***DIGITAL to ANALOG CONVERSION* (DAC) dan BUNYI**

1. **Tujuan**

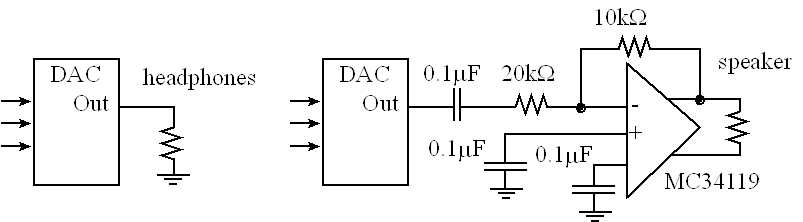
* Mengembangkan sarana untuk komputer digital untuk berinteraksi dengan dunia analog
* Memahami digitisasi: kuantisasi, rentang, presisi dan resolusi
* Memahami *sampling* dan teorema Nyquist
* Memahami dasar-dasar bunyi: elektromagnet, pengeras suara, daya AC vs DC, persepsi bunyi
* Memahami bagaimana menciptakan bunyi: *loudness*, *pitch*, *envelope*, dan *shape*
* Menggunakan SysTick untuk menciptakan bunyi dengan pemrograman variabel frekuensi

1. **Peralatan yang digunakan**

* Keil μVision v5
* Tiva C Series LaunchPad
* Stellaris® ICDI Drivers
* Resistor 11 kΩ
* Resistor 22 kΩ
* *Speaker*
* Osiloskop

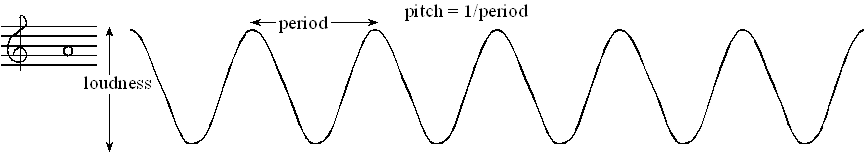
1. **Landasan teori**

Sebagian besar perangkat musik digital mengandalkan DAC kelajuan tinggi untuk menciptakan gelombang analog yang diperlukan untuk menghasilkan bunyi berkualitas tinggi. Sistem pembangkitan bunyi yang sangat sederhana terdiri dari antarmuka DAC dan pengeras suara. *Headphone* dapat digerakkan secara langsung dari *output* DAC, tetapi untuk menggerakkan *speaker* biasa, dibutuhkan penguat audio, seperti terlihat pada Gambar 7.



Gambar 7. DAC memungkinkan perangkat lunak menciptakan musik

Kualitas musik akan tergantung pada faktor perangkat keras dan perangkat lunak. Presisi DAC, *noise* eksternal, dan rentang dinamis *speaker* adalah beberapa faktor perangkat keras. Faktor perangkat lunak adalah laju *output* DAC dan kompleksitas data bunyi yang tersimpan. Jika *output* deretan bilangan DAC yang membentuk gelombang sinus, maka akan terdengar not kontinu pada *speaker*, seperti terlihat pada Gambar 8.



Gambar 8. *Loudness* dan *pitch* dikendalikan oleh amplitudo dan frekuensi

*Loudness* not ditentukan oleh amplitudo gelombang. *Pitch* ditentukan oleh frekuensi gelombang. Tabel 2 mengandung nilai frekuensi untuk not dalam satu oktaf.

Tabel 2. Frekuensi dasar not musik standar

|  |  |
| --- | --- |
| Not | Frekuensi (Hz) |
| C  B  Bb  A  Ab  G  Gb  F  E  Eb  D  Db  C | 523  494  466  440  415  392  370  349  330  311  294  277  262 |

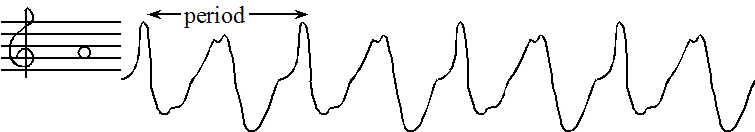
Frekuensi gelombang fsin akan ditentukan oleh frekuensi *interrupt* fint dibagi oleh ukuran tabel n. Ukuran tabel dalam Program 1 adalah n=16.

