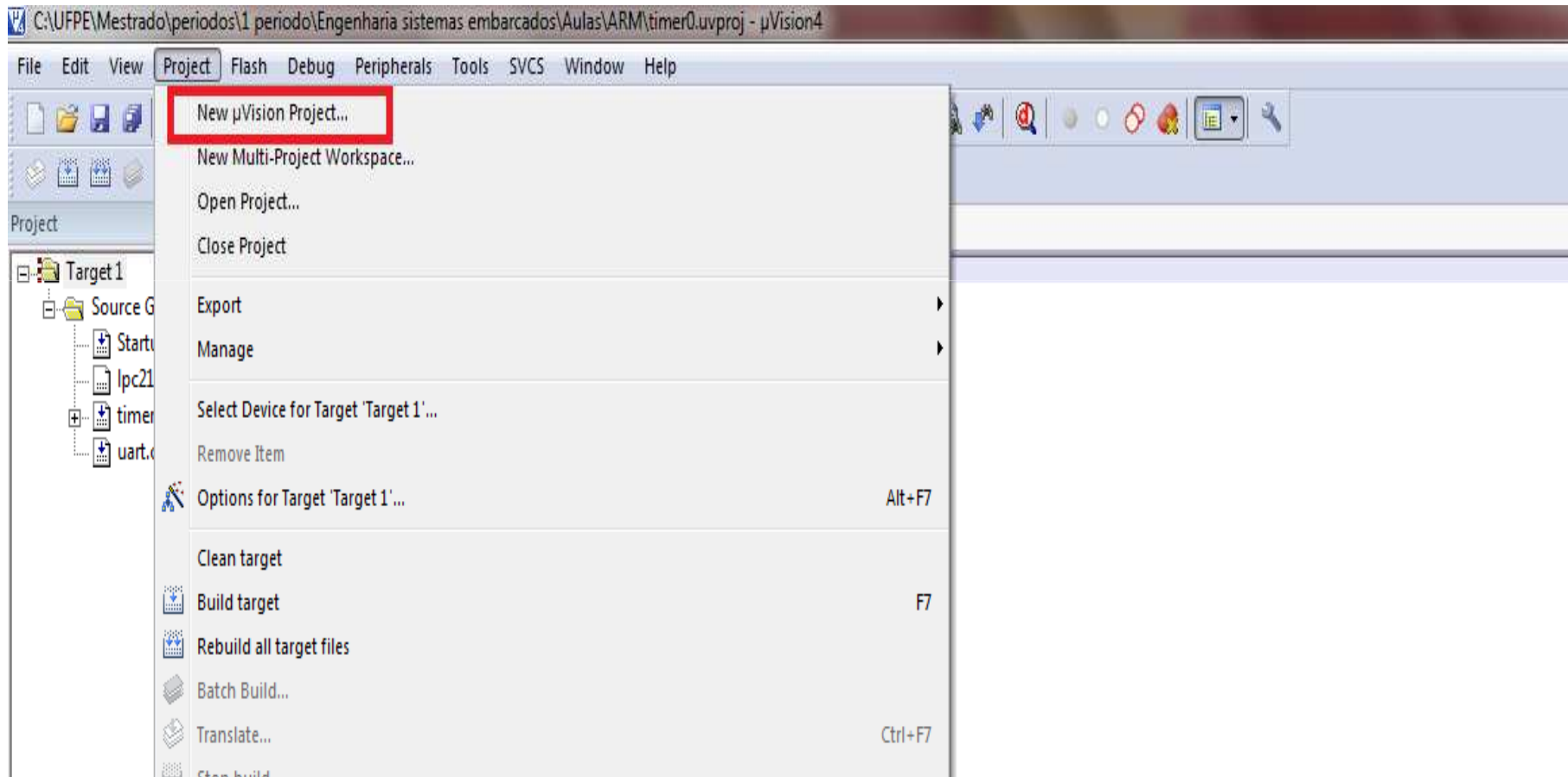


# Programação ARM

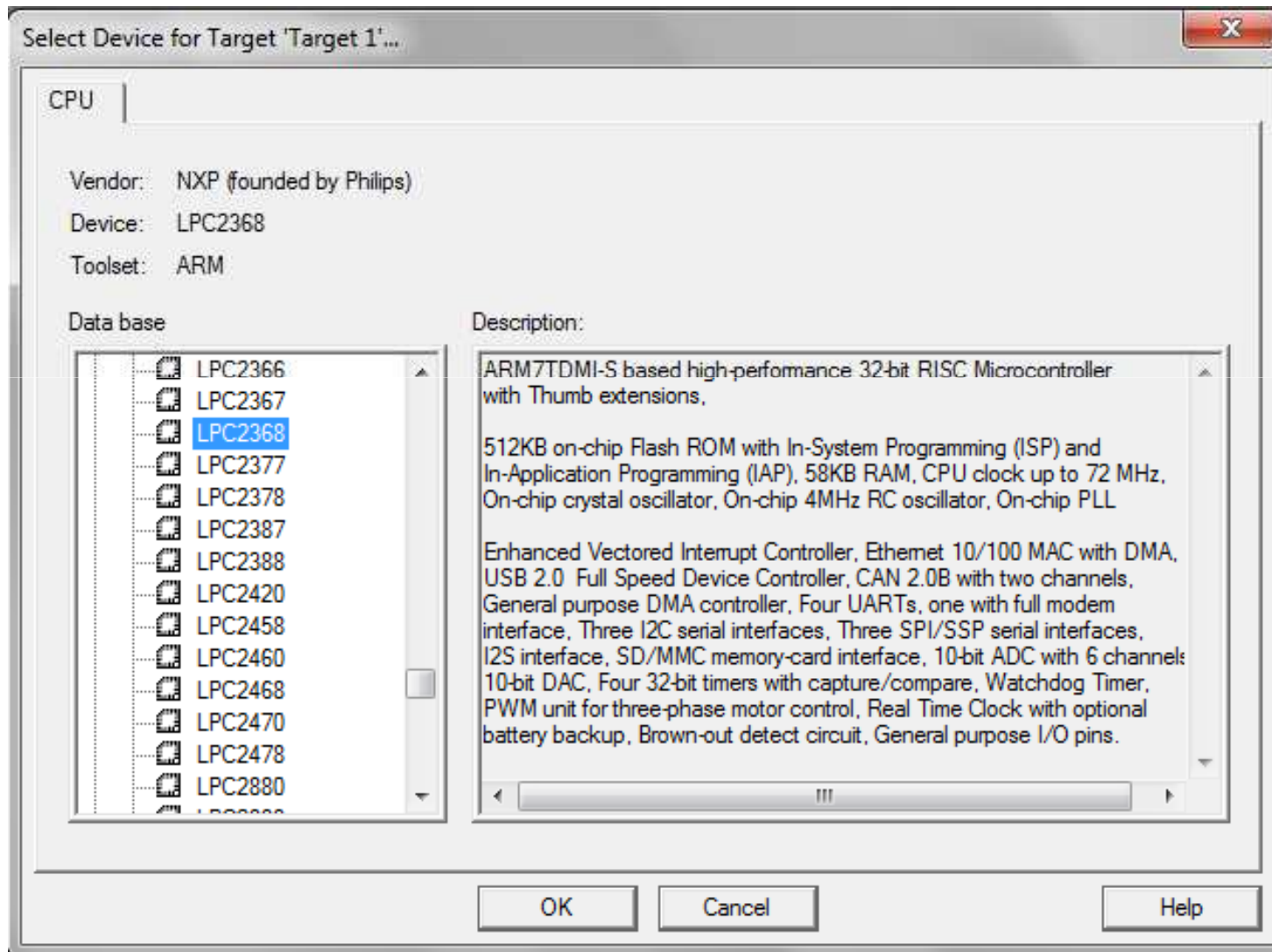
Engenharia de Sistemas Embarcados

Prof. Abel Guilhermino

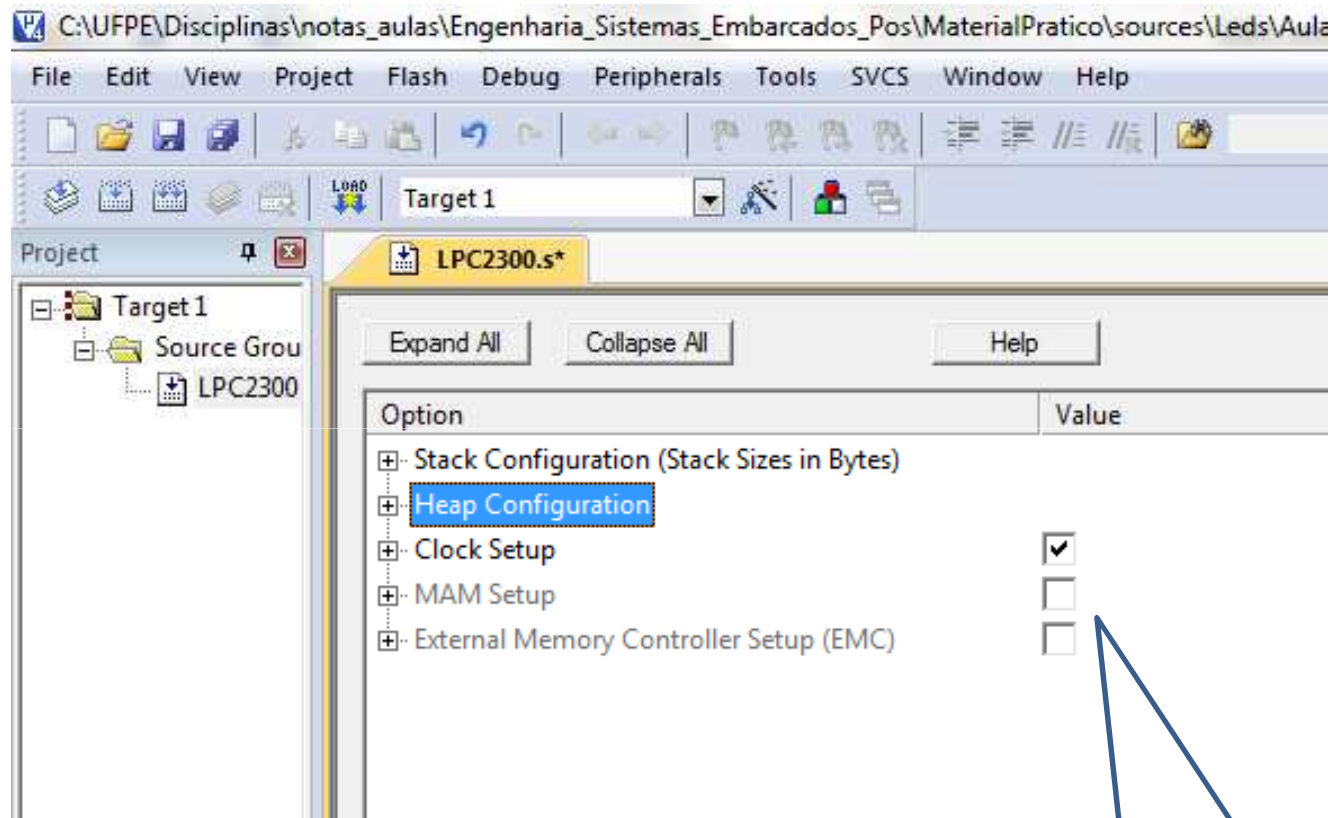
# Criação de Novo Projeto



# Criação de Novo Projeto

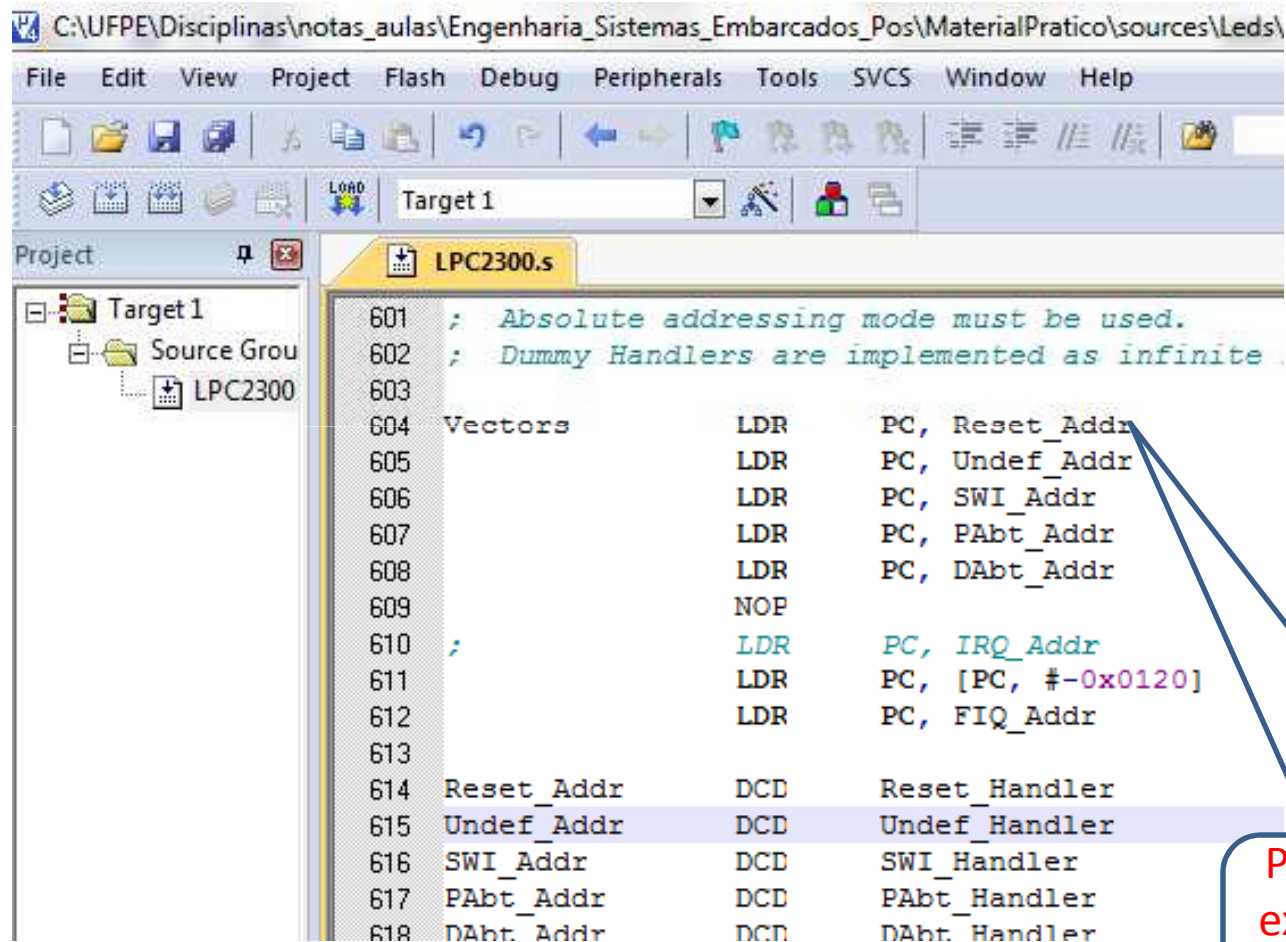


# Criação de Novo Projeto



Desmarcar MAM e  
External Memory e  
Salvar

# Criação de Novo Projeto



```
C:\UFPE\Disciplinas\notas_aulas\Engenharia_Sistemas_Embarcados_Pos\MaterialPratico\sources\Leds\
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
LPC2300.s
601 ; Absolute addressing mode must be used.
602 ; Dummy Handlers are implemented as infinite
603
604 Vectors      LDR    PC, Reset_Addr
605             LDR    PC, Undef_Addr
606             LDR    PC, SWI_Addr
607             LDR    PC, PAbt_Addr
608             LDR    PC, DAbt_Addr
609             NOP
610 ;           LDR    PC, IRQ_Addr
611             LDR    PC, [PC, #-0x0120]
612             LDR    PC, FIQ_Addr
613
614 Reset_Addr  DCD    Reset_Handler
615 Undef_Addr  DCD    Undef_Handler
616 SWI_Addr    DCD    SWI_Handler
617 PAbt_Addr   DCD    PAbt_Handler
618 DAbt_Addr   DCD    DAbt_Handler
```

