects to get parts of a system up and running quickly as a proof of concept of overall system design.

Also consider the fact that in the ultimate system, which uses an RTOS, all the shared code needs to be "re-entrant", and communication via global variables should be kept to a minimum. Any communication between threads via global variables will need to be carefully examined to see if semaphores are needed. Timing and priority of threads will also be an issue.

Consult the Project Marking Scheme so you can manage your time and focus on areas where the maximum marks will be awarded for appropriate effort.