ETIN80 — Algorithms in Signal Processors

Development Environment

Tekn.Dr. Mikael Swartling

Lund Institute of Technology
Department of Electrical and Information Technology

January 27, 2014

Learning the Environment

- Visual DSP++ 5.0 Getting Started Guide
  The guide shows to how to create and setup a project, and how to configure a simulator or emulator session.
- Follow the exercises to get familiar with Visual DSP++ 5.0.
- Important:
  - A *simulator* session can be used without a DSP.
  - An *emulator* session requires a DSP to be connected.
  - Replace Blackfin ADSP-BF533 with SHARC ADSP-21262.
  - Select *ADSP-21262 via HPUSB-ICE* emulator platform.
  - Copy the tutorial projects from the specified location to your own folder if you intend to change or build it.
- The simulator platform does not provide audio or keyboard interrupts.
- An active session is only required to run a program.

Software Framework

- The provided framework configures the DSP.
- Add the supplied source and header files to your project.

  the `main.c` file can be used as a base example for your project
- Functions to control or query the framework:
  - `dsp_init` – *call once at the beginning to initialize the framework*
  - `dsp_start` – *call to start the serial ports connected to the codecs*
  - `dsp_stop` – *call to stop the serial ports connected to the codecs*
  - `dsp_get_audio` – *returns a pointer to the current audio block to process*
  - `dsp_get_keys` – *returns a bitmask of keys currently pressed*
- Interrupts raised by the framework:
  - `SIG_SP1` – *interrupt from the audio codec to process audio data*
  - `SIG_USR0` – *interrupt when a keyboard button is pressed*
  - `SIG_TMZ` – *interrupt from the timer (not a framework interrupt, but useful)*

Project Settings and Flash Loader

- The DSP does not have internal flash memory.
- To run from Visual DSP++, make an executable file.
  - *Project* menu, select *Project Options*.
  - *Project* settings, set *Type* to *Executable file*.
- To run stand-alone, make a loader file.
  - *Project* menu, select *Project Options*.
  - *Project* settings, set *Type* to *Loader file*.
  - *Project/Load* settings, set *Boot Type* to *SPI flash*.

Project Settings and Flash Loader

- Ensure that the DSP is connected and a session is active.
- To write the application to flash memory:
  - *Tools* menu, select *Flash Programmer*.
  - Select the *Flash* tab, the *OTP* tab is not relevant.
  - In the *Driver* tab:
    - Select the supplied driver file in *Driver file*.
      the supplied loader file is `21262EzFlashDriver_Serial.dxe`
    - Press *Load Driver*.
  - In the *Programming* tab:
    - Select your compiled application in *Data file*.
      the application is the `.ldr` *file in your project's* `Release` *or* `Debug` *folder*
    - Press *Program* to write your application to flash memory.
- Disconnect the session and power-cycle the DSP.

Configuring the Framework

- The header file has four values to configure to your needs.
  - sampling rate in Hz
  - block size samples
  - mic or line input gain in dB
  - speaker output attenuation in dB

```
#define DSP_SAMPLE_RATE         16000
#define DSP_BLOCK_SIZE          32
#define DSP_INPUT_GAIN          20
#define DSP_OUTPUT_ATTENUATION  0
```

Block Buffers and the Framework

- The framework handles one block per interrupt.
- Input and output uses a cyclic set of three buffers:
  - the input block is recorded
  - the process block is processed by the application
  - the output block is played
- Total delay is two blocks, plus additional codec delay.
- If no processing, the framework is pass-through.
- Halting the DSP *does not* halt the I/O processor!
  - beware of acoustic feedback
  - halting does not stop codecs from recording and playing

Block Buffers and the Framework

- The function `dsp_get_audio` returns a pointer to an array of `DSP_BLOCK_SIZE` integer-valued stereo samples.
- A block is processed in-place.

  the output is written to where the input is
- Sample values are 32-bit signed integers.
  - sample values range from -2147483648 to 2147483647
  - typecast the pointer to a `fract *`, or
  - scale the sample values to a `float`
- Assuming that

  ```
  sample_t *audio = dsp_get_audio();
  ```

  then

  ```
  audio[n].left
  audio[n].right
  ```

  is valid for `n` from 0 to `DSP_BLOCK_SIZE-1`.

Software Framework

```
#include "framework.h"

void process(int sig)
{
    int n;
    sample_t *audio = dsp_get_audio();

    for(n=0; n<DSP_BLOCK_SIZE; ++n) {
        audio[n].left = audio[n].left << 1;   // half volume on left channel
        audio[n].right = audio[n].right >> 1; // double volume on right channel
    }
}

void main()
{
    dsp_init();
    interrupt(SIG_SP1, process);
    dsp_start();

    for(;;) {
        idle();
    }
}
```