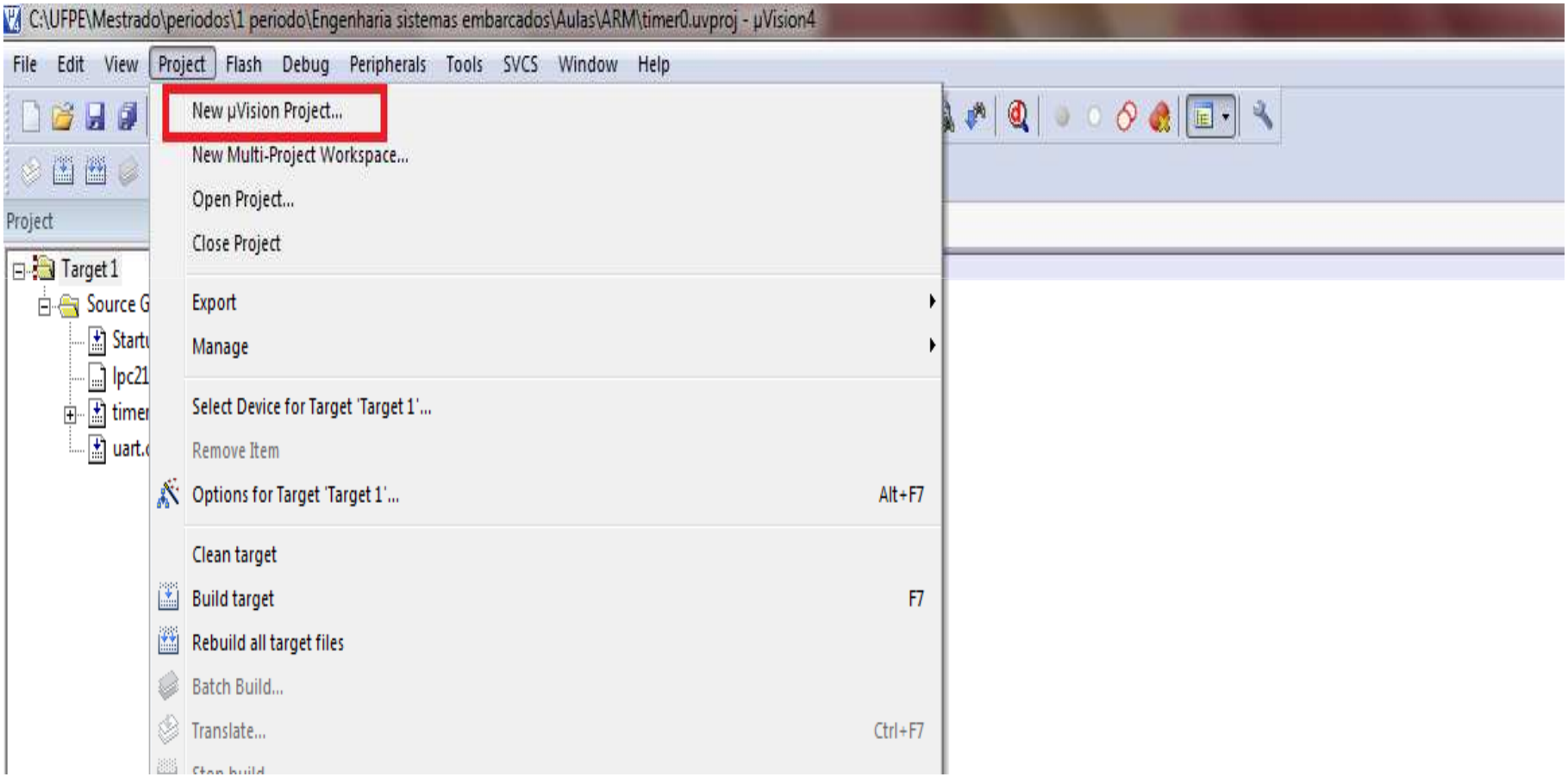
Programação ARM

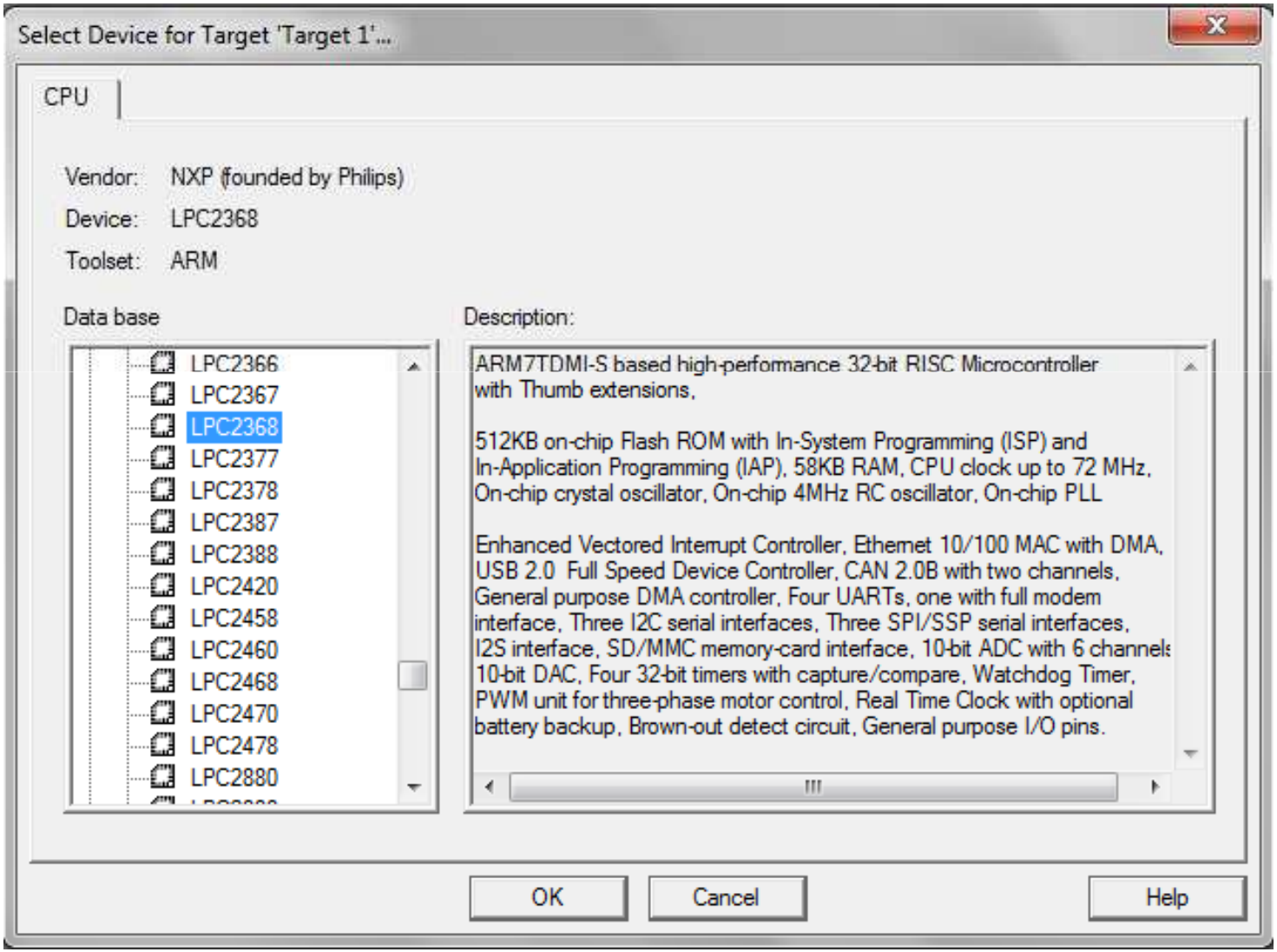
Engenharia de Sistemas Embarcados

Prof. Abel Guilhermino

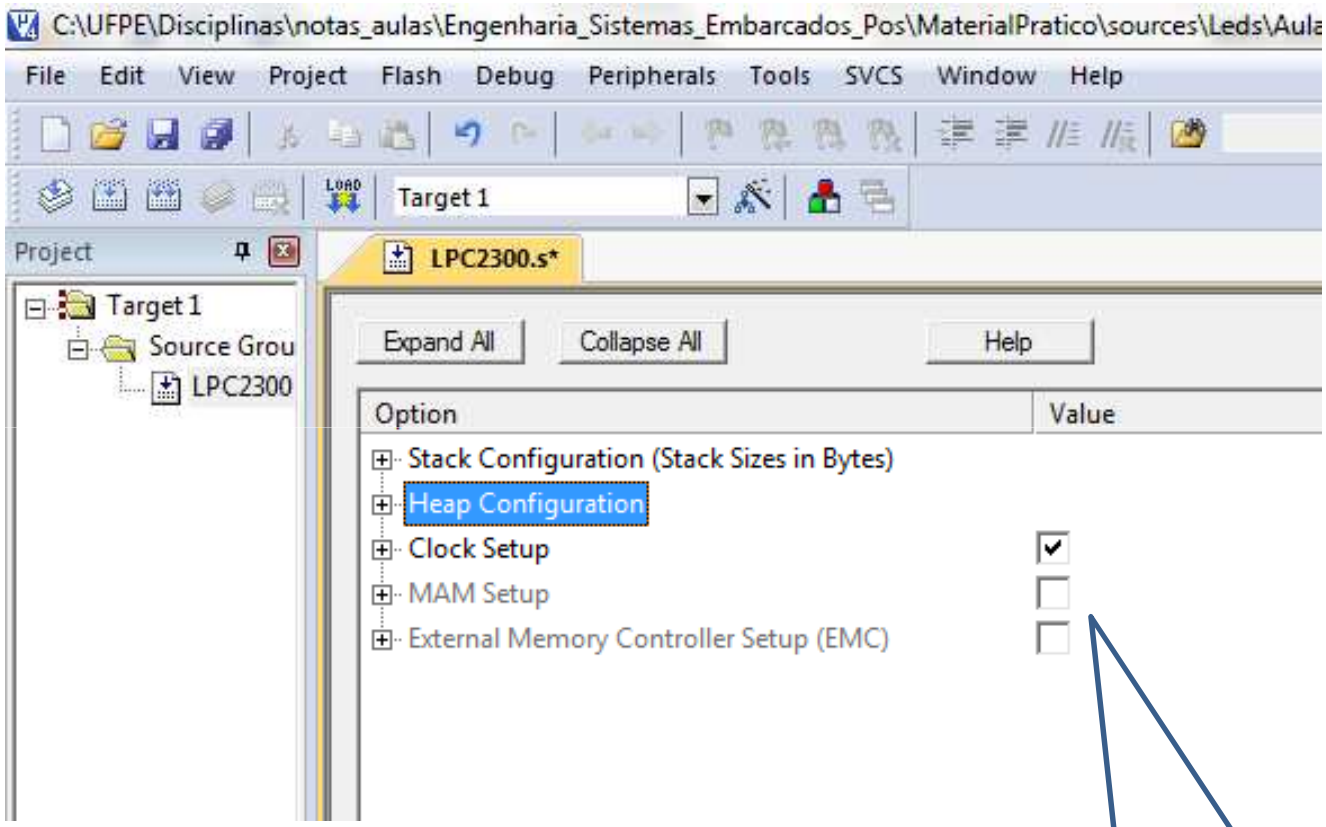