|  |
| --- |
| **const unsigned char SineWave[16] = {4,5,6,7,7,7,6,5,4,3,2,1,1,1,2,3};**  **unsigned char Index=0;           // Index varies from 0 to 15**  **// \*\*\*\*\*\*\*\*\*\*\*\*\*\*DAC\_Init\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **// Initialize 3-bit DAC**  **// Input: none**  **// Output: none**  **void DAC\_Init(void){unsigned long volatile delay;**  **SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOB; // activate port B**  **delay = SYSCTL\_RCGC2\_R;    // allow time to finish activating**  **GPIO\_PORTB\_AMSEL\_R &= ~0x07;      // no analog**  **GPIO\_PORTB\_PCTL\_R &= ~0x00000FFF; // regular GPIO function**  **GPIO\_PORTB\_DIR\_R |= 0x07;      // make PB2-0 out**  **GPIO\_PORTB\_AFSEL\_R &= ~0x07;   // disable alt funct on PB2-0**  **GPIO\_PORTB\_DEN\_R |= 0x07;      // enable digital I/O on PB2-0**  **}**    **// \*\*\*\*\*\*\*\*\*\*\*\*\*\*Sound\_Init\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **// Initialize Systick periodic interrupts**  **// Input: interrupt period**  **//        Units of period are 12.5ns**  **//        Maximum is 2^24-1**  **//        Minimum is determined by length of ISR**  **// Output: none**  **void Sound\_Init(unsigned long period){**  **DAC\_Init();          // Port B is DAC**  **Index = 0;**  **NVIC\_ST\_CTRL\_R = 0;         // disable SysTick during setup**  **NVIC\_ST\_RELOAD\_R = period-1;// reload value**  **NVIC\_ST\_CURRENT\_R = 0;      // any write to current clears it**  **NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R&0x00FFFFFF)|0x20000000; // priority 1**  **NVIC\_ST\_CTRL\_R = 0x0007; // enable SysTick with core clock and interrupts**  **}**  **// \*\*\*\*\*\*\*\*\*\*\*\*\*\*DAC\_Out\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **// output to DAC**  **// Input: 3-bit data, 0 to 7**  **// Output: none**  **void DAC\_Out(unsigned long data){**  **GPIO\_PORTB\_DATA\_R = data;**  **}**  **// the sound frequency will be (interrupt frequency)/(size of the table)**  **void SysTick\_Handler(void){**  **Index = (Index+1)&0x0F;      // 4,5,6,7,7,7,6,5,4,3,2,1,1,1,2,3,...**  **DAC\_Out(SineWave[Index]);    // output one value each interrupt**  **}**  **void main(void){**  **PLL\_Init();          // bus clock at 80 MHz**  **Switch\_Init();       // Port F is onboard switches, LEDs, profiling**  **Sound\_Init(50000);   // initialize SysTick timer, 1.6kHz**  **while(1){**  **}**  **}** |

Program 1. *Interrupt* periodik mengeluarkan satu nilai ke DAC (*C13\_R2R\_3bit*)

Frekuensi masing-masing not musik dapat dihitung dengan cara mengalikan frekuensi sebelumnya dengan . Metode ini dapat digunakan untuk menentukan frekuensi not tambahan diatas dan dibawah dalam Tabel 2. Terdapat 12 not dalam 1 oktaf, sehingga menaikkan satu oktaf melipatgandakan frekuensi.

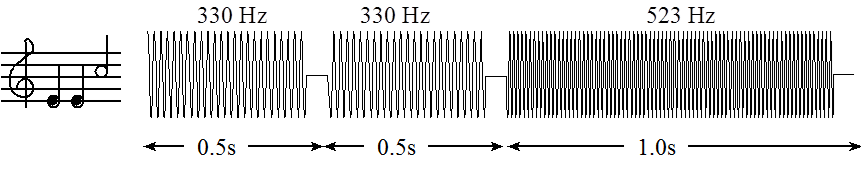
Gambar 9 memperlihatkan konsep instrumen.



