```
8/16 BIT Division
                         CUSTOMER ERRATA AND INFORMATION SHEET
                               Part: HC12BE32.01L43M.A
                                 General Business Use
                     Report Generated: Wed Aug 23, 2006, 03:04:18
                                        Page 1
                 ========================================================
                 |              HC12BE32.01L43M.A  Modules              |
                 ========================================================
                 |               Current Module Revision                |
                 ========================================================
                 |    ATD8B8C.2.14                                      |
                 |    BDLC.1.6                                          |
                 |    BDM256.3.1                                        |
                 |    BKP.3.0                                           |
                 |    CGM_B32.2.2                                       |
                 |    CORNER.2.9                                        |
                 |    CPU.3.8                                           |
                 |    ECT16B8C.1.5                                      |
                 |    EE768.2.4                                         |
                 |    INT.4.2                                           |
                 |    IOB.2.12                                          |
                 |    LIM_BE32.2.1                                      |
                 |    MEBI_B32.4.0                                      |
                 |    MMI_BE32.2.0                                      |
                 |    MSI1C1P.2.3                                       |
                 |    PADS_BE32.2.2                                     |
                 |    PWM8B4C.4.0                                       |
                 |    RAM1K.2.2                                         |
                 |    ROM32K.1.2                                        |
                 |    VDD.3.7                                           |
                 |    VDDA.3.7                                          |
                 |    VDDX.3.8                                          |
                 |    VPP.3.3                                           |
                 |    VREF.1.3                                          |
                 |    VSS.3.4                                           |
                 |    VSSA.3.5                                          |
                 |    VSSX.2.5                                          |
                 |    WCR.2.3                                           |
                 |    XTAL.2.7                                          |
                 ========================================================

                             ERRATA AND INFORMATION SUMMARY

AR_630  (ATD8B8C,ATD10B8C) Word read of $x06E($x1EE) will return $0000 regardless
AR_311  (ATD) conversion of the (Vrh-Vrl)/2 internal ref voltage returns $7F not $80
AR_629  (BDLC) 300us separation causes invalid symbol after SOF
AR_493  (BDLC) Ignored receiver error kills pending transmission
AR_519  (BDLC) CRC error in received message does not prevent IFR transmission
AR_520  (BDLC) Problem receiving type 2 IFR bytes after winning arbitration
AR_652  (CGM_B32) XIRQ during last cycle of STOP instruction causes run away
AR_644  (ECT16B8C) PA Overflow flag not set when event is concurrent w/ write of $FFFF
AR_528  (INT) Edge sensitive IRQ does not work correctly during STOP
AR_600  (INT_4) WAIT can not be exited if XIRQ/IRQ Level deasserts within window of time
AR_662  (INT) XIRQ after IRQ may cause vector fetch from bad address
AR_527  (INT) Disabling interrupt with I mask bit clear can cause SWI
AR_649  (MSI1C1P) SPI data gets corrrupted after disabling during trans in master mode
AR_333  (PWM8B4C) Concatenation implementation not compatible with HC11 PWM.
AR_148  (PWM) Duty register implementation not compatible with HC11 PWM definition
AR_327  (PWM8B4C) Scaled clock (S0,S1) equations incorrect in HC12 documentation
AR_328  (PWM8B4C) Scaled clock (S0,S1) equations not compatible with HC11 PWM.
AR_329  (PWM8B4C) Center-Aligned Mode equations incorrect in HC12 documentation
AR_330  (PWM8B4C) Period equations incorrect in HC12 documentation
AR_638  (WCR/CGM_D60/CGM_Dx128) Exit from pseudo-stop in limp home
```

DETAILED ERRATA DESCRIPTIONS

HC12_AR_630 Customer Erratum ATD8B8C.2.14

BRIEF DESCRIPTION:
(ATD8B8C,ATD10B8C) Word read of $x06E($x1EE) will return $0000 regardless

DESCRIPTION:
In the I/O register space, a word read of reserved location $x06E ($x1EE) will return a value of $00 for location $x06F ($x1EF), the Port AD Data Input Register, regardless of the logic levels present on the Port AD input pins. Note: address locations in parentheses apply to the second ATD converter module if present.