Criação de Novo Projeto



Criação de Novo Projeto

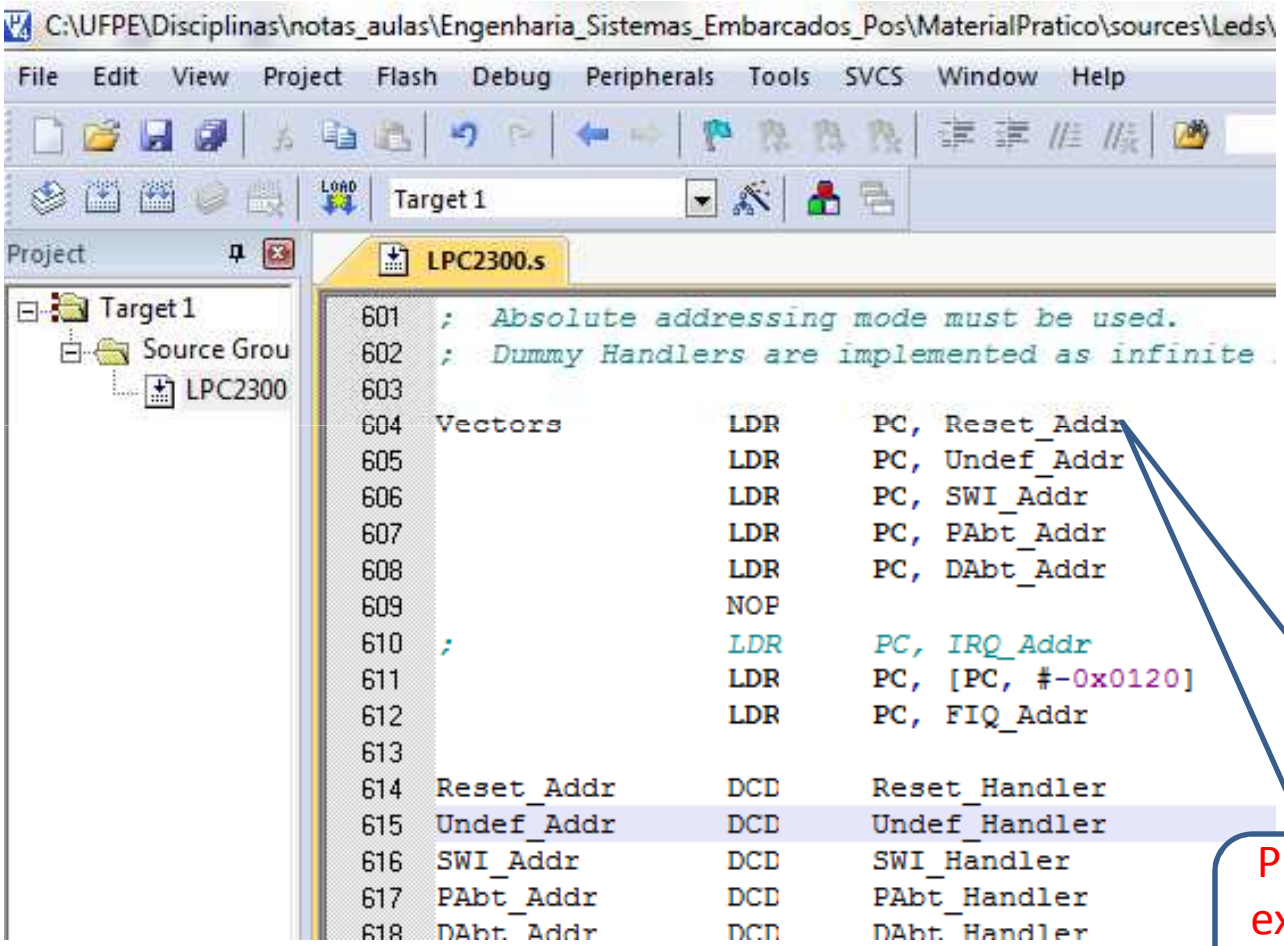


Criação de Novo Projeto



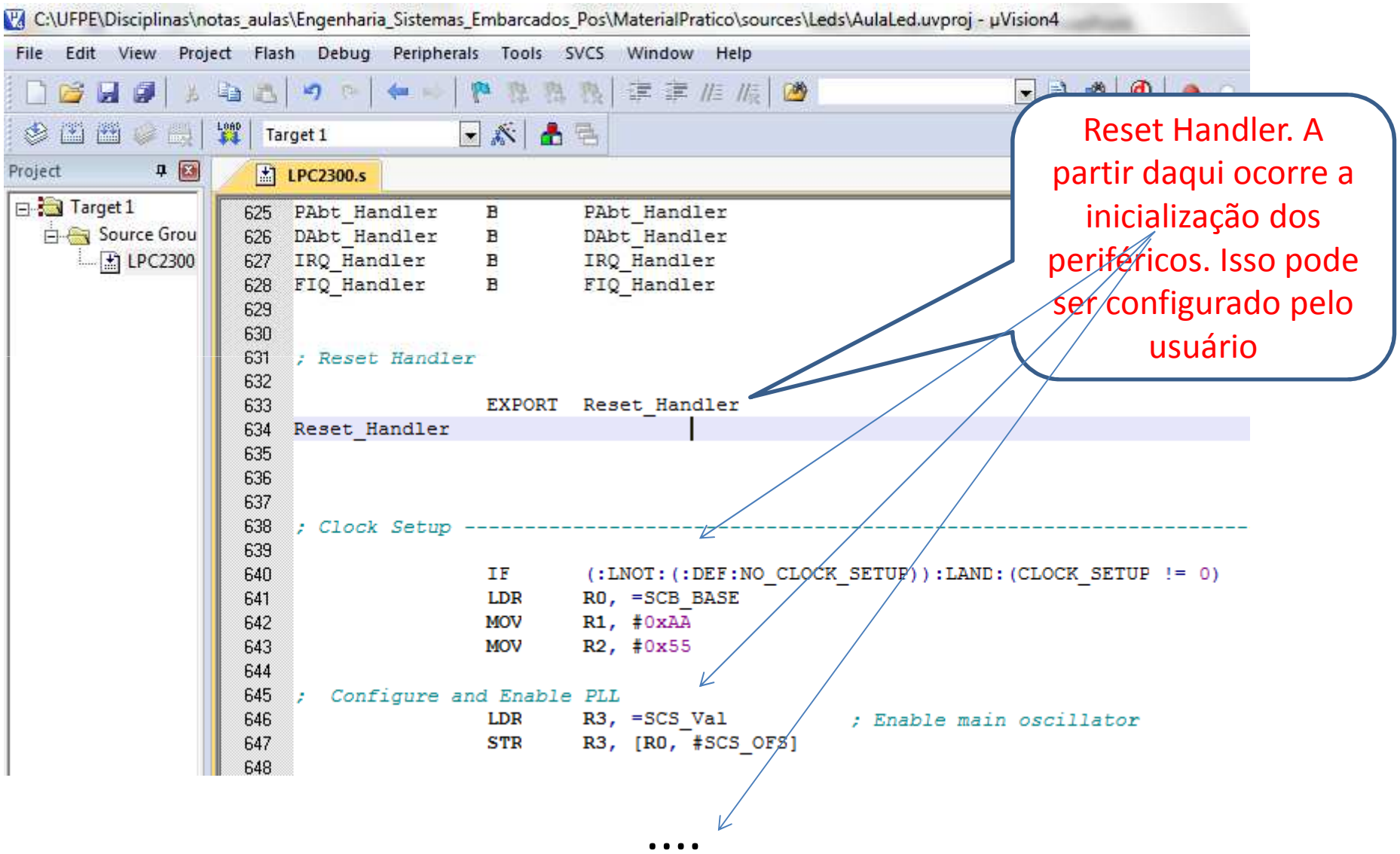
Desmarcar MAM e External Memory e Salvar