Gambar 9. Bentuk gelombang yang membangkitkan bunyi terompet

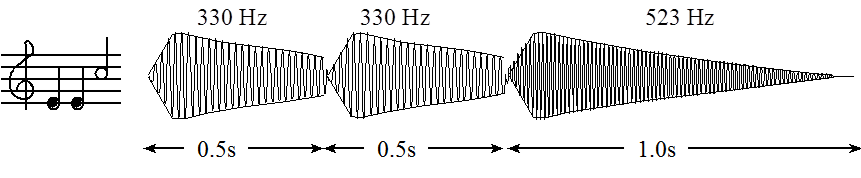
Tipe bunyi dapat ditentukan oleh tegangan versus waktu bentuk gelombang. Alat musik tiup memiliki frekuensi harmonis pertama yang sangat besar.

Tempo musik menentukan kelajuan lagu. Pada musik 2/4, ¾ atau 4/4, *beat* ditentukan sebagai not seperempat. Tempo moderat adalah 120 *beat*/menit, yang berarti bahwa not seperempat memiliki durasi ½ detik. Deretan not dapat dipisah oleh *pause* (hening) sehingga masing-masing not terdengar terpisah. *Envelope* not ditentukan oleh hubungan amplitudo versus waktu. *Envelope* yang sangat sederhana terlihat pada Gambar 10.



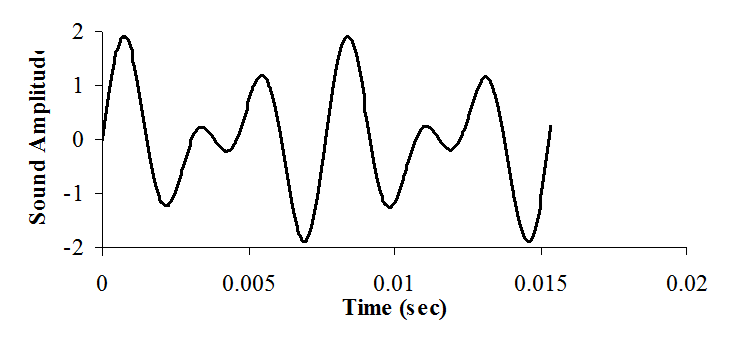
Gambar 10. Not E-E-C

Prosesor CortexTM – M memiliki banyak daya untuk menciptakan tipe-tipe gelombang di atas. *Envelope* berbentuk halus, seperti terlihat pada Gambar 11, menyebabkan *staccato* yang kurang dan bunyi yang merdu.



Gambar 11. Amplitudo dawai yang dipetik menurun secara eksponensial menurut waktu

Tipe pembangkitan bunyi di atas dapat dihasilkan secara *real time* pada mikrokontroler CortexTM – M. *Chord* diciptakan oleh permainan banyak not secara simultan. Ketika dua tuts piano ditekan secara simultan, kedua not tercipta, dan bunyi tercampur secara aritmatika. Gambar 12 memperlihatkan penjumlahan matematis gelombang sinus 262 Hz (not C) dan 392 Hz (not G), menghasilkan *chord* yang sederhana.



Gambar 12. *Chord* sederhana dari pencampuran not C dan G

1. **Eksperimen**

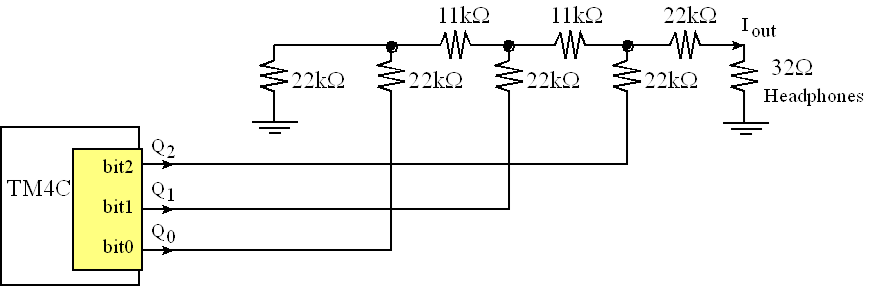
Desain suatu DAC R-2R 3 bit dan gunakan untuk menciptakan gelombang sinus 100 Hz.

Tabel 3 memperlihatkan spesifikasi desain.

