R1610/R1620/R2010/R2020C
Programming Draft

Version 0.13
February 24, 2003

Content Index:

1. How to initialize RDC MAC

This chapter, we describe some necessary operation.

1-1 How to reset MAC controller

Software reset is a necessary action to make sure your chip works on the default state. Software reset sequence is 1). Set MSCTL register bit0. 2). Wait MSCTL bit0 clear. 3) Reset complete.

<Example>

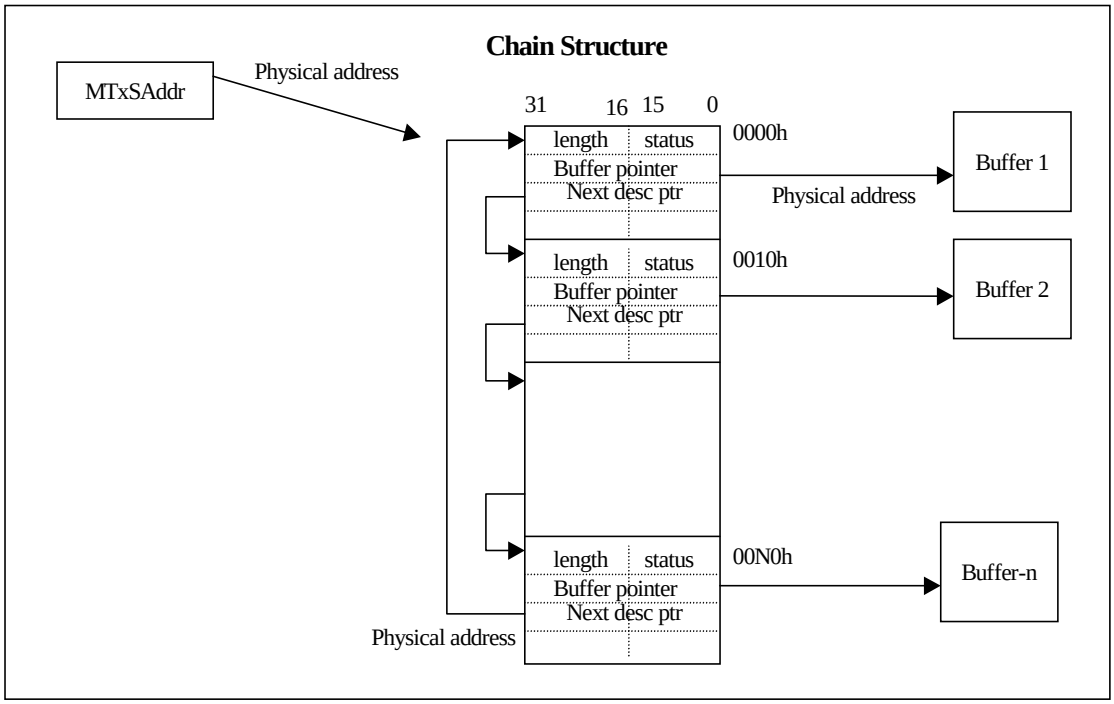
```
; Issue RESET command
      mov       ax, 0001h
      mov       dx, MSCTL
      out       dx, ax

; Wait RESET complete
reset_wait:
      in        ax, dx
      test      ax, 0001h
      jnz       reset_wait

; RESET complete
```

1-2 How to initialize TX descriptor chain

For data transmission, driver needs to prepare a descriptor chain structure like as the following figure. Buffer is not necessary in the initialization step. If you want to hook your sent buffer directly, you don't need to allocate buffer and initialize <buffer pointer>. <status> OWN_BIT(bit 15) must be zero because you have no data to send so far. About descriptor format and MTxSAddr register, please see the data sheet.

