anadigm

App Note - 209

Application Note: Developing a State-Driven Embedded System Using the ATmega128

Rev: 1.0.0

Date: 14th July 2009

This application note contains a source code files, if you have only this pdf text document, go here
http://www.anadigm.com/ /sup_AppNoteLib
to find the source code.

References:

Mega128-Dev Development Board Manual –
http://www.prllc.com/Mega128_Dev.pdf

Software Included with this Document:

AN221_Mega128 – project file, source file, make file, .HEX file, readme file

Software Required:

IDE – AVR Studio v3.56
Compiler – avr-gcc v3.2 (June 25 2002)
Bootloader – AVR Bootloader v2.0.1

TABLE OF CONTENTS

1 Purpose

This document describes how to create a state-driven embedded system consisting of the Anadigm AN221E04 FPAA and the Atmel ATmega128 microcontroller. This system will output a sinewave whose frequency will be selectable using 1 of 4 pushbuttons. The document describes how to: set up the hardware, how to set up the IDE and compiler, how to open the project, edit it and build it, how to download the compiled and assembled code to the microcontroller, and how to run the demonstration.

The points covered in this document include:

- Creating the primary configuration data
- Writing the code to execute a primary configuration on power up
- Creating the reconfiguration data
- Writing the state driven reconfiguration code

2 Building state driven code for the Atmel Mega128

2.1 Hardware Setup

The serial port of the PC is connected via a serial cable to P9 of the Mega128-Dev board and +10V DC is applied to its screw terminal power socket. All the jumpers are removed from J18 of the AN221D04-DEV board and port A (P2) of the Mega128 board is connected by 8 flying leads to the digital pins of the AN221 board, as illustrated in figure 1. 2 more flying leads connect 0 and +5V from the Mega128 board to the AN221 board. The digital connections are shown in table 1. Note that the connection to LCCb is not used in this demo.

ATmega128	AN221E04	ATmega128	AN221E04
PA0	EXECUTE	PA4	LCCb (unused)
PA1	PORb	PA5	DIN
PA2	ACTIVATE	PA6	DCLK
PA3	ERRb	PA7	CS2b

