COP8SBR9/SCR9/SDR9 User Information Sheet Rev. E

This user information sheet is intended to aid users of the COP8SBR9/SCR9/SDR9 series of products by documenting known problems and limitations of the C revision of this device and of product documentation for the device.

Number: 1

Name: Maximum Current Through The V_{CC} or Ground Pin

Description: The device specification states that the absolute maximum total current through the V_{CC} or ground pin of the device must be less than 200mA. This specification correctly applies to devices in the 68 pin package which has two V_{CC} and two ground connections. Devices in packages which have only one V_{CC} or one ground connection must be limited to 100mA. Exceeding this limit may affect long term reliability of the device.

Workaround: The user should be careful to ensure that the total of I/O source or sink current, plus supply current, does not exceed 100mA/V_{CC} or Ground pin.

Plan: The design of the device will be changed in the next revision to allow the specified current.

Number: 2

Name: USART Lockup after Framing Error

Description: The USART receive state machine can hang after receiving a frame error or detecting a Line Break condition.

Workaround: The lockup can be cleared by setting the USART in internal loopback mode and switching to synchronous operation for a short time. For faster baud rates, there is no need for a timing loop. The code that is included works for 300 baud with the device running on a 10MHz crystal. (2MHz instruction clock, $2 \times 10^6 / (16 \times 300) = 417$ instruction cycles for 1/16 bit time delay). At baud rates >= 9600 baud (>= 4800 baud with a 5MHz xtal) no timing loop is required.

In the following example the UART was initialized as follows:

```
PSR   =     C9
BAUD  =     40
ENU   =     01              ; 8 bits, no parity
;Remember that at the end of the code, the PSR and ENU must be restored to
;whatever setting the application requires.
CLERR:
      LD    ENU, #018       ; Temporarily go into loop back mode
      SBIT  SSEL,ENUI       ; Then into synchronous mode
```

```
        LD      0F1,#69         ;Delay loop timing - (6*69-2)/2 = 206uS
LOOP3:                          ;Bit time/16=208.33uS
        DRSZ    0F1             ;Following instructions make up the difference
        JP      LOOP3
        LD      A, ENUR         ;Clear the error flags
        LD      A, RBUF         ;Clear RBFL
        RBIT    SSEL,ENUI       ;Back to asynch mode
        LD      PSR,#000        ;Stop USART clock before restarting
        LD      ENU, #000       ; Replace with normal ENU value
        LD      PSR,#0C9        ;Restore the PSR to restart the USART
        RET
```

The same subroutine works for a break detect, but you need to be careful. If the break is still in force when the PSR is restored, another break detect will occur. The part could easily end up spending all of it's cycles just handling break detects. The user may wish to delay handling break detects until RDX goes high (poll the pin periodically, or use MIWU).

Plan: This will be corrected in the next revision of the device.

Number: 3

Name: Interrupts after the Key is set

Description: Interrupts may occur between setting the Key and executing the JSRB instruction to initiate Flash memory reads or writes. If an interrupt occurs at this time, the Key will expire before the completion of the interrupt service. The JSRB execution will be invalid and the jump will be to the same location in the Flash.

Workaround: It is recommended that the user globally disable interrupts before setting the key. Interrupts will be latched in their respective pending bits and will occur when the user enables interrupts immediately on return to the program. For example, the following instruction sequence can be used to read a byte of Flash memory:

```
RBITGIE,PSW  ;TEMPORARILY DISABLE INTS
LD ISPKEY,#KEY;SET THE KEY
JSRBCREADBF  ;GO READ THE FLASH
SBITGIE,PSW  ;RE-ENABLE INTERRUPTS
```

Plan: This information has been incorporated in the user documentation.

Number: 4

Name: Device failures after forced Boot ROM execution

Description: The device may be forced into Boot ROM execution for the purpose of MICROWIRE/PLUS In System Programming by the application of a high voltage on the G6 pin during and after Reset. The intent of this action is to account for program errors during development of customer ISP code and for recovery from catastrophic ISP failures. The device may be damaged if a high voltage is applied to the G6 pin prior to the application of V_{CC}.

Workaround: V_{CC} must be valid and stable before the application of the high voltage to the G6 pin. The following sequence should be followed to force execution from the Boot ROM MICROWIRE/PLUS ISP program:

1. Disconnect G6 from the source of data for MICROWIRE/PLUS ISP.

2. Apply V_{CC} to the device.

3. Pull RESET Low.

4. After V_{CC} is valid and stable, connect a voltage between 2 X V_{CC} and V_{CC}+7V to the G6 pin. Ensure that the rise time of the high voltage on G6 is slower than the minimum in the Electrical Specifications.