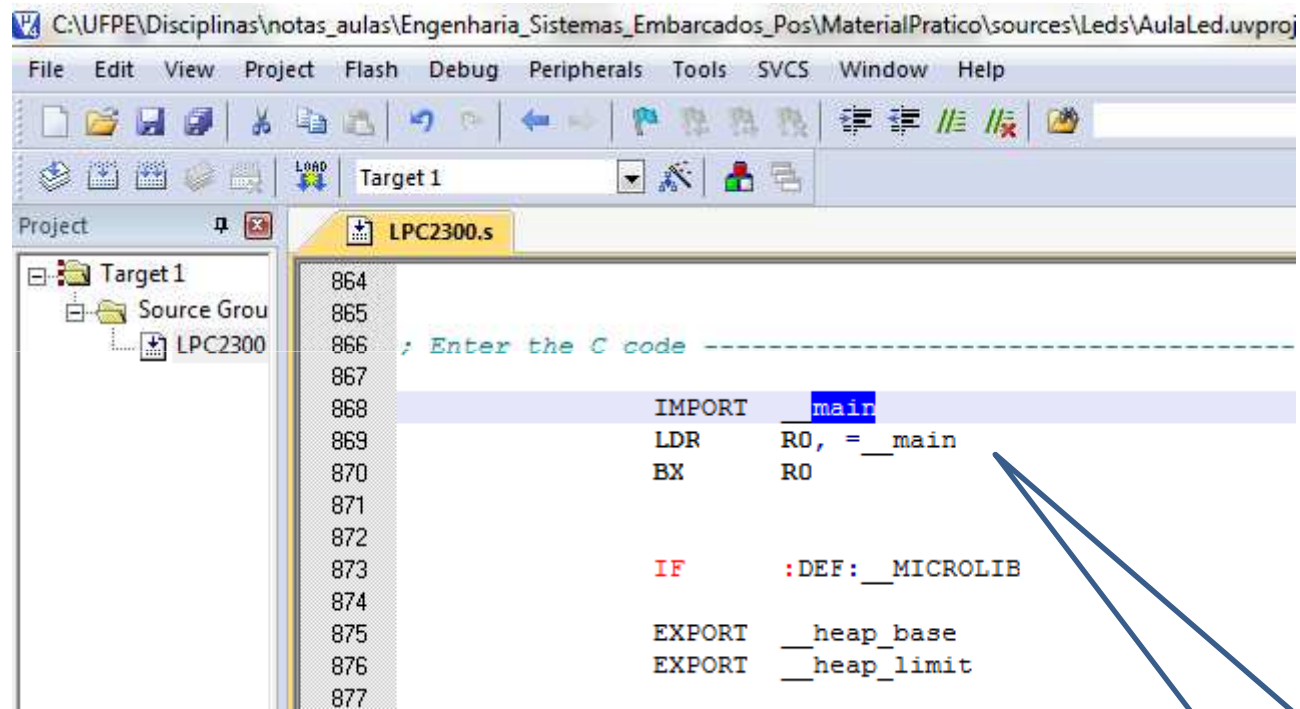
Primeira instrução a ser executada -> Jump para (0x0000 0000) = Reset Handler

# Criação de Novo Projeto

```
C:\UFPE\Disciplinas\notas_aulas\Engenharia_Sistemas_Embarcados_Pos\MaterialPratico\sources\Leds\AulaLed.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
  Target 1
    Source Grou
      LPC2300
625 PAbt_Handler  B    PAbt_Handler
626 DAbt_Handler  B    DAbt_Handler
627 IRQ_Handler   B    IRQ_Handler
628 FIQ_Handler   B    FIQ_Handler
629
630
631 ; Reset Handler
632
633 EXPORT Reset_Handler
634 Reset_Handler
635
636
637
638 ; Clock Setup -----
639
640 IF      (:LNOT:(:DEF:NO_CLOCK_SETUP)):LAND:(CLOCK_SETUP != 0)
641 LDR    R0, =SCB_BASE
642 MOV    R1, #0xAA
643 MOV    R2, #0x55
644
645 ; Configure and Enable PLL
646 LDR    R3, =SCS_Val      ; Enable main oscillator
647 STR    R3, [R0, #SCS_OFS]
648
....
```

Reset Handler. A partir daqui ocorre a inicialização dos periféricos. Isso pode ser configurado pelo usuário

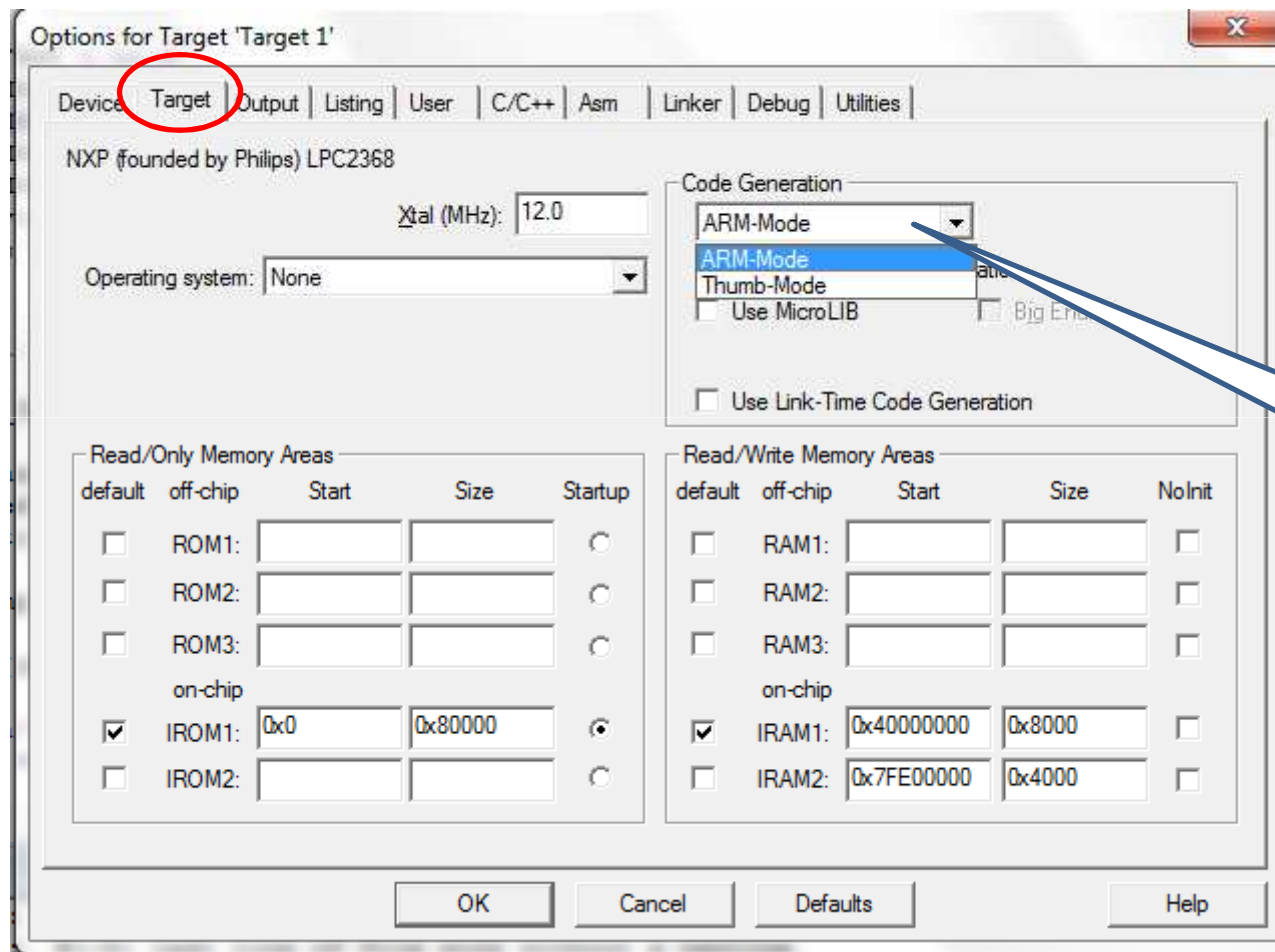
# Criação de Novo Projeto



```
864
865
866 ; Enter the C code -----
867
868 IMPORT __main
869 LDR R0, =__main
870 BX R0
871
872
873 IF :DEF: __MICROLIB
874
875 EXPORT __heap_base
876 EXPORT __heap_limit
877
```

Jump para Código do  
Usuário (default na  
Flash)

