Anexo VI: DTC_0B

```
;/////////////////////////////////////////////////////////////////////////////////////////
; PROJECT INFORMATION:
;        Name:                                  soft_test_ver_1.s
;        Author                                 J. Granado
;        Date of creation                       Dec. 16, 1998
;        Date of last review                    Jan. 20, 1999
;        Version                                1.0
;        Purpose:                    hardware- software  co-simulation
;
;/////////////////////////////////////////////////////////////////////////////////////////
; SUBROUTINES INCLUDED:
;        IRQ_Handler
;        RESET_HANDLER
;        IMC_TEST
;        DAC_ADC_TEST
;        UART_TEST
;        SPI_TEST
;
;/////////////////////////////////////////////////////////////////////////////////////////
; HARDWARE HYPOTESIS
;        Based on: Policom Data Sheet of December 1998
;
;/////////////////////////////////////////////////////////////////////////////////////////
; COMMENTS:
;        SWI Handler not installed
;        FIQ Handler not installed
;        IRQ Handler uses its hidden features
;        Ready to simulate
;/////////////////////////////////////////////////////////////////////////////////////////
;  RESULTS:
;        - this software shows by the BPP what is being tested
;        - at the end of the simulation the internal RAM will show a
;                compleat report about the test performed.
;/////////////////////////////////////////////////////////////////////////////////////////


             AREA    rom, CODE, READONLY



;--------------------------------
;-       variables          -
;--------------------------------

Mode_USR        EQU     0x10
Mode_IRQ         EQU     0x12
Mode_SVC        EQU     0x13

I_Bit            EQU     0x80
F_Bit           EQU     0x40


;*******************************************************
;                           MEMORY MAP
;*******************************************************

;-------------------------- RAM-----------------------------------------------------------------------------
RAM_INT_Base    EQU     0x100000                                ; RAM: from 100_000 to 102_000
RAM_INT_Limit   EQU     RAM_INT_Base + RAM_SIZE - 4
RAM_SIZE EQU      0x2000                                        ; 2 x 4 KBytes as Internal Ram

;-------------------------- ROM-----------------------------------------------------------------------------
ROM_INT_Limit      EQU      0x10000                     ; Internal ROM limit
                                                                    ; internal ROM located at 0x000000
; --------------------------stacks ------------------------------------------------------------------------
--
STACK_SIZE         EQU      400                                    ; 1 KByte as stack size
IRQ_Stack       EQU     RAM_INT_Limit                   ;  IRQ stack at top of memory
SVC_Stack       EQU     IRQ_Stack-STACK_SIZE            ; followed by SVC stack
USR_Stack       EQU         SVC_Stack-STACK_SIZE        ; followed by USR stack
STACK_Limit        EQU      USR_Stack-STACK_SIZE

;-------------------------------- Interrupt vector and priority Tables---------------------------------------
--------
EXCEPTION_VECTOR_Base         EQU     RAM_INT_Base
EXCEPTION_VECTOR_Limit        EQU     EXCEPTION_VECTOR_Base+(IRQ_SOURCES_NUMBER-1)*4
;*************************************************************************
EXCEPTION_PRIORITY_Base       EQU     AIC_Base+0x300
;EXCEPTION_PRIORITY_Limit EQU         EXCEPTION_PRIORITY_Base+(IRQ_SOURCES_NUMBER-1)
;        ***************************************
;        Remember that this version does not implement
;        the PVT into the RAM.
;        ***************************************
```

```
;*************************************************************************
SWI_VECTOR_Base              EQU     EXCEPTION_VECTOR_Limit+4 ;
SWI_VECTOR_Limit    EQU     SWI_VECTOR_Base + 4*32 -4
;
;
;-------------------------- special addresses-------------------------------------------------------
----------
RESERVED_MEMORY_Base        EQU      SWI_VECTOR_Limit+4
RESERVED_MEMORY_Limit       EQU      RESERVED_MEMORY_Base+4*32-4
;
;-------------------------- external memory---------------------------------------------------------
----------
ROM_EXT_Base       EQU      0x200000
RAM_EXT_Base       EQU      0x300000
;
;
;-------------------------- User data for general purposes -----------------------------------------
----------
INT_RAM_USER_DATA_Base      EQU      RESERVED_MEMORY_Limit+4
INT_RAM_USER_DATA_Limit EQU STACK_Limit-4
;
;-------------------------- scape ------------------------------------------------------------------
-------------
FIQ_ESCAPE         EQU      ROM_EXT_Base


;*******************************************************
;                            PERIPHERAL MAP
;        Based on in Data-sheet (Dec.98 version)
;
;*******************************************************

;-------------------------- ASB map ---------------------------

EMC_Base  EQU      0x400000
AIC_Base  EQU      0X410000
APB_Base  EQU      0x420000

;-------------------------- APB map---------------------------
UART_Base EQU      0x420000
BPP_Base  EQU      0x430000
ADC_Base  EQU      0x440000
DAC_Base  EQU      0x450000
TIMER_Base         EQU      0x460000
WD_Base   EQU      0x470000
SBM_Base  EQU      0x480000
SPI_Base  EQU      0x490000
EXT_PER_0_Base     EQU      0x4A0000
EXT_PER_1_Base     EQU      0x4B0000
EXT_PER_2_Base     EQU      0x4C0000
EXT_PER_3_Base     EQU      0x4D0000

;--------------------------E M C--------------------------
emc_BROM  EQU      EMC_Base
emc_STROM EQU      EMC_Base+0x4
emc_ATROM EQU      EMC_Base+0x8
emc_HTROM EQU      EMC_Base+0xc
emc_BRAM  EQU      EMC_Base+0x10
emc_STRAM EQU      EMC_Base+0x14
emc_ATRAM EQU      EMC_Base+0x18
emc_HTRAM EQU      EMC_Base+0x1c

;--------------------------A I C ----------------------------                    -
aic_IRS            EQU      AIC_Base
aic_ISS            EQU      AIC_Base+0x4
aic_IER            EQU      AIC_Base+0x8
aic_IES            EQU      AIC_Base+0x8
aic_IEC            EQU      AIC_Base+0xC
aic_SIT            EQU      AIC_Base+0x10
aic_FSS            EQU      AIC_Base+0x100
aic_FRS            EQU      AIC_Base+0x104
aic_FER            EQU      AIC_Base+0x108
aic_FES            EQU      AIC_Base+0x108
aic_FEC            EQU      AIC_Base+0x10C
aic_PRS            EQU      AIC_Base+0x200
;--------------------------------
;    AIC BIT IDENTIFICATION
;--------------------------------

FIQ                EQU      0x0
PI                 EQU      0x1
COMMRX             EQU      0x2
COMMTX             EQU      0x3

TIMER1             EQU      0x4
TIMER2             EQU      0x5
```

```
PCM_A               EQU     0x6
PCM_B               EQU     0x7

SER_A               EQU     0x8
SER_B               EQU     0x9
BPP                 EQU     0xA
ADC                 EQU     0xB

SBM                 EQU     0xC
SPI                 EQU     0xD
EP1                 EQU     0xE
EP2                 EQU     0xF

EP3                 EQU     0x10
EP4                 EQU     0x11

