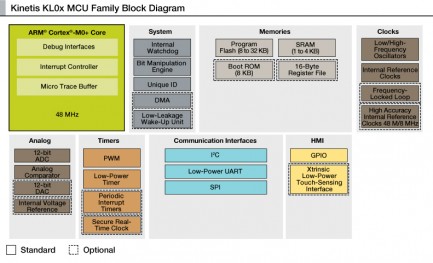
**Autores : Gabriel Silva Marcatto e Rodrigo Maximiano Antunes de Almeida**

É importante ressaltar que este tipo análise, apesar de específico para o KL05z, obedece o mesmo procedimento para qualquer microcontrolador.

**Entendendo o KL05z**

A configuração básica de um microcontrolador é uma etapa importante para garantir o bom funcionamento da placa. Como se trata das configurações iniciais, é comum que sejam alteradas de acordo com o desenvolvimento do produto, principalmente quando novas funcionalidades são adicionadas no projeto. Em geral, o datasheet do produto explicita todas as configurações necessárias e como alterá-las.

Abaixo é apresentado o diagrama de blocos do KL05 que apresenta todos os periféricos disponíveis.



Para a grande maioria dos microcontroladores do mercado, os pontos mais críticos são: o clock, os sistemas de interrupção e o watchdog. Eles podem ser considerados críticos pois configurações erradas nestes periféricos vão criar problemas no funcionamento da placa, bugs intermitentes ou até a total paralisação do software.

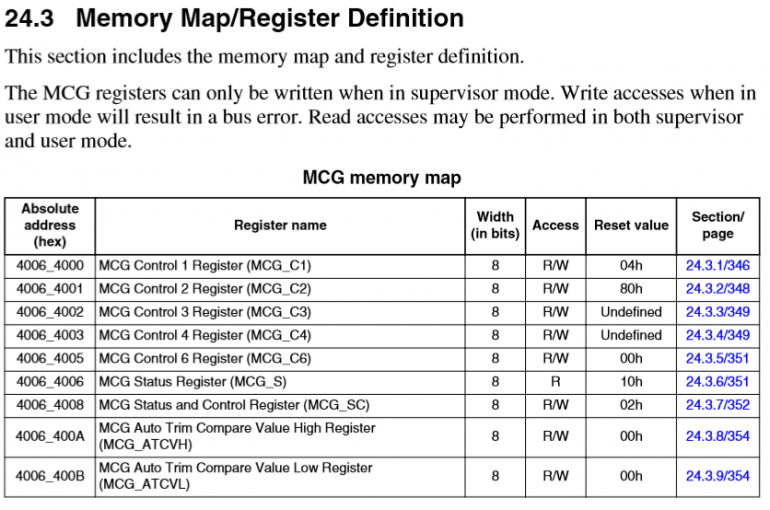
**A fonte de clock**

De acordo o capítulo 5 do datasheet, que apresenta a arquitetura do clock para o microcontrolador, o *Cortex M0+* possui um core síncrono, permitindo que os clocks do processador, bus masters, flash e periféricos sejam configurados de modo independentes. A seleção e multiplexação do clock do sistema é controlado e programado via o módulo **MCG**, já o clock dos drivers são programados via o módulo **SIM**. Esse tipo de arquitetura é denominada de *clock gating* e permite uma maior economia de energia. Normalmente o clock nos módulos são desativados como padrão.

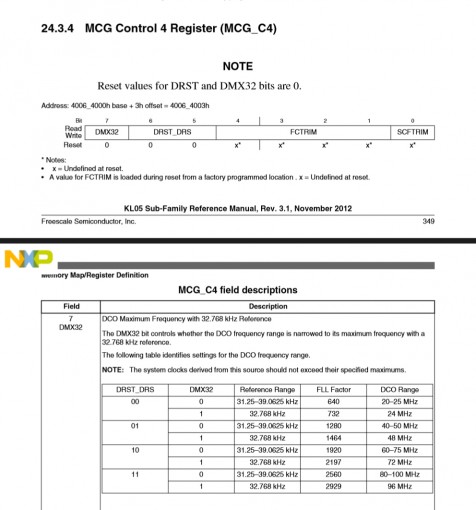
**Configurando o clock do sistema**

O MCG (*Multipurpose Clock Generator*) é um módulo que permite diversas fontes de clock para o microcontrolador. Ele contém um frequency-locked loop (FLL). O FLL é controlado pela referência interna ou externa de clock. O módulo pode escolher o FLL interno ou externo como fonte de clock para o sistema do MCU. O MCG opera em conjunto com um cristal oscilador, que permite outra fonte de clock externa produza o clock de referência [página 343 do datasheet].