WORKAROUND:
To ensure reading the proper data from the Port AD Data Input Register, only access location $x06F ($x1EF) using byte read operations. Do not use a word read.

HC12_AR_311 Customer Information ATD8B8C.2.14

BRIEF DESCRIPTION:
(ATD) conversion of the (Vrh-Vrl)/2 internal ref voltage returns $7F not $80

DESCRIPTION:
The (VRH-VRL)/2 internal reference conversion result may be $7F, $80 or $81.

WORKAROUND:
If the (VRH-VRL)/2 internal reference is used (perhaps for system diagnostics), expected pass result may be $7F, $80 or $81.

HC12_AR_629 Customer Erratum BDLC.1.6

BRIEF DESCRIPTION:
(BDLC) 300us separation causes invalid symbol after SOF

DESCRIPTION:
If two messages are received at 300us IFS separation (+/-1us, as measured at the RX pin), the second message's SOF symbol generates an "invalid symbol" interrupt. This invalid symbol interrupt results in the second message being lost by the application software. This is the result of a race condition within the BDLC where it is changing states in its receive State Machine at the same time a transition occurs on the RX pin (the beginning of the second SOF).

WORKAROUND:
1) Ensure that no nodes on the J1850 network will transmit a message at 300us IFS separation from another message. Be certain that physical layer error is taken into account when calculating this case, as temperature changes and ground shifts can shift the timing seen at the RX pin of the microcontroller. Motorola silicon implementations of J1850 have not been shown to retransmit any faster than 320us, and are therefore not likely to cause this behavior. 2) Design messaging and application software to properly handle loss of messages in the system. This is safe programming practice in any case and will protect the integrity of the system in the event of a lost message.

HC12_AR_493 Customer Erratum BDLC.1.6

BRIEF DESCRIPTION:
(BDLC) Ignored receiver error kills pending transmission

DESCRIPTION:
To transmit a message using the BDLC, the user writes the first byte of the message to be transmitted into the BDLC data register (BDR). This will initiate the transmission process at the beginning of the next idle bus state. An invalid symbol being received by the BDLC clears any byte that had been previously written to the BDR. This will inhibit the transmission process until the user writes another byte to the BDR. The following scenario describes an event sequence that would prevent the user from knowing that an invalid symbol was received and that the BDR had been cleared.

WORKAROUND:
A two level strategy has been developed that positively signals the need to restart the transmission of a message. The first level looks for the special case of reception of an illegal symbol with a byte pending transmission in the BDLC data register, as described above. The second level uses a transmit watchdog timer to spot any case of a transmission not occurring within a maximum amount of time. 1a) If an illegal symbol interrupt occurs with a byte pending transmission in the BDR, reload the BDR with the first byte of thatmessage to restart transmission. 1b) Do not load the first byte of a message to transmit into the BDR when the IMSG bit is set in BCR1, wait until IMSG is cleared by reception of the next message on the bus. This will prevent the case of an illegal symbol interrupt clearing the BDR but no interrupt indication being given to the host due to IMSG blocking the interrupts. 1c) If a byte is pending transmission in the BDR, do not set the IMSG bit in BCR1. This will require the continued handling of each byte received in the current message, but will allow the host to see an illegal symbol interrupt and perform operation 1a, above. 2) When the first byte of a message to transmit is loaded into the BDR, start a 16.5 msec (minimum) transmit watchdog timer. This is the worst case time before the first TDRE interrupt of a transmission should occur. It is the sum of symbol times of a maximum length message (consisting of an SOF, 12 bytes of long symbols, and an IFS), and the SOF of the message to be transmitted. The worst case symbol receive times, as shown in Table A-2 of the BDLC Reference Manual, were used to calculate the watchdog time-out. When the first TDRE is received this watchdog is deactivated. If the watchdog times out, the first byte of the message to transmit is reloaded into the BDR and the watchdog restarted. This timer will prevent all cases of the driver software transmit state machine getting out of synchronization with the BDLC and hanging up, either due to this issue or any other unexpected condition that causes the software to think that a transmission is pending when it is not.