Tabel 3. Spesifikasi DAC R-2R 3 bit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Q2 | Q1 | Q0 | Iout (μA) |
| 0  1  2  3  4  5  6  7 | 0  0  0  0  3,3  3,3  3,3  3,3 | 0  0  3,3  3,3  0  0  3,3  3,3 | 0  3,3  0  3,3  0  3,3  0  3,3 | 0,0  12,5  25,0  37,5  50,0  62,5  75,0  87,5 |

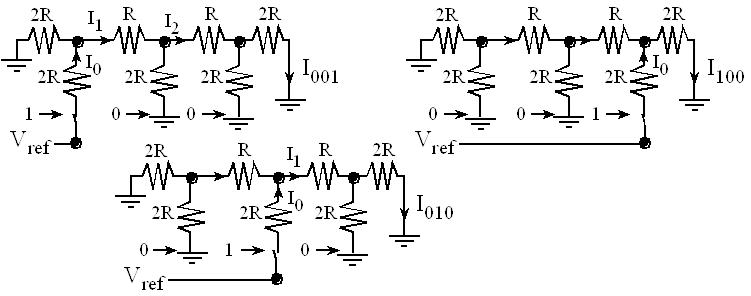
Sistem akan dapat membangkitkan 8 arus *output* yang berbeda dengan resolusi sekitar 12,5 μA. Rangkaian seperti terlihat pada Gambar 13.



Gambar 13. DAC R-2R 3 bit

Asumsikan *output high voltage* (VOH) mikrokontroler adalah 3,3 V, dan *output low voltage* (VOL) adalah 0 V. DAC R-2R 3 bit akan membutuhkan dua resistor R (11 kΩ) dan lima resistor 2R (22 kΩ), seperti terlihat pada Gambar 13. Arus *output* maksimum menjadi .

Cara menganalisis DAC R-2R adalah menerapkan Hukum Superposisi. Misalkan terdapat tiga unsur dasar (1, 2, dan 4). Jika ketiganya didemonstrasikan, maka Hukum Superposisi menjamin lima yang lain. Ketika salah satu *input* digital adalah *true* maka Vref (3,3 V) terhubung ke R-2R, dan ketika *input* digital *false*, maka terhubung ke *ground*. Perhatikan Gambar 14.



Gambar 14. Analisis tiga unsur dasar {001, 010, 100} DAC R-2R 3 bit

Berhubung 22.000 lebih besar daripada 32, abaikan resistansi 32 Ω ketika kalkulasi arus.

Untuk membangkitkan bunyi, dibutuhkan tabel data dan *interrupt* periodik. Program 1 di atas memperlihatkan kode C yang menciptakan gelombang sinus 3 bit 16 unsur. Frekuensi bunyi menjadi frekuensi *interrupt* dibagi 16 (ukuran tabel). Untuk menciptakan gelombang 100 Hz dibutuhkan SysTick *interrupt* pada 16×100 Hz = 1600 Hz. Jika *bus clock* adalah 80 MHz, maka inisialisasi dipanggil dengan parameter nilai *input* 50000. **const** *modifier* akan menempatkan data dalam ROM. Perangkat lunak *interrupt* akan *output* satu nilai ke DAC. Perhatikan Gambar 15.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Gambar 15. DAC dan *interrupt* periodik digunakan untuk menciptakan bunyi

Pada eksperimen ini, DAC 3 bit berasal dari *output pin* PB2 – PB0. Untuk *output* ke DAC, maka tulis ke *Port* B. Untuk menciptakan bunyi, *output* satu bilangan ke DAC pada setiap *interrupt*. Rentang DAC adalah 0 – 87,5 μA.

**Instruksi:**

* Buat rangkaian pada *protoboard* berdasarkan Gambar 13.
* Pasang *jumper* ke PB2 (bit 2), PB1 (bit 1), dan PB0 (bit 0) TM4C123.
* Jalankan C:\Keil\TExaSware\C13\_R2R\_3bit\R2R.uvproj.
* Periksa kesesuaian Program 1 dengan R2R.uvproj.
* Hubungkan kanal 1 osiloskop ke *output* pada Gambar 13.
* Hubungkan kanal 2 osiloskop ke PF3 TM4C123.
* Tekan saklar SW1 pada TM4C123.
* Amati apa yang terlihat pada osiloskop, dan apa yang terdengar pada *speaker*.

1. **Bibliografi**

<http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C13_DACSound.htm>