***FINITE STATE MACHINE***

1. **Tujuan**

* Mempelajari bagaimana mengaktifkan PLL sehingga mikrokontroler memiliki basis waktu akurat
* Menggunakan SysTick untuk menghasilkan waktu tunda akurat
* Mempelajari bagaimana mengelola data pada komputer menggunakan struktur
* Mengembangkan strategi desain untuk pembangunan Finite State Machine

1. **Peralatan yang digunakan**

* Keil μVision v5
* Tiva C Series LaunchPad
* Stellaris® ICDI Drivers
* 2 LED merah
* 2 LED kuning
* 2 LED hijau
* 2 saklar

1. **Landasan teori**

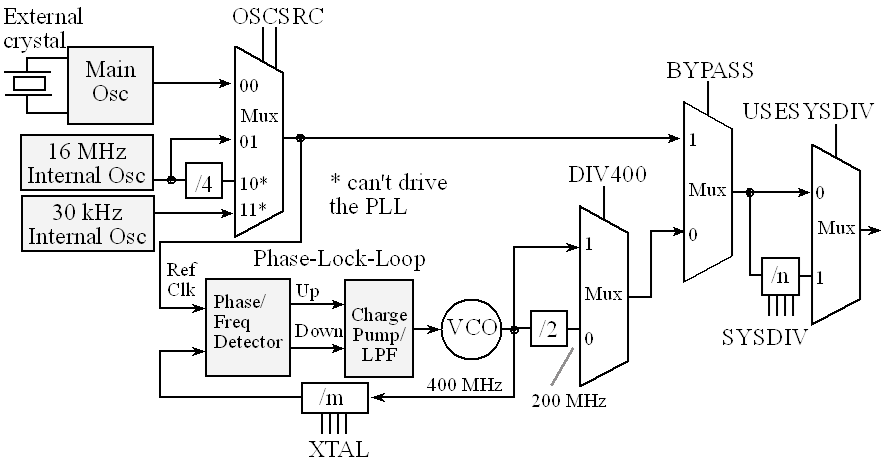
Waktu adalah elemen penting sistem embedded. Dalam modul ini kita memperkenalkan phase-lock-loop (PLL) yang memungkinkan perangkat lunak memanfaatkan kristal akurat dan memilih seberapa cepat komputer mengeksekusi.

Pada umumnya, laju eksekusi mikrokontroler ditentukan oleh kristal eksternal. Keping Stellaris® EK-TM4C123GXL memiliki kristal 16 MHz. Kebanyakan mikrokontroler dilengkapi phase-lock-loop (PLL) yang memungkinkan perangkat lunak menyesuaikan laju eksekusi komputer. Pada khususnya, pemilihan frekuensi melibatkan tradeoff antara laju eksekusi perangkat lunak dan daya listrik. Dengan kata lain, memperlambat clock bus akan membutuhkan lebih sedikit daya untuk beroperasi dan menghasilkan lebih sedikit panas. Mempercepat clock bus jelas memungkinkan untuk lebih banyak kalkulasi per detik, membutuhkan lebih banyak daya untuk beroperasi dan menghasilkan lebih banyak panas.

Laju bus standar untuk osilator internal TM4C adalah 16 MHz ± 1%. Osilator internal secara signifikan kurang presisi daripada kristal, tetapi membutuhkan daya lebih kecil dan tidak membutuhkan kristal eksternal. Jika kita ingin memiliki kontrol waktu akurat, kita akan mengaktifkan kristal eksternal (disebut osilator utama) menggunakan PLL untuk memilih laju bus yang diinginkan.

Ada dua cara mengaktifkan PLL. Kita dapat memanggil fungsi pustaka, atau kita dapat mengakses register clock secara langsung. Pada umumnya, penggunaan fungsi pustaka menciptakan desain yang lebih baik karena solusi akan lebih stabil (sedikit bug) dan akan lebih portabel (lebih mudah mengaktifkan mikrokontroler). Penampilan akses langsung menggambarkan beberapa konsep PLL.

Kristal eksternal melekat pada mikrokontroler TM4C, seperti ditunjukkan pada Gambar 1.



Gambar 1. Diagram blok pohon clock utama pada TM4C termasuk PLL

Tabel 1 memperlihatkan register clock yang digunakan untuk menentukan berapa laju prosesor beroperasi.