Table 1: Digital connections

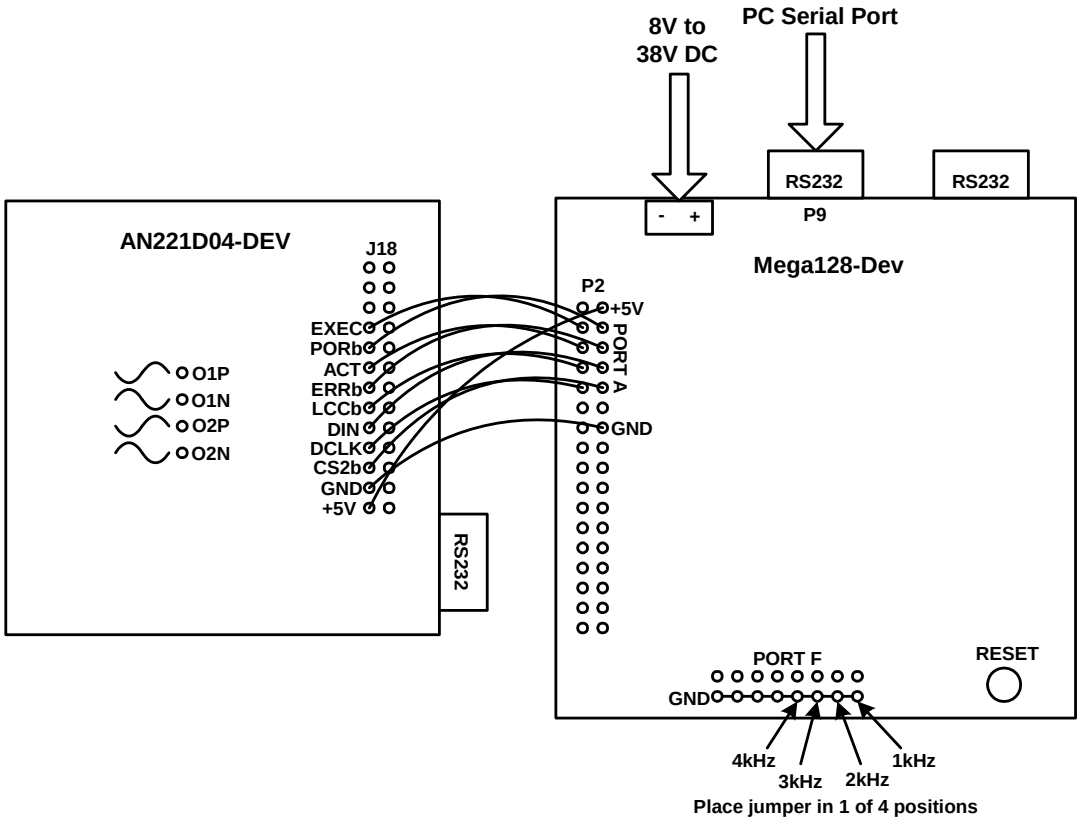

Figure 1: Hardware layout

.

2.2 Opening & Building the Project

2 pieces of software need to be downloaded from the web and installed. The first is the IDE (Integrated Development Environment) which allows editing of the source files. It is called 'AVR Studio v3.56' and can be downloaded free from the following website:

http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2724

Download the .exe file, double click on it and follow the instructions.

The second is the compiler which compiles and assembles the source files into downloadable .hex files. It is called 'avr-gcc v3.2 (June 25 2002)' and can be downloaded free from the following website:

http://www.avrfreaks.net/AVRGCC/index.php

Download the .exe file, double click on it and follow the instructions.

The next step depends on your operating system:

Win9x: go to the following directory: 'C:\avrgcc\avrfreaks' and copy the file called 'gcc_cmp.bat' to the directory 'c:\windows'.

Win2000 or WinXP: go to the following directory: 'C:\avrgcc\avrfreaks\win2000' and copy the 2 files called 'gcc_cmp.bat' and 'gcc_cmp2.bat' to the directory 'c:\WINNT'.

Now open up AVR Studio, go to the Project/Open menu and select the project file called 'AN221_Mega128.apr'. The project files will appear in the project window in the IDE. There are just 2 files which are:

AN221_Mega128.c – the source file

makefile – a make file for performing the compilation

Go to Project/Build in the menus and the project will compile. The compilation output will appear in a new window. Hopefully there are no errors and a new .hex file will be generated.

2.3 Downloading the Software to the Mega128-Dev

Download the AVR Bootloader v2.0.1 from the following site (it is free):

http://www.prllc.com/Mega128-dev.htm

Click on the link called 'AVRBL Windows Application' and download the zip file. Double click on the 'Setup.exe' file in the zip window and follow the instructions.

Open up the AVR Bootloader. Opposite where it says 'Application code', browse to the .hex file which you want to download. No other settings have to be changed. Make sure the Mega128.Dev board is powered up and that it is connected via a serial cable to the PC as shown in figure 1. Click the button that says 'Wait for Bootloader then Download' and then press the reset button on the Mega128-Dev board. The software will download to the board and start running.

2.4 Testing the Software

Monitor one of the analog outputs with a scope. When the demo starts, there should be a 1kHz sinewave appearing on these pins. Place a jumper on one of the 4 positions shown in figure 1. These 4 positions correspond to (R to L) 1kHz, 2kHz, 3kHz, and 4kHz. If you leave the jumper in any of these positions, you will see that a new configuration takes place every second. You will also see the yellow LED flashing while configuration is taking place, and then the green LED turning on to show configuration successful.

2.5 Editing the Project

The user can edit the source code and replace the existing configurations with those of his own, created from his own circuits using AnadigmDesigner2. This can be done as follows:

Open up AnadigmDesigner2 and create the first circuit. Go to 'Configure/Write configuration data to a file…' in the menus. Save the circuit as a .AHF file. Create the second circuit in AnadigmDesigner2 and do the same, but using a different .AHF filename. Repeat for each circuit that is required.

Go to 'Dynamic Config./State-driven method…' in the menus. Click on 'Add Circuits' and select all of the .AHF files that have just been created. Check the box next to each of the files. Click the 'Generate…' button. Select the C-code tab and browse to the appropriate directory if it is not already pointing to it. Click on the 'Generate' button.