O capítulo 24, referente ao MCG, também oferece um *Memory Map/ Register definition*, que mostra quais registros pertencem ao módulo e quais possuem acesso de leitura e/ou escrita.



Verificando a função de cada um dos registros, descobre-se que através do registro **MCG\_C4** é possível configurar o clock do sistema:



Para determinar o clock do sistema é necessário manipular os bits 7, 6 e 5 do registro. Ao ligarmos o bit 7, o clock da placa passa a ser 24 MHz. Então se realizarmos uma operação do tipo OU (OR) com valor 0x80u (0b10000000) no ponteiro MCG\_C4, estaremos ligando o bit 7. As outras opções como FCTRIM e SCFTRIM não são necessárias para a configuração inicial. Logo, o código responsável por essa operação é:

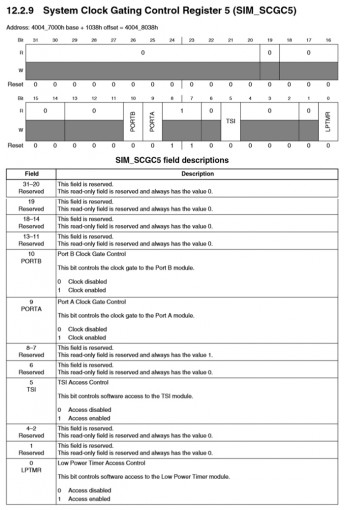
|  |  |
| --- | --- |
|  | MCG\_C4      |= 0x80u; |

A biblioteca do MCU oferece máscaras e *structs* que permitem uma facilidade na programação. Utilizando essas ferramentas disponíveis, o código anterior se transforma em:

|  |  |
| --- | --- |
|  | MCG\_BASE\_PTR ->C4 |= MCG\_C4\_DMX32\_MASK; |
|  |  |

**Configurando o clock gating das portas**

Após configurar o sistema, é necessário ativar o clock nas portas A e B. O clock pode ser habilitado ou desabilitado através do registro **SCGC5[PORTx]** no módulo **SIM**. Esses bits são sempre limpos após a qualquer tipo de reset, o que desabilita o clock do módulo correspondente para economizar energia [página 147 do datasheet].



Para ligar o clock nas portas A e B é necessário manipular os bits 9 e 10. Para que isso ocorra é preciso realizar uma operação do tipo OU (OR) com o valor 0x600u (0b11000000000) no registro SIM\_SCGC5. O TSI e LPTMR não são necessários para a configuração inicial, o valor “padrão serve”. Logo, o código responsável por essa operação é:

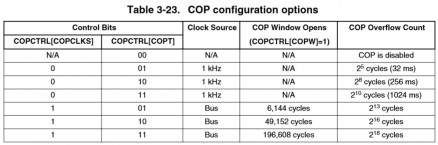
|  |  |
| --- | --- |
|  | SIM\_SCGC5 |= 0x600u; |

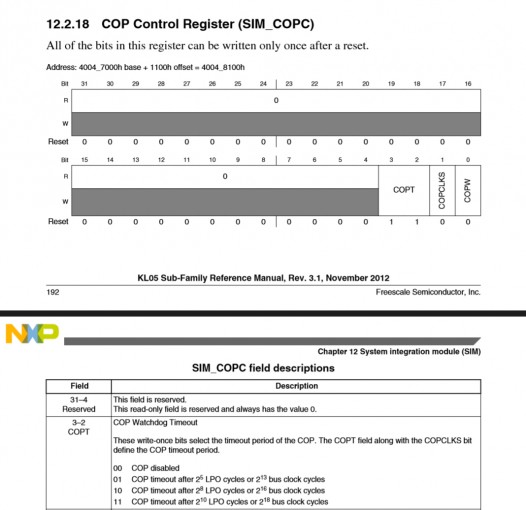
Utilizando as máscaras e *structs* fornecidas, o código acima se transforma em:

|  |  |
| --- | --- |
|  | SIM\_BASE\_PTR ->SCGC5 |= (SIM\_SCGC5\_PORTA\_MASK | SIM\_SCGC5\_PORTB\_MASK); |

**Watchdog**

O COP watchdog tem função de forçar um reset no sistema quando o software falha em executar como esperado. Para evitar o acionamento do watchdog, e a consequente reinicialização da placa, a aplicação deve resetar o contador do COP periodicamente. Depois de qualquer reset, o COP Watchdog é habilitado. Caso a aplicação não utilize o Watchdog, ele pode ser desabilitado limpando os btis do **COPCTRL[COPT]** no **SIM** [página 64 do datasheet]. O capítulo também mostra uma tabela com as configs do COP.