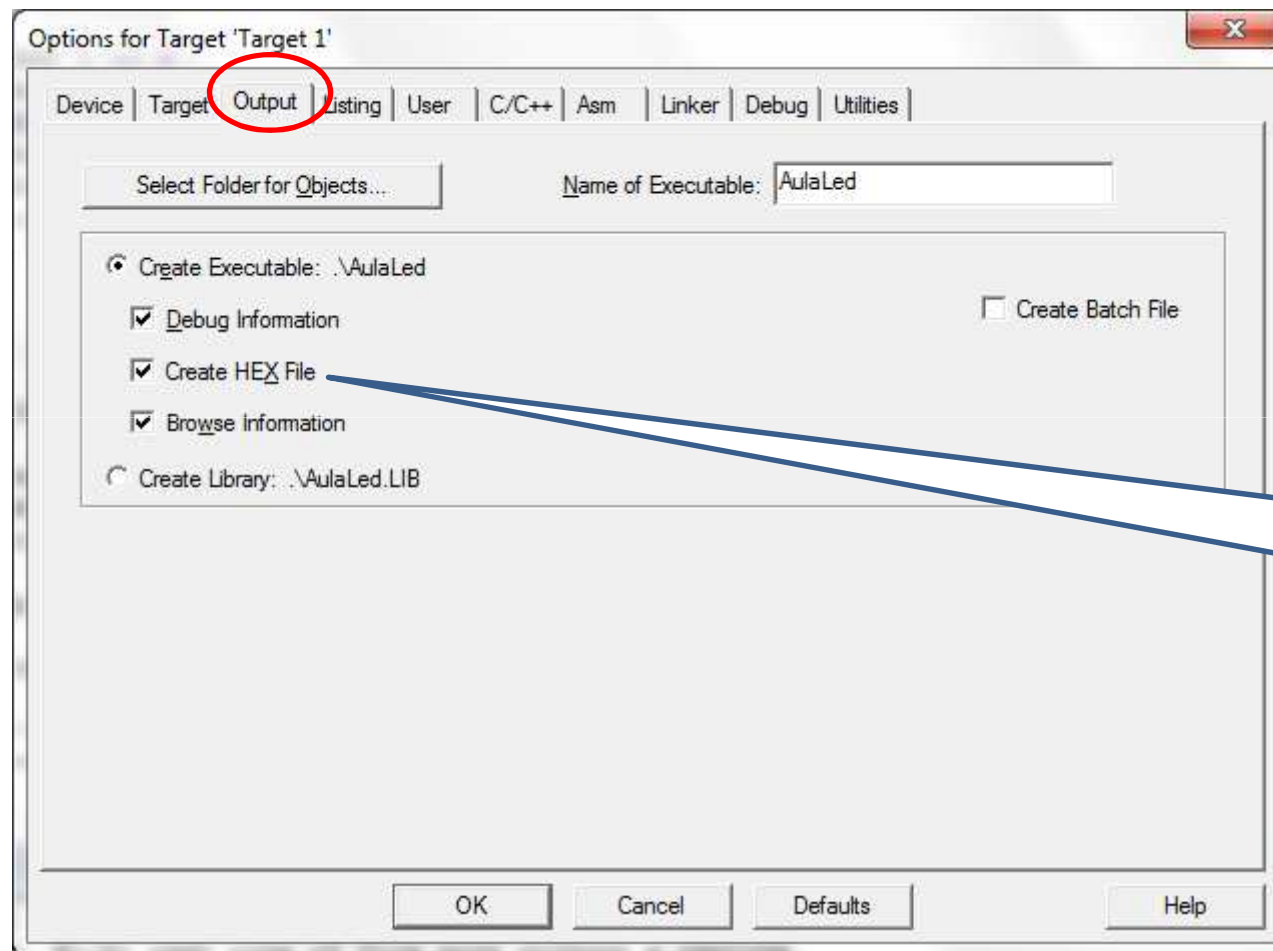
# Criação de Novo Projeto



Alterar Geração de  
Código ARM/THUMB

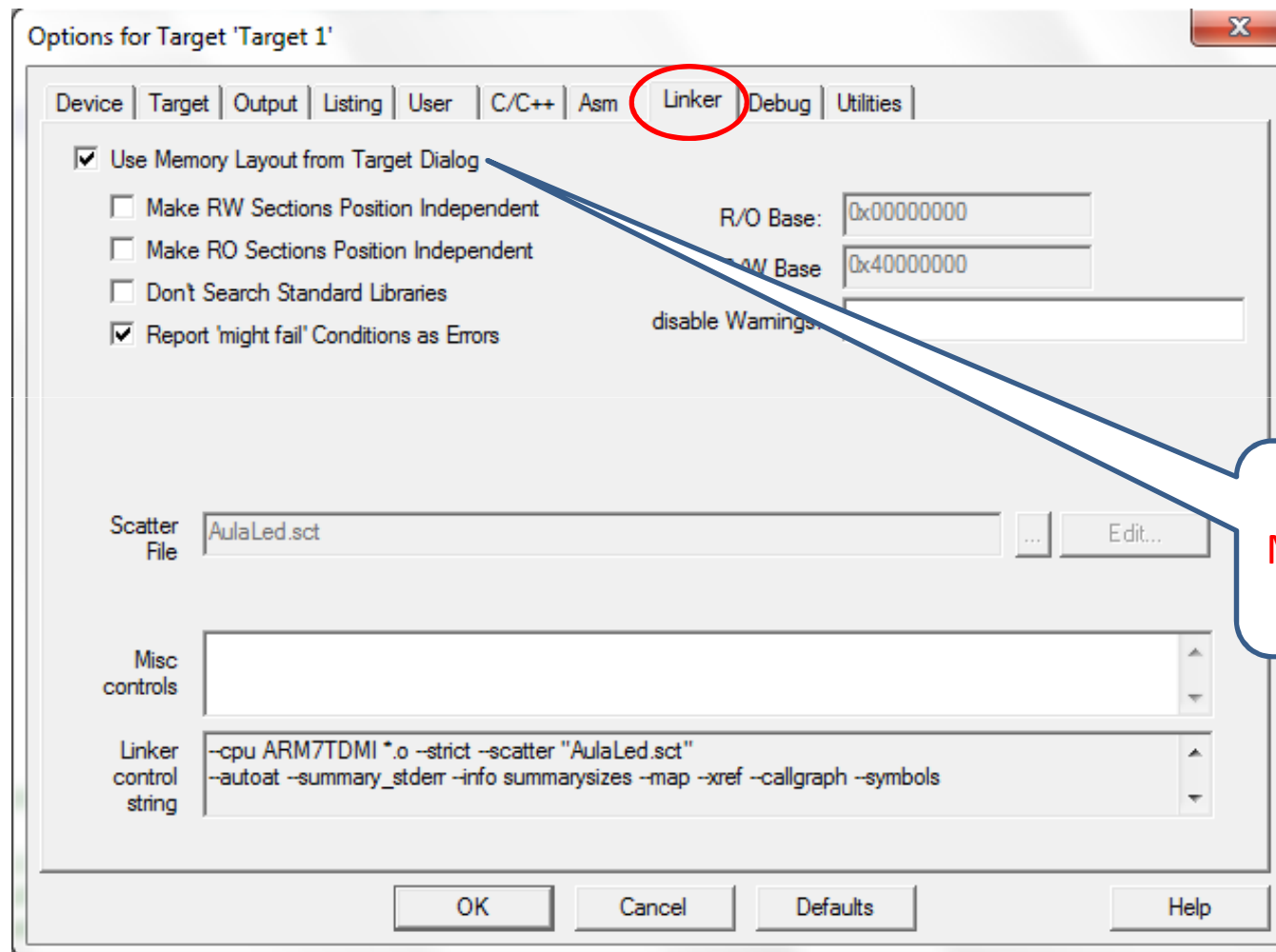


# Criação de Novo Projeto



Selecionar para Criar  
Hex File.

# Criação de Novo Projeto



Pegar Layout de  
Memória (endereços)  
Pré-determinado

# Criação de Novo Projeto

Options for Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | **Debug** | Utilities

Use Simulator  
 Limit Speed to Real-Time

Settings  Use: ULINK ARM Debugger Settings

Load Application at Startup  Run to main()  Load Application at Startup  Run to main()

Initialization File:

Restore Debug Session Settings

Breakpoints  Toolbd  
 Watch Windows & Performance  
 Memory Display

CPU DLL: SARM.DLL Parameter: -cLPC236x

Dialog DLL: DARMP.DLL Parameter: -pLPC2368

ARM Target Driver Setup

ULINK USB - JTAG Adapter

Serial No: V0187LBE

ULINK Version: ULINK2

Device Family: ARM

Firmware Version: V1.42

Max JTAG Clock: 100kHz

JTAG Device Chain

IDCODE	Device Name	IR len	Move	
TDO	0x4F1F0F0F	ARM7TDMI-S Core	4	Up
TDI				Down

Automatic Detection ID CODE:   
 Manual Configuration Device Name:   
IR len:

Add Delete Update

Debug

Cache Options

Cache Code  
 Cache Memory

Download Options

Verify Code Download  
 Download to flash

Misc Options

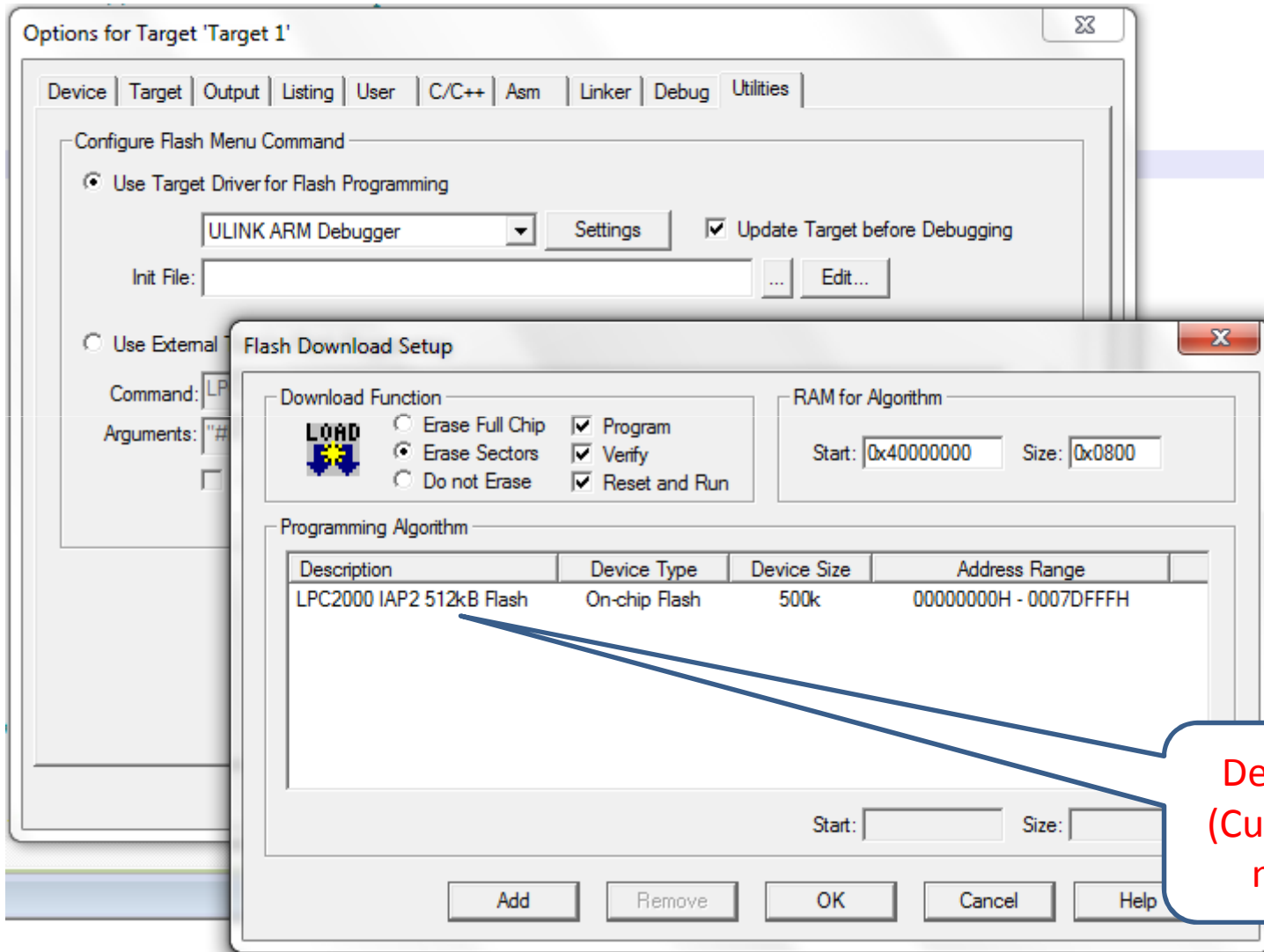
Use Reset at StartUp

OK Cancel

Selecionar Simulador ou Debugger

Selecionar 100kHz  
Selecionar Download to flash  
Use Reset at StartUp

# 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 = Saída)
012 FIO2DIR = 0xFFFFFFFF; // (FastIO por exemplo)
013
014 // Para colocar 1 em um pino de saída
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)

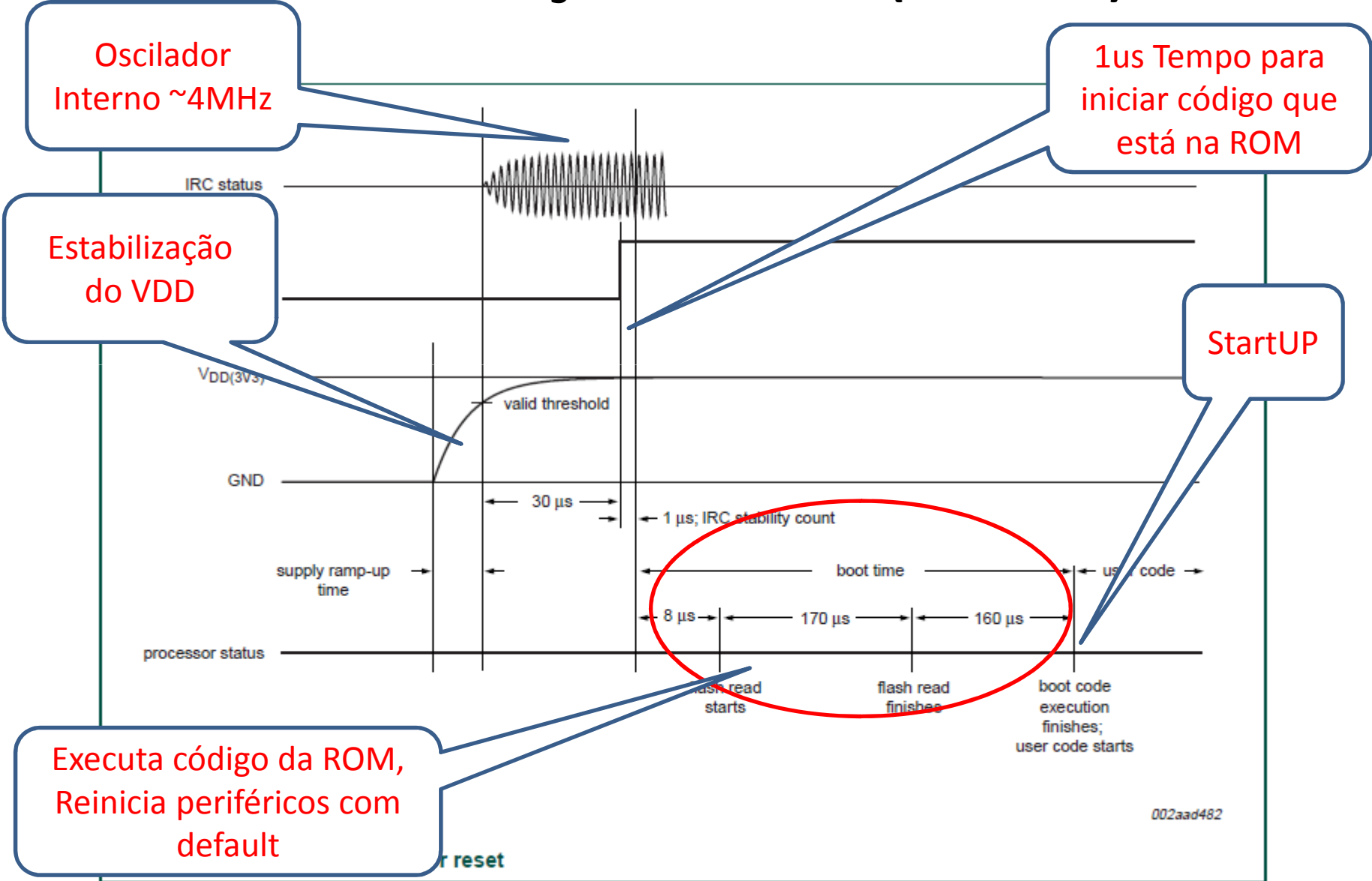
```
007
008 void main(){
009 //int a;
010 char apenas_byte;
011
012 // Definir as direções dos pinos (0 = Entrada, 1 = saída)
013 FIO2DIR = 0xFFFFFFFF; // (FastIO porta 2)
014
015 // Para colocar 1 em um pino de saída
016 a = 0x00000001;
017 FIO2SET = a ;
018
019
020 FIO2SET = 0x01; // Ligar Led P2.0
021 FIO2SET = 0x02; // Ligar Led P2.1 e manter 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){
034
```

Setar apenas um bit da porta

Obter conteúdo de uma porta

Obter 1 byte de 32 bits

# Inicialização ARM (Reset)



Frequência do PLL

# PLL

Frequência de entrada do Core

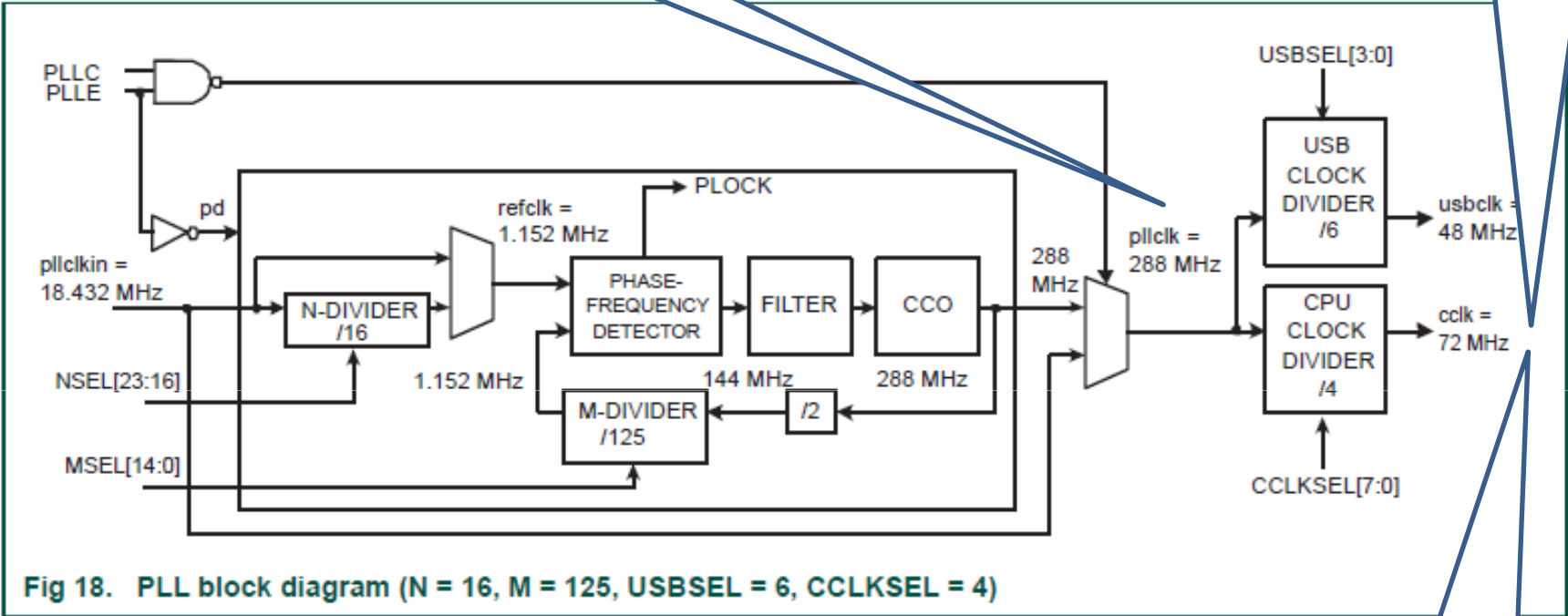


Fig 18. PLL block diagram (N = 16, M = 125, USBSEL = 6, CCLKSEL = 4)

Valor pode ser configurado no debugger

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

$$FCCO = 2 * (12\text{MHz} * (11+1)) / (0+1) = 288\text{MHz}$$



# PLL (Debugger)

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

Precisa desconectar o PLL para configurar

The screenshot displays the PLL (Debugger) interface with several configuration windows:

- Clock Source Selection:** Shows the clock source selected as 'Main Oscillator' (CLKSRCSEL: 0x01).
- Phase Locked Loop (PLL):** Shows the PLL control register (PLLCON: 0x03) with PLLC checked, and the configuration register (PLLCFG: 0x0000000B) with MSEL: 0x000B and NSEL: 0x00.
- Clock Dividers:** Shows the CPU clock configuration (CCLKSEL: 0x05) and USB clock configuration (USBSEL: 0x5).
- Peripheral Clock Selection:** Shows a table of peripheral clock selections.

Peripheral	Selection	Clock [MHz]
WDT	CCLK/4	12.000000
TIMERO	CCLK/4	12.000000
UART0	CCLK/4	12.000000
UART1	CCLK/4	12.000000
PWM1	CCLK/4	12.000000
I2CO	CCLK/4	12.000000
SPI	CCLK/4	12.000000

Configurar: MSEL e NSEL

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

# PLL (Debugger)

The image shows a debugger's PLL configuration interface with several windows:

- Clock Source Selection:** Shows the clock source set to 'Main Oscillator' with a frequency of 12.000000 MHz. Other options include IRC (4.000000 MHz) and RTCX (32.768 kHz).
- Phase Locked Loop (PLL):** Shows the PLL configuration registers, including PLLCON (0x03), PLLCFG (0x0000000B), MSEL (0x000B), NSEL (0x00), and PLLIN (12.000000 MHz). It also shows the PLL output frequency (HCLK) as 288.000000 MHz.
- Clock Dividers:** Shows the CPU Clock Configuration (CCLKCFG: 0x05, CCLKSEL: 0x05) resulting in a CPU clock of 48.000000 MHz. The USB Clock Configuration (USBCLKCFG: 0x05, USBSEL: 0x5) also results in a USB clock of 48.000000 MHz.
- Peripheral Clock Selection:** A table showing the clock selection for various peripherals. The selected peripheral is PCLK\_TIMER0, which is currently set to CCLK/4.

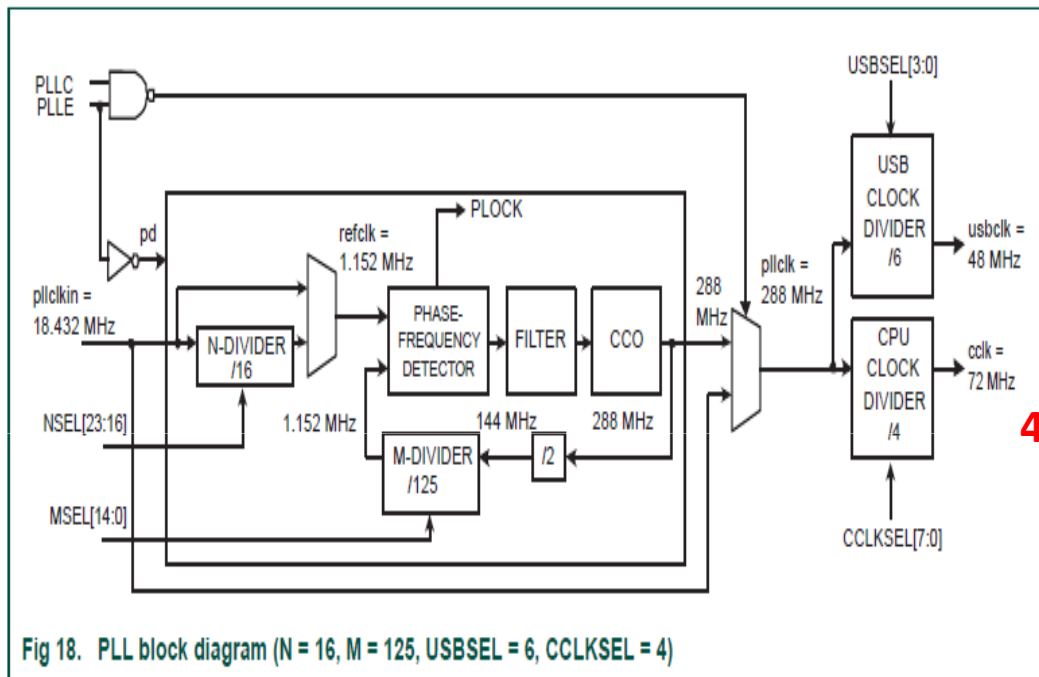
Peripheral	Selection	Clock [MHz]
WDT	CCLK/4	12.000000
TIMER0	CCLK/4	12.000000
TIMER1	CCLK/4	12.000000
UART0	CCLK/4	12.000000
UART1	CCLK/4	12.000000
PWM1	CCLK/4	12.000000
I2C0	CCLK/4	12.000000
SPI	CCLK/4	12.000000

The 'Selected Peripheral' section shows the following settings:

- PCLK\_TIMER0: CCLK/4
- PCLKSEL0: 0x00000000
- PCLKSEL1: 0x00000000

Escolhar a frequência do Periférico (Ex: TIMER0)

# Timer



Registrador:  
Divisor de Freq.  
Para o periférico

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

48MHz

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

Fixo, 32 bits

**TOPR**

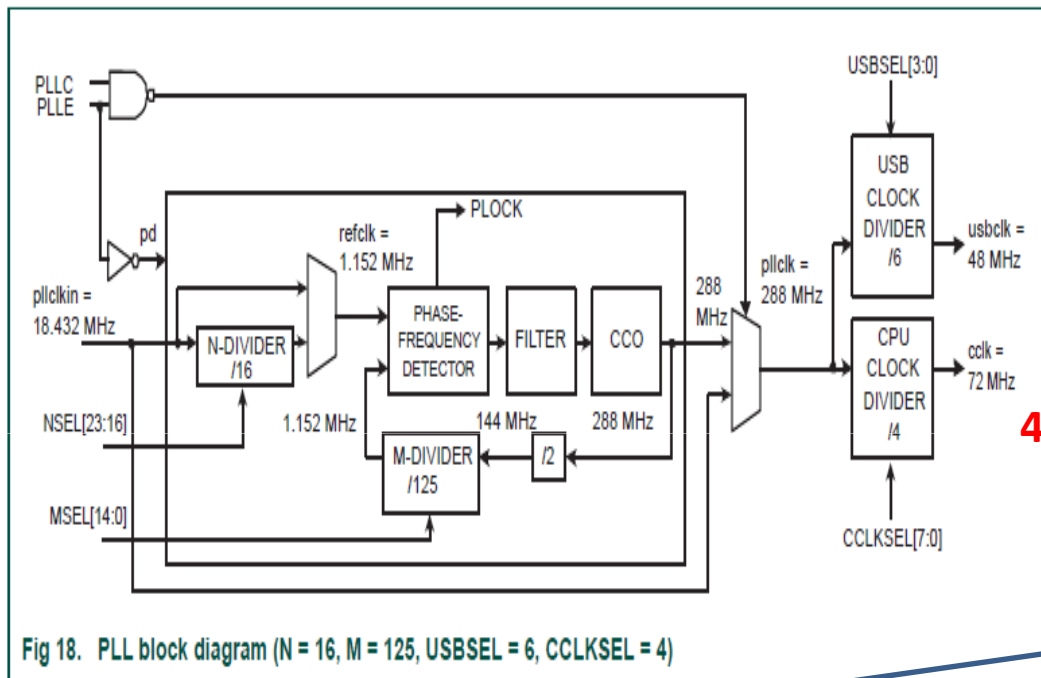
Contador 32 bits de 0 até atingir TOPR

**TOPC**

0 TRPR

Se TOPC = TOPR

# Timer



Registrador:  
Divisor de Freq.  
Para o periférico

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

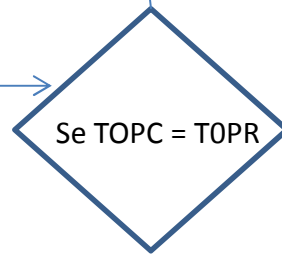
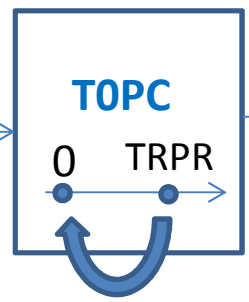
48MHz

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

Fixo, 32 bits  
**TOPR**  
TOPR=11

1MHz

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

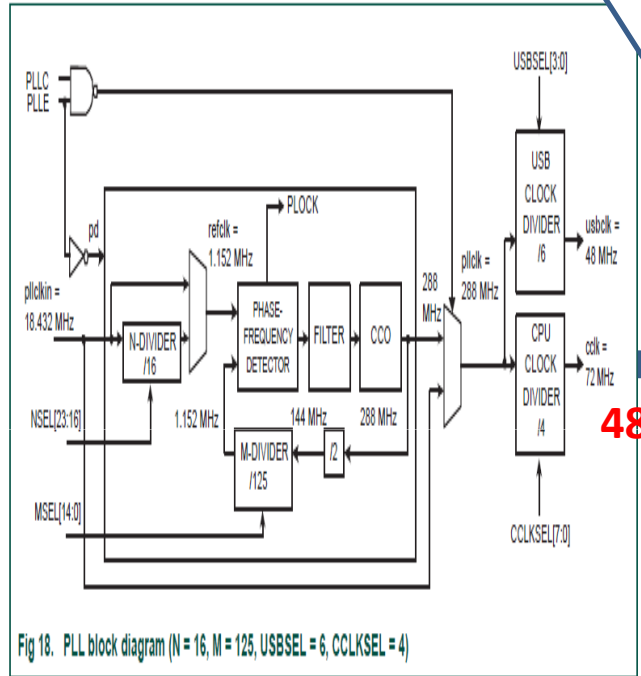


Registrador:  
Divisor de Freq.  
Para o periférico

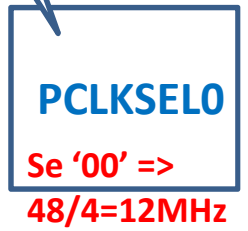
# Timer

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

Precisa zerar  
contador via  
software com 0



48MHz



Fixo, 32 bits

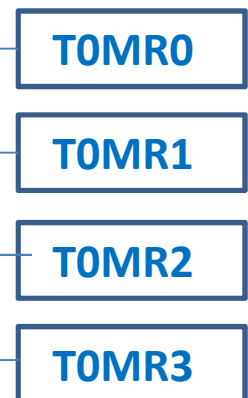
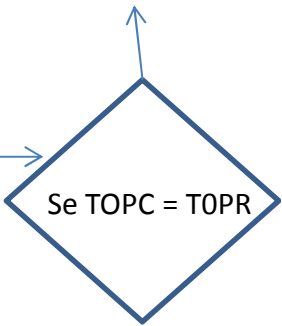
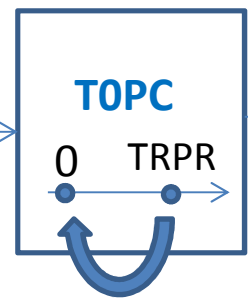


1MHz

Contador



Interrupt



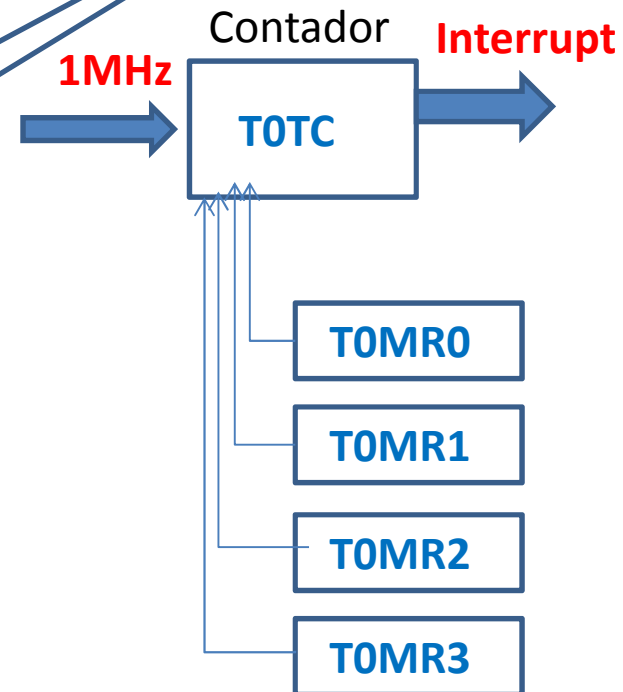
Fixo, 32 bits

# TOMCR

```
// 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 TOTC é Reiniciado quando valor se iguala ao do registrado MRN
//        0 -> O registrador TOTC não é reiniciado quando o valor se iguala ao do registrador MRN
// MRNI : 1 -> Gera interrupção
//        0 -> Não gera interrupção
```

```
TOMCR = 0x00000003;
```

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



# Configuração de Interrr. Timer

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

    // Gera interrup, TOTC é reiniciado qdo valor iguala a MRN
    TOMCR = 0x00000003; // Controle
    TOMRO = 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.
    TOTCR = 0x01;
    while(1){ }
}
```

```
void ISR (void) __irq {
    unsigned char IR_TMP = 0;
    static int pin_set = 0, flag = 0 ;
    IR_TMP = TOIR;   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 interrupção ativa
    if ((IR_TMP & 0x01) == 0x01 )
        TOIR = 0x00000001;
    if ((IR_TMP & 0x02) == 0x02 )
        TOIR = 0x00000010;           // Limpa vetor

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

# Configuração Serial

```
void confUart() {  
  // Seleciona o pino para utilização da UART0  
  PINSELO = 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 <sup>2</sup> C0)
		10	Capture 0.0 (Timer 0)
		11	Reserved
7:6	P0.3	00	GPIO Port 0.3
		01	SDA0 (I <sup>2</sup> C0)
		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  
  PINSELO = 0x00000050; //P0.0 TxD0 = 01, P0.1 RxD0 = 01  
  //Uart Fifo control Register  
  UOFCR = 0x00;  
  //Uart Line Control Register  
  UOLCR = 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 UODLL e UODLM  
//                      : 1 -> Ativa permissão para alterar UODLL e UODLM  
//Break Control : 0 ->  
//              : 1 ->  
//Parity Select : 00 -> Paridade impa  
//              01 -> Paridade par  
//              10 -> ...  
//              11 -> ...  
//Parity Enable : 0 -> Desativada paridade  
//              1 -> Ativada 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
PINSELO = 0x00000050; //P0.0 TxD0 = 01, P0.1 RxD0 = 01
//Uart Fifo control Register
UOFCR = 0x00;
//Uart Line Control Register
UOLCR = 0x83;
//Uart Divisor Latch register (Configurando para 9600 bps)
UODLM = 0x00;
UODLL = 0x4E;
    
```

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

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

Desired baudrate	MULVAL = 0 DIVADDVAL = 0		Optimal MULVAL & DIVADDVAL			
	UODLM:UODLL hex <sup>[2]</sup>	UODLL dec <sup>[1]</sup>	% error <sup>[3]</sup>	UODLM:UODLL dec <sup>[1]</sup>	Fractional pre-scaler value MULDIV MULDIV + DIVADDVAL	% error <sup>[3]</sup>
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  
  PINSELO = 0x00000050; //P0.0 TxD0 = 01, P0.1 RxD0 = 01  
  //Uart Fifo control Register  
  UOFCR = 0x00;  
  //Uart Line Control Register  
  UOLCR = 0x83;  
  //Uart Divisor Latch register (Configurando para 9600 bps)  
  UODLM = 0x00;  
  UODLL = 0x4E;  
  //Uart Line Control Register  
  UOLCR = 0x03;  
  //Interrupt Enable Register  
  UOIER = 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 -> Desativada 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  
    UOFCR = 0x00;  
    //Uart Line Control Register  
    UOLCR = 0x83;  
    //Uart Divisor Latch register (Configurando para 9600)  
    UODLM = 0x00;  
    UODLL = 0x4E;  
    //Uart Line Control Register  
    UOLCR = 0x03;  
    //Interrupt Enable Register  
    UOIER = 0x00000001;  
    // Possui o endereço da rotina de tratamento  
    VICVectAddr6 = (unsigned ) ISR_UART;
```

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

Channel	Source	Name	Type	Vector	IntEnable	RawInt
0	Watchdog	WDINT	IRQ 15	00000000H	0	0
1	SW Interrupt		IRQ 15	00000000H	0	0
2	DbgCommRx		IRQ 15	00000000H	0	0
3	DbgCommTx		IRQ 15	00000000H	0	1
4	Timer 0		IRQ 15	00000000H	0	0
5	Timer 1		IRQ 15	00000000H	0	0
6	UART0		IRQ 15	00000000H	0	0
7	UART1		IRQ 15	00000000H	0	0
8	PWM1		IRQ 15	00000000H	0	0
9	I2C0		IRQ 15	00000000H	0	0
10	SPI		IRQ 15	00000000H	0	0
10	SSP0		IRQ 15	00000000H	0	0

Selected Interrupt: Watchdog

IntEnable     IntSelect    IRQPriority0: 15    VICVectAddr0: 0x00000000

SoftInt     RawInt

VICAddress: 0x00000000    VICIntSelect: 0x00000000    VICRawIntr: 0x00001008

VICSoftInt: 0x00000000    VICIntEnable: 0x00000000    VICIRQStatus: 0x00000000

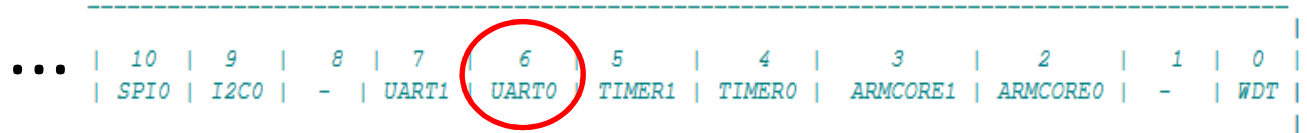
VICSoftIntClear: 0x00000000    VICIntEnClr: 0x00000000    VICFIQStatus: 0x00000000

VICSWPriMask: 0x0000FFFF    VICProtection: 0x00000000

# Configuração Serial

```
void confUart() {  
    // Seleciona o pino para utilização da UART0  
    PINSELO = 0x00000050; //P0.0 TxD0 = 01, P0.1 RxD0 = 01  
    //Uart Fifo control Register  
    UOFCR = 0x00;  
    //Uart Line Control Register  
    UOLCR = 0x83;  
    //Uart Divisor Latch register (Configurando para 9600 bps)  
    UODLM = 0x00;  
    UODLL = 0x4E;  
    //Uart Line Control Register  
    UOLCR = 0x03;  
    //Interrupt Enable Register  
    UOIER = 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



# Rotina da Interrupção Serial

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

    char_tmp = UORBR;        // 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;
    }
    TOIR = 0x00000001;          // Limpar vetor de interrup.
    VICVectAddr = 0x00000000;
}
```


# Estrutura do Case

```
#include "lpc23xx.h"

void ISR_UART (void) __irq
{
    (.....)
    VICVectAddr = 0x00000000;
}
```



```
void ISR_TIMER (void) __irq
{
    //(.....)
    VICVectAddr = 0x00000000;
}
```



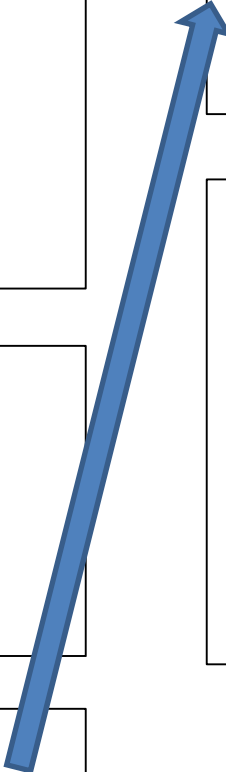
```
void confUart() {
{
}
```

```
void confTimer() {
{
}
```



```
int main()
{
    confUart();
    confTimer();

    while(1) { }
    return 0;
}
```





# Ambiente p/ Teste com Serial