Tabel 1. Register clock utama (RCC2 dalam TM4C)

|  |  |  |  |
| --- | --- | --- | --- |
| *XTAL* | *Crystal Freq (MHz)* | *XTAL* | *Crystal Freq (MHz)* |
| 0x0 | Reserved | 0x10 | 10.0 MHz |
| 0x1 | Reserved | 0x11 | 12.0 MHz |
| 0x2 | Reserved | 0x12 | 12.288 MHz |
| 0x3 | Reserved | 0x13 | 13.56 MHz |
| 0x4 | 3.579545 MHz | 0x14 | 14.31818 MHz |
| 0x5 | 3.6864 MHz | 0x15 | 16.0 MHz |
| 0x6 | 4 MHz | 0x16 | 16.384 MHz |
| 0x7 | 4.096 MHz | 0x17 | 18.0 MHz |
| 0x8 | 4.9152 MHz | 0x18 | 20.0 MHz |
| 0x9 | 5 MHz | 0x19 | 24.0 MHz |
| 0xA | 5.12 MHz | 0x1A | 25.0 MHz |
| 0xB | 6 MHz (reset value) | 0x1B | Reserved |
| 0xC | 6.144 MHz | 0x1C | Reserved |
| 0xD | 7.3728 MHz | 0x1D | Reserved |
| 0xE | 8 MHz | 0x1E | Reserved |
| 0xF | 8.192 MHz | 0x1F | Reserved |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Address | 26-23 | 22 | 13 | 11 | 10-6 | 5-4 | Name |
| $400FE060 | SYSDIV | USESYSDIV | PWRDN | BYPASS | XTAL | OSCSRC | SYSCTL\_RCC\_R |
| $400FE050 |  |  |  |  | PLLRIS |  | SYSCTL\_RIS\_R |
| Address | 31 | 30 | 28-22 | 13 | 11 | 6-4 | Name |
| $400FE070 | USERCC2 | DIV400 | SYSDIV2 | PWRDN2 | BYPASS2 | OSCSRC2 | SYSCTL\_RCC2\_R |

Output osilator utama (Main Osc) adalah clock pada frekuensi yang sama dengan kristal. Dengan mengatur bit OSCSRC ke 0, kontrol multiplekser akan memilih osilator utama sebagai sumber clock.

Osilator utama untuk LaunchPad TM4C adalah 16 MHz. Ini berarti clock referensi (Ref Clk) input ke detektor fase/frekuensi adalah 16 MHz. Untuk kristal 16 MHz, kita atur bit XTAL menjadi 10101 (lihat Tabel 1). Dengan cara ini, output 400 MHz voltage controlled oscillator (VCO) akan menghasilkan clock 16 MHz pada input lain detektor fase/frekuensi. Jika clock 400 MHz terlalu lambat, sinyal **naik** akan ditambah ke pompa muatan, menaikkan input ke VCO, menyebabkan peningkatan frekuensi 400 MHz. Jika clock 400 MHz terlalu cepat, sinyal **turun** akan berkurang dari pompa muatan, menurunkan input ke VCO, menyebabkan penurunan frekuensi 400 MHz. Karena clock referensi stabil, loop umpan balik dalam PLL akan menggerakkan output ke frekuensi 400 MHz stabil.

Program 1 memperlihatkan langkah 0-6 untuk mengaktifkan LaunchPad TM4C123 dengan osilator utama 16 MHz untuk bekerja pada 80 MHz.

|  |
| --- |
| **void PLL\_Init(void){   // 0) Use RCC2   SYSCTL\_RCC2\_R |=  0x80000000;  // USERCC2   // 1) bypass PLL while initializing   SYSCTL\_RCC2\_R |=  0x00000800;  // BYPASS2, PLL bypass   // 2) select the crystal value and oscillator source   SYSCTL\_RCC\_R = (SYSCTL\_RCC\_R &~0x000007C0)   // clear XTAL field, bits 10-6                  + 0x00000540;   // 10101, configure for 16 MHz crystal   SYSCTL\_RCC2\_R &= ~0x00000070;  // configure for main oscillator source   // 3) activate PLL by clearing PWRDN   SYSCTL\_RCC2\_R &= ~0x00002000;   // 4) set the desired system divider   SYSCTL\_RCC2\_R |= 0x40000000;   // use 400 MHz PLL   SYSCTL\_RCC2\_R = (SYSCTL\_RCC2\_R&~** **0x1FC00000)  // clear system clock divider                   + (4<<22);      // configure for 80 MHz clock   // 5) wait for the PLL to lock by polling PLLLRIS   while((SYSCTL\_RIS\_R&0x00000040)==0){};  // wait for PLLRIS bit   // 6) enable use of PLL by clearing BYPASS   SYSCTL\_RCC2\_R &= ~0x00000800; }** |