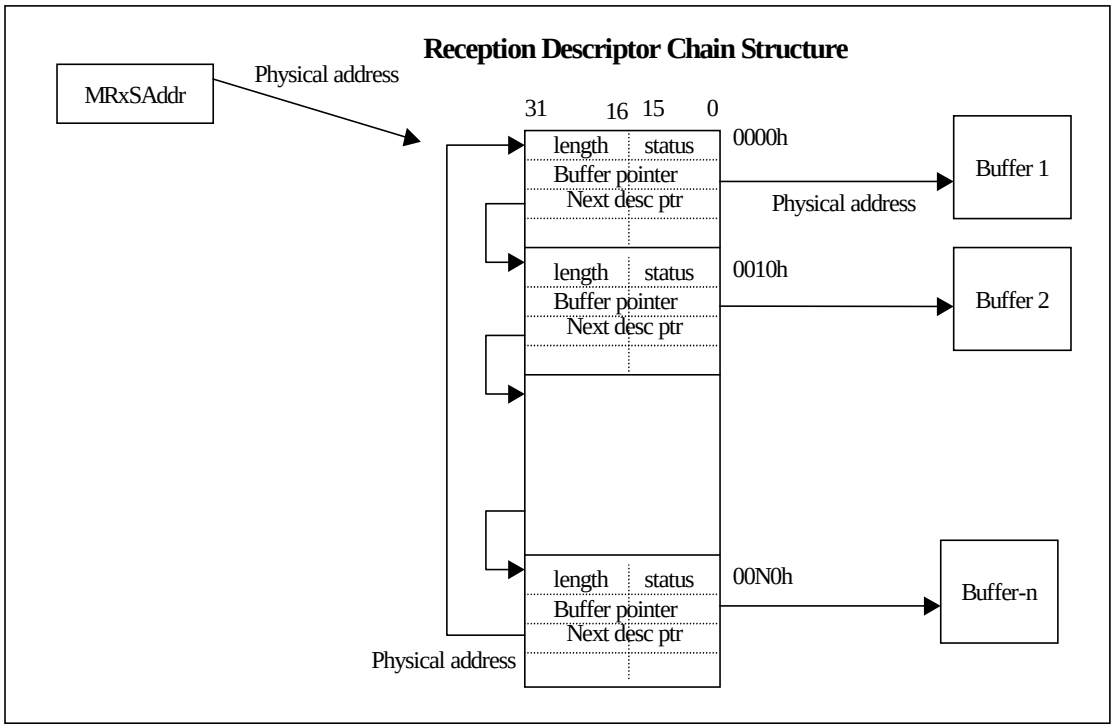


Descriptor must be double word aligned. TX buffer can be at any byte address. MTxSAddr, Buffer_pointer, Next_desc_ptr must be the physical address.

Note: When you are using R2010/R2020's chip that can be addressing to 24 bits mode.
Please be sure your physical address setting correctly. R16xx's physical address is segment base left shift 4 bits add offset value. R20xx is segment base left shift 8 bits add offset value.

1-3 How to initialize RX descriptor chain

For data reception, driver needs to prepare a descriptor chain structure like as the following figure. Buffer is necessary. But you can dynamically allocate. Or like as figure, statically allocate buffer. <status> OWN_BIT(bit 15) can be set if you allocate buffer. You must specify the buffer size into register MRBS. About descriptor format, MRxSAddr and MRBS register, please see the data sheet.



RX Descriptor and buffer must be double word aligned.
MRxSAddr, Buffer_pointer, Next_desc_ptr must be the physical address.

Note: When you are using R2010/R2020's chip that can be addressing to 24 bits mode.
Please be sure your physical address setting correctly. R16xx's physical address is segment base left shift 4 bits add offset value. R20xx is segment base left shift 8 bits add offset value.

1-4 How to initialize MAC register

There are some registers must be initialized before driver activates MAC.