Para desligar o COP watchdog é necessário limpar os bits 3 e 2. Para que isso ocorra é preciso realizar uma operação do tipo NAND com o valor 0xCu (0b1100) no registro SIM\_COPC. Logo, o código responsável por essa operação é:

|  |  |
| --- | --- |
|  | SIM\_COPC  &=~ 0xCu; |

  Utilizando as máscaras e *structs* fornecidas, o código acima se transforma em:

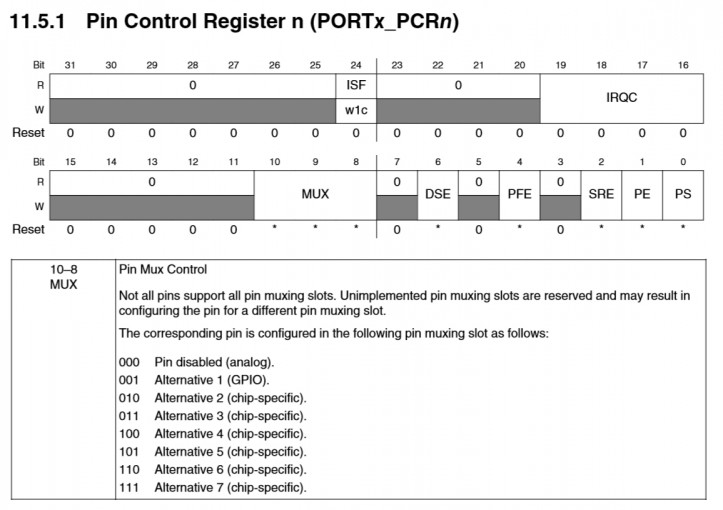
|  |  |
| --- | --- |
|  | SIM\_BASE\_PTR -> COPC &=~ SIM\_COPC\_COPT\_MASK; |

**Non-Maskable Interrupt**

O NMI (*Non-Maskable Interrupt*) da placa, por configuração padrão,vem conectado no pino 13. Isso pode causar problemas pois qualquer flutuação no pino pode disparar o interrupt. Para arrumar esse inconveniente é necessário desconectar o NMI do pino.



O microcontrolador multiplexa diversas funções para um único pino, permitindo uma maior customização. Essas opções podem ser selecionadas através do registro **PORTx\_PCRn.**



Para remover o NMI da PORTB5 é necessário escolher a alternativa 1 do Mux. Então é necessário manipular os bits 10, 9 e 8. As outras configurações do registro não são necessárias incialmente. Logo, o código responsável por essa operação é :

|  |  |
| --- | --- |
|  | PORTB\_PCR5 = 0x100u; |

Utilizando as máscaras e *structs* fornecidas, o código acima se transforma em:

|  |  |
| --- | --- |
|  | PORTB\_BASE\_PTR-> PCR[5] = PORT\_PCR\_MUX(1); |

Deste modo o código mínimo de configuração do sistema, levando em conta os periféricos que devem ser minimamente inicializados para o bom funcionamento do microcontrolador, é apresentado abaixo.

|  |  |
| --- | --- |
|  | //Autor: Gabriel Silva Marcatto  //    Rodrigo Almeida  #include "MKL05Z4.h"  #define bitSet(arg,bit) ((arg) |= (1<<bit))  #define bitClr(arg,bit) ((arg) &= ~(1<<bit))  static int i = 0;  void main(void){  //-------------------------Config para boot minimo da placa -------------------------------------------------------     MCG\_BASE\_PTR ->C4 |= MCG\_C4\_DMX32\_MASK;     SIM\_BASE\_PTR ->SCGC5 |= (SIM\_SCGC5\_PORTA\_MASK | SIM\_SCGC5\_PORTB\_MASK);     PORTB\_BASE\_PTR-> PCR[5] = PORT\_PCR\_MUX(1);     SIM\_BASE\_PTR -> COPC &=~ SIM\_COPC\_COPT\_MASK;  //-----------------------------------------------------------------------------------------------------------     PORTB\_BASE\_PTR ->PCR[10] = (PORT\_PCR\_MUX(1) | PORT\_PCR\_DSE\_MASK);     bitSet(PTB\_BASE\_PTR ->PDDR, 10);     for(;;){         bitSet(PTB\_BASE\_PTR ->PDOR, 10);         for(float i = 0; i<100000; i++);         bitClr(PTB\_BASE\_PTR ->PDOR, 10);         for(float i = 0; i<100000; i++);     }  } |