IRQ_SOURCES_NUMBER EQU 20 ; tiene que ser multiplo de 4


;----------------------------UART----------------------------
uart_RBR EQU      UART_Base+0
uart_THR EQU      UART_Base+0
uart_DLL EQU      UART_Base+0
uart_IER EQU      UART_Base+1
uart_DLM EQU      UART_Base+1
uart_IIR EQU      UART_Base+2
uart_FCR EQU      UART_Base+2
uart_LCR EQU      UART_Base+3
uart_MCR EQU      UART_Base+4
uart_LSR EQU      UART_Base+5
uart_MSR EQU      UART_Base+6
uart_SCR EQU      UART_Base+7

;----------------------------B P P----------------------------
bpp_IAMR            EQU     BPP_Base
bpp_IMR             EQU     BPP_Base+0x4
bpp_ISR             EQU     BPP_Base+0x8
bpp_IAR             EQU     BPP_Base+0xC
bpp_IIMR EQU      BPP_Base+0x10
bpp_IIECREQU      BPP_Base+0x14
bpp_IIDCREQU      BPP_Base+0x18
bpp_IOLR EQU      BPP_Base+0x1c
bpp_OLR             EQU     BPP_Base+0x20

;----------------------------A / D C----------------------------
adc_DR              EQU     ADC_Base+0x0
adc_CR              EQU     ADC_Base+0x4
adc_CE              EQU     ADC_Base+0x4
adc_CD              EQU     ADC_Base+0x8

;----------------------------D / A C----------------------------
dac_DR              EQU     DAC_Base+0x0
dac_CR              EQU     DAC_Base+0x4
dac_CE              EQU     DAC_Base+0x4
dac_CD              EQU     DAC_Base+0x8

;----------------------------T I M E R----------------------------
timer_TC0_LR        EQU     TIMER_Base+0x0
timer_TC0_CV        EQU     TIMER_Base+0x4
timer_TC0_CI        EQU     TIMER_Base+0x8
timer_TC0_CR        EQU     TIMER_Base+0xC
timer_TC1_LR        EQU     TIMER_Base+0x20
timer_TC1_CV        EQU     TIMER_Base+0x24
timer_TC1_CI        EQU     TIMER_Base+0x28
timer_TC1_CR        EQU     TIMER_Base+0x2C
timer_TC2_LR        EQU     TIMER_Base+0x40
timer_TC2_CV        EQU     TIMER_Base+0x44
timer_TC2_CI        EQU     TIMER_Base+0x48
timer_TC2_CR        EQU     TIMER_Base+0x4C

;----------------------------W D----------------------------
wd_CV               EQU     WD_Base+0x0
wd_CCV   EQU      WD_Base+0x0
wd_LR               EQU     WD_Base+0x4
wd_ER               EQU     WD_Base+0x8

;----------------------------S P I----------------------------                    -
spi_RBR  EQU      SPI_Base
spi_THR  EQU      SPI_Base
spi_DLR  EQU      SPI_Base+1
spi_FCR  EQU      SPI_Base+2
spi_LCR  EQU      SPI_Base+3
spi_FSR  EQU      SPI_Base+4
spi_LSR  EQU      SPI_Base+5


;---------------------------- S B M ----------------------------
```

```
sbm_RBR_0 EQU      SBM_Base
sbm_RBR_1 EQU      SBM_Base+0x04
sbm_IIR   EQU      SBM_Base+0x08
sbm_IER   EQU      SBM_Base+0x0C
sbm_MOD_0 EQU      SBM_Base+0x10
sbm_MOD_1 EQU      SBM_Base+0x14
sbm_LSR_0 EQU      SBM_Base+0x20
sbm_LSR_1 EQU      SBM_Base+0x24
sbm_NSE_00         EQU      SBM_Base+0x28
sbm_NSE_01         EQU      SBM_Base+0x2C
sbm_NSE_10         EQU      SBM_Base+0x30
sbm_NSE_11         EQU      SBM_Base+0x34
sbm_CCR   EQU      SBM_Base+0x50
sbm_BDT   EQU      SBM_Base+0x54
sbm_BPT   EQU      SBM_Base+0x58
sbm_THR   EQU      SBM_Base+0x5C

;*********************************************************
;                          TEST PARAMETERS
;*********************************************************

;----------------------------------------
;         BPP     messages
;----------------------------------------
MSG_TESTING_RAM             EQU      0x1
MSG_UART_TEST               EQU      0x2
MSG_SPI_TEST                EQU      0x4
MSG_DAC_ADC_TEST   EQU      0x8
MSG_IMC_TEST                EQU      0x10

;----------------------------------------
;         Memory FLAGS
;----------------------------------------
RAM_TEST_OK                 EQU      0xFF00
RAM_TEST_FAILED             EQU      0x00FF
UART_TEST_OK                EQU      0xFF00
UART_TEST_FAILED            EQU      0x00FF
SPI_TEST_OK                          EQU      0xFF00
SPI_TEST_FAILED             EQU      0x00FF
IMC_TEST_OK                          EQU      0xFF00
IMC_TEST_FAILED             EQU      0x00FF
DAC_ADC_TEST_OK             EQU      0xFF00
DAC_ADC_TEST_FAILED         EQU      0x00FF

;----------------------------------------
;         Test sizes
;----------------------------------------
FLAG_AREA_Size              EQU      0x40
UART_TEST_Size              EQU      0x4       ;(64)
SPI_TEST_Size                        EQU      0x80    ;(128bytes)
IMC_TEST_BYTES_Size         EQU      0x100     ;(256)
IMC_TEST_HALFWS_Size        EQU      0x80
IMC_TEST_WORDS_Size         EQU      0x40
IMC_TEST_Size               EQU      (IMC_TEST_WORDS_Size*4)+(IMC_TEST_HALFWS_Size*2)+IMC_TEST_BYTES_Size
DAC_ADC_TEST_Size           EQU      0x8
EMC_TEST_Size               EQU      0x0
FLAG_TEST_Size              EQU      0x14

;----------------------------------------
;         Test AREAS
;----------------------------------------
FLAG_TEST_AREA_Begin                              EQU      INT_RAM_USER_DATA_Base

IMC_TEST_TARGET_AREA_Begin          EQU      FLAG_TEST_AREA_Begin+FLAG_AREA_Size
IMC_TEST_SOURCE_AREA_Begin          EQU      0x84

EMC_TEST_TARGET_AREA_Begin          EQU      IMC_TEST_TARGET_AREA_Begin+IMC_TEST_Size
EMCI_TEST_SOURCE_AREA_Begin         EQU      0x84

DAC_ADC_TEST_TARGET_AREA_Begin      EQU      EMC_TEST_TARGET_AREA_Begin+EMC_TEST_Size
DAC_ADC_TEST_SOURCE_AREA_Begin      EQU      0x84

UART_TEST_TARGET_AREA_Begin         EQU      DAC_ADC_TEST_TARGET_AREA_Begin+DAC_ADC_TEST_Size
UART_TEST_SOURCE_AREA_Begin         EQU      0x84

SPI_TEST_TARGET_AREA_Begin          EQU      UART_TEST_TARGET_AREA_Begin+UART_TEST_Size
SPI_TEST_SOURCE_AREA_Begin          EQU      0x84


;----------------------------------------
;         Test FLAGs
;----------------------------------------
RAM_TEST_FLAG_Addr EQU      FLAG_TEST_AREA_Begin
IMC_TEST_FLAG_Addr EQU      FLAG_TEST_AREA_Begin+0x04
EMC_TEST_FLAG_Addr EQU      FLAG_TEST_AREA_Begin+0x08
DAC_ADC_TEST_FLAG_Addr      EQU      FLAG_TEST_AREA_Begin+0x0C
UART_TEST_FLAG_Addr         EQU      FLAG_TEST_AREA_Begin+0x10
```

```
SPI_TEST_FLAG_Addr          EQU      FLAG_TEST_AREA_Begin+0x14


;*******************************************************
;                           ARM CODE
;       Begins with the exceoption vectors
;*******************************************************


       ENTRY
;---------------------------initial vector table (BEGIN)---------------------------
       B       Reset_Handler
       B       Reset_Handler
       B       Reset_Handler
       B       Reset_Handler
       B       Reset_Handler
       NOP                         ; Reserved vector
       B       IRQ_Handler
       B       Reset_Handler

;---------------------------initial vector table (END)-----------------------------


;*****************************************************
; SUBROUTINE INFORMATION:
;        Name:                      RESET_HANDLER_ver_3
;        Author:                    J. Granado
;        Date of creation           Dec. 16, 1998
;        Last review
;        Version                    ver 3.0
;        Comments:
;
;
;
;*****************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;*****************************************************
; EXECUTION CONTEXT
;        Reset state.