1) Register MTxSHAddr:MTxSLAddr must point to the start address of the transmission descriptor chain.
2) Register MRxSHAddr:MRxSLAddr must point to the start address of the reception descriptor chain.
3) Decide register MCTL <full duplex bit>. It is depended on PHY's status. If PHY is on full duplex mode, this bit must set. Otherwise, this bit must clear.
4) If your MAC address and multi-cast address are less than or equal 4 groups, you can directly use register MIDxx. If over 4 groups, you must set hash table by register MHMARxx and set MCTL bit to enable hash function.
5) If you want to use the interrupt driven to transfer packets, please register your interrupt handler before you turn on register MintMsk. If you use the polling method for transmission and reception, you need to turn off MintMsk.
6) Decide MAC operation mode by register MCTL. Please see the data sheet and set the function bit that you want to use.
7) Set MCTL bit0 RCV_EN and bit12 XMT_EN to activate MAC transmission and reception.

<Example>
Map MAC address 00-01-02-03-04-05 to MAC0 MID0

```
        mov     dx, 0FD68h      ; MID0 low byte offset
        mov     ax, 0100h
        out     dx, ax
        mov     dx, 0FD6Ah      ; MID0 middle byte offset
        mov     ax, 0302h
        out     dx, ax
        mov     dx, 0FD6Ch      ; MID0 high byte offset
        mov     ax, 0504h
        out     dx, ax
```

2. Transmit packets

Transmit a packet include two part. One is prepare data and send. Another is the sent status check.

2-1 How to transmit packet

1) Get the first free transmission descriptor
2) If there is a buffer hook on this descriptor, you can copy the send data into this buffer. If not, you can directly hook your send data buffer into descriptor <DaPtH:DaPtL>. But you must be care, <DaPth:DaPtL> must be the physical address.
3) Set the send data length into descriptor <Length> field. **If your send data length is less than 60 byte, please put 60 into this field because MAC doesn't do the padding action.**
4) If want to disable CRC padding, please let <Status> bit13 DISCRC set. If DISCRC is 0, MAC will automatically append CRC at the packet end.
5) Set <Status> OWN_BIT. OWN_BIT is set, it means this descriptor belongs to MAC. OWN_BIT is clear, it means this descriptor belongs to host.
6) Issue transmission-polling command. Out register XMTPC to signal MAC to send. Please note MCTL bit 12 must be set, otherwise MAC do not send.

2-2 how knows the transmitted status

Driver how to know MAC sent status? MAC will clear descriptor <status> OWN_BIT after transmit complete and fill the sent status into <status> field. After complete above action, MAC will fetch the next descriptor and check <status> OWN_BIT. If OWN_BIT is set, MAC will send it. If OWN_BIT is clear, MAC will be idle state to wait next polling command.

So you can use the polling method to the sent descriptor <status> OWN_BIT. If this bit is 0, it means this data was sent completely. Then driver can check <status> to know the sent status.

If you want to use the interrupt driven method to check send status, you can register an interrupt handler and turn on register MintMsk <XPF_EN>. MAC will generate an interrupt and set register MintSt <XPKT_FINISH> after sent completely.

3. Receive packets

MAC will clear the reception descriptor <status> OWN_BIT after received completely. Now the received data is in the address that the descriptor <DaPtL> point. The received data length is in the descriptor <Length> field. The received data status is in the descriptor <Status> field. MAC will fetch the next descriptor after complete this reception. If next descriptor <status> OWN_BIT is 1, MAC will receive the next packet. If next descriptor <status> OWN_BIT is 0, MAC will enter the waiting state. Next packet coming will trigger MAC to check descriptor available again.

3-1 How to receive packet

The simple method is the polling method. Driver keep to check the reception descriptor <status> OWN_BIT, until OWN_BIT is 0. It means a packet received already.
Another method, driver can register an interrupt and turn on register MintMsk <RPF_EN>. MAC will generate an interrupt request and set MintSt <RPKT_FINISH> after receive a packet.

3-2 How to re-use reception descriptor

Driver must take care of the data of received descriptor. Driver can copy the data from the received buffer to driver private buffer. Then set descriptor <status> OWN_BIT to 1 to let this descriptor can receive data again.

