Migration Guide (XC88x to XC878) V1.0

Prepared by:
Lim Lay Mui
14 May 2008

Never stop thinking

List of Differences

- Memory Extension in XC878
- Flash Architecture
- Clock System
- Pinout
- XC878 EVR and Interrupts
- Bootstrap Load Mode (BSL)

Memory Extension in XC878

XC88x:

- All memory addresses are mapped within 64KBytes (Bank 0:0000_H–$FFFF_H$)

XC878:

- 52KBytes Flash Variants
  - ✓ Mapped within 64KBytes (Bank 0:0000_H–$FFFF_H$)
- 64KBytes Flash Variant
  - ✓ Code Space: XRAM and BootROM are mapped to Bank 2 (2'0000_H-2'$FFFF_H$)
  - ✓ Data Space: XRAM is mapped to Bank F (F'0000_H-F'$FFFF_H$)
- Memory Extension Registers (MEX1,MEX2,MEX3) are required to access the extended memory space in 64K Flash Variant
- Startup file handles all the settings of the MEX registers unless codes are executed from XRAM
- Following instructions are affected by the memory extension
  - ✓ MOVC
  - ✓ MOVX
  - ✓ LJMP
  - ✓ LCALL & ACALL
  - ✓ RET & RETI

Flash Architecture

XC88x:

- 6 P-Flash Bank, 2 D-Flash Bank
- Each P-Flash bank has one 3.75 Kbyte sectors and two 128 byte sectors
- Each F-Flash bank has two 1 Kbyte sectors, two 512 byte sectors, two 256 byte sectors and four 128 byte sectors
- Min. erase width : 1 sector
- Min. program width : 64 bytes (P-Flash), 32 bytes (D-Flash)
- Flash execution time: 3xCCLK (1 Wait state)
- Background flash programming/erasing via FSM

XC878:

- 1 P-Flash Bank , 1 D-Flash Bank
- P-Flash has 120 pages, 8 wordlines/page, 64 bytes/wordline
- D-Flash has 64 pages, 2 wordlines/page, 32 bytes/wordline
- Min. erase width : 1 page
- Min. program width : 2 bytes (P-Flash), 1 bytes (D-Flash)
- Flash execution time : 1xCCLK (Zero Wait state)
- Flash programming/erasing via CPU using software routines
- Distinguish between 1 bit corrected error and 2 bit uncorrected error

Clock System

XC88x:

- Free-running frequency during loss of lock
- PLL lock detection and oscillator (int. & ext.) run detection
- PLL frequency : 96 MHz; CPU frequency : 24 MHz (default)
- Only MultiCan has the option to run at double the CPU frequency via CMCON register
- CLKOUT could be internal(9.6 MHz) or external oscillator when COUTS="00"

XC878:

- No free-running frequency; Switch to on-chip oscillator during loss of clock
- PLL lock detection, external oscillator watchdog and PLL watchdog
- PLL frequency : 144 MHz; CPU frequency : 24 MHz (default), 26.7 MHz(max)
- MultiCan, CORDIC, MDU, CCU6 and T2CCU has the option to run at double the CPU frequency via CMCON and CR_MISC registers
- CLKOUT can only be internal oscillator(4 MHz) when COUTS="00"
- Feedback resistor is recommended for the external resonator circuitry

Differences in Pinout

XC88x:

- Pin 57 is NC (no connection).
- Port 2 has 8 input pins which is also ADC analog inputs(AN0-AN7)

XC878:

- Pin 57 is the TM (Test Mode). A pull low device is needed.
- No Port 2. Only 8 ADC analog inputs (AN0-AN7)
  - ✓ Port 2 alternate functions are mapped to P5.3-P5.5
- Bi-directional pad has programmable drive strength control

XC878 EVR and Interrupts

EVR:

- PMCON0.VDDPWARN is added to indicate the VDDP threshold level
- VDDC pre-warning feature not available

Interrupts:

- For edge detection of EXINT0/1, bit IT0/IT1 in register TCON must be set to falling edge triggered
- Extra interrupt controls added in UART1 (SCON1 register), SSC (MODIEN register) and T2 (T2CON1 register) for individual enabling/disabling of interrupts.