;
;*****************************************************
; LABELS
;
;
;*****************************************************



Reset_Handler
Reset_Handler_ver_3_Begin

;++++++++++++++++++++++++++++++++
; ARM state:       IRQ disable
;                  FIQ disable
;                  Supervision Mode
;++++++++++++++++++++++++++++++++
         ;  Register used:
         ;        R0  -> data
         ;        R10 -> bpp output level register

;---------------------------
; ---> Initialise registers
;---------------------------

         mov      r0,#0
         mov      r1,#0
         mov      r2,#0
         mov      r3,#0
         mov      r4,#0
         mov      r5,#0
         mov      r6,#0
         mov      r7,#0
         mov      r8,#0
         mov      r9,#0
         mov      r10,#0
         mov      r11,#0
         mov      r12,#0
         mov      r13,#0
         mov      r14,#0

;-----------------------------------------------------
; Enter IRQ mode and set up the IRQ stack pointer
;---------------------------;-------------------------
       MOV     R0, #Mode_IRQ:OR:I_Bit:OR:F_Bit ; No interrupts
       MSR     CPSR, R0
       LDR     R13, =IRQ_Stack
```

```
;-------------------------------------------------------------------
; Set up the SVC stack pointer last and return to SVC mode
;-------------------------------------------------------------------
       MOV     R0, #Mode_SVC:OR:I_Bit:OR:F_Bit ; No interrupts
       MSR     CPSR, R0
       LDR     R13, =SVC_Stack

;-------------------------------------------------
;      Disable watchdog
;-------------------------------------------------
         LDR      R0, =wd_ER
         MOV      R1,#0x0
         STR      R1,[R0]

;++++++++++++++++++++++++++++++++++++++++
; ARM STATE:       - IRQ disabled and FIQ enabled
;                  - supervision mode
;++++++++++++++++++++++++++++++++++++++++

;-------------------------------------------------------------------------------
;     Now check and clear Internal RAM
;-------------------------------------------------------------------------------
         ;  Register used:
         ;        R0  -> Internal ram base addr
         ;        R1  -> not used
         ;        R2  -> 0000
         ;        R3  -> fail counter
         ;        R4  -> not used
         ;        R5  -> value read from ram
         ;        R6  -> int ram test ok msg
         ;        R7  -> int ram test failed msg
         ;        R9  -> counter
         ;        R10 -> ram test flag addr
         ;        R11 -> RAM LIMIT

;        ------------------------------------------------------
         LDR      R4,=bpp_LMR       ; LSB bpp pins as outputs
         MOV      R1,#0x00FF
         STR      R1,[R4]
         LDR      R4,=bpp_OLR
         MOV      R1,#MSG_TESTING_RAM
         STR      R1,[R4]
;        ------------------------------------------------------

         LDR      R0, =RAM_INT_Base
         MOV      R2, #0x0
         MOV      R3, #0
         MOV      R6, #RAM_TEST_OK
         MOV      R7, #RAM_TEST_FAILED
         MOV      R9, #0x0
         LDR      R10, =RAM_TEST_FLAG_Addr
         LDR      R11, =RAM_INT_Limit

Reset_Handler_ver_3_label_2
         STR      R9, [R0], #4
         ADD      R9,R9,#1
         CMP      R0,R11
         BLE      Reset_Handler_ver_3_label_2

         MOV      R9, #0
         LDR      R0, =RAM_INT_Base

Reset_Handler_ver_3_label_3
         LDR      R5,[R0]
         CMP      R5,R9
         ORRNE    R3,R3,#0x1
         STR      R2, [R0], #4
         ADD      R9, R9, #1
         CMP      R0, R11
         BLE      Reset_Handler_ver_3_label_3

         TST      R3,#0x01
         STRNE    R7,[R10]  ; ram test has failed
         STREQ    R6,[R10]  ; ram test has finished OK

;-----------------------------------------------------
; Enable Watchdog with 45sec aprox
;-----------------------------------------------------


         LDR      R0,=wd_LR
         LDR      R1,=0x249F0       ; 25 milisec msec as time out
         STR      R1,[R0]

         LDR      R0, =wd_ER
         MOV      R1,#0xFF
```

```
        STRB    R1,[R0]

        LDR     R0,=wd_CCV      ; reset wd
        MOV     R1,#0
        STR     R1,[R0]

Reset_Handler_ver_3_End

        BL      USER_CODE; jump to user code


;++++++++++++++++++++++++++++++++++++++++
; ARM STATE:      - IRQ disabled and FIQ enabled
;                 - supervision MODE
;++++++++++++++++++++++++++++++++++++++++

;******(END of ROUTINE)*****************************

;**************************************************
; SUBROUTINE INFORMATION:
;       Name:                   USER_CODE
;       Author:                 J. Granado
;       Date of creation        Dec. 21, 1998
;       Last review (4.0)  Jan. 20, 1999
;       Version                 ver 1.0
;       Comments:
;
;**************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;**************************************************
; EXECUTION CONTEXT
;
;**************************************************
USER_CODE
USER_CODE_Begin

        BL      IMC_TEST
;       BL      EMC_TEST
        BL      DAC_ADC_TEST
        BL      UART_TEST
        BL      SPI_TEST

USER_CODE_End
;********(END of ROUTINE*****************************




;**************************************************
; SUBROUTINE INFORMATION:
;       Name:                   IRQ_Handler
;       Author:                 J. Granado
;       Date of creation        July 7, 1998
;       Last review (4.0)  Nov. 4, 1998
;       Version                 ver 4.0
;       Comments:
;                tested (ARMulator + Verilog )