Another method, driver can directly take off this buffer from the descriptor. Then allocated a new buffer and hook this new buffer into the descriptor. Finally, set descriptor <status> OWN_BIT to 1.

4. How to access PHY register

R1610/R1620/R2010/R2020C MAC supports the useful registers to read or write PHY registers. User only needs to specify which PHY registers those user wants to read or write and issues a read or write command.
R1610/R1620/R2010/R2020C MAC will do the all detail steps.
MMDIO register offset 20h, MII command register
MMIIRDATA register, offset 24h, data read from PHY
MMIIWDATA register, offset 28h, data write to PHY
About above register description, please refer R1610/R1620/R2010/R2020C data sheet.

4-1 How to read PHY register

1). Put the PHY register address that you want to read into MMDIO bit4:0, Put PHY address into MMDIO bit12:8. Set MII read command on MMDIO bit13.
2). Wait read complete after MMDIO bit13 is 0.
3). Now PHY data is on MMIIRDATA register.

<Example>
Assume PHY address is 01h, read PHY register 02h

```
; Set PHY address, register address and issue read command
      mov     ax, 2102h
      mov     dx, MMDIO
      out     dx, ax

; Wait read command complete
read_wait:
      in      ax, dx
      test    ax, 2000h
      jnz     read_wait

; PHY register data on MMIIRDATA
      mov     dx, MMIIRDATA
      in      ax, dx
```

4-2 How to write PHY register

1). Put the written data into MMIIWDATA.
2). Put the PHY register address that you want to write into MMDIO bit4:0, Put PHY address into MMDIO bit12:8. Set MII write command on MMDIO bit14.

3). Wait the write command complete after MMDIO bit14 is 0.

<Example>

Assume PHY address is 01h, write 0061h to PHY register 04h

```
; Put the written data to MMIIWDATA
        mov     ax, 0061h
        mov     dx, MMIIWDATA
        out     dx, ax
; Set PHY address, register address and issue write command
        mov     ax, 4104h
        mov     dx, MMDIO
        out     dx, ax

; Wait write command complete
write_wait:
        in      ax, dx
        test    ax, 4000h
        jnz     write_wait
```

5. How to use hash function for address filter

R1610/R1620/R2010/R2020C has 4 address set. MID 0,1,2,3, every one can be a multi-cast address or uni-cast address. If your address is over this, you can use hash function for multi-cast address and uni-cast address.
R1610/R1620/R2010/R2020C MAC supports 4 hash table registers and each register is 16 bits. So the hash table is 64-bits width. It needs 6 bit to index this hash table.

5-1 How to enable hash function

Hash function can be for multi-cast and uni-cast address. You just need to set MCTL FC_EN bit to enable the function. This function is disabled after reset. About hash functin for uni-cast address, please see 5-4.

<For example>

```
mov     dx, MCTL
in      ax, dx
or      ax, 0100h
out     dx, ax
```

Note. If you want to set hash bit in hash table. You MUST enable hash function first.

5-2 How to calculate the hash table index from your multi-cast address

You can use the following program to get a 6-bit hash index from a multi-cast address. Then use this hash index to map hash table register.

Hash index algorithm:

Buffer: 6 byte multi-cast address

```
{
   unsigned long int  Crc, Carry;
   unsigned int       i, j, Hash_Index;
   unsigned char      CurByte;

   Crc = 0xffffffffUL;

   for (i = 0; i < 6; i++) {
      CurByte = Buffer[i];

      for (j = 0; j < 8; j++) {
         Carry = ((Crc & 0x80000000UL) ? 1 : 0) ^ (CurByte & 0x01);
         Crc <<= 1;
         CurByte >>= 1;

         if (Carry) {
            Crc = (Crc ^ 0x04c11db6UL) | Carry;
         }
      }
   }
```

```
   Hash_Index = (unsigned int) ( (Crc>>26) & 0x3F
}
```