Criação de Novo Projeto

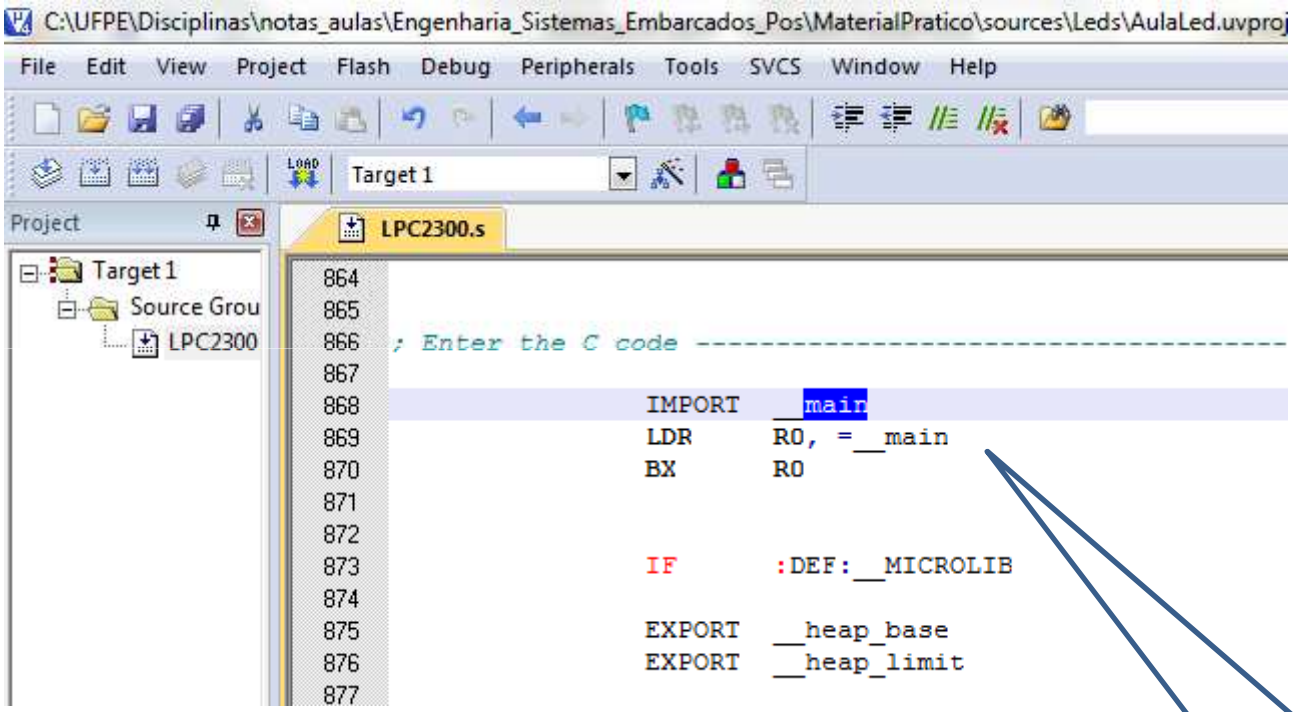


Primeira instrução a ser executada -> Jump para (0x0000 0000) = Reset Handler

Criação de Novo Projeto

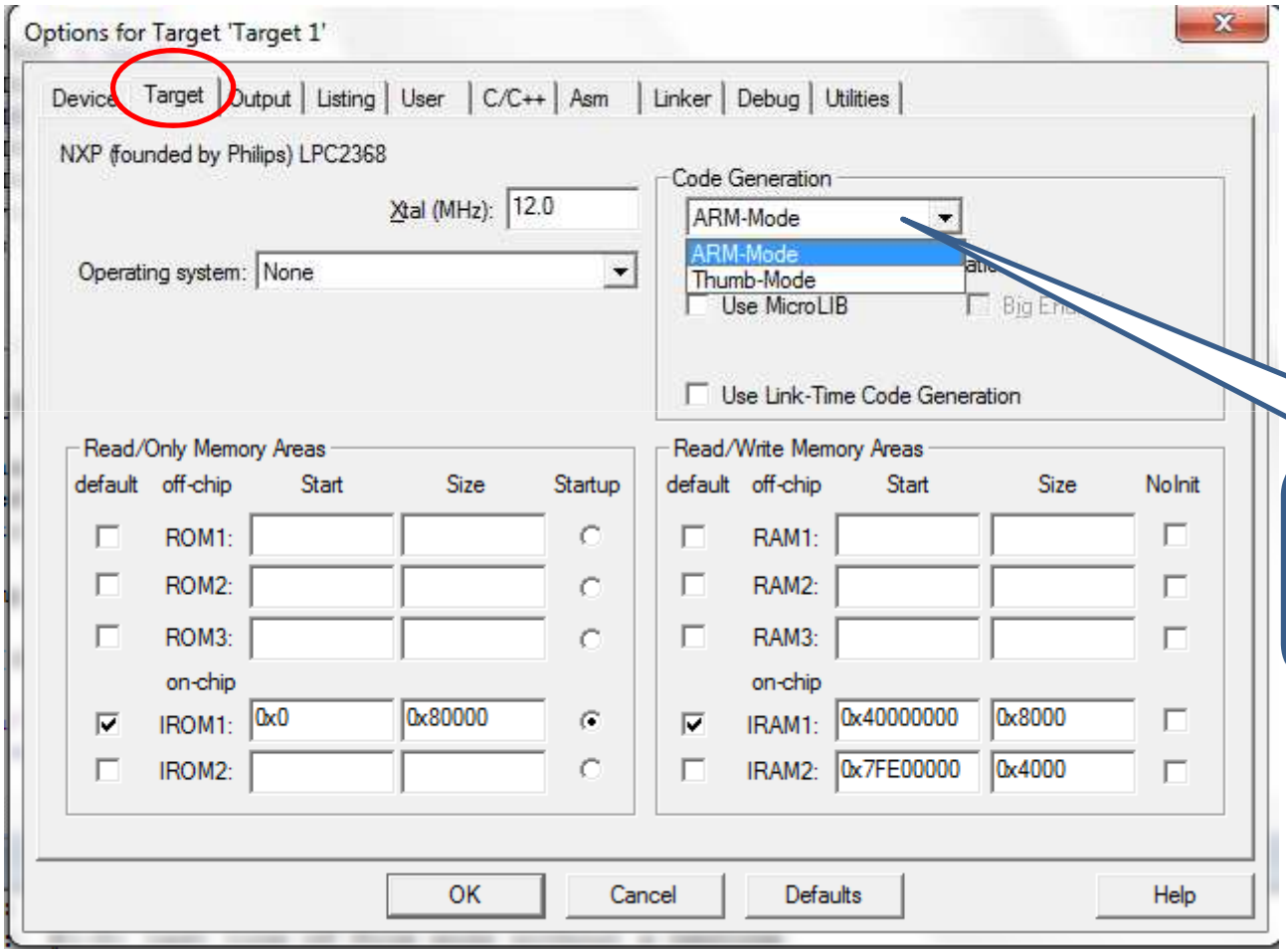


Criação de Novo Projeto



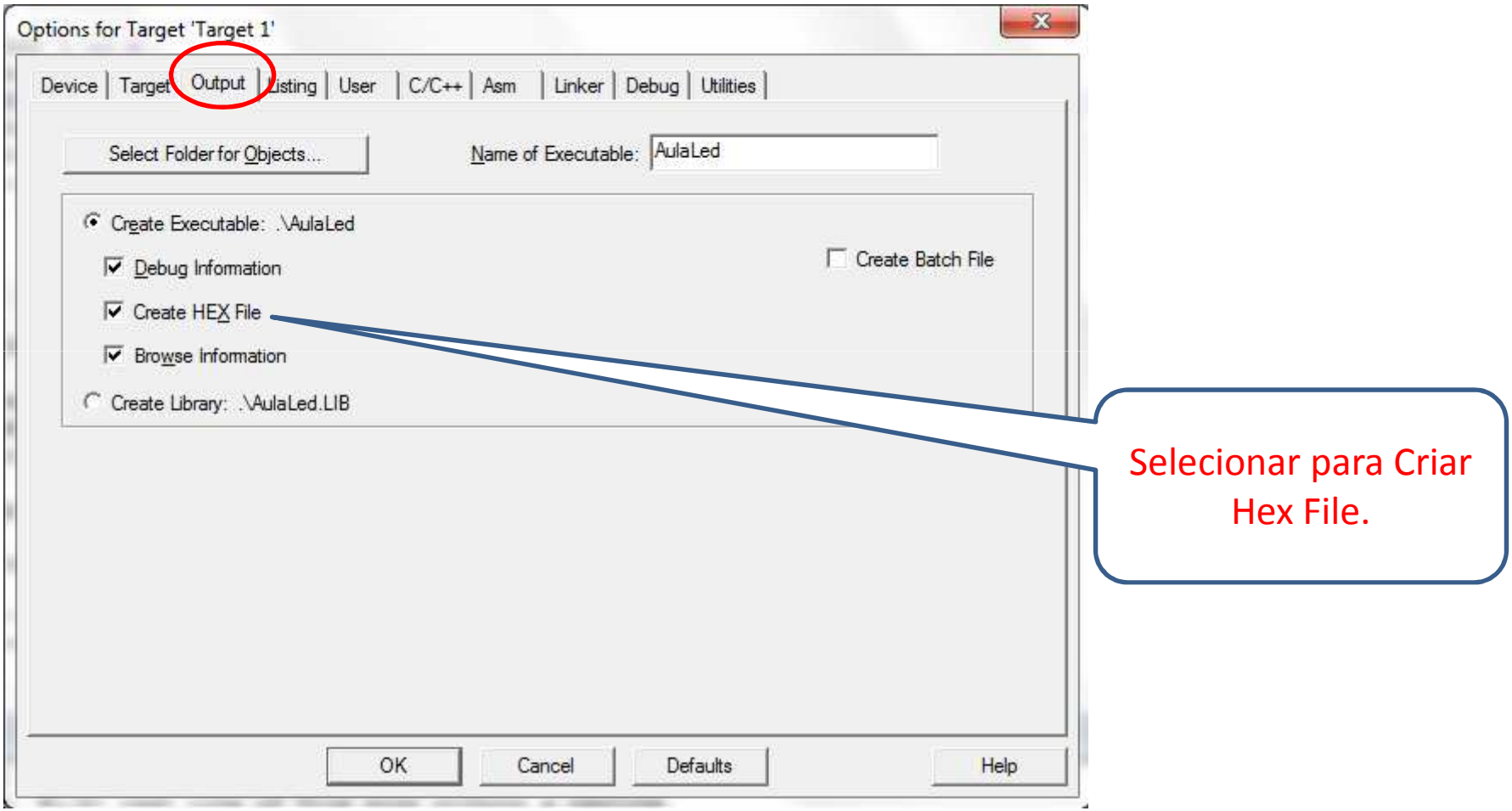
Jump para Código do Usuário (default na Flash)