;                compatible with software aic implementation
;                (POLICOM DIGITAL SPECIFICATIONS ver 7.0, Nover, 1998)
;                - Hardware Priority Implementation (hidden feratures)
;
;**************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;**************************************************
; EXECUTION CONTEXT
;                - SPSR contains old CPSR
;                - IRQ's are disabled
;                - LR contains return PC
;                - ARM mode
;
;**************************************************
; LABELS
;
;
;**************************************************
IRQ_Handler

IRQ_Handler_ver_4_0_begin

        SUB     LR,LR,#4        ; first of all we will save arm state
        STMDB   SP!, {LR}       ; return address and working registers
        MRS     R14, SPSR       ;
        STMDB   SP!, {R0-R12,R14}
```

```
;        ******************
;        * RESET WATCHDOG *
;        ******************

         LDR      R0,=wd_CCV         ; reset wd
         MOV      R1,#0
         STR      R1,[R0]

;          *   Used registers
;        -R0 -->
;        -R1 -->
;        -R2 -->
;        -R3 --> aic PRS
;        -R4 --> aic IRS
;        -R5 -->
;        -R6 --> Exception Vector Base addr
;        -R7 --> Vector
;        -R8 --> aic PRS addr
;        -R9 --> aic IRS addr
;        -R10 ->

         LDR      R6, =EXCEPTION_VECTOR_Base
         LDR      R8,=aic_PRS
         LDR      R9,=aic_IRS

         LDR      R4,[R9]
         LDR      R3,[R8]
         LDR      R7,[R6,R3,LSL#2]

         TEQ      R4,#0
         BEQ      IRQ_Handler_ver_4_0_end

         MOV      LR,PC                  ;branching and linking
         MOV      PC, R7

IRQ_Handler_ver_4_0_end


         LDMIA    SP!, {R0-R12,r14}
         MSR      SPSR, R14
         LDMIA    SP!, {PC}^ ; cuidadín con el ^


;******(END of ROUTINE)*****************************


;*************************************************
; SUBROUTINE INFORMATION:
;        Name:                      UART_TEST
;        Author:                    J. Granado
;        Date of creation           August 27, 1998
;        Last review (new ver)      Jan 20, 1999
;        Version                     2.0
;        Comments:
;        - Based on UART_TEST_ver_2_0
;
;*************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;*************************************************
; EXECUTION CONTEXT
;                 - works with a IRQ handler
;*************************************************
; LABELS
;
;
;*************************************************

UART_TEST
UART_TEST_ver_2_0_Begin

         STMDB    SP!, {R14}         ; storing return addr.
         LDR      R0, =wd_CCV        ; reset wd
         MOV      R1,#0
         STR      R1,[R0]
         MOV      R0, #Mode_SVC:OR:I_Bit              ; Disable IRQ in arm
         MSR      CPSR, R0
;        ------------------------------------------------------
         LDR      R4,=bpp_LMR        ; LSB bpp pins as outputs
         MOV      R1,#0x00FF
         STR      R1,[R4]
         LDR      R4,=bpp_OLR
         MOV      R1,#MSG_UART_TEST
         STR      R1,[R4]
;        ------------------------------------------------------
;         SET UP AIC PARAMETERS
```

```
;       ------------------------------------------------

        ; Register used:
        ;       R0  -> data
        ;       R1  -> IRQ NUMBER
        ;       R8  -> IRQ VECTOR
        ;       R9  -> priority table
        ;       R10 -> addr of int. enable set, evt

        LDR     R10, =aic_IES
        MOV     R0, #0x1
        MOV     R0, R0, LSL  #SER_A
        STR     R0, [R10]                                       ;enable uart irq in aic

        LDR     R10, =EXCEPTION_VECTOR_Base
        LDR     R9, =EXCEPTION_PRIORITY_Base
        LDR     R8, =UART_TEST_IRQ_HANDLER

        MOV     R0, #0x1
        LDR     R1,=SER_A
        MOV     R1, R1, LSL #2
        STR     R0,[R9,R1]          ; priority
        STR     R8, [R10,R1]        ; vector

UART_TEST_ver_2_0_Label_1

        ; UART PARAMETERS

        ; Register used:
        ;       R1  -> DATA
        ;       R5  -> uart line control register
        ;       R6  -> uart divisor latch
        ;       R7  -> uart int enable register
        ;       R8  -> uart modem control register
        ;       R9  -> addr of reserved memory

        LDR     R5, =uart_LCR
        LDR     R6, =uart_DLL
        LDR     R7, =uart_IER
        LDR     R8, =uart_MCR

        MOV     R1, #0x80
        STRB    R1, [R5]        ;DLAB=1


        MOV     R1, #0x27; 9600 bps
        STRB    R1, [R6]
        MOV     R1, #0
        STRB    R1,[R6,#1]

        MOV     R1, #0x7
        STRB    R1,[R5]
                ; 8 bits per character
                ; 1 stop bit
                ; odd parity

        MOV     R1, #0x3  ; enable irq for tx and rx
        STRB    R1, [R7]

        MOV     R1, #0x13; enable loopback mode
        STRB    R1, [R8]  ; dtr = 1; rts=1

        ; Now reset first positions of reserved memory area.
        ; This area will contain two data:
        ;       - number of received bytes
        ;       - number of transmitted bytes
        ; These bytes con only be modify by irq handler

        LDR     R9, =RESERVED_MEMORY_Base
        MOV     R1, #0x0
        STR     R1, [R9]

        ; Now enable ARM irq

        MOV     R0, #Mode_SVC                                   ; Enable IRQ in arm
        MSR     CPSR, R0

        LDR     R0, =0x1388                                                     ;wait 5000usec.