5-3 How to map Hash register

Hash index 00 maps to Hash Table word1 bit 00
Hash index 15 maps to Hash Table word1 bit 15
Hash index 16 maps to Hash Table word2 bit 00
Hash index 31 maps to Hash Table word2 bit 15
Hash index 32 maps to Hash Table word3 bit 00
Hash index 47 maps to Hash Table word3 bit 15
Hash index 48 maps to Hash Table word4 bit 00
Hash index 63 maps to Hash Table word4 bit 15

5-4 How to filter uni-cast packet by hash table

If you want to filter uni-cast packet by hash table, you can set MAC control register 1's bit 15(AUCP) on. And when this bit turning on please check you have set the related bit in hash table. Otherwise the uni-cast packet will not coming.

5-5 How to write Rx packet's hash index to Rx descriptor

If you want to get the receive packet's hash value, you can set MAC control register 1's bit 14(WIDX) on. It will automatic write the hash index to Rx descriptor HIDX[5:0] field.

6. Flow control function

R1610/R1620/R2010/R2020C MAC supports 802.3X flow-control function for full duplex mode.

6- 1 How to enable flow control function

Enable flow control function, you just need to set MCTL FC_EN bit. And this function is only work on full duplex mode. If you turn on this bit on half duplex mode, it does not work.

<For example>

```
mov     dx, MCTL
in      ax, dx
or      ax, 0200h
out     dx, ax
```

6-2 MAC how to handle the received pause frame

When MAC receive a pause frame with time not zero, MAC will temporary stop to send after current transmitting completely. Until timeout or receive a pause frame with time 00, MAC will keep to send.

6-3 MAC how to send pause frame and how to stop

MRDCR(1Ah) bit7..0 is RXDESCPAN, RX descriptor available count for flow-control. Bit15..8 is RXPT, RX descriptor threshold. Please see data sheet for details.

MAC will automatically send the pause frame when RXDESCPAN is less or equal RXPT. MAC automatically decreases RXDESCPAN after received a packet. If RXDESCPAN equals to RXPT, MAC will set MSCTL TPF bit then send a pause frame with time FFFFh. Please note transmission must be enabled (MCTL XMT_EN is set), otherwise MAC can't send the pause frame. MAC will keep sending the pause frame after timeout. Driver need to signal MAC to increase RX descriptor count after RX descriptor is available. MAC will stop to send PF and clear TPF flag when RXDESCPAN is larger than RXPT.

MRDCR register is a special register. You can specify RXDESPAN and RXPT into this register when RX_EN is 0. If RX_EN is 1 (RX machine activates), MAC will automatically increase 1 to RXDESCPAN when driver does an I/O write to this register. Any time, driver does an I/O read to this register to get the current value.

<For example>

```
; One RX descriptor is available again & Signal MAC to increase RXDESCPAN
mov     RX_descripotr_OWN, 1
mov     dx, MRDCR
out     dx, ax
```

7. How to reduce interrupt amount

7-1 Host interrupt

R1610/R1620/R2010/R2020C MAC 1 and 2 share the interrupt 4. You can check the interrupt status register (FF30h) to identify this interrupt caused by MAC1 or MAC2. If bit4 is set, it means MAC1 has an interrupt request. If bit5 is set, it means MAC2 has an interrupt request. Those two bits can set both when MAC1 and 2 have the interrupt request at the same time.

7-2 MAC interrupt

When packet transmit and receive too fast, it will generate many interrupts. If interrupts are too many, OS has no time to service other process because OS always service interrupts. When this condition happens, R1610/R1620/R2010/R2020C MAC has a good method to reduce transmit and receive interrupt amount.

There are two interrupt control registers. One is for transmission and another is for reception.

How to use the interrupt control register to reduce the interrupt amount. There are two fields ,INTC and TIMER, in this control register.

If INTC is 0, it means disable interrupt control function. MAC will generate interrupt after every transmission or reception.

If INTC is not zero, it means MAC will generate an interrupt after transmit or receive packets reach INTC.

For example, TXINTC is 6. MAC will generate an interrupt after transmit 6 packets. In this case, we reduce 5 interrupts. OS get more time to service other process. Do not waste too much time on content switch.