Bootstrap Loader Mode (BSL)

XC88x:

- BSL Mode 4 is used to erase Flash memory
- Flash (un)protection scheme via Moe 6
  - ✓ PASSWORD is 8 bit
  - ✓ Hardware reset is needed to (de)activate all types of protection
  - ✓ Not able to program and erase P-Flash with hardware protection mode 0/1
- MultiCAN BSL support ext-osc frequency of 4, 6, 8(default) and 12 MHz

XC878:

- New Mode : Alternate BSL mode (a user-defined BSL mode)
- BSL Mode 16 is used to erase Flash memory
- Flash (un)protection scheme via Mode 6
  - ✓ PASSWORD is 16 bit
  - ✓ Hardware reset is needed to (de)activate the hardware protection only
  - ✓ Able to program and erase P-Flash with hardware protection 0/1
- MultiCAN BSL support ext-osc frequency of 4, 5, 8(default) and 12 MHz

Appendix

Code example : MOVX – XC878 64 k Flash

Accessing XRAM data in bank F in data space

```
      MOV MEX3, #0x1F    ;set MXM and select bank F
      MOVX A, @DPTR

;code loaded in FLASH
      ORG  0x0000
      LJMP START_MAIN

      ORG  0x0100
START_MAIN:
      MOV  P3_DIR, #0xff
      MOV  P3_DATA, #0x01
      MOV  MEX3, #0x1F     ;Set MXM bit, select  bank
      MOV  DPTR, #0xF000   ;F in data code space
      MOVX A, @DPTR
      CJNE A, #0x55, ERR
      SETB P3_DATA.1
      INC  DPTR
      MOVX A, @DPTR
      CJNE A, #0x66, ERR
      SETB P3_DATA.2
      SJMP $
ERR:
      SETB P3_DATA.7
      SJMP $
```

```
      ORG  0xF000
DT2: DB    0xAA, 0xBB

;code loaded in XRAM
      ORG  0xF000
DT1:  DB   0x55, 0x66

;Result: P3 = 0x07
```

Code example : LJMP – XC878 64 k Flash

LJMP to XRAM code space in bank 2 from FLASH code space in bank 0 and LJMP back to FLASH code space in bank 0

```
;code loaded in FLASH
        ORG     0x0000
        LJMP    START

        ORG     0x0100
START:
        MOV     P3_DIR, #0xff
        MOV     P3_DATA, #0x01
        MOV     MEX1, #0x02   ;jump to XRAM code space
        LJMP    0xF000        ;in bank 2

        ORG     0x2000
        SETB    P3_DATA.2
        SJMP    $

;code loaded in XRAM
        ORG     0xF000
        SETB    P3_DATA.1
        MOV     MEX, #0x00    ;jump to FLASH code space
        LJMP    0x2000        ;in bank 0

;Result: P3 = 0x07
```

Code examples : CALL/RET/JMP – XC878 64 k Flash

CALL/RET

LCALL to routine in XRAM and FLASH

```
;code loaded in FLASH
      ORG    0x0000
      LJMP    START

      ORG    0x0100
START:
      MOV    P3_DIR, #0xff
      MOV    P3_DATA, #0x01
      MOV    MEX1, #0x02     ;call to routine in XRAM
      LCALL  0xF000          ;code space in bank 2
      SETB   P3_DATA.2
      MOV    MEX1, #0x00     ;call to routine in Flash
      LCALL  0x0200          ;code space in bank 0
      SETB   P3_DATA.4
      SJMP   $

      ORG    0x0200
      SETB   P3_DATA.3
      RET

;code loaded in XRAM
      ORG    0xF000
      SETB   P3_DATA.1
      RET

;Result: P3 = 0x1F
```

JMP/CALL/RET

LCALL to routine in FLASH code space in bank 0 from XRAM code space in bank 2

```
;code loaded in FLASH
      ORG    0x0000
      LJMP    START

      ORG    0x0100
START:
      MOV    P3_DIR, #0xff
      MOV    P3_DATA, #0x01
      MOV    MEX1, #0x02     ;jump to XRAM code space
      LJMP   0xF000          ;in bank 2

      ORG    0x2000
      SETB   P3_DATA.2
      RET

;code loaded in XRAM
      ORG    0xF000
      SETB   P3_DATA.1
      MOV    MEX1, #0x00     ;call to routine in flash
      LCALL  0x0200          ;code space in bank 0
      SETB   P3_DATA.3
      SJMP   $

;Result: P3 = 0x0F
```