        BL      WAIT_ver_3_0

        ; Disable irq in arm

        MOV     R0, #Mode_SVC:OR:I_Bit                          ;Disable IRQ in arm
        MSR     CPSR, R0

        LDR     R10, =aic_IEC                                           ; Clear bit mask in aic
        MOV     R1, #0x1
```

```
        MOV     R1, R1, LSL #SER_A
        STR     R1, [R10] ; clear request aic


        ; now we will check results

        ;  Register used:
        ;       R0  -> data
        ;       R1  -> msg UART_test OK
        ;       R2  -> msg UART test failed
        ;       R3  -> flag
        ;       R4  -> counter
        ;       R5  -> not used
        ;       R6  -> bpp output mode register
        ;       R7  -> bpp output line register
        ;       R8  -> user data base memory (target buffer)
        ;       R9  -> source buffer base addr
        ;       R10 -> received data
        ;       R11 -> sent data

        MOV     R1, #UART_TEST_OK
        MOV     R2, #UART_TEST_FAILED
        MOV     R3, #0x0
        MOV     R4, #0x0
        LDR     R7, =UART_TEST_FLAG_Addr
        LDR     R8, =UART_TEST_TARGET_AREA_Begin
        LDR     R9, =UART_TEST_SOURCE_AREA_Begin

UART_TEST_ver_2_0_Label_2

        LDRB    R10, [R8, R4]
        LDRB    R11, [R9, R4]

        CMP     R11,R10
        MOVNE   R3,#0x1

        ADD     R4,R4,#0x1
        CMP     R4,#UART_TEST_Size
        BLT     UART_TEST_ver_2_0_Label_2

        TST     R3, #0x1
        STREQ   R1, [R7]
        STRNE   R2, [R7]

UART_TEST_ver_2_0_End

        LDMIA   SP!, {PC}

;******(END of ROUTINE)*****************************


;*****************************************************
; SUBROUTINE INFORMATION:
;       Name:                   UART_TEST_IRQ_HANDLER_ver_3
;       Author:                 J. Granado
;       Date of creation        Sep. 27, 1998
;       Last review (new ver)   Dec.18, 1998
;       Version                  3.0
;       Comments:
;
;
;*****************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;*****************************************************
; EXECUTION CONTEXT
;
;*****************************************************
; LABELS
;
;
;*****************************************************

UART_TEST_IRQ_HANDLER
UART_TEST_IRQ_HANDLER_Begin

        STMDB   SP!, {LR}

UART_TEST_IRQ_HANDLER_ver_3_0_Begin
        ;       R0  -> [R6], bytes counter
        ;       R1  -> received or trasmitted bytes
        ;       R2  -> [R8]
        ;       R3  -> number of received bytes
        ;       R4  -> number of transmitted bytes
        ;       R5  -> int enable reg. for next time
        ;       R6  -> aic int request status addr
        ;       R7  -> uart interrupt enable register addr
```

```
        ;       R8  -> uart interrupt identification reg
        ;       R9  -> uart rx buffer reg / tx holding reg
        ;       R10 -> reserved memory base
        ;       R11 -> ram int base memory
        ;       R12 ->

        LDR     R6, =aic_IRS
        LDR     R7, =uart_IER
        LDR     R8, =uart_IIR
        LDR     R9, =uart_RBR
        LDR     R10,=RESERVED_MEMORY_Base
        LDR     R11,=UART_TEST_TARGET_AREA_Begin

        ; first test aic to find uart irq source
        ; if there is no irq source active, jump to end

        LDR     R0, [R6]
        MOV     R1, #0x1                        ; test aic
        MOV     R1,R1,LSL #SER_A
        TST     R1,R0
        BEQ     UART_TEST_IRQ_HANDLER_ver_3_0_End


        ; if there is uart irq source, check to find
        ; if it is rx or tx irq
        ; First rx


UART_TEST_IRQ_HANDLER_ver_3_0_Label_0

        LDRB    R2, [R8]        ; read interrupt  identification registro
        LDRB    R3, [R10]       ; numero de bytes hasta ahora recibidos
        LDRB    R4, [R10,#1]            ; numero de bytes hasta ahora transmitidos

UART_TEST_IRQ_HANDLER_ver_3_0_Label_1

        TST     R2, #0x04                       ; RX interrupt
        BEQ     UART_TEST_IRQ_HANDLER_ver_3_0_Label_2

        LDRB    R1, [R9]        ; read uart buffer
        STRB    R1,[R11, R3]            ; store read byte into RAM
        ADD     R3,R3,#0x1              ; READ=READ++
        STRB    R3, [R10]       ; --> RAM
        CMP     R3, #UART_TEST_Size     ;  has been received
        BLT     UART_TEST_IRQ_HANDLER_ver_3_0_Label_2
        LDRB    R5,[R7]
        BIC     R5,R5,#0x1              ;disable rx interrupt
        STRB    R5, [R7]

UART_TEST_IRQ_HANDLER_ver_3_0_Label_2

        TST     R2, #0x02       ; TX Interrupt ?
        BEQ     UART_TEST_IRQ_HANDLER_ver_3_0_Label_3

        LDRB    R1, [R4,#UART_TEST_SOURCE_AREA_Begin]       ; read from rom source buffer
        STRB    R1, [R9]        ; TX
        ADD     R4,R4,#1        ; written = written ++
        STRB    R4,[R10,#1]             ; -->RAM
        CMP     R4, #UART_TEST_Size     ; if the target buffer has been completed
        BLT     UART_TEST_IRQ_HANDLER_ver_3_0_Label_3
        LDRB    R5,[R7]
        BIC     R5,R5,#0x2              ;disable rx interrupt
        STRB    R5, [R7]

UART_TEST_IRQ_HANDLER_ver_3_0_Label_3

        TST     R2, #0x06
        BNE     UART_TEST_IRQ_HANDLER_ver_3_0_Label_0


UART_TEST_IRQ_HANDLER_ver_3_0_End
        LDMIA   SP!, {PC}


;******(END of ROUTINE)*****************************



;*******************************************************
; SUBROUTINE INFORMATION:
;       Name:                   SPI_TEST
;       Author:                 J. Granado
;       Date of creation        Oct. 27, 1998
;       Last review (new ver)   Dec.18, 1998
;       Version                  2.0
;       Comments:
;
```

```
;
;*****************************************************
; SUBROUTINE PARAMEMTERS:
;                 -  no argument needed

;*****************************************************
; EXECUTION CONTEXT
;        -  works with a irq handler
;        -  uses wait subroutine
;*****************************************************
; LABELS
;
;
;*****************************************************

SPI_TEST
SPI_TEST_ver_2_0_Begin

        STMDB   SP!, {R14}        ; storing return addr.

        LDR     R0, =wd_CCV       ; reset wd
        MOV     R1,#0
        STR     R1,[R0]

        MOV     R0, #Mode_SVC:OR:I_Bit              ; Disable IRQ in arm
        MSR     CPSR, R0

;       -----------------------------------------------------
        LDR     R4,=bpp_IMR       ; LSB bpp pins as outputs
        MOV     R1,#0x00FF
        STR     R1,[R4]
        LDR     R4,=bpp_OLR
        MOV     R1,#MSG_SPI_TEST
        STR     R1,[R4]
;       -----------------------------------------------------

        ; SET UP AIC PARAMETERS
        ; Register used:
        ;       R0  -> data
        ;       R1  -> IRQ NUMBER
        ;       R8  -> IRQ VECTOR
        ;       R9  -> priority table
        ;       R10 -> addr of int. enable set, evt

        LDR     R10, =aic_IES
        MOV     R0, #0x1
        MOV     R0, R0, LSL  #SPI
        STR     R0, [R10]                                   ;enable SPI irq in aic

        LDR     R10, =EXCEPTION_VECTOR_Base
        LDR     R9, =EXCEPTION_PRIORITY_Base
        LDR     R8, =SPI_TEST_IRQ_HANDLER_ver_2_0

        MOV     R0, #0x1
        LDR     R1,=SPI
        MOV     R1, R1, LSL #2
        STR     R0,[R9,R1]                                  ;write  priority and vector
        STR     R8, [R10,R1]      ; vector

SPI_TEST_ver_2_0_Label_1

        ; SPI PARAMETERS

        ; Register used:
        ;       R0  -> data
        ;       R5  -> SPI divisor latch reg addr.
        ;       R6  -> SPI fifo contro reg addr.
        ;       R7  -> SPI line control reg addr.

        LDR     R5, =spi_DLR
        LDR     R6, =spi_FCR
        LDR     R7, =spi_LCR

        MOV     R0, #0xA4 ; rx fifo size = A
        STRB    R0, [R6]          ; tx fifo size = 4
        MOV     R0, #0x60 ; rx & tx irq enable
        STRB    R0, [R7]          ; parity disable
        MOV     R0, #0x0  ; 1 Mbps
        STRB    R0, [R5]

        ; Now reset first positions of reserved memory area.