TIMER is another way to generate interrupt. In upper case, if MAC only send 3 packets, there is no data to send a long time. MAC will generate an interrupt after TIMER is timeout. TIMER is based on TX clock or RX clock. Driver can specify the timeout time by program TIMER field.

Timer activated by packet transmission or reception. So if no packet transmits or receives, the timer doesn't work and doesn't generate interrupt.

8. How to enable cache function

R1610/R1620/R2010/R2020C serial support 8K bytes uniform cache. That can enhance your system's performance. It supports I-cache and D-cache. It also supports 4 non-cache region and 1 write-invalid region.
If enables I-cache mode, the code will be stored into the cache.
If enable D-cache mode, the data will be stored into the cache.
The non-cache region is for any memory location which CPU's peripheral will modify it. Like MAC's descriptor and buffer location. Set non-cache region will prevent cache and memory data not coherence.
The write invalid region is for Flash Rom area use. When data in this region will not caching, but instruction will caching. It will guarantee when user issue write command directly writing to Flash Rom.
And the non-cache/write-invalid region can "NOT" runtime change. So you need to set non-cache region before cache function enable.

8-1 How to set the non-cache region

Every non-cache region has 4 registers to define it. It was Non-Cache region Start Address Low, Start Address High, End Address Low and End Address High. You need to calculate the physical address for those registers.
<Example>
Set 81007~8FFFF(physical address) for Non-cache region 0.

```
Mov  dx, 0FEC2h ;(NCR0 start low)
Mov  ax, 1007h
And  ax, 0FFF8h ;(bit 0~2 is reserved)
Out  dx, ax
Mov  dx, 0FEC4h ;(NCR0 start high)
Mov  ax, 0008h
Out  dx, ax
Mov  dx, 0FEC6h ;(NCR0 end low)
Mov  ax, 0FFFFh
And  ax, 0FFF8h ;(bit 0~2 is reserved)
Out  dx, ax
Mov  dx, 0FEC8h ;(NCR0 end high)
Mov  ax, 0008h
```

```
Out   dx, ax
```

8-2 How to set write invalid region

The write invalid region is the same as non-cache region. It also needs 4 register to define it.

<Example>

Assume flash memory is locate at 80000~FFFFF(physical address).

```
Mov  dx, 0FEE2h ;Write invalid region start low
Mov  ax, 0000h
Out   dx, ax
Mov  dx, 0FEE4h ;Write invalid region start high
Mov  ax, 0008h
Mov  dx, 0FEE6h ;Write invalid region end low
Mov  ax, 0FFFFh
And  ax, 0FFF8h ;(bit 0~2 is reserved)
Out   dx, ax
Mov  dx, 0FEE8h ;Write invalid region end high
Mov  ax, 000Fh
Out   dx, ax
```