A pair of files will have appeared in the working directory called 'StateDrivenCode.c' and 'StateDrivenCode.h'. Open the 'StateDrivenCode.c' file using WordPad and copy and paste each of the configuration bitstreams into the 'AN221Mega128.c' file, replacing the bitstreams that are already there. Copy both the primary configuration bitstreams, and the much shorter reconfiguration (or update) bitstreams into the appropriate places in the C code.

Note that the demo software contains 4 primary configurations and 4 reconfigurations (updates). These have been included for illustration. The user may decide that he only needs 1 primary configuration and multiple reconfigurations. Reconfigurations do not require a reset or dummy bytes at the start, and the bitstreams are much shorter. Also, a reconfiguration does not cause a break in the analog function as a primary configuration does.

It is also important to change the array sizes to the new lengths of the bitstreams. The bitstream length for each bitstream is located in a structure near the bottom of the 'StateDrivenCode.c' file. Change the bitstream lengths for both the primary configurations and the reconfigurations (updates). To do this, change the following lines near the top of the source code to reflect the new bitstream lengths.

```
#define PrimarySize1        155     //155 bytes in primary bit stream for circuit #1
#define PrimarySize2        155     //155 bytes in primary bit stream for circuit #2
#define PrimarySize3        155     //155 bytes in primary bit stream for circuit #3
#define PrimarySize4        155     //155 bytes in primary bit stream for circuit #4

#define UpdateSize1         34      //34 bytes in reconfig bit stream for circuit #1
#define UpdateSize2         34      //34 bytes in reconfig bit stream for circuit #2
#define UpdateSize3         34      //34 bytes in reconfig bit stream for circuit #3
#define UpdateSize4         34      //34 bytes in reconfig bit stream for circuit #4
```

Note that in the example demo, all of the primary configurations are the same length, as are all the reconfigurations (updates). This is not always the case. The length of the primary configurations depend on the complexity of the circuit, and the length of the reconfiguration updates depends on how many parameters are being changed.

The user can also change the code. He may, for example, want to change the conditions that determine which circuit gets downloaded.

2.6 Starting a New Project

In order to create a new project from scratch rather than just opening and editing an existing project, there are certain things that need to be done. Here is the procedure for creating a new project:

1. Create a working directory.
2. Go to 'C:\avrgcc\avrfreaks' and copy the makefile into the working directory.

3. Edit the makefile and change the line below to reflect the target MCU, in this case it is the ATmega128 (use lower case as shown).

```
#put the name of the target mcu here (at90s8515, at90s8535, .....
MCU = atmega128
```

4. Edit the makefile and change the line below to reflect the target file name. Change the 'gcctest1' to the required name without the extension.

```
#put the name of the target file here (without extension)
TRG         = gcctest1
```

5. Open up AVR Studio. Go to 'Project/New…' in the menu. Enter the project name, browse to the location, and select 'Generic 3rd party C compiler'. Click on OK.
6. In the project window, right click on 'Target:Debug' to get the target options window. Uncheck the box marked 'Run compile on each file in Source Files group'. In the box headed 'Command line:', replace the text 'Your linker/build stage commands goes here' with 'gcc_cmp.bat'. Click on OK.
7. Right click over 'Other Files' in the project window, select 'Add File…' and select the makefile in the working directory. Click on OK.
8. To add a .C file, either create it outside of AVR Studio and then do a right click over 'Source Files' in the project window of AVR Studio and add it to the project, or right click over Source Files in the project window of AVR Studio and select 'Create New File…', give it a name with a .C extension, and then click on OK. If it does not appear under 'Source Files' in the project window, select it and drag it there.
9. Enter the lines below into the empty .C file. Note that the compiler seems to require that the main() function has an integer return type. This is the reason for the redundant line 'return 1'.

```
int main(void)
{
    return 1;
}
```

10. Go to Project/Build in the menu and the project should compile without errors and a .hex file be created.

go here
http://www.anadigm.com /sup_AppNoteLib
to find the example source code.

For more product information, please visit Anadigm® at: www.anadigm.com

Or email support@anadigm.com