        ; This area will contain two data:
        ;       - number of received bytes
        ;       - number of transmitted bytes
        ; These bytes con only be modify by irq handler

        LDR     R9, =RESERVED_MEMORY_Base
```

```
        MOV     R1, #0x0
        STR     R1, [R9]
        STR     R1,[R9,#4]


        ; Now enable ARM irq

        MOV     R0, #Mode_SVC               ; Enable IRQ in arm
        MSR     CPSR, R0

        LDR     R0, =0x5DC                          ;wait 1500 usec.(h_0x5DC)
        BL      WAIT_ver_3_0

        ; Disable irq in arm
        MOV     R0, #Mode_SVC:OR:I_Bit       ;Disable IRQ in arm
        MSR     CPSR, R0

        LDR     R10, =aic_IEC                       ; Clear bit mask in aic
        MOV     R1, #0x1
        MOV     R1, R1, LSL #SPI
        STR     R1, [R10]                   ; clear request aic

        LDR     R7, =spi_LCR                        ; set SS signal to 1 (inactive)
        MOV     R1, #0x1
        STRB    R1,[R7]

;       ------------------------------------------------
        ; read residue in spi rx buffer
;       ------------------------------------------------

        ;       R6  -> aic int request status addr
        ;       R7  -> Line controregister addr
        ;       R8  -> Line status  reg addr
        ;       R9  -> SPI  rx buffer reg / tx holding reg
        ;       R10 -> reserved memory base
        ;       R11 -> ram int base memory
        ;       R12 -> FIFO status reg.

        LDR     R6, =aic_IRS
        LDR     R7, =spi_LCR
        LDR     R8, =spi_LSR
        LDR     R9, =spi_RBR
        LDR     R10,=RESERVED_MEMORY_Base
        LDR     R11,=SPI_TEST_TARGET_AREA_Begin
        LDR     R12, =spi_FSR
        LDR     R3, [R10]                   ; numero de bytes hasta ahora recibidos

                                                    ; read residue in spi rx buffer
SPI_TEST_ver_2_0_Label_1_1

        LDR     R0, =SPI_TEST_Size
        CMP     R3,R0                               ; if the source buffer has been received
        BGE     SPI_TEST_ver_2_0_Label_1_3


        LDRB    R1, [R9]          ; read SPI  buffer
        STRB    R1,[R11, R3]                ; store read byte into RAM

        ADD     R3,R3,#0x1                  ; READ=READ++
        STR     R3, [R10]         ; --> RAM

SPI_TEST_ver_2_0_Label_1_2
        LDRB    R0, [R12]
        MOV     R0,R0,LSR #4
        CMP     R0,#0x0           ; Have I read all the buffer ? if no, read again.
        BGT     SPI_TEST_ver_2_0_Label_1_1

SPI_TEST_ver_2_0_Label_1_3
        ; now we will check results
        ;  Register used:
        ;       R0  -> data
        ;       R1  -> msg SPI_test OK
        ;       R2  -> msg SPI  test failed
        ;       R3  -> flag
        ;       R4  -> counter
        ;       R5  -> not used
        ;       R7  -> falg addr
        ;       R8  -> spi target data addr. (target buffer)
        ;       R9  -> source buffer base addr
        ;       R10 -> received data
        ;       R11 -> sent data
        ;       R12 ->

        MOV     R1, #SPI_TEST_OK
        MOV     R2, #SPI_TEST_FAILED
        MOV     R3, #0x0
        MOV     R4, #0x0
```

```
        LDR     R7, =SPI_TEST_FLAG_Addr
        LDR     R8, =SPI_TEST_TARGET_AREA_Begin
        LDR     R9, =SPI_TEST_SOURCE_AREA_Begin

SPI_TEST_ver_2_0_Label_2

        LDRB    R10, [R8, R4]
        LDRB    R11, [R9, R4]

        CMP     R11,R10
        MOVNE   R3,#0x1

        ADD     R4,R4,#0x1

        LDR     R0, =SPI_TEST_Size
        CMP     R4,R0
        BLT     SPI_TEST_ver_2_0_Label_2

        TST     R3, #0x1
        STREQ   R1, [R7]
        STRNE   R2, [R7]

SPI_TEST_ver_2_0_End

        LDMIA   SP!, {PC}

;******(END of ROUTINE)*****************************



;*****************************************************
; SUBROUTINE INFORMATION:
;       Name:                   SPI_TEST_IRQ_HANDLER_ver_2_0
;       Author:                 J. Granado
;       Date of creation        Oct. 8, 1998
;       Last review (new ver)   Dec.18, 1998
;       Version                  2.0
;       Comments:
;
;
;*****************************************************
; SUBROUTINE PARAMEMTERS:
;               -  no argument needed

;*****************************************************
; EXECUTION CONTEXT
;       -  irq handler
;
;*****************************************************
; LABELS
;
;
;*****************************************************
SPI_TEST_IRQ_HANDLER_ver_2_0
        STMDB   SP!, {LR}

SPI_TEST_IRQ_HANDLER_ver_2_0_Begin

        ;       R0  -> [R6], bytes counter, bytes number in FIFO
        ;       R1  -> received or trasmitted bytes
        ;       R2  -> [R8]
        ;       R3  -> number of received bytes
        ;       R4  -> number of transmitted bytes
        ;       R5  -> int enable reg. for next time
        ;       R6  -> aic int request status addr
        ;       R7  -> Line controregister addr
        ;       R8  -> Line status  reg addr
        ;       R9  -> SPI  rx buffer reg / tx holding reg
        ;       R10 -> reserved memory base
        ;       R11 -> ram int base memory
        ;       R12 -> FIFO status reg.

        LDR     R6, =aic_IRS
        LDR     R7, =spi_LCR
        LDR     R8, =spi_LSR
        LDR     R9, =spi_RBR
        LDR     R10,=RESERVED_MEMORY_Base
        LDR     R11,=SPI_TEST_TARGET_AREA_Begin
        LDR     R12, =spi_FSR

        ; first test aic to find SPI  irq source
        ; if there is no irq source active, jump to end

        LDR     R0, [R6]
        MOV     R1, #0x1                        ; test aic
        MOV     R1,R1,LSL #SPI
        TST     R1,R0
```

```
        BEQ     SPI_TEST_IRQ_HANDLER_ver_2_0_End


        ; if there is SPI irq source, check to find
        ; if it is rx or tx irq
        ; First rx


SPI_TEST_IRQ_HANDLER_ver_2_0_Label_0

        LDRB    R2, [R8]          ; read interrupt  identification registro
        LDR     R3, [R10]         ; numero de bytes hasta ahora recibidos
        LDR     R4, [R10,#4]              ; numero de bytes hasta ahora transmitidos
SPI_TEST_IRQ_HANDLER_ver_2_0_Label_1 ; RX ?

        TST     R2, #0x04                                       ; RX interrupt if LSR==40
        BEQ     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_1_1
        LDRB    R1, [R9]                  ; read SPI  buffer
        STRB    R1,[R11, R3]              ; store read byte into RAM

        ADD     R3,R3,#0x1                ; READ=READ++
        STR     R3, [R10]         ; --> RAM

        LDR     R0, =SPI_TEST_Size
        CMP     R3,R0             ; if the source buffer has been received
        BLT     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_1_2

        LDRB    R5,[R7]
        BIC     R5,R5,#0x20               ;disable rx interrupt if  transfer is completed
        STRB    R5, [R7]
        B       SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_1_2
        LDRB    R0, [R12]
        MOV     R0,R0,LSR #4
        CMP     R0,#0x0           ; Have I read all the buffer ? if no, read again.