Criação de Novo Projeto

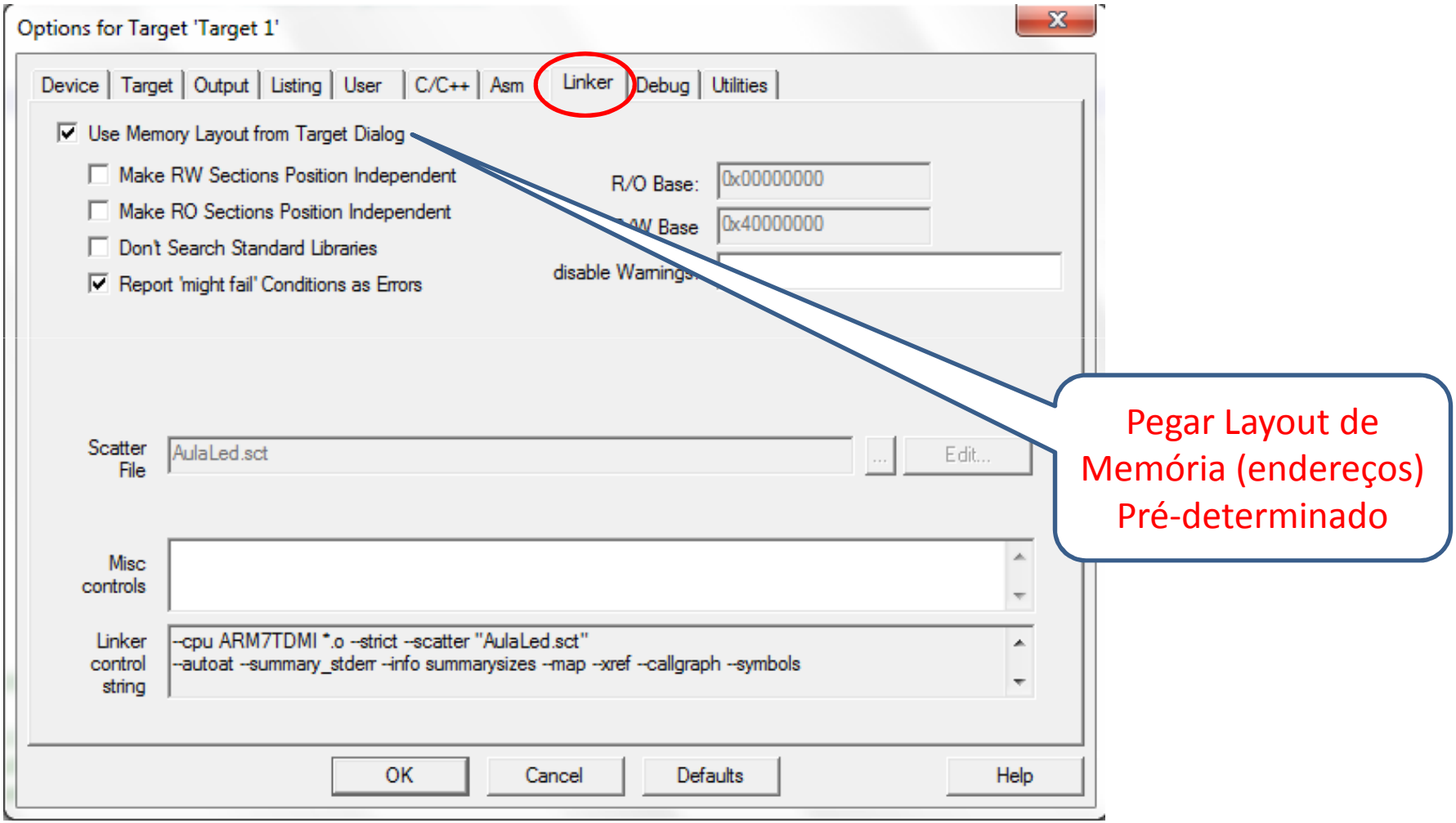


Alterar Geração de Código ARM/THUMB

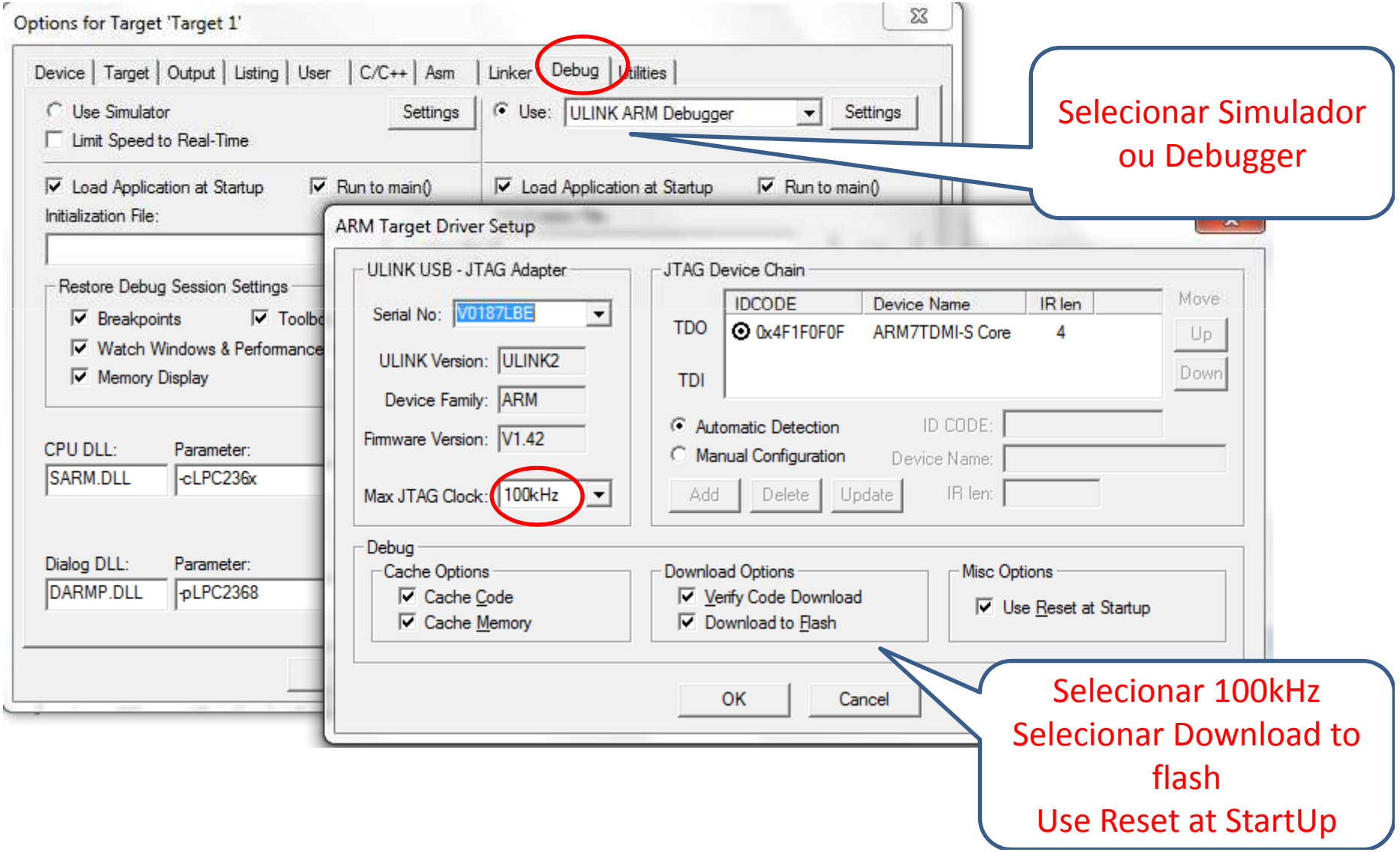
Criação de Novo Projeto



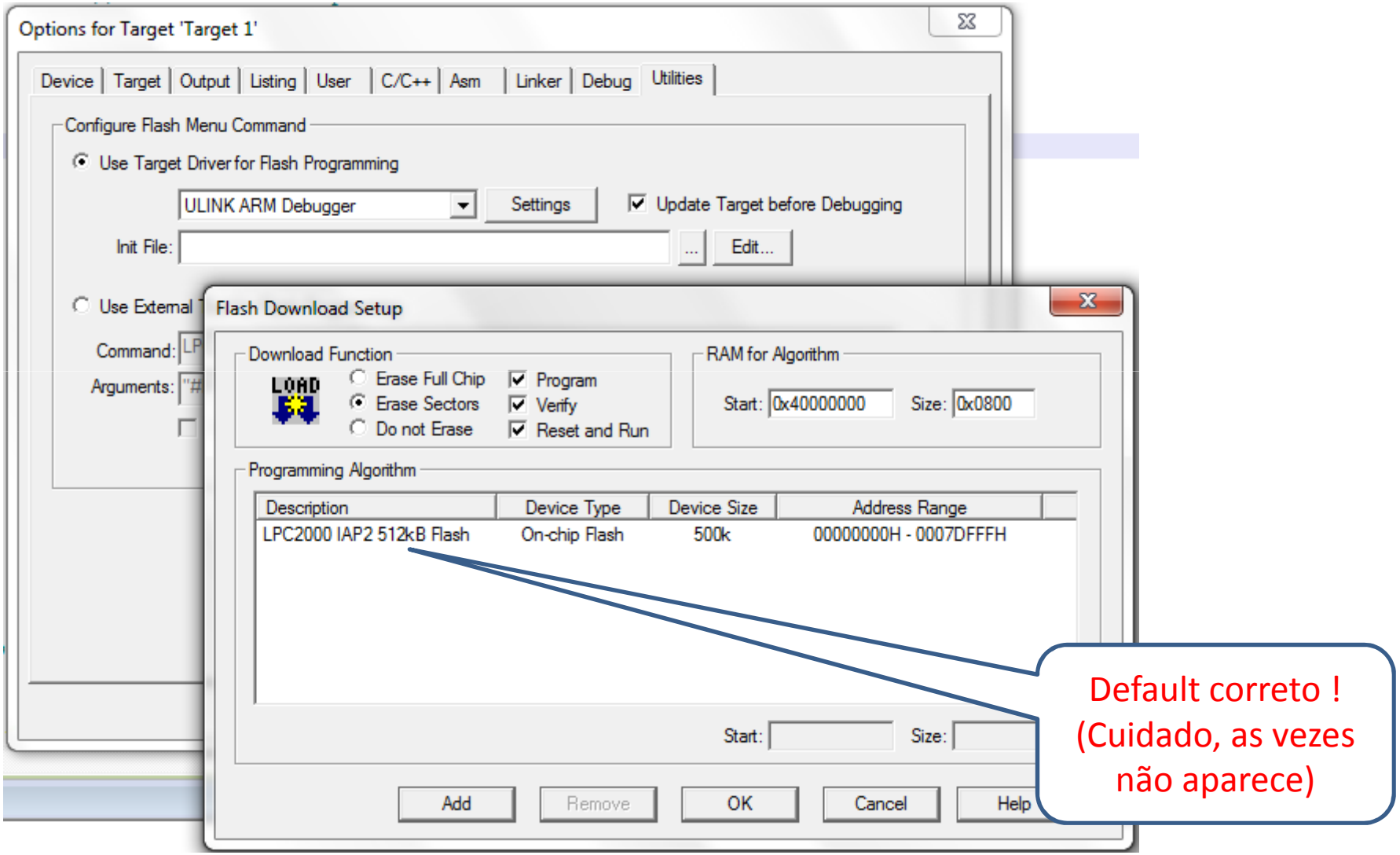
Criação de Novo Projeto



Criação de Novo Projeto



Criação de Novo Projeto



Ligando LED (FIO2SET)

```
001  #include "lpc23xx.h"    // header definido pela keil
002
003  // Na placa da Keil tem disponível LEDS de P2.0 até P2.7
004
005
006
007
008  void main(){
009  int a;
010
011  // Definir as direções dos pinos    (0 = Entrada,  1
012  FIO2DIR = 0xFFFFFFFF;                //(FastIO po
013
014  // Para colocar 1 em um pino de saíd
015  a = 0x00000001;
016  FIO2SET =  a ;
017
018
019  FIO2SET = 0x01;    // Ligar Led P2.0
020  FIO2SET = 0x02;    // Ligar Led P2.1 e mantém estado anterior
021
022
023
024  while(1){
025
026  }
027
028
029  }
```

HEXAdecimal

Uso da Porta P2

FIO2SET, Seta em 1 o pino.