HC12_AR_519 Customer Erratum BDLC.1.6

BRIEF DESCRIPTION:
(BDLC) CRC error in received message does not prevent IFR transmission

DESCRIPTION:
CRC error in received message does not prevent IFR transmission.

WORKAROUND:
In response to the CRC Error interrupt ($18 in BSVR), immediately write an $FF byte into the BDR, and then clear the lower four bits in BCR2 (TSIFR, TMIFR0, TMIFR1, TEOD). This will cancel the IFR transmission.

HC12_AR_520 Customer Erratum BDLC.1.6

BRIEF DESCRIPTION:
(BDLC) Problem receiving type 2 IFR bytes after winning arbitration

DESCRIPTION:
When programmed to transmit a single byte Type 2 IFR (byte loaded in the BDR and TSIFR bit set in BCR2), the BDLC will attempt to transmit a single IFR byte following the EOD of the message currently being received. If the byte to be transmitted loses arbitration, the BDLC will continue to retry transmission until it is successful or an error occurs or the TEOD bit is set. When the BDLC does win arbitration and transmits its single byte IFR, it does not receive additional IFR bytes transmitted from other nodes properly, generating an invalid symbol interrupt. This errata does not affect the actual symbols on the bus, the original requesting node receives all IFR bytes from all responders properly.

WORKAROUND:
(Short form) When the first RXIFR interrupt occurs the requester message has been received without errors (invalid symbol or CRC) . When the BDLC receives back correctly the IFR byte is was trying to transmit, then response by this node is complete and the requester message received can be passed to the application layer. Ignore any invalid symbol error that occurs after the first RXIFR interrupt, since transmission of IFRs are not retried.

HC12_AR_652 Customer Erratum CGM_B32.2.2

BRIEF DESCRIPTION:
(CGM_B32) XIRQ during last cycle of STOP instruction causes run away

DESCRIPTION:
If an XIRQ interrupt occurs during the execution of the STOP instruction with the control bit DLY=0 (located in the INTCR register), the CPU may not run the software code as designed.

WORKAROUND:
1) Set the delay control bit DLY=1 so that a delay will be imposed prior to coming out of STOP. Must also implement workaround for AR#566 (Exiting STOP with DLY=1) as well. 2) If using XIRQ with a stable external clock and DLY=0, contact Motorola Applications Department for a detailed workaround.

HC12_AR_644 Customer Erratum ECT16B8C.1.5

BRIEF DESCRIPTION:
(ECT16B8C) PA Overflow flag not set when event is concurrent w/ write of $FFFF

DESCRIPTION:
When the value $FFFF is written to PACA or PACB and at the same time an external clocking pulse is applied to the PAC, the pulse accumulator may overflow from $FFFF to $0000, but the pulse accumulator overflow flag [PAFLG,PBFLG] is not set. Same situation may happen with 8-bit pulse accumulators PAC1 and PAC3.

WORKAROUND:
The input capture function for the subject channel be enabled prior to writing a value to the PACA or PACB. Write to the pulse accumulator register. Then do one NOP (to allow the input capture to update the interrupt flag) followed by a read of the input capture interrupt flag to see if it set. If yes, a check must be made for a missing pulse accumulator event. Steps for software workaround to see if event

happens while writing to PAC: 1) Enable Input Capture on same pin as the pulse accumulator (and same type of event). 2) Clear the appropriate CxF in the timer interrupt flag register. 3) Read PAC and store as "Old PAC". 4) Calculate desired PAC value and write it to the PAC. 5) Execute 1 NOP. PLEASE SEE http://beosweb.sps.mot.com/Design/HC12/HC12_erratas/PDFs/ FOR A COMPLETE VIEW OF THIS ERRATA. 6) Read CxF in the ti

---

HC12_AR_528 Customer Erratum INT.4.2

BRIEF DESCRIPTION:
(INT) Edge sensitive IRQ does not work correctly during STOP

DESCRIPTION:
When using an edge sensitive IRQ signal to trigger an interrupt service routine and the IRQ is not released during servicing, the part will not enter stop mode if the following executed instruction is STOP.