8-3 How to enable cache function

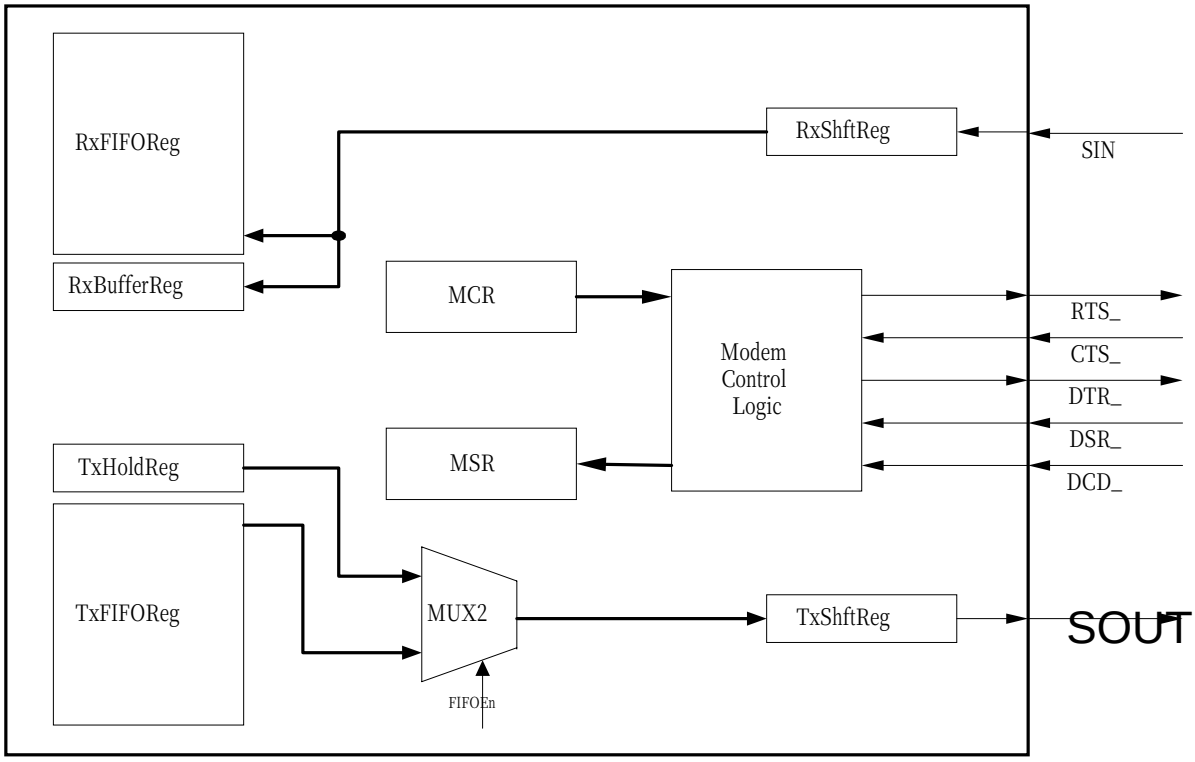
The register FEC0 is cache control register. In this register you can set I-cache enable, D-cache enable and NCR0~3 enable. And this register will be clear after system reset.

<Example>

Assume non-cache region 0, 1 and 2 are set already. And I want to enable I-cache and D-cache. So we need to set cache control register's ICE, DCE, NCR0, NCR1, NCR2 on.

```
Mov  dx, 0FEC0h
Mov  ax, 0C700h
Out   dx, ax
```

9. Use UART hardware flow control function



UART function block diagram

Now R1610/R1620/R2010/R2020C has support UART hardware flow control function.
When you enable auto flow control function:
RTS_ signal will de-asserted when receiver FIFO reached a trigger level of 1, 4, or 8. If the receiver FIFO level set to 14, the RTS_ will de-assert after the 16th character's first bit coming.
If CTS_ (input signal) was set to high, the UART will NOT send character to the others. Until the other side give to CTS low signal.

Note. If you want to use R2010C UART function, please see Programming Notes (10-3) point 5.

9-1 How to enable UART flow control function

It can be setting to Auto-RTS and Auto-CTS enable, Auto-CTS enable only, or disable flow control. The bits define at UART Modem Control Register's (FF88h, FF18h) ACE(bit5) and MCR(bit1). If both of ACE and MCR bit set to 1, that

means Auto-RTS/CTS enabled. If ACE is 1, MCR is 0. It will enable Auto-CTS only. If ACE is 0, the Auto-RTS/CTS will disable.