Ligando LED (FIO2SET)

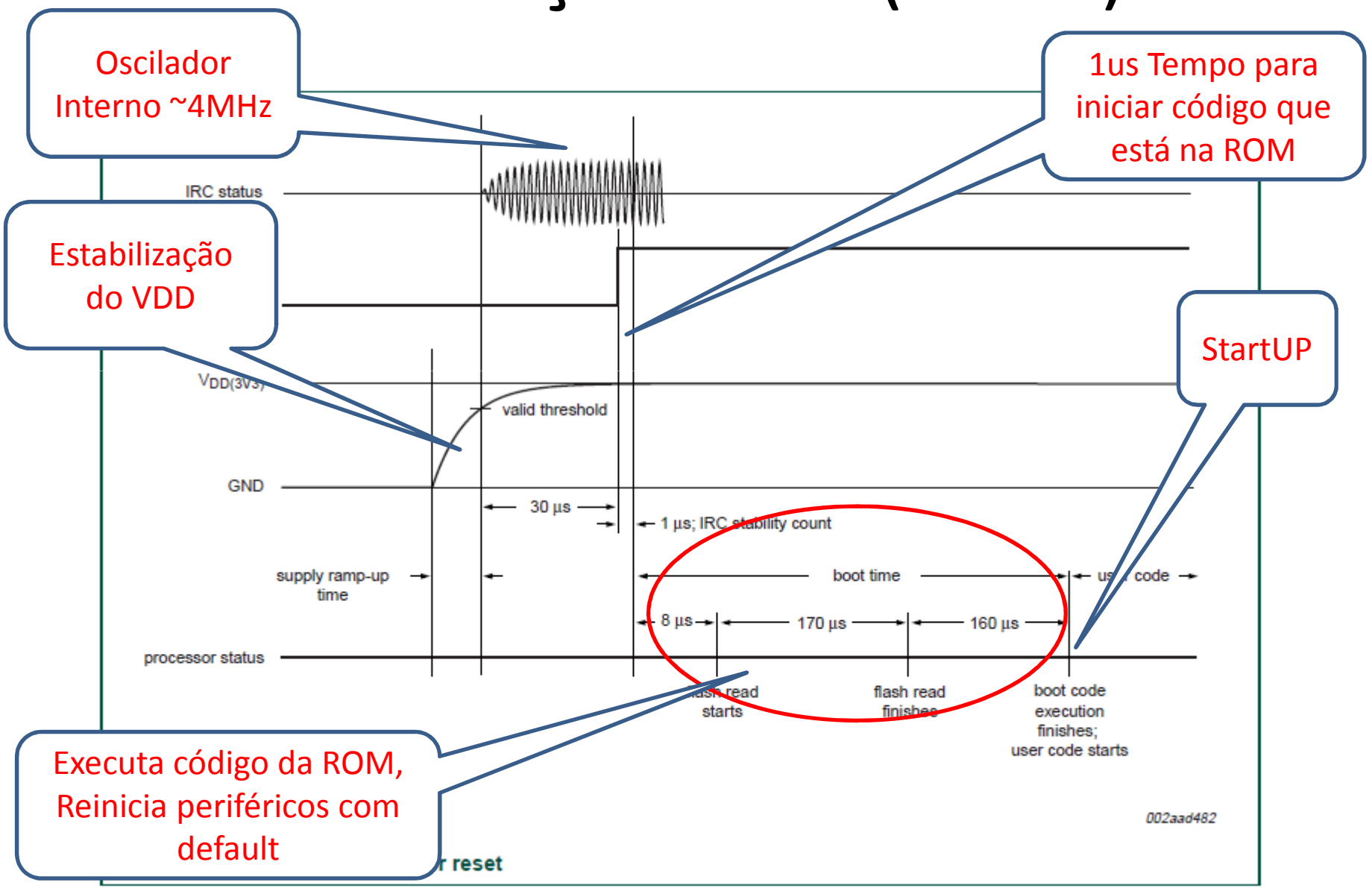
```
007
008 void main(){
009 //int a;
010 char apenas_byte;
011
012 // Definir as direções dos pinos    (0 = Entrada,  1 = saída)
013 FIO2DIR = 0xFFFFFFFF;               //(FastIO porta 2)
014
015 // Para colocar 1 em um pino de saída
016 a = 0x00000001;
017 FIO2SET =  a ;
018
019
020 FIO2SET = 0x01;    // Ligar Led P2.0
021 FIO2SET = 0x02;    // Ligar Led P2.1 e mantém estado anterior
022
023 FIO2DIR = FIO2DIR | (0x00000001 < 1);  // Pega apenas o segundo bit menos
024                                         // significativo e diz que é de saída
025
026 // Pegar o conteúdo da Porta 2
027 a = FIO2PIN;
028
029 // Máscara
030 apenas_byte = 0x000000FF & a;
031
032
033 while(1){
```

Setar apenas um bit da porta

Obter conteúdo de uma porta

Obter 1 byte de 32 bits

Inicialização ARM (Reset)



PLL

Frequência do PLL

Frequência de entrada do Core

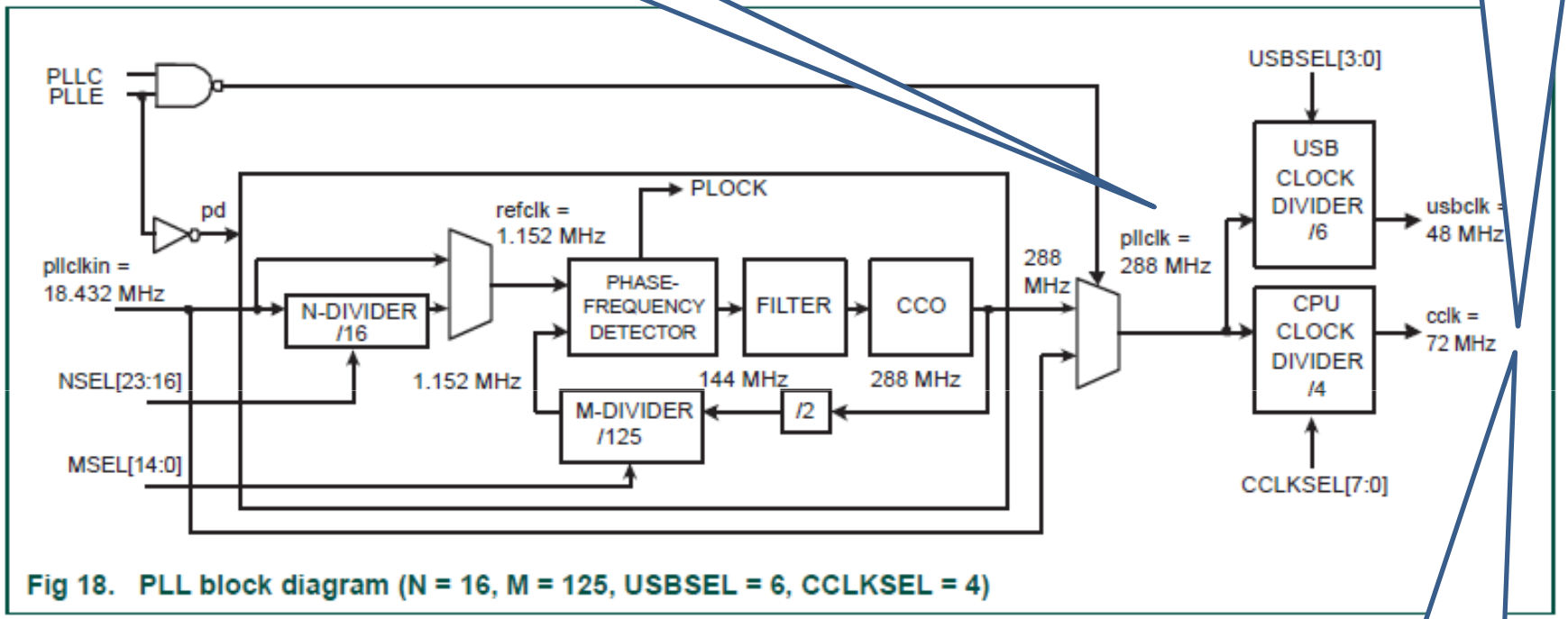


Fig 18. PLL block diagram (N = 16, M = 125, USBSEL = 6, CCLKSEL = 4)

Valor pode ser configurado no debugger

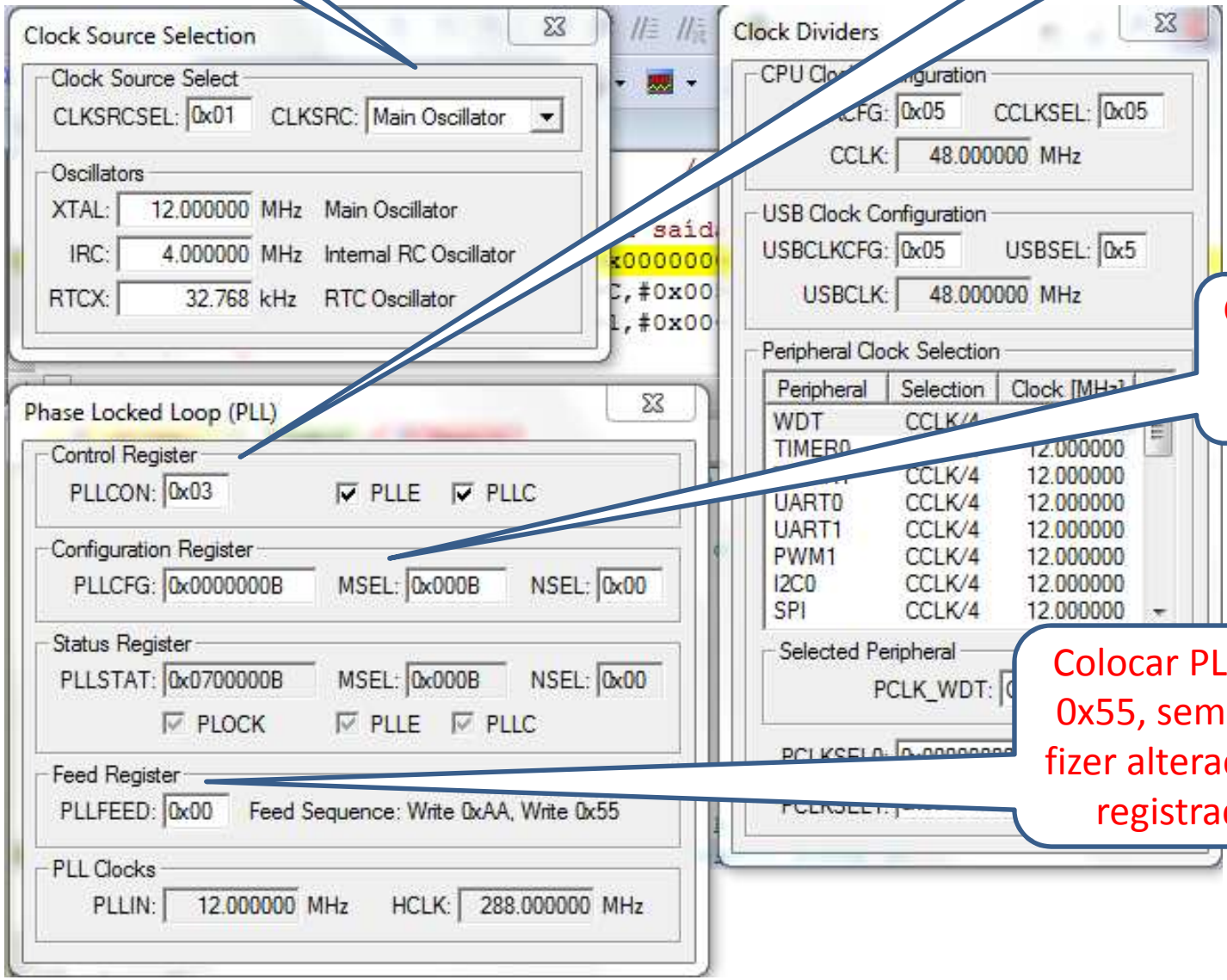
FCCO = 2*(pllclkin * (MSEL+1))/(NSEL+1)