Code example : Flash Program/Erase using user routines – XC878 64 k Flash

Program 10 bytes of data in DFLASH location 0xF000 to 0xF009 and verify that the data is programmed correctly. After that, erase the programmed data and verify that the DFLASH data is erased accordingly.

```
FLASH_PROGRAM       EQU     0xDFF6
FLASH_ERASE         EQU     0xDFF9
FLASH_PADDR         EQU     0xF000
START_XRAM          EQU     0xF112
COMPARE_LEN         EQU     10
COMPARE_DATA        DATA    0x7f

        ORG   0x0000
        LJMP  START_MAIN

        ORG   0x0100
START_MAIN:
        MOV   P3_DIR, #0xFF
        MOV   P3_DATA, #0x01
        ;Write data to be programmed in XRAM
        MOV   COMPARE_DATA, #0x55
        ACALL  WRITE_XRAM

;Program data
        MOV   DPTR, #FLASH_PADDR ;Start address for
        ORL   EO, #0x01          ;programming
        MOV   DPTR, #START_XRAM  ;Start of XRAM
        ANL   EO, #0xFE          ;containing flash data
        MOV   R1, #10
        MOV  MEX1, #0x02         ;User routine in bank 2
        LCALL FLASH_PROGRAM      ;Call Flash Program
                                 ;User routine
        MOV  MEX1, #0x00         ;Reset MEX1
        JC    ERR
        SETB  P3_DATA.1          ;Indicate program done

        ;Verify programmed data
        LCALL  VERIFY_FLASH_DATA
        SETB  P3_DATA.2          ;Indicate program ok

        ;Erase programmed data
        MOV   DPTR, #FLASH_PADDR
        CLR   A
        MOV  MEX1, #0x02         ;User routine in bank 2
        LCALLFLASH_ERASE         ;Call Flash Erase
                                 ;User routine
        MOV  MEX1, #0x00         ;Reset MEX1
        JC    ERR2
        SETB  P3_DATA.3          ;Indicate Erase done

;Verify data erased
        MOV   COMPARE_DATA, #0xFF  ;Erase data 0xFF
        LCALL  VERIFY_FLASH_DATA
```

Code example : Flash Program/Erase using user routines (Continue…)

```
      SETB  P3_DATA.4          ;Indicate Erase ok
      SJMP  $

ERR:
      SETB  P3_DATA.7          ;Indicate programmed
      SJMP  $                  ;data incorrect

ERR2:
      SETB  P3_DATA.5          ;Indicate erased
      SJMP  $                  ;data incorrect



;Routine to verify data content
VERIFY_FLASH_DATA:
      MOV   R0, #COMPARE_LEN
      MOV   DPTR, #FLASH_PADDR
CHECK_NEXT:
      CLR   A
      MOVC  A, @A+DPTR
      CJNE  A, COMPARE_DATA, ERR1
      INC   DPTR
      DJNZ  R0, CHECK_NEXT
      RET
ERR1:
      SETB  P3_DATA.6
      SJMP  $
```

```
;Routine to write data to be programmed in flash
;to XRAM
WRITE_XRAM:
      MOV   MEX3, #0x1F
      MOV   R0, #COMPARE_LEN
      MOV   DPTR, #START_XRAM
      MOV   A, COMPARE_DATA
WRITE_NEXT:
      MOVX  @DPTR, A
      INC   DPTR
      DJNZ  R0, WRITE_NEXT
      RET


;Result: P3 = 0x1F
```

Code example : Changing PLL parameters

XC88x:

```
      MOV   SCU_PAGE, #1
      ORL   PLL_CON, #0x08        ;VCOBYP = 1
      ANL   PLL_CON, #0xFB        ;OSCDISC = 0
      MOV   PASSWD, #0x98         ;Disable bit protect
                                  ;for 32 CCLKs

;change PLL parameters
      ANL   PLL_CON, #0x0F
      ORL   PLL_CON, #NDIV

;detect PLL lock
      ORL   PLL_CON, #0x02        ;RESLD = 1
CHECK_LOCK:
      MOV   A, PLL_CON
      JNB   ACC.0, CHECK

;reconnect to PLL
      ANL   PLL_CON, #0xF7        ;VCOBYP = 0
      MOV   SCU_PAGE, #0
```