5. Pull RESET High.

6. After a delay of at least three instruction cycles, remove the high voltage from G6.

7. Connect G6 to the source of data for MICROWIRE/PLUS ISP.

Plan: This information has been incorporated in the user documentation.

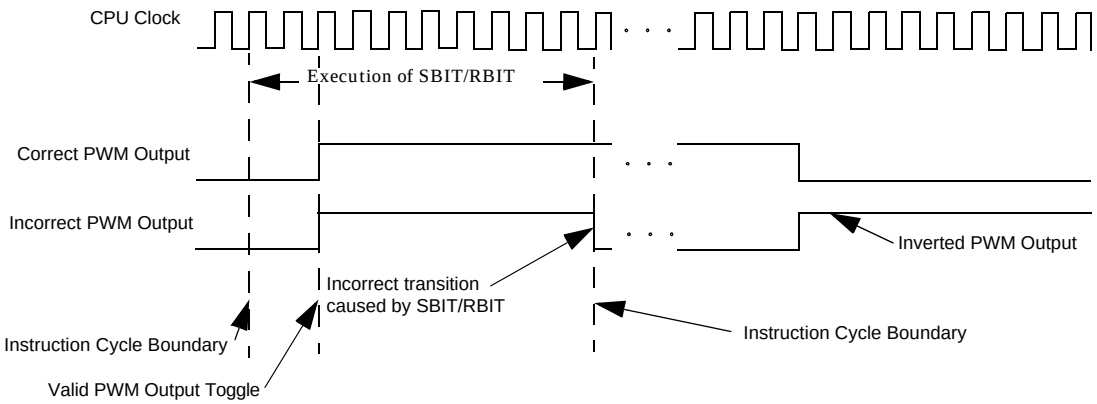
Number: 5

Name: High Speed Timer/Port L Interaction

Description: If either Timer T2 or T3 is used in High Speed PWM mode and an SBIT or RBIT instruction operates on any other bit of the PORT L Data Register, the PWM output may **appear** to miss a toggle and thus be inverted. If the timer causes the PWM output to toggle in the middle of an SBIT or RBIT operation on the PORTLD Register, the PWM output may be set back to its state before the output toggle by the operation of the SBIT/RBIT. This can have the effect of generating a shortened pulse (less than one instruction cycle in width) on the PWM output and inverting the PWM duty cycle.

If the PWM Timer is used in low speed mode or if the PWM output toggle is synchronous with the end of the instruction cycle, this problem is not seen.

The following figure illustrates the PWM output when the failure is seen.



Workaround: The user should be aware of the state of Timers T2 and T3 before any SBIT or RBIT instructions are executed which operate on the PORTLD register. If the PWM output is close to toggling, the user should delay the SBIT or RBIT instruction.

The following program sequence works to delay the operation. The user may wish to experiment with other sequences to see which best fits the application and to make sure that the time between the completion of the tests and the modification of PORTLD is not too long. The sequence can easily be modified to work with Timer T3.

```
        LD    B,#TMR2HI     ;POINT B TO THE TIMER
        LD    A,[B-]        ;GET THE VALUE IN THE TIMER
        IFGT  A,#0          ;IF NON ZERO
        JP    GOOD          ;WE HAVE TIME
WAIT:   IFBIT 6,[B]         ;TEST BIT 6 OF THE TIMER
        JP    GOOD          ;TIME TO GET IT DONE SAFELY
        JP    WAIT          ;WAIT A WHILE
GOOD:   SBIT  2,PORTLD      ;GO AHEAD AND SET THE BIT
```

1. The above program uses specific bits of the port for explanation purposes only.

2. The above program uses the SBIT instruction in the way of example. The RBIT instruction will have the same effect.

3. The above sequence will not work properly for PWM times shorter than 64 CPU Clock cycles.

4. The choice of TMR2LO bit 6 works, but may introduce delay at the wrong time in some applications, particularly if bit 7 is a one.

5. The above example shows the workaround if only one timer (T2 or T3) is used in high speed PWM mode. If both Timers T2 and T3 are used in high

speed PWM mode, the program becomes significantly more complicated, since the execution of the SBIT or RBIT instruction must be delayed until the PWM output of neither T2 nor T3 is likely to change during the execution of the instruction.

Plan: This will be corrected in the next revision of the device.