FCCO = 2*(12MHz * (11+1))/(0+1) = 288MHz

PLL (Debugger)

Definir qual vai ser a entrada do PLL (Há 3 opções)

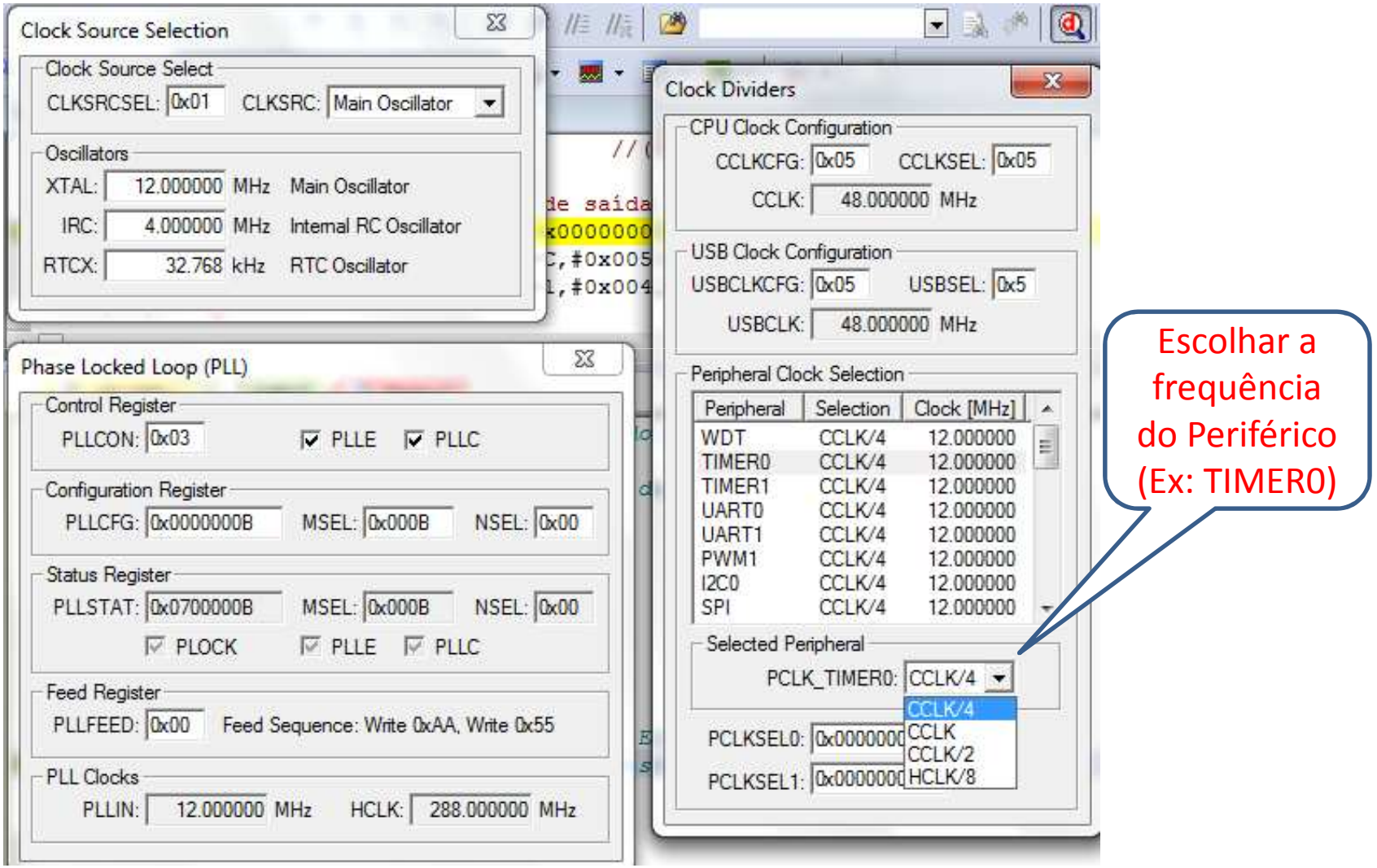
Precisa desconetar o PLL para configurar



Configurar: MSEL e NSEL

Colocar PLLFEED = 0x55, sempre que fizer alterações nos registradores

PLL (Debugger)



Timer

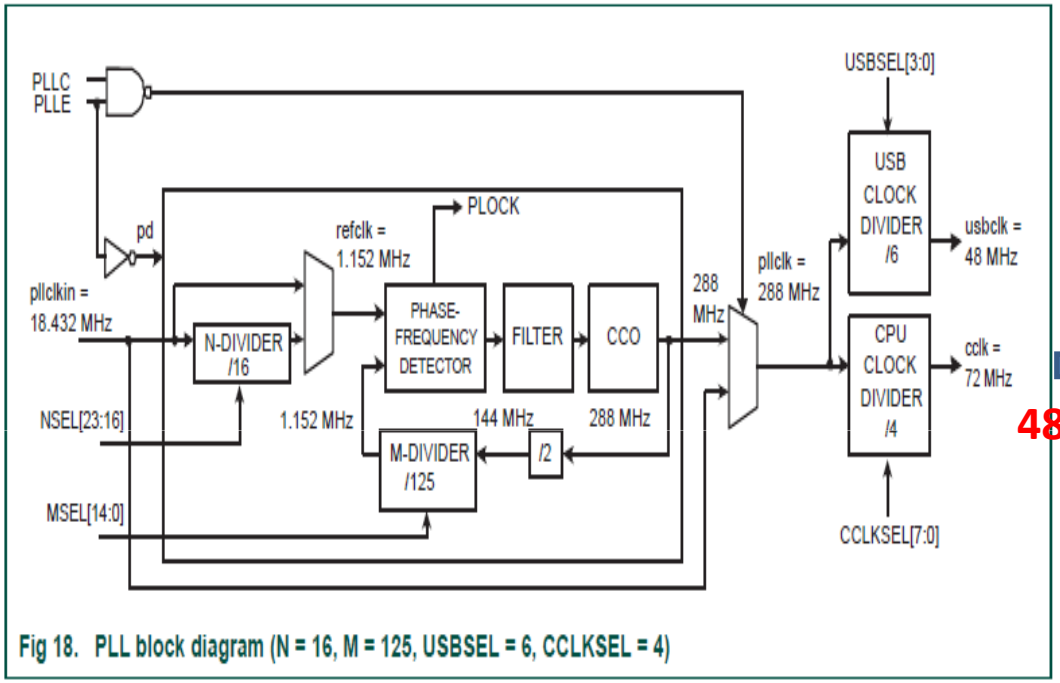


Fig 18. PLL block diagram (N = 16, M = 125, USBSEL = 6, CCLKSEL = 4)

Registrador:
Divisor de Freq.
Para o periférico

00 = CCLK/4
01 = CCLK
10 = CCLK/2
11 = CCLK/8
(*CAN cclk/6)

48MHz

PCLKSEL0

Se '00' =>
48/4=12MHz

Fixo, 32 bits

T0PR

Contador 32 bits de 0 até atingir T0PR

T0PC

0 TRPR

Se TOPC = T0PR

Timer

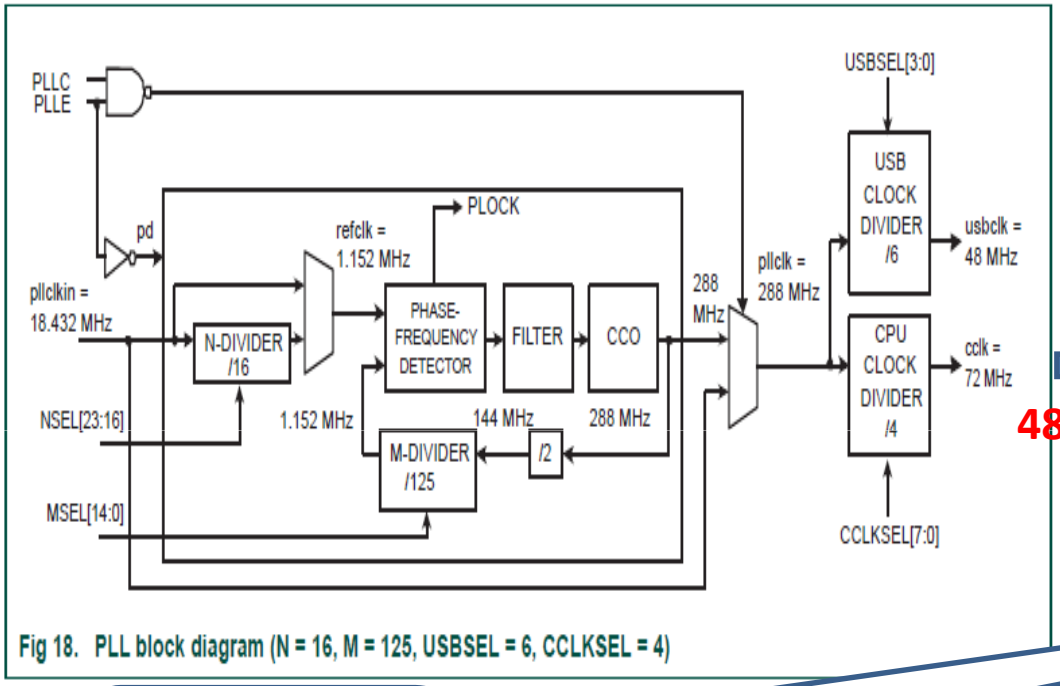


Fig 18. PLL block diagram (N = 16, M = 125, USBSEL = 6, CCLKSEL = 4)