WORKAROUND:
When IRQ is set to be edge sensitive, release pin before executing STOP instruction.

HC12_AR_600 Customer Erratum INT.4.2

BRIEF DESCRIPTION:
(INT_4) WAIT can not be exited if XIRQ/IRQ Level deasserts within window of time

DESCRIPTION:
Depending on deassertion timing of XIRQ or Level IRQ with respect to exiting the WAIT instruction, the part can get stuck in WAIT. Only reset will allow you to recover. Noise/bounce on the pins could also cause this problem to manifest itself.

WORKAROUND:
Here are some suggestions: 1. Use edge-triggered IRQ (IRQE=1) instead of XIRQ or level-triggered IRQ. 2. Use RTI, timer interrupts, KWU or other interrupts (except level-sensitive IRQ or XIRQ) to exit WAIT. If using RTI, it must be enabled in WAIT (RSWAI=0) and the COP must be disabled (CME=0). 3. Assert XIRQ or level-sensitive IRQ until the interrupt subroutine is entered. 4. Add de-bouncing logic to prevent inadvertent highs when exiting WAIT.

HC12_AR_662 Customer Erratum INT.4.2

BRIEF DESCRIPTION:
(INT) XIRQ after IRQ may cause vector fetch from bad address

DESCRIPTION:
Asserting XIRQ interrupt at a certain time after an IRQ interrupt may cause a false interrupt vector to be sent to the CPU. The false interrupt vector occurs where the IRQ vector would be expected. The MCU performs stacking for both IRQ and XIRQ vectors, executes the XIRQ ISR, then uses an invalid return address from the XIRQ ISR.

WORKAROUND:
Hardware workaround: clock the input of the XIRQ pin so that it only asserts at a known clock edge. The best time for XIRQ to assert is at T2 (rising ECLK.)
Software workaround: add code to the XIRQ ISR to do the following: 1. Determine whether an XIRQ has interrupted an IRQ, by comparing the stack. If the stacked A:B, X, Y values are the same, and if the CCR values are the same excluding the I bit, then it is likely that the XIRQ interrupted an IRQ interrupt. 2. Adjust the

stack pointer to point to the IRQ interrupt stack. When the XIRQ interrupt finishes, the CPU will use the return address stacked for the IRQ interrupt, and should continue normal execution.

HC12_AR_527 Customer Information INT.4.2

BRIEF DESCRIPTION:
(INT) Disabling interrupt with I mask bit clear can cause SWI

DESCRIPTION:
If the source of an interrupt is taken away by disabling the interrupt without setting the I mask bit in the CCR, an SWI interrupt may be fetched instead of the vector for the interrupt source that was disabled. NOTE: THIS ERRATA HAS BEEN ADDED TO THE LATEST REVISION OF THE TECHNICAL DATABOOK FOR DT128A.

WORKAROUND:
Before disabling an interrupt using a local interrupt control bit, set the I mask bit in the CCR.

HC12_AR_649 Customer Erratum MSI1C1P.2.3

BRIEF DESCRIPTION:
(MSI1C1P) SPI data gets corrrupted after disabling during trans in master mode

DESCRIPTION:
SPI data gets corrupted after disabling SPI during transmission in master mode

WORKAROUND:
Make sure that SPE-bit in SPICR1 is not cleared before current transmission is complete (SPIF in SPISR is set).

HC12_AR_333 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM8B4C) Concatenation implementation not compatible with HC11 PWM.

DESCRIPTION:
HC11 code for the PWM module is not directly portable to the HC12. The PWM Concatenation (16-bit) mode implementation is not compatible with the HC11 PWM. When operating in Concatenation mode, the HC11 PWM channel output pin and clock source are both controlled by the "low-order" byte. i.e. if concatenate channels 1(3) & 2(4) the output pin is channel 2(4) and the clock source for the concatenated counter is the clock source for channel 2(4). When operating in Concatenation mode, the HC12 PWM channel output pin is the pin associated with the "high-order" byte and the clock source is controlled by the "low-order" byte. i.e. if concatenate 01(23), then pin 0(2) is the output and channel 1(2) controls the clock source.