        BGT     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_1_1

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2
        TST     R2, #0x08         ; TX Interrupt ?
        BEQ     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_3

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2_1
        LDRB    R1, [R4,#SPI_TEST_SOURCE_AREA_Begin]          ; read from rom 0x84
        STRB    R1, [R9]                                                ; TX

        ADD     R4,R4,#1          ; written = written ++
        STR     R4,[R10,#4]               ; -->RAM

        LDR     R0, =SPI_TEST_Size
        CMP     R4, R0                              ; if the target buffer has been completed ???
        BLT     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2_2

        LDRB    R5,[R7]
        BIC     R5,R5,#0x40               ;disable tx interrupt if Buffer has been transmitted compleatly
        STRB    R5, [R7]
        B          SPI_TEST_IRQ_HANDLER_ver_2_0_Label_3

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2_2
        LDRB    R0, [R12]
        AND     R0,R0,#0x0F
        CMP     R0,#0xF           ; Have I precharged  all the tx buffer ? if no,write  again.
        BLT     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_2_1

SPI_TEST_IRQ_HANDLER_ver_2_0_Label_3
        TST     R2, #0xC
        BNE     SPI_TEST_IRQ_HANDLER_ver_2_0_Label_0

SPI_TEST_IRQ_HANDLER_ver_2_0_End
        LDMIA   SP!, {PC}

;******(END of ROUTINE)*******************************



;*****************************************************
; SUBROUTINE INFORMATION:
;       Name:                       IMC_TEST
;       Author:                     J. Granado
;       Date of creation            Dec 18, 1998
;       Last review (new ver)
;       Version                      1.0
;       Comments:
;
;
```

```
;*****************************************************
; SUBROUTINE PARAMEMTERS:
;                 -  no argument needed

;*****************************************************
; EXECUTION CONTEXT
;        -
;
;*****************************************************
; LABELS
;
;
;*****************************************************

IMC_TEST
IMC_TEST_ver_1_0_Begin

;        ----------------------------
;        * RESET WATCHDOG *
;        ----------------------------

         LDR      R0,=wd_CCV         ; reset wd
         MOV      R1,#0
         STR      R1,[R0]
         STMDB    SP!, {r14}          ;save return addr.

         LDR      R4,=bpp_IMR        ; all bpp pins as outputs
         MOV      R1,#0xFF
         STR      R1,[R4]
         LDR      R4,=bpp_OLR
         MOV      R1,#MSG_IMC_TEST
         STR      R1,[R4]

;        ------------------------------------------
;        Registers:
;        R1-----> 
;        R2-----> data
;        R3-----> 
;        R4-----> transfer size
;        R5-----> counter
;        R6----->IMC SOURCE BUFFERaddr
;        R7----->IMC TARGET BUFFER addr
;        R8-----> IMC_TEST_OK
;        R9-----> IMC_TEST_FAILED
;        R10---->IMC_TEST_FLAG_Addr.
;        R11---->
;        R12---->
;        ------------------------------------------

         MOV      R2,#0x0
         MOV      R3,#0x0
         MOV      R4,#0x0
         MOV      R5,#0x0
         LDR      R6, =IMC_TEST_SOURCE_AREA_Begin
         LDR      R7, =IMC_TEST_TARGET_AREA_Begin
         MOV      R8, #IMC_TEST_OK
         MOV      R9, #IMC_TEST_FAILED
         LDR      R10, =IMC_TEST_FLAG_Addr

;        ------------------------------------------
;        Testing BYTE transfer
;        ------------------------------------------

         MOV      R4,#IMC_TEST_BYTES_Size

IMC_TEST_ver_1_0_Label_1

         LDRB     R2,[R6,R5]
         STRB     R2,[R7,R5]
         ADD      R5,R5,#1
         CMP      R4,R5
         BGT      IMC_TEST_ver_1_0_Label_1
         ADD      R6,R6,R5
         ADD      R7,R7,R5


;        ------------------------------------------
;        Testing HALFWORD transfer
;        ------------------------------------------

         MOV      R4,#IMC_TEST_HALFWS_Size
         MOV      R5,#0x0
         MOV      R4,R4,LSL#1

IMC_TEST_ver_1_0_Label_2

         LDRH     R2,[R6,R5]
```

```
        STRH    R2,[R7,R5]
        ADD     R5,R5,#2
        CMP     R4,R5
        BGT     IMC_TEST_ver_1_0_Label_2
        ADD     R6,R6,R5
        ADD     R7,R7,R5

;       ----------------------------------------
;       Testing WORD  transfer
;       ----------------------------------------

        MOV     R4,#IMC_TEST_WORDS_Size
        MOV     R5,#0x0

IMC_TEST_ver_1_0_Label_3

        LDR     R2,[R6,R5, LSL #2]
        STR     R2,[R7,R5, LSL#2]
        ADD     R5,R5,#1
        CMP     R4,R5
        BGT     IMC_TEST_ver_1_0_Label_3

;       ----------------------------------------
;       Checking results (byte by byte)
;       ----------------------------------------
        MOV     R1,#0x0
        MOV     R2,#0x0
        MOV     R3,#0x0
        MOV     R4, #IMC_TEST_Size
        MOV     R5,#0x0
        LDR     R6, =IMC_TEST_SOURCE_AREA_Begin
        LDR     R7, =IMC_TEST_TARGET_AREA_Begin

IMC_TEST_ver_1_0_Label_4

        LDRB    R2, [R6, R5]
        LDRB    R3, [R7, R5]

        CMP     R3,R2
        MOVNE   R1,#0x1
        ADD     R5,R5,#0x1

        CMP     R4,R5
        BGT     IMC_TEST_ver_1_0_Label_4

        TST     R1, #0x1
        STREQ   R8, [R10]
        STRNE   R9, [R10]

IMC_TEST_ver_1_0_End

        LDMIA   SP!, {PC}

;******(END of ROUTINE)*****************************




;*******************************************************
; SUBROUTINE INFORMATION:
;       Name:                   DAC_ADC_TEST
;       Author:                 J. Granado
;       Date of creation        Dec.. 18, 1998
;       Last review (new ver)
;       Version                 1.0
;       Comments:
;
;
;*************************************************
; SUBROUTINE PARAMEMTERS: NONE
;
;*************************************************
; EXECUTION CONTEXT
;
;*************************************************
; LABELS
;
;
;*************************************************
DAC_ADC_TEST
DAC_ADC_TEST_ver_1_Begin


;       ----------------------------
;       * RESET WATCHDOG *
```

```
;       ------------------------------

        LDR     R0,=wd_CCV        ; reset wd
        MOV     R1,#0
        STR     R1,[R0]
        STMDB   SP!, {r14}         ;save return addr.