Registrador:
Divisor de Freq.
Para o periférico

00 = CCLK/4
01 = CCLK
10 = CCLK/2
11 = CCLK/8
(*CAN cclk/6)

48MHz

PCLKSEL0

Se '00' =>
48/4=12MHz

Fixo, 32 bits

T0PR

T0PR=11

1MHz

T0PC

0 TRPR

Se TOPC = T0PR

Supor 1MHz:
Fout = Fin/C
1MHz=12Mhz/C
C=12

Timer

Registrador:
Divisor de Freq.
Para o periférico

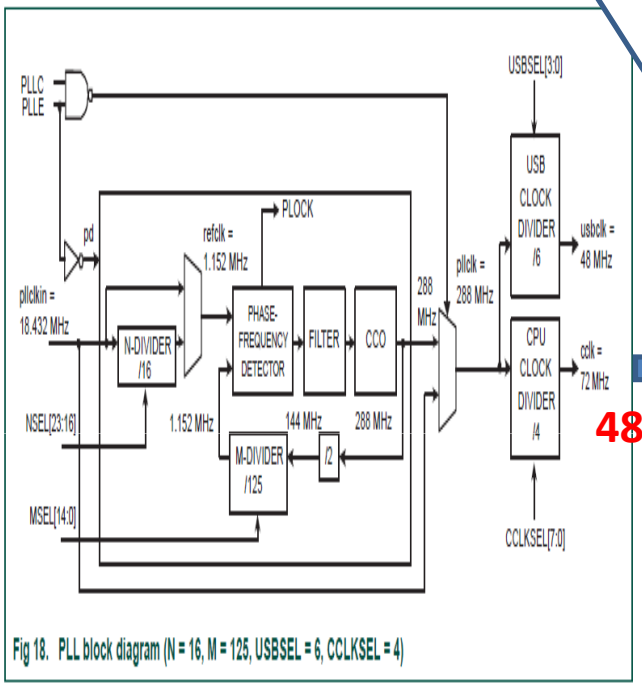


Fig 18. PLL block diagram (N = 16, M = 125, USBSEL = 6, CCLKSEL = 4)

00 = CCLK/4
01 = CCLK
10 = CCLK/2
11 = CCLK/8
(*CAN cclk/6)

48MHz

PCLKSEL0
Se '00' =>
48/4=12MHz

Fixo, 32 bits

T0PR
T0PR=11

1MHz

Contador

T0TC

Interrupt

Precisa zerar contador via software com 0

T0PC
0 TRPR

Se TOPC = T0PR

T0MR0
T0MR1
T0MR2
T0MR3

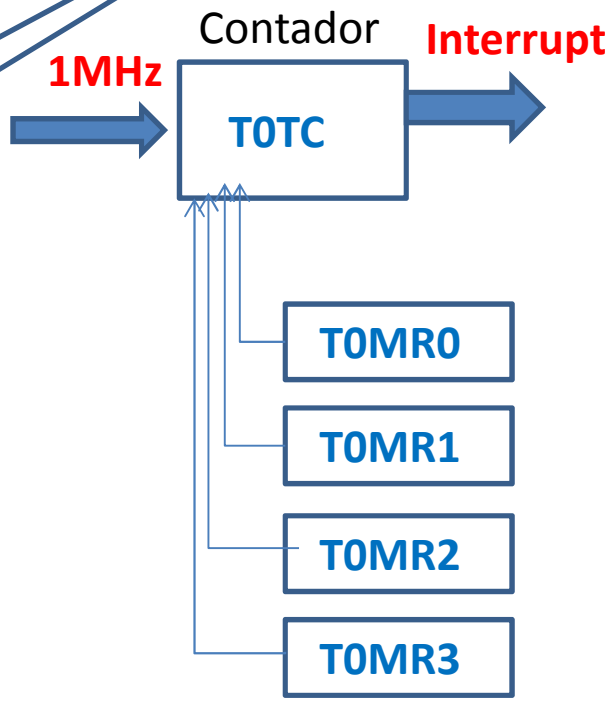
Fixo, 32 bits

T0MCR

```
// Match Control Register
//  ___________________________________________________________________________________
// |                                                                                   |
// |   15:12     |  11    10     9  |  8      7     6  |  5      4      3  |  2     1     0  |
// | RESERVADO   | MR3S |MR3R | MR3I | MR2S | MR2R | MR2I | MR1S |MR1R | MR1I | MR0S | MR0R | MR0I |
// |___________________________________________________________________________________|
// MRNS :  1 -> Para de contar quando ocorre se iguala ao registrado MRN
//         0 -> Não para de conta quando o valor se iguala ao valor contido no registrador MRN
// MRNR :  1 -> O registrador T0TC é Reiniciado quando valor se iguala ao do registrado MRN
//      :  0 -> O registrador T0TC não é reiniciado quando o valor se iguala ao do registrador MRN
// MRNI :  1 -> Gera interrupção
//      :  0 -> Não gera interrupção


T0MCR = 0x00000003;
```

Cada 3 bits de T0MCR indica a ação que o Timer deve fazer



Configuração de Interrr. Timer

```
int main (){
  T0PR = 0x000005;  //  Até onde o TRPC vai contar
  T0CTCR = 0x00;  // Timer Mode
  T0TCR = 0x02;    // Desativa Timer, Não incrementa

  // Gera interrup, T0TC é reiniciado qdo valor iguala a MRN
  T0MCR = 0x00000003;  //  Controle
  T0MR0 = 0x0000FFFF;

 // Possui o endereço da rotina de tratamento
  VICVectAddr4 = (unsigned ) ISR;

 // VICVectPriority4  (3..0)=> Prioridade variando de 0 a 16.
  VICVectPriority4 = 0x0000000024;

 // Habilitar Quinto bit do registrador, manter os outros (timer)
   VICIntEnable = VICIntEnable | 0x10;

 // Não resetar timer, Habilitar timer.
  T0TCR = 0x01;
  while(1){ }
}
```

```
void ISR (void) __irq {
unsigned char IR_TMP = 0;
static int pin_set = 0, flag = 0 ;
IR_TMP = T0IR;     FIO2CLR = 0xFFFFFFFF;

if(flag == 0) {
      FIO2SET = 1 << pin_set;
      pin_set++;  }
  else  {
      FIO2SET = 1 << pin_set;
      pin_set--;   }
  if(pin_set == 7)
      flag=1;
      if(pin_set == 0)
        flag=0;

// Se tiver mais de uma interrução ativa
if ((IR_TMP & 0x01) == 0x01 )
  T0IR = 0x00000001;
if ((IR_TMP & 0x02) == 0x02 )
  T0IR = 0x00000010;                  // Limpa vetor

//Dummy Acknowledgement
VICVectAddr = 0x00000000;
}
```

Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
```

Seleciona o pino para utilização da UART0

Bit	Symbol	Value	Function
1:0	P0.0	00	GPIO Port 0.0
		01	TXD (UART0)
		10	PWM1
		11	Reserved
3:2	P0.1	00	GPIO Port 0.1
		01	RxD (UART0)
		10	PWM3
		11	EINT0
5:4	P0.2	00	GPIO Port 0.2
		01	SCL0 (I^2C0)
		10	Capture 0.0 (Timer 0)
		11	Reserved
7:6	P0.3	00	GPIO Port 0.3
		01	SDA0 (I^2C0)
		10	Match 0.0 (Timer 0)
		11	EINT1
9:8	P0.4	00	GPIO Port 0.4
		01	SCK0 (SPI0)
		10	Capture 0.1 (Timer 0)
		11	AD0.6
11:10	P0.5	00	GPIO Port 0.5
		01	MISO0 (SPI0)
		10	Match 0.1 (Timer 0)
		11	AD0.7

Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
//Uart Fifo control Register
U0FCR = 0x00;
//Uart Line Control Register
U0LCR = 0x83;
```

Registrador de Controle de Linha da UART

```
//Uart Line Control Register
//
//  _______________________________________________________________________________________________
// |      7       |      6        |      5:4       |       3         |       2         |    1:0      |
// |Divisor Latch | Break Control | Parity Select | Parity Enable   | Stop Bit Select | Word Length |
// |______________________________________________________________________________________________|
//Divisor Latch (DLAB) : 0 -> Desativa permissão para alterar U0DLL e U0DLM
//                     : 1 -> Ativa permissão para alterar U0DLL e U0DLM
//Break Control : 0 ->
//              : 1 ->
//Parity Select : 00 -> Paridade impa
//                01 -> Paridade par
//                10 -> ...
//                11 -> ...
//Parity Enable : 0 ->  Desativada paridade
//                1 ->  Ativda paridade
//
//Stop Bit Select : 0 -> 1 stop bit
//                  1 -> 2 stop bits
//Word Length : 00 ->  5 bit
//            : 01 ->  6 bit
//            : 10 ->  7 bit
//            : 11 ->  8 bit
```

É preciso ativar esta permissão para alterar em seguida o baudrate.

Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
//Uart Fifo control Register
U0FCR = 0x00;
//Uart Line Control Register
U0LCR = 0x83;
//Uart Divisor Latch register (Configurando para 9600 bps)
U0DLM = 0x00;
U0DLL = 0x4E;
```

$$UART0_{baudrate} = \frac{PCLK}{16 \times (16 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Table 102: Baudrates available when using 20 MHz peripheral clock (PCLK = 20 MHz)

Desired baudrate	MULVAL = 0 DIVADDVAL = 0 U0DLM:U0DLL hex[2]	dec[1]	% error[3]	Optimal MULVAL & DIVADDVAL U0DLM:U0DLL dec[1]	Fractional pre-scaler value MULDIV / (MULDIV + DIVADDVAL)	% error[3]
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
//Uart Fifo control Register
U0FCR = 0x00;
//Uart Line Control Register
U0LCR = 0x83;
//Uart Divisor Latch register (Configurando para 9600 bps)
U0DLM = 0x00;
U0DLL = 0x4E;
//Uart Line Control Register
U0LCR = 0x03;
//Interrupt Enable Register
U0IER = 0x00000001;
```

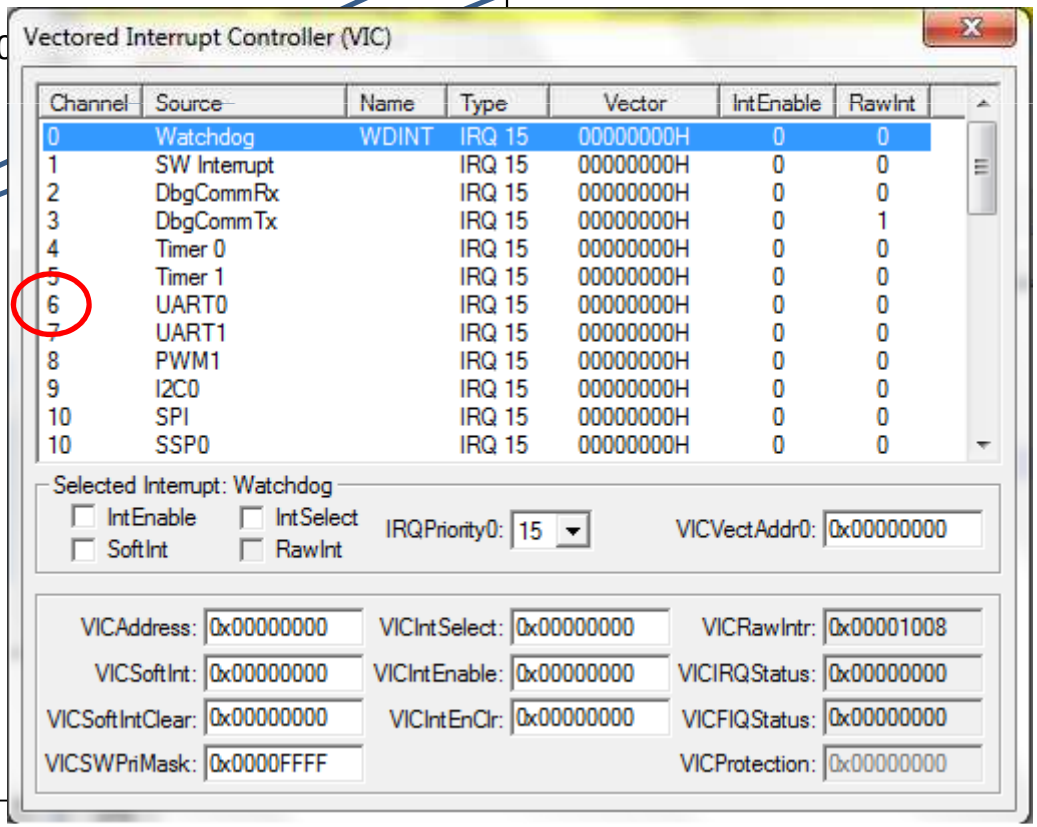
Neste exemplo, está sendo ativo interrupções para recebimento e desativando para envio.

```
// Interrupt Enable Register
//  ____________________________________________________________________________
// |   7 : 3  |                2                |          1          |          0          |
// | Reserved | Rx Line Status Interrupt Enable | THRE Interrupt Enable | RBR Interrupt Enable |
// |__________|_________________________________|_______________________|_____________________|
//Rx Line Status Interrupt Enable : 0 -> Desativa interrupções referente ao status
// Thre Interrupt Enable : 0 -> Desativda Interrupções para envio
//                         1 -> Ativa interrupções para envio
// RBR interrupt Enable : 0 -> Desativa interrupções para recebimento
//                        1 -> Ativa interrupções para recebimento
```

Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
//Uart Fifo control Register
U0FCR = 0x00;
//Uart Line Control Register
U0LCR = 0x83;
//Uart Divisor Latch register (Configurando para 960
U0DLM = 0x00;
U0DLL = 0x4E;
//Uart Line Control Register
U0LCR = 0x03;
//Interrupt Enable Register
U0IER = 0x00000001;
// Possui o endereço da rotina de tratamento
VICVectAddr6 = (unsigned ) ISR_UART;
```

Registrador de 32 bits.
Cada interrupção tem uma numeração (serial =6)



Configuração Serial

```
void confUart() {
// Seleciona o pino para utilização da UART0
PINSEL0 = 0x00000050;   //P0.0  TxD0 = 01,   P0.1  RxD0 = 01
//Uart Fifo control Register
U0FCR = 0x00;
//Uart Line Control Register
U0LCR = 0x83;
//Uart Divisor Latch register (Configurando para 9600 bps)
U0DLM = 0x00;
U0DLL = 0x4E;
//Uart Line Control Register
U0LCR = 0x03;
//Interrupt Enable Register
U0IER = 0x00000001;
// Possui o endereço da rotina de tratamento
VICVectAddr6 = (unsigned ) ISR_UART;
// Registrador indica prioridade
VICVectPriority6 = 0x00000001;
// Interrupt Enable Register
VICIntEnable = VICIntEnable | 0x40;  // Mantenho os anteriores
}
```

Habilitando a interrupção da Serial no registrador de 32 bits
(0x40)hex = (1000000)bin

...

10	9	8	7	6	5	4	3	2	1	0
SPI0	I2C0	-	UART1	UART0	TIMER1	TIMER0	ARMCORE1	ARMCORE0	-	WDT

Rotina da Interrupção Serial

```
#include "lpc23xx.h"
void ISR_UART (void) __irq
{
  char char_tmp = 0x00;
  unsigned int uar_intr = 0;
  uar_intr = U0IIR;                // Ler status de quem gerou a interrupção

  char_tmp = U0RBR;          // Dado recebido

  switch(char_tmp)
  {
      case 'a':     active = 1;   // Para habilitar algo
                    break;
      case 'd':
                   active = 0;    // Para desabilitar algo
                   break;
  }
  VICVectAddr = 0x00000000;
}
```

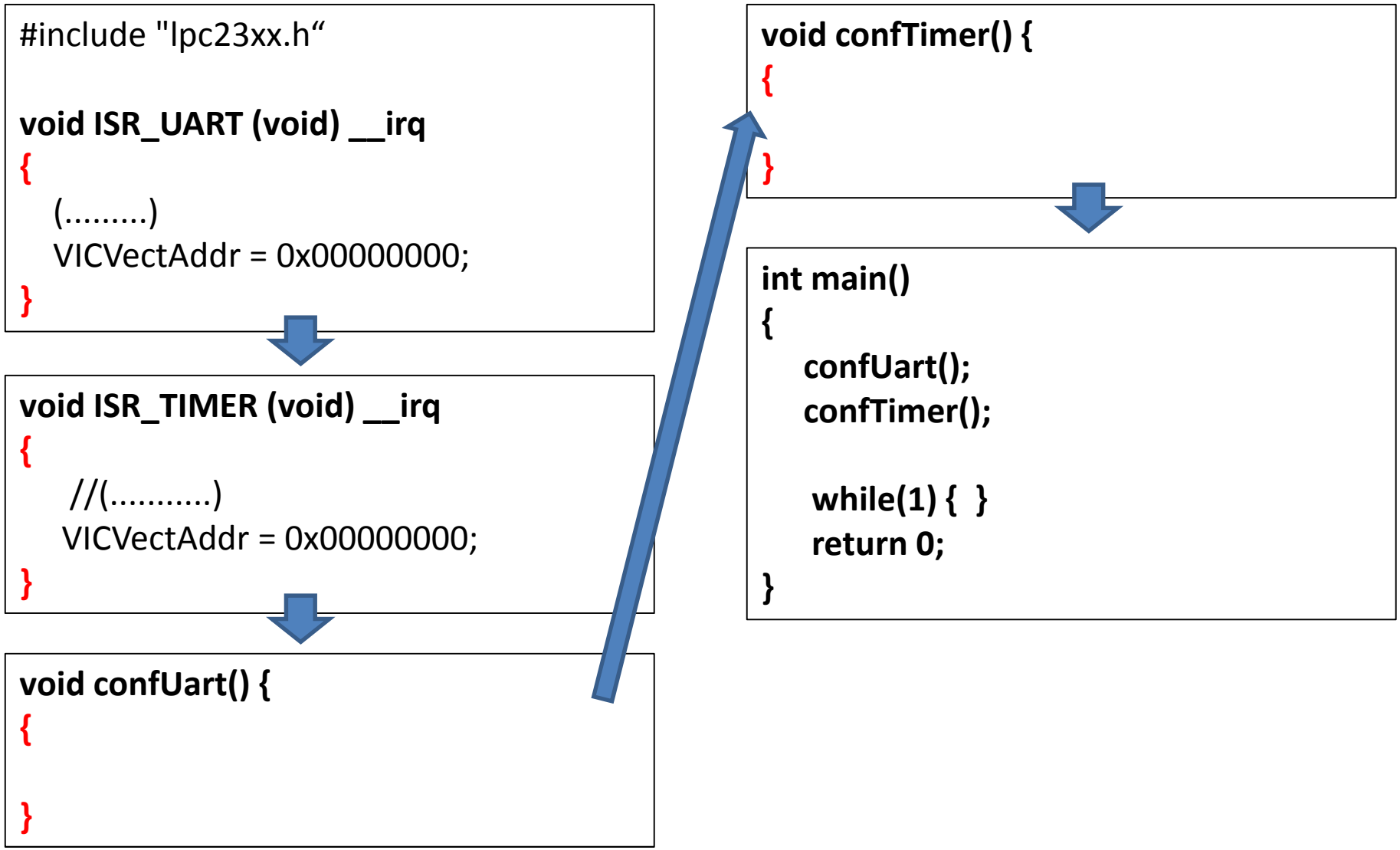
Rotina da Interrupção Timer

```
void ISR_TIMER (void) __irq
{
 unsigned char IR_TMP = 0;
 static int pin_set = 0, flag = 0;

 if(active == 1)
 {
   FIO2DIR = 0xFFFFFFFF;        // Porta 2 como saída
   FIO2CLR = 0xFFFFFFFF;        // Seto todos os pinos para 0
   if(flag == 0) {
         FIO2SET = 1 << pin_set;
         pin_set++;
   }
   else {
         FIO2SET = 1 << pin_set;
         pin_set--;
   }

  if(pin_set == 7)
      flag=1;
   if(pin_set == 0)
      flag=0;
 }
T0IR = 0x00000001;                 // Limpar vetor de interrup.
VICVectAddr = 0x00000000;
}
```

Estrutura do Case



Ambiente p/ Teste com Serial