WORKAROUND:
Modify system configuration based on HC12 specification information.

HC12_AR_148 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM) Duty register implementation not compatible with HC11 PWM definition

DESCRIPTION:
HC11 code for the PWM module is not directly portable to the HC12.

WORKAROUND:
Modify all HC11 duty register values to be the desired duty value - 1 on the HC12.

HC12_AR_327 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM8B4C) Scaled clock (S0,S1) equations incorrect in HC12 documentation

DESCRIPTION:
The PWM scaled clock (S0,S1) equations are incorrect in the HC12 documentation.
The incorrect equations are: Clock S0 = A / 2 * PWSCAL0 Clock S1 = B / 2 *
PWSCAL1

WORKAROUND:
The correct PWM scale clock (S0,S1) equations are: Clock S0 = A / 2 * (PWSCAL0 +
1) Clock S1 = B / 2 * (PWSCAL1 + 1) Therefore, programming $FF is full scale
divide of 256.

HC12_AR_328 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM8B4C) Scaled clock (S0,S1) equations not compatible with HC11 PWM.

DESCRIPTION:
HC11 code for the PWM module is not directly portable to the HC12. The PWM scaled
clock (S0,S1) equations are not compatible with the HC11 PWM. The HC11 PWM scaled
clock equations are: Clock S0 = A / 2 * PWSCAL0 Clock S1 = B / 2 * PWSCAL1 The
HC12 PWM scaled clock equations are: Clock S0 = A / 2 * (PWSCAL0 + 1) Clock S1 =
B / 2 * (PWSCAL1 + 1)

WORKAROUND:
Modify all HC12 PWM scaled clock (S0,S1) values to use the following equations:
Clock S0 = A / 2 * (PWSCAL0 + 1) Clock S1 = B / 2 * (PWSCAL1 + 1) Therefore,
programming $FF is full scale divide of 256.

HC12_AR_329 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM8B4C) Center-Aligned Mode equations incorrect in HC12 documentation

DESCRIPTION:
The PWM Center-Aligned Mode Duty Cycle equations are incorrect in the HC12
documentation. The incorrect equations are: Center-Aligned-Ouput Mode: (center=1)
Duty cycle = ((PWPERx - PWDTYx) / (PWPERx + 1)) X 100% for Polarity = 1 Duty cycle
= ((PWDTYx + 1) / (PWPERx + 1)) X 100% for Polarity = 0

WORKAROUND:
The correct PWM duty cycle equations are: Duty cycle = [(PWPERx - PWDTYx) /
(PWPERx)] X 100% for Polarity = 0 Duty cycle = {(PWDTYx) / (PWPERx)] X 100% for
Polarity = 1

HC12_AR_330 Customer Information PWM8B4C.4.0

BRIEF DESCRIPTION:
(PWM8B4C) Period equations incorrect in HC12 documentation

DESCRIPTION:

The PWM Period equations are incorrect in the HC12 documentation. The incorrect equations are: Period = (Channel-Clock-Period / (PWPER + 1)) for center = 0 Period = (Channel-Clock-Period / (2 * (PWPER + 1))) for center = 1

WORKAROUND:
The correct PWM Period equations are: Period = [Channel-Clock-Period * (PWPER + 1)] for center = 0 Period - [Channel-Clock-Period * (2 * PWPER)] for center = 1

HC12_AR_638 Customer Information WCR.2.3

BRIEF DESCRIPTION:
(WCR/CGM_D60/CGM_Dx128) Exit from pseudo-stop in limp home

DESCRIPTION:
Customer information: If the MCU is operating with the crystal oscillator driving the bus clock (BCSP=0) as it enters PSEUDO-STOP, the chip goes through LIMP HOME mode after it wakes up. This is not necessary since the crystal oscillator will be running and available immediately. The LIMP HOME bus frequency will not be as accurate as the crystal driven frequency.

WORKAROUND:
N/A