XC878:

```
      MOV   SCU_PAGE, #1;
      MOV    PASSWD, #0xC0        ;Disable bit protect
      ORL   OSC_CON, #0x40        ;PLLBYP = 1
      ORL   OSC_CON, #0x20        ;PLLPD = 1

;change PLL parameters
      MOV   PLL_CON, #NDIV_XC878_PLL_CON
      MOV   PLL_CON1, #NDIV_XC878_PLL_CON1
      MOV   A, #PDIV_XC878_PLLCON1
      ORL    PLL_CON1, A
      ANL    OSC_CON,#0xDF        ;PLLPD = 0

;detect PLL lock
      ORL    OSC_CON, #0x80       ;PLLRDRES = 1
CHECK_RUN:
      MOV   A, PLL_CON
      JNB   ACC.1, CHECK_RUN      ;Check PLLR
CHECK_LOCK:
      MOV   A, PLL_CON
      JNB   ACC.0, CHECK_LOCK     ;Check PLL_LOCK
      ANL   OSC_CON, #0xBF        ;PLLBYP = 0
      MOV    PASSWD, #0xC3        ;Enable bit protect
      MOV   SCU_PAGE, #0
```

Code example : Select EXT-OSC as PLL input source

XC88x:

```
;switching to external xtal
      MOV    SCU_PAGE, #1
      ORL    PLL_CON, #0x08     ;VCOBYP = 1
      ORL    PLL_CON, #0x04     ;OSCDISC = 1
      ORL    OSC_CON, #0x04     ;OSCSS = 1
      ANL    OSC_CON, #0xF7      ;XPD = 0

;wait for XTAL to stabilise
      MOV    R0, #0xFF
DECR0:
      MOV    R1, #0x20
      DJNZ   R1, $
      DJNZ   R0, DECR0

;detection of oscillation
OSCRNOTSET:
      ORL      OSC_CON, #0x02    ;ORDRES = 1
WAIT_ORDRES_0:
      MOV    R0, #0xFF
DECR1:
      MOV    R1, #0x20
      DJNZ   R1, $
      DJNZ   R0, DECR1
      MOV    A, OSC_CON
      ANL    A, #0x02
      JNZ    WAIT_ORDRES_0
      MOV    A, OSC_CON
      JNB    ACC.0, OSCRNOTSET

;VCOBYP to change ndiv
      ORL    PLL_CON, #0x08     ;VCOBYP = 1
      ANL    PLL_CON, #0xFB     ;OSCDISC = 0
      MOV    PASSWD, #0x98      ;access protected bit
      ANL    PLL_CON, #0x0F
      ORL    PLL_CON, #0xA0      ;8MHz crystal

;detect PLL lock
      ORL    PLL_CON, #0x02

WAIT_LOCK:
      MOV    A, PLL_CON
      JNB    ACC.0, WAIT_LOCK

;reconnect to PLL
      ANL    PLL_CON, #0xF7      ;VCOBYP = 0
      MOV    SCU_PAGE, #0
```

Code example : Select EXT-OSC as PLL input source

XC878:

```
;switching to external xtal
      MOV    SCU_PAGE, #1
      MOV    PASSWD, #0xC0       ;Disable bit protect
      ORL    OSC_CON, #0x40      ;PLLBYP = 1
      ORL    OSC_CON, #0x20      ;PLLPD = 1
      ANL    OSC_CON, #0xF7      ;XPD = 0

;wait for XTAL to stabilise
      MOV    R0, #30
DECR0:
      MOV    R1, #100
      DJNZ   R1, $
      DJNZ   R0, DECR0

;detection of oscillation
      ORL    OSC_CON, #0x02      ;EORDRES = 1
      MOV    R0, #65
      DJNZ   R0, $
OSCRNOTSET:
      MOV    A, OSC_CON
      JNB    ACC.0, OSCRNOTSET
      ORL    OSC_CON, #0x04      ;OSCSS = 1
```

```
;VCOBYP to change ndiv
      MOV    PLL_CON,
      MOV    PLL_CON1,
      ANL    OSC_CON, #0xDF      ;PLLPD = 0

;detect PLL run
      ORL    OSC_CON, #0x80      ;PLLRDRES = 1
CHECK_RUN:
      MOV    A, PLL_CON
      JNB    ACC.1, CHECK_RUN

CHECK_LOCK:
       MOV   A, PLL_CON
       JNB   ACC.0, CHECK_LOCK

      ANL    PLL_CON, #0xBF      ;VCOBYP = 0
      MOV    PASSWD, #0xC3       ;Enable bit protect
      MOV    SCU_PAGE, #0
```