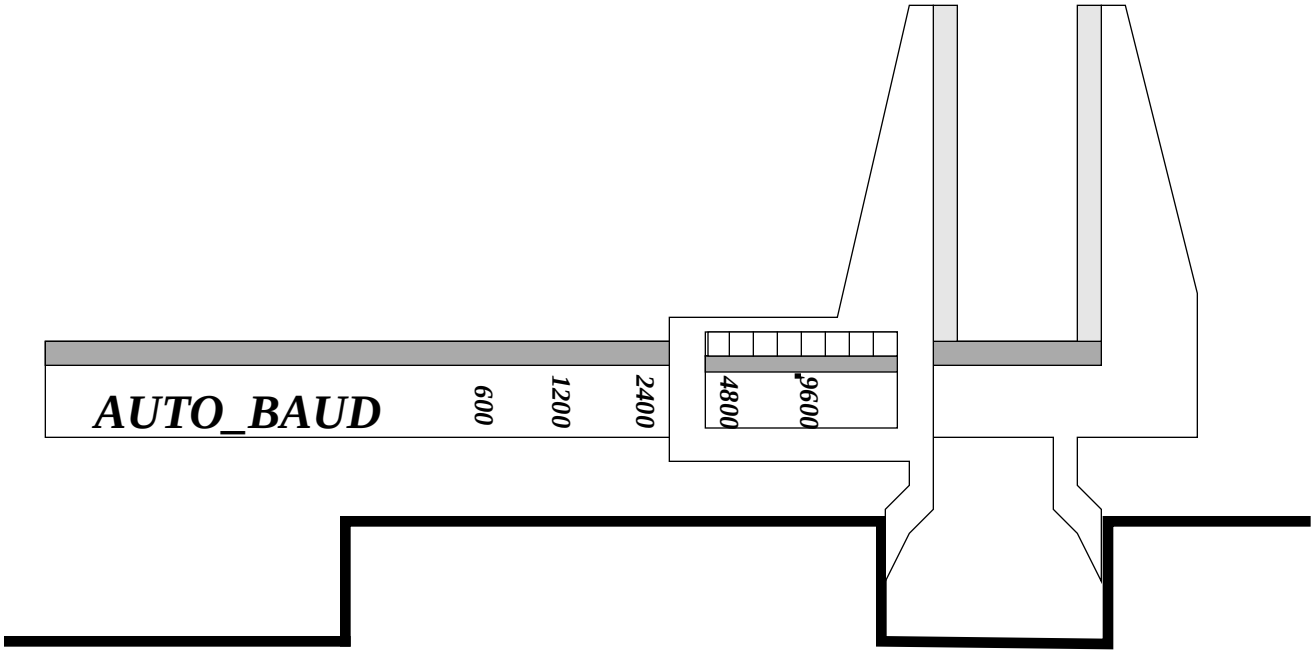
MOTOROLA

MICROPROCESSOR & MEMORY
TECHNOLOGIES GROUP

MC68302 Auto Baud Support Package Specifications

Revision 2.0



MC68302 AutoBaud Specification - Rev.2.0

May 1, 1994

Table of Contents

1.0 AutoBaud controller

This document describes the operation, registers, and programming of the Autobaud Microcode Package Revision 2.0. Below are the major features of Revision 2.0:

- Downloads into the MC68302 dual-port RAM.
- Requires 16 x the maximum baudrate for the sampling clock (can be internally or externally generated).
- Supports parity determination and character length detection on reception for 7-bit characters with parity, 8-bit characters with and without parity.
- Supports parity generation on transmission
- Supports 2 echo modes (automatic and smart echoing).
- Supports determining baudrates up to 115.2 kbaud.

1.1 Overview

The support for the AutoBaud function is contained in a RAM microcode that is downloaded into the dual-port RAM of the MC68302. