1.a)Write a program to use byte transfer of data using loops

b) Write a C program to beep a buzzer connected to port B11.

1a)

**PRESERVE8 ; Indicate the code here preserve**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x20000010 ; Source address**

**LDR r1,=0x20000040 ; Destination address**

**LDR r2,=10; number of bytes to copy**

**copy\_loop**

**LDRB r3, [r0] ; read 1 byte**

**ADDS r0, r0, #1 ; increment source pointer**

**STRB r3, [r1] ; write 1 byte**

**ADDS r1, r1, #1 ; increment destination pointer**

**SUBS r2, r2, #1 ; decrement loop counter**

**BNE copy\_loop ; loop until all data copied**

**stop B stop**

**END**

2b)

**//**

**// Smpl\_GPIO\_Buzzer : GPB11 low-active output control Buzzer**

**// Note: Nu-LB-NUC140 R1 should be 0 ohm**

**//**

**#include <stdio.h>**

**#include "NUC1xx.h"**

**#include "Driver\DrvSYS.h"**

**#include "Driver\DrvGPIO.h"**

**#include "Driver\DrvADC.h"**

**int main (void)**

**{**

**UNLOCKREG();   // unlock register for programming**

**DrvSYS\_Open(48000000);     // set System Clock to run at 48MHz**

**LOCKREG();   // lock register from programming**

**DrvGPIO\_Open(E\_GPB, 11, E\_IO\_OUTPUT); // initial GPIO pin GPB11 for controlling Buzzer**

**while(1) {**

**DrvGPIO\_ClrBit(E\_GPB,11); // GPB11 = 0 to turn on Buzzer**

**DrvSYS\_Delay(100000);     // Delay**

**DrvGPIO\_SetBit(E\_GPB,11); // GPB11 = 1 to turn off Buzzer**

**DrvSYS\_Delay(100000);     // Delay**

**}**

**}**

2. a)Write a program to use Labels to calculate the sum of say 10 numbers

b) Write a program to use switch case

2a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**DataIn EQU 0x20000000**

**Sum EQU 0x20000040**

**\_\_main**

**LDR r0,=DataIn; Get the address of variable 'DataIn'**

**MOVS r1, #10 ; loop counter**

**MOVS r2, #0 ; Result - starting from 0**

**add\_loop**

**LDM r0!,{r3} ; Load result and increment address**

**ADDS r2, r3 ; add to result**

**SUBS r1, #1 ; increment loop counter**

**BNE add\_loop**

**LDR r0,=Sum ; Get the address of variable 'Sum'**

**STR r2,[r0] ; Save result to Sum**

**stop B stop**

**END**

2b)

**\_\_main**

**LDR R0, =0**

**CMP R0, #3 ; Compare input to maximum valid choice**

**BHI default\_case ; Branch to default case if higher than 3**

**MOVS R2, #4 ; Multiply branch table offset by 4**

**MULS R0, R2, R0 ; (size of each entry)**

**LDR R1, =BranchTable ; Get base address of branch table(0x284)**

**LDR R2,[R1,R0] ; Get the actual branch destination**

**BX R2 ; Branch to destination**

**ALIGN 4 ; Alignment control. The table has**

**BranchTable ; to be word aligned to prevent unaligned read ;table of each destination address**

**DCD Dest0**

**DCD Dest1**

**DCD Dest2**

**DCD Dest3**

**default\_case**

**stop B stop; Instructions for default case**

**Dest0 ldr r0, =10**

**stop1 B stop1 ; Instructions for case ‘0’**

**Dest1 ldr r0, =20**

**stop2 B stop2 ; Instructions for case ‘1’**

**Dest2 ldr r0, =30**

**stop3 B stop3 ; Instructions for case ‘2’**

**Dest3 ldr r0, =40**

**stop4 B stop4 ; Instructions for case ‘3’**

**END**

3. a)Write an Assembly level program to copy multiple data at a time say 128 bytes of data in two iteration

b) Write a program to calculate factorial

3a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x20000000 ; Source address**

**LDR r1,=0x20000120 ; Destination address**

**LDR r2,=128 ; number of bytes to copy, also**

**copy\_loop ; acts as loop counter**

**LDMIA r0!,{r4-r7} ; Read 4 words and increment r0**

**STMIA r1!,{r4-r7} ; Store 4 words and increment r1**

**LDMIA r0!,{r4-r7} ; Read 4 words and increment r0**

**STMIA r1!,{r4-r7} ; Store 4 words and increment r1**

**LDMIA r0!,{r4-r7} ; Read 4 words and increment r0**

**STMIA r1!,{r4-r7} ; Store 4 words and increment r1**

**LDMIA r0!,{r4-r7} ; Read 4 words and increment r0**

**STMIA r1!,{r4-r7} ; Store 4 words and increment r1**

**SUBS r2, r2, #64 ; Each time 64 bytes are copied**

**BNE copy\_loop ; loop until all data copied**

**stop B stop**

**END**

3b)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT main**

**; Start of CODE area**

**main\_\_**

**MOVS R6,#05   ; factorial of no**

**MOVS R4, R6**

**SUBS R4,R4,#1**

**LOOP**

**MOVS R7,R4**

**MULS R7,R6,R7**

**MOVS R6,R7**

**SUBS R4,R4,#1**

**BNE LOOP**

**stop B stop ; R7 ANSWER**

**END**

4. a)Write a program to use BICS and MVN

b) Write a C program to Light a RGBLED connected to port A12-14.

4a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0xF0 ; and(r0,not(r1)**

**LDR r1,=0xC0**

**BICS r0,r0,r1; result 30**

**LDR r2,=0xFF ;**

**LDR r3,=0xC0; and(r2,not(r3)3F**

**BICS r2,r2,r3**

**LDR r4,=0xFF ;**

**LDR r5,=0xC0;**

**MVNS r4,r5; (not(r5)**

**stop B stop**

**END**

4b)

**//**

**// Smpl\_GPIO\_RGBled : GPA12,13,14 output control RGB LED**

**//                    output low to enable LEDs**

**#include <stdio.h>**

**#include "NUC1xx.h"**

**#include "Driver\DrvGPIO.h"**

**#include "Driver\DrvUART.h"**

**#include "Driver\DrvSYS.h"**

**// Initial GPIO pins (GPA 12,13,14) to Output mode**

**void Init\_LED()**

**{**

**// initialize GPIO pins**

**DrvGPIO\_Open(E\_GPA, 12, E\_IO\_OUTPUT); // GPA12 pin set to output mode**

**DrvGPIO\_Open(E\_GPA, 13, E\_IO\_OUTPUT); // GPA13 pin set to output mode**

**DrvGPIO\_Open(E\_GPA, 14, E\_IO\_OUTPUT); // GPA14 pin set to output mode**

**// set GPIO pins output Hi to disable LEDs**

**DrvGPIO\_SetBit(E\_GPA, 12); // GPA12 pin output Hi to turn off Blue  LED**

**DrvGPIO\_SetBit(E\_GPA, 13); // GPA13 pin output Hi to turn off Green LED**

**DrvGPIO\_SetBit(E\_GPA, 14); // GPA14 pin output Hi to turn off Red   LED**

**}**

**int main (void)**

**{**

**UNLOCKREG();   // unlock register for programming**

**DrvSYS\_Open(48000000);     // set System Clock to run at 48MHz (PLL with 12MHz crystal input)**

**LOCKREG();   // lock register from programming**

**Init\_LED();**

**while (1)**

**{**

**// GPA12 = Blue,  0 : on, 1 : off**

**// GPA13 = Green, 0 : on, 1 : off**

**// GPA14 = Red,   0 : on, 1 : off**

**// set RGBled to Blue**

**DrvGPIO\_ClrBit(E\_GPA,12); // GPA12 = Blue,  0 : on, 1 : off**

**DrvGPIO\_SetBit(E\_GPA,13);**

**DrvGPIO\_SetBit(E\_GPA,14);**

**DrvSYS\_Delay(1000000);**

**// set RGBled to Green**

**DrvGPIO\_SetBit(E\_GPA,12);**

**DrvGPIO\_ClrBit(E\_GPA,13); // GPA13 = Green, 0 : on, 1 : off**

**DrvGPIO\_SetBit(E\_GPA,14);**

**DrvSYS\_Delay(1000000);**

**// set RGBled to Red**

**DrvGPIO\_SetBit(E\_GPA,12);**

**DrvGPIO\_SetBit(E\_GPA,13);**

**DrvGPIO\_ClrBit(E\_GPA,14); // GPA14 = Red,   0 : on, 1 : off**

**DrvSYS\_Delay(1000000);**

**// set RGBled to off**

**DrvGPIO\_SetBit(E\_GPA,12); // GPA12 = Blue,  0 : on, 1 : off**

**DrvGPIO\_SetBit(E\_GPA,13); // GPA13 = Green, 0 : on, 1 : off**

**DrvGPIO\_SetBit(E\_GPA,14); // GPA14 = Red,   0 : on, 1 : off**

**DrvSYS\_Delay(1000000);**

**}**

**}**

**///SIMPLIFIED APPROACH**

**#include <stdio.h>**

**#include "NUC1xx.h"**

**#include "Driver\DrvGPIO.h"**

**#include "Driver\DrvUART.h"**

**#include "Driver\DrvSYS.h"**

**// Initial GPIO pins (GPA 12,13,14) to Output mode**

**void redON()**

**{**

**DrvGPIO\_ClrBit(E\_GPA,12);**

**}**

**void blueON()**

**{**

**DrvGPIO\_ClrBit(E\_GPA,13);**

**}**

**void greenON()**

**{**

**DrvGPIO\_ClrBit(E\_GPA,14);**

**}**

**void redOFF()**

**{**

**DrvGPIO\_SetBit(E\_GPA,12);**

**}**

**void blueOFF()**

**{**

**DrvGPIO\_SetBit(E\_GPA,13);**

**}**

**void greenOFF()**

**{**

**DrvGPIO\_SetBit(E\_GPA,14);**

**}**

**void Init\_LED()**

**{**

**// initialize GPIO pins**

**DrvGPIO\_Open(E\_GPA, 12, E\_IO\_OUTPUT); // GPA12 pin set to output mode**

**DrvGPIO\_Open(E\_GPA, 13, E\_IO\_OUTPUT); // GPA13 pin set to output mode**

**DrvGPIO\_Open(E\_GPA, 14, E\_IO\_OUTPUT); // GPA14 pin set to output mode**

**// set GPIO pins output Hi to disable LEDs**

**redOFF();**

**greenOFF();**

**blueOFF();**

**}**

**int main (void)**

**{**

**UNLOCKREG();   // unlock register for programming**

**DrvSYS\_Open(48000000);     // set System Clock to run at 48MHz (PLL with 12MHz crystal input)**

**LOCKREG();   // lock register from programming**

**Init\_LED();**

**while (1)**

**{**

**// GPA12 = Blue,  0 : on, 1 : off**

**// GPA13 = Green, 0 : on, 1 : off**

**// GPA14 = Red,   0 : on, 1 : off**

**// set RGBled to Blue**

**redON();**

**blueOFF();**

**greenOFF();**

**DrvSYS\_Delay(1000000);**

**// set RGBled to Green**

**redOFF();**

**blueON();**

**greenOFF();**

**DrvSYS\_Delay(1000000);**

**// set RGBled to Red**

**redOFF();**

**blueOFF();**

**greenON(); // GPA14 = Red,   0 : on, 1 : off**

**DrvSYS\_Delay(1000000);**

**// set RGBled to off**

**redOFF();**

**blueOFF();**

**greenOFF(); // GPA14 = Red,   0 : on, 1 : off**

**DrvSYS\_Delay(1000000);**

**}**

**}**

5. a)Write a program to illustrate ASR and LSLS instruction

b) . Write a program to illustrate the processing of data in a stack and realizing of stack using another file say(processing x to read 2x+9)

5a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**\_\_main**

**LDR r2,=0x00000080;**

**ASRS r0,r2,#04;00000001000**

**LDR r2,=0x80000000;**

**ASRS r0,r2,#04;0xF8000000**

**LDR r2,=0xFFFFFFF8**

**ASRS r0,r2,#05;0xFFFFFFFF**

**LDR r2,=0x00000080;**

**LSLS r0,r2,#04;0x00000800**

**LDR r2,=0x80000000;0x00000000**

**LSLS r0,r2,#04;0x**

**LDR r2,=0xFFFFFFF8**

**LSLS r0,r2,#05;0xFFFFFF00**

**stop B stop**

**END**

5b)

**Main.asm**

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**EXTERN func**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x10;**

**BL func**

**stop B stop**

**END**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func**

**func**

**push {r0}**

**LDR R2,[r13];**

**MOVS R3,#2**

**MULS R2,R3,R2**

**ADDS r2,r2,#9**

**STR r2,[R13]**

**pop {r1}**

**BX LR**

**END**

6. a)Write a program to illustrate LSRS and RORS

b) Write a program to use TST and Push and Pop with stack initialization

6a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT \_\_main**

**\_main**

**LDR r0,=0x80000001;**

**LSRS r2,r0,#31**

**LDR r0,=0x80000001;**

**LSRS r2,r0,#30**

**LDR r0,=0x80000001;**

**MOVS r2,#31;32-1**

**RORS r0,r2**

**LDR r0,=0x80000001;**

**MOVS r2,#30**

**RORS r0,r2**

**stop B stop**

**END**

6b)

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

\_\_main

LDR r0,=0xF0000000;

LDR r2,=0xF0000000;

TST r0,r2;

MRS r3,XPSR;updating only N and Z flags not C

LDR r0,=0x70000000;

LDR r2,=0x70000000;

TST r0,r2;

MRS r4,XPSR;updating only N and Z flags not C

stop B stop

END

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

\_\_main

LDR r3,=0x20000100

LDR r0,=0x20000050

LDMIA r3!,{r1,r2}

mov SP,r0

PUSH {r1,r2}

POP {r4,r5}

stop B stop

END

7. a)Write a program to Use of Rev ,Rev16,SXTH, SXTB,UXTB, UXTH

b) Write a program to illustrate function using BL and BX instructions, also illustrate a push and a

pop by decrementing / incrementing the address.

7a)

**PRESERVE8**

**THUMB**

**AREA |.TEXT|, CODE, READONLY**

**EXPORT \_\_main**

**\_\_main**

**LDR R0,=0x11223344**

**REV R1, R0**

**LDR R0,=0x11223344**

**REV16 R1, R0**

**LDR R0,=0x55AA8765**

**SXTB R1, R0**

**LDR R0,=0x55AA8765**

**SXTH R1, R0**

**LDR R0,=0x55AA8765**

**UXTH R1, R0**

**UXTB R1, R0**

**stop B stop**

**END**

7b)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**function1**

**SUB SP, SP, #0x8 ; Reserve 2 words of stack ;(8 bytes) for local variables ;Data processing in function**

**MOVS r0, #0x12 ; set a dummy value**

**STR r0, [sp, #0] ; Store 0x12 in 1st local variable**

**STR r0, [sp, #4] ; Store 0x12 in 2nd local variable**

**LDR r1, [sp, #0] ; Read from 1st local variable**

**LDR r2, [sp, #4] ; Read from 2nd local variable**

**ADD SP, SP, #0x8; Restore SP to original position**

**BX LR**

**\_\_main**

**BL function1**

**stop B stop**

**END**

8. a)Write a program to illustrate the nested function and the nested function being imported from other files.

b) Write a program to illustrate ASRS and RORS

8a)

**Main.asm**

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**EXTERN func**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x10;**

**BL func**

**stop B stop**

**END**

**Function2.asm**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func**

**EXTERN func2**

**func**

**push{LR}**

**MOVS R1,#08**

**BL func2**

**pop{PC}**

**END**

**Function3.asm**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func2**

**func2**

**MOVS r2,#08**

**BX LR**

**END**

8b)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB     ; Indicate THUMB code is used**

**AREA    |.text|, CODE, READONLY**

**EXPORT \_\_main**

**\_main**

**LDR r0,=0x80000001;**

**ASRS r2,r0,#31**

**LDR r0,=0x80000001;**

**ASRS r2,r0,#30**

**LDR r0,=0x80000001;**

**MOVS r2,#31;32-1**

**RORS r0,r2**

**LDR r0,=0x80000001;**

**MOVS r2,#30**

**RORS r0,r2**

**stop B stop**

**END**

9 a)Write a program to illustrate the processing of data in a stack and realizing of stack using another file say(processing x to read 2x+9)

b) Write a program to Use of Rev ,Rev16,SXTH, SXTB,UXTB, UXTH

9a)

**Main.asm**

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**EXTERN func**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x10;**

**BL func**

**stop B stop**

**END**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func**

**func**

**push {r0}**

**LDR R2,[r13];**

**MOVS R3,#2**

**MULS R2,R3,R2**

**ADDS r2,r2,#9**

**STR r2,[R13]**

**pop {r1}**

**BX LR**

**END**

9b)

**PRESERVE8**

**THUMB**

**AREA |.TEXT|, CODE, READONLY**

**EXPORT \_\_main**

**\_\_main**

**LDR R0,=0x11223344**

**REV R1, R0**

**LDR R0,=0x11223344**

**REV16 R1, R0**

**LDR R0,=0x55AA8765**

**SXTB R1, R0**

**LDR R0,=0x55AA8765**

**SXTH R1, R0**

**LDR R0,=0x55AA8765**

**UXTH R1, R0**

**UXTB R1, R0**

**stop B stop**

**END**

10.a) Write a program to clear and extract the bits with starting position P and width of the bits W

b) Write a C program to Light a LED connected to port C12.

10a)

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**; Start of CODE area**

**\_\_main**

**;extracting**

**LDR r0,=0xFFC0FFFF**

**LSLS r0,r0,#(32-16-8)//(32-W-P)//**

**LSRS r0,r0,#(32-8); clearing**

**LDR r0,=0xFFC0FFFF**

**MOVS r1,#16**

**MOVS R2,#08;(32-16-08)//( 32-W-P)**

**MOVS r3,#08**

**RORS r0,r0,r1**

**LSRS r0,r0,r2**

**RORS r0,r0,r3**

**stop B stop**

**END**

10b)

**//**

**// Smpl\_GPIO\_LED : GPC12 to control on-board LEDs**

**//                 low-active output to control Red LEDs**

**//**

**#include <stdio.h>**

**#include "NUC1xx.h"**

**#include "Driver\DrvGPIO.h"**

**#include "Driver\DrvUART.h"**

**#include "Driver\DrvSYS.h"**

**void Init\_LED() // Initialize GPIO pins**

**{**

**DrvGPIO\_Open(E\_GPC, 12, E\_IO\_OUTPUT); // GPC12 pin set to output mode**

**DrvGPIO\_SetBit(E\_GPC, 12);   // output Hi to turn off LED**

**}**

**int main (void)**

**{**

**UNLOCKREG(); // unlock register for programming**

**DrvSYS\_Open(48000000);// set to run at 48MHz**

**// 12MHz crystal input, PLL output 48MHz**

**LOCKREG();   // lock register  from programming**

**Init\_LED(); // Initialize LEDs (four on-board LEDs)**

**while (1) //   forever loop to keep flashing four LEDs one at a time**

**{**

**DrvGPIO\_ClrBit(E\_GPC, 12); // output Low  turn on LED**

**DrvSYS\_Delay(30000); // delay**

**DrvGPIO\_SetBit(E\_GPC, 12); // output Hi  turn off LED**

**DrvSYS\_Delay(300000);  // delay**

**}**

**}**

11.a)Write a program to use TST and Push and Pop with stack initialization.

b) Write a C program to Light a LEDs connected to port C12-15 using macros.

11a)

????????????????????????????????????????

11b)

**#include <stdio.h>**

**#include "NUC1xx.h"**

**#include "Driver\DrvGPIO.h"**

**#include "Driver\DrvUART.h"**

**#include "Driver\DrvSYS.h"**

**#define  INIT\_LED0 DrvGPIO\_Open(E\_GPC, 12, E\_IO\_OUTPUT)**

**#define  INIT\_LED1 DrvGPIO\_Open(E\_GPC, 13, E\_IO\_OUTPUT)**

**#define  INIT\_LED2 DrvGPIO\_Open(E\_GPC, 14, E\_IO\_OUTPUT)**

**#define  INIT\_LED3 DrvGPIO\_Open(E\_GPC, 15, E\_IO\_OUTPUT)**

**#define  LED0\_ON   DrvGPIO\_ClrBit(E\_GPC, 12)**

**#define  LED0\_OFF  DrvGPIO\_SetBit(E\_GPC, 12)**

**#define  LED1\_ON   DrvGPIO\_ClrBit(E\_GPC, 13)**

**#define  LED1\_OFF  DrvGPIO\_SetBit(E\_GPC, 13)**

**#define  LED2\_ON   DrvGPIO\_ClrBit(E\_GPC, 14)**

**#define  LED2\_OFF  DrvGPIO\_SetBit(E\_GPC, 14)**

**#define  LED3\_ON   DrvGPIO\_ClrBit(E\_GPC, 15)**

**#define  LED3\_OFF  DrvGPIO\_SetBit(E\_GPC, 15)**

**#define  DELAY     DrvSYS\_Delay(300000)**

**// Initial GPIOs**

**void Init\_LED()**

**{**

**// initialize GPIO pins to OUTPUT mode**

**INIT\_LED0;**

**INIT\_LED1;**

**INIT\_LED2;**

**INIT\_LED3;**

**// set GPIO pins to output Low**

**LED0\_OFF;**

**LED1\_OFF;**

**LED2\_OFF;**

**LED3\_OFF;**

**}**

**int main (void) {**

**UNLOCKREG(); // unlock register for programming**

**DrvSYS\_Open(48000000);// set  to run at 48MHz**

**LOCKREG();     // lock register from programming**

**Init\_LED();  // Initialize LEDs (four on-board LEDs)**

**while(1)  {**

**LED0\_ON;**

**DELAY;**

**LED0\_OFF;**

**DELAY;**

**LED1\_ON;**

**DELAY;**

**LED1\_OFF;**

**DELAY;**

**LED2\_ON;**

**DELAY;**

**LED2\_OFF;**

**DELAY;**

**LED3\_ON;**

**DELAY;**

**LED3\_OFF;**

**DELAY;**

**}**

**}**