10. Application Notes

10-1 Difference between R1610/R1620C and R1620B

1. Register FFF2h bit15-11, 0 specifies individual PCSx bus width. Bit7 USIZ is read only now.
2. Register FFAAh specify PCS t1 wait clock. (R1620B t1 wait clock is in FFA4h bit6-4)
3. Register FF34h, 36h, 38h, 3Ah, 3Ch and 3Eh (INT5, 6, 0, 1 and 2) support edge selects.
4. Register FFA8h: PCS_n and MCS_n Auxiliary register supports
5. Register FFA6h: MCS_n Midrange Memory Chip Select Register
6. Register FF88h/18h MCR bit5 ACE, Auto flow control function.
7. Register FD04h/FE04h MCR1 bit15 AUCP, bit14 WIDX
8. Register FFF2h bit7 USIZ is read only now.
9. Register FEF0h not present.
10. Register FEC0h~FEC8h for cache function
11. Register FFF4h Processor Release Level Register is 12D9h
12. Register FFEAh bit6-3 support the external independent address and control signal.
13. Register FFACh specify MCS wait clcok.
14. Register FFE2h Clock Prescalar Register default is 0080h
15. Register FFE4h Enable RCU Register default is 8000h (RCU enabled).

10-2 Difference between R2020C and R1620B

R2020C difference with R1620B included the items in 10-1.

1. Register FFAAh to specify LCS/UCS/PCS chip select size multipliers and PCS wait clock of t1. (R1620B t1 wait clock is in FFA4h bit6-4)
2. Register FFA4h Peripheral Chip Select Register, bit7-4 BA[23:20]
3. DMA parts

   Register FFC6h DMA0 Destination Address High Reg, bit 7-0, DDA[23:16]
   Register FFC2h DMA0 Source Address High Register, bit 7-0, DSA[23:16]
   Register FFD6h DMA1 Destination Address High Reg, bit 7-0, DDA[23:16]
   Register FFD2h DMA1 Source Address High Register, bit 7-0, DSA[23:16]

3. MAC parts

   RX descriptor: descriptor pointer is expanded to 24 bits

Buffer pointer is expanded to 24 bits
TX descriptor: descriptor pointer is expanded to 24 bits
Buffer pointer is expanded to 24 bits
FD30h/FE30h: TX descriptor start address1 register, bit 7-0, TDSA[23:16]
FD38h/FE38h: RX descriptor start address1 register, bit 7-0, RDSA[23:16]

10-3 Difference between R2020C and R2020A

1. Register FFE2h Clock Prescalar Register default is 0080h
2. Register FFE4h Enable RCU Register default is 8000h (RCU enabled).
3. UART with FIFO and HW flow control (FF88h bit5 AFE)
4. PCSx Wait state supports up to 255 clocks
5. R2020C supports SLA18-0. R2010C supports SLA11-0.

10-3 Programming Notes

1. Because this version have cache memory inside. So every reset system need 256 clock to reset cache data.
2. Now PCS's wait state is extend to 255(Maximum value). There have some different with R1620-B. So please be sure your wait setting correctly.
3. After cache enabled, program can't disabled it. When cache hit, data will come from cache. If you want to read the data from SDRAM, but the data is already in cache memory. You must let cache entry's (Which was mapping to the memory address) V-bit equal 0. Then the data will get from SDRAM. Or you can use DT program's cache test mode function. You will see the data that coming from the SDRAM.
4. If you want to use MCS. Don't forget both of FFA6 (Midrange Chip Select Register) and FFA8 (PCS_n and MCS_n auxiliary Register) need to initial it. If not, the MCS will not working.
5. Because R2010C default will set slow bus to 16 bit mode. And SAD8~15 will share with UART0. So if you want to use UART0, you need to disable slow bus first. (Port 0xFFEA out 0x8000)
6. How to invalid cache content?
   6-1 Port 0xFEC0, out 0x0000 Disable cache function first
   6-1 Port 0xFEEA, out 0xA000 invalid cache content start
   6-2 Port 0xFEEA, out 0x0000 invalid cache content end
   At this time all of cache content will disappear.
   After the above procedure, you can specify what kind of cache function you need.

Index

Errata:

V0.10 10/14	Start up. The document modification is base on R1620-B programming draft.
V0.11 11/15	Add R1620C difference item with R1620B
V0.12 01/03	1.Add Enable Hash table sequence 2.Add R2010C programming notes
V0.13 02/24/03	Add programming notes6