Program 1. Aktivasi TM4C dengan kristal 16 MHz untuk bekerja pada 80 MHz (C10\_PLL)

1. Gunakan RCC2 karena menyediakan untuk pilihan lainnya.
2. Langkah pertama adalah mengatur BYPASS2 (bit 11).

Pada titik ini PLL dilewati dan tidak ada pembagi clock sistem.

1. Langkah kedua adalah menentukan frekuensi kristal dalam empat bit XTAL menggunakan kode dalam Tabel 1.

Bit OSCSRC2 dihapus untuk memilih osilator utama sebagai sumber clock osilator.

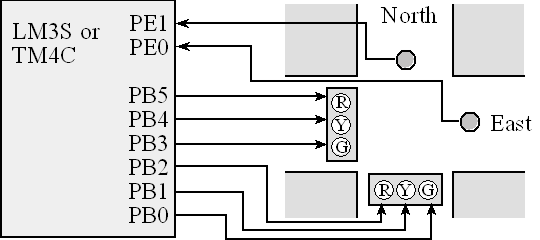
1. Langkah ketiga adalah membersihkan PWRDN2 (bit 13) untuk aktivasi PLL.
2. Langkah keempat adalah mengkonfigurasi dan enable pembagi clock menggunakan 7-bit SYSDIV2.

Jika 7-bit SYSDIV2 adalah **n**, maka clock akan dibagi oleh **n**+1. Untuk mendapatkan 80 MHz yang diinginkan dari PLL 400 MHz, kita membagi dengan 5. Jadi, kita tulis 4 dalam SYSDIV2.

1. Langkah kelima adalah menunggu PLL stabil dengan menunggu PLLRIS (bit 6) dalam **SYSCTL\_RIS\_R** menjadi tinggi.
2. Langkah terakhir adalah menghubungkan PLL dengan membersihkan bit BYPASS2.
3. **Eksperimen**

Desain pengendali lampu lalu lintas untuk persimpangan dua jalan sibuk satu arah. Tujuannya adalah memaksimalkan arus lalu lintas, meminimalkan waktu tunggu saat lampu merah, dan menghindari kecelakaan.

Persimpangan memiliki dua jalan satu arah dengan jumlah lalu lintas yang sama: North dan East, seperti terlihat dalam Gambar 6.



Gambar 6. Antarmuka lampu lalu lintas dengan dua sensor dan 6 LED

Dalam sistem ini, keadaan menjelaskan jalan mana yang memiliki otoritas untuk menyeberangi persimpangan. Ide dasarnya adalah untuk mencegah mobil arah selatan memasuki persimpangan pada saat yang sama dengan mobil arah barat. Dalam sistem ini, pola lampu mendefinisikan jalan yang memiliki hak jalan atas yang lain. Sistem akan memiliki dua input (sensor mobil di jalan North dan East) dan enam output (satu untuk setiap lampu dalam sinyal lalu lintas). Enam lampu lalu lintas dihubungkan ke Port B bit 5-0, dan dua sensor dihubungkan ke Port E bit 1-0

**Instruksi**:

* Buat rangkaian pada protoboard berdasarkan Gambar 6.
* Hubungkan saklar North ke Port PE1, sedangkan saklar East ke Port PE0.
* Hubungkan LED merah barat ke Port PB5, sedangkan LED merah selatan ke Port PB2.
* Hubungkan LED kuning barat ke Port PB4, sedangkan LED kuning selatan ke Port PB1.
* Hubungkan LED hijau barat ke Port PB3, sedangkan LED hijau selatan ke Port PB0.
* Jalankan C:\Keil\TExaSware\C10\_TableTrafficLight\TableTrafficLight.uvproj.
* Periksa kesesuaian Program 1 dengan TableTrafficLight.uvproj.
* Gunakan simulator, amati apa yang terjadi.
* Tekan saklar North, amati apa yang terjadi.
* Tekan saklar East, amati apa yang terjadi.
* Tekan kedua saklar North dan East bersamaan, amati apa yang terjadi.

1. **Bibliografi**

<http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C10_FiniteStateMachines.htm>