        LDR     R4,=bpp_IMR       ; all bpp pins as outputs
        MOV     R1,#0xFF
        STR     R1,[R4]
        LDR     R4,=bpp_OLR
        MOV     R1,#MSG_DAC_ADC_TEST
        STR     R1,[R4]

;       Registers used:
        ;       R0  -> data
        ;       R1  -> data
        ;       R2  -> data
        ;       R3  -> timer control
        ;       R4  -> timer current value
        ;       R5  -> timer load value
        ;       R6  -> source buffer addr
        ;       R7  -> target buffer adddr
        ;       R8  -> adc control
        ;       R9  -> adc data
        ;       R10 -> dac data
        ;       R11 -> dac control
        ;       R12 ->

        MOV     R0,#0x0
        MOV     R1,#0x0
        MOV     R2,#0x0
        LDR     R3, =timer_TC0_CR
        LDR     R4, =timer_TC0_CV
        LDR     R5, =timer_TC0_LR
        LDR     R8, =adc_CR
        LDR     R9, =adc_DR
        LDR     R10, =dac_DR
        LDR     R11, =dac_CR

;       -------------------------------
;       Enable converters
;       -------------------------------

        MOV     R0, #0x78 ; Set up timer
        STR     R0, [R5]
        MOV     R0,#0x80
        STR     R0, [R3]
        MOV     R0,#0x01  ; Enable adc
        STR     R0,[R8]
        MOV     R0,#0x01  ; Enable dac
        STR     R0,[R11]

DAC_ADC_TEST_ver_1_Label_0

        LDR     R1, [R4]
        CMP     R1,#0x8000
        BLT     DAC_ADC_TEST_ver_1_Label_0

        MOV     R3, #0x0
        MOV     R2,#DAC_ADC_TEST_Size
        LDR     R6, =DAC_ADC_TEST_SOURCE_AREA_Begin
        LDR     R7, =DAC_ADC_TEST_TARGET_AREA_Begin

DAC_ADC_TEST_ver_1_Label_1

        LDRB    R1,[R6,R3]
        STRB    R1,[R10]

        LDR     R0, =0xA                    ;wait 10 usec.(h_0x15E)
        BL      WAIT_ver_3_0

        MOV     R0,#0x5
        STR     R0,[R8]

DAC_ADC_TEST_ver_1_Label_2

        LDR     R1,[R8]
        TST     R1,#0x4
        BEQ     DAC_ADC_TEST_ver_1_Label_2

        LDRB    R0,[R9]
        STRB    R0,[R7,R3]
        ADD     R3,R3,#0x1
        CMP     R2,R3
        BGT     DAC_ADC_TEST_ver_1_Label_1
```

```
        LDR     R11,=dac_CD
        MOV     R0,#0x1
        STR     R0,[R11]
        LDR     R11,=adc_CD
        MOV     R0,#0x1
        STR     R0,[R11]

;       -------------------------
;       Checking results
;       -------------------------

        LDR     R1, =DAC_ADC_TEST_OK
        LDR     R2, =DAC_ADC_TEST_FAILED
        LDR     R6, =DAC_ADC_TEST_SOURCE_AREA_Begin
        LDR     R7, =DAC_ADC_TEST_TARGET_AREA_Begin
        LDR     R8, =DAC_ADC_TEST_FLAG_Addr

        MOV     R4,#0x0
        MOV     R3,#0x0

DAC_ADC_TEST_ver_1_Label_3

        LDRB    R10, [R6, R4]
        LDRB    R11, [R7, R4]

        CMP     R11,R10
        MOVNE   R3,#0x1

        ADD     R4,R4,#0x1
        CMP     R4,#DAC_ADC_TEST_Size
        BLT     DAC_ADC_TEST_ver_1_Label_3

        TST     R3, #0x1
        STREQ   R1, [R8]
        STRNE   R2, [R8]

DAC_ADC_TEST_ver_1_End

        LDMIA   SP!, {PC}

;******(END of ROUTINE)*****************************


;++++++++++++++++++++++++++++++++++++++
; SubROUTINE name: WAIT_ver_3_0
; Author:J. Granado
; Change Log:
;                Created: Sept. 10, 1998
;                Updating:        ---------
;                Reviewed:Oct. 26, 1998
; Version:       3.0
; Comments:      tested
;++++++++++++++++++++++++++++++++++++++
; R0 -> contains the delay value in microsec
; No return
; Max value = 10900 usec.
;++++++++++++++++++++++++++++++++++++++

WAIT_ver_3_0
        STMDB   SP!, {R0-R12,LR}

WAIT_ver_3_0_Begin

        ; Used registers:
        ;       R0  -> used as a call parameter. It contains number
        ;              of microsec to wait
        ;       R1  -> overflow value corresponding to 1msec (PCLK)
        ;       R2  -> total value to precharge counter r2=r0*r1
        ;       R4  -> transient data
        ;       R5  -> timer control register addr.
        ;       R6  -> timer load register addr
        ;       R7  -> timer current value
        ;       R8  ->[R7]

        MOV     R1,#0x6         ;0x6 = 1usec (PCLK)
        MUL     R2,R1,R0 ;

        LDR     R5,=timer_TC0_CR
        LDR     R6,=timer_TC0_LR
        LDR     R7,=timer_TC0_CV

        MOV     R4,#0x0
        STR     R4,[R5]         ; Disable and load timer
        STR     R2,[R6]

        MOV     R4,#0x80 ; enable timer with: PCLK
                         ; free running
```

```
        STR     R4,[R5]
        LDR     R0, =0xF000
WAIT_ver_3_0_label_0

        LDR     R8,[R7]           ; remains into tho loop until
        CMP     R8, R0            ; timer == 0
        BLT     WAIT_ver_3_0_label_0

        MOV     R0,#0x0           ; disable timer
        STR     R0,[R5]

WAIT_ver_3_0_End
        LDMIA           SP!, {R0-R12,PC}



;*************************************************
; SUBROUTINE INFORMATION:
;       Name:                   ROM_COF_2x8
;       Author:                 J. Granado
;       Date of creation  Mar. 29, 1999
;       Last review (4.0)  Mar. 29, 1999
;       Version                 ver 1.0
;       Comments:         this routine must be
;                               called from address 0x0
;                               in external boot.
;
;*************************************************
; SUBROUTINE PARAMEMTERS: NONE
; RETURN: 0x20 (reset handler)
;*************************************************
; EXECUTION CONTEXT:
;       - after reset and before executing reset handler
;
;*************************************************
ROM_COF_2x8
ROM_COF_2x8_Begin

        LDR     R6,=emc_ROMCF
        MOV     R0,#FFD
        STR     R0,[R6]

ROM_COF_2x8_End
;********(END of ROUTINE*****************************



;*************************************************
; SUBROUTINE INFORMATION:
;       Name:                   ROM_COF_1x16
;       Author:                 J. Granado
;       Date of creation  Mar. 29, 1999
;       Last review (4.0)  Mar. 29, 1999
;       Version                 ver 1.0
;       Comments:         this routine must be
;                               called from address 0x0
;                               in external boot.
;
;*************************************************
; SUBROUTINE PARAMEMTERS: NONE
; RETURN: 0x20 (reset handler)
;*************************************************
; EXECUTION CONTEXT:
;       - after reset and before executing reset handler
;
;*************************************************
ROM_COF_1x16
ROM_COF_1x16_Begin

        LDR     R6,=emc_ROMCF
        MOV     R0,#FFE
        STR     R0,[R6]

ROM_COF_1x16_End
;********(END of ROUTINE*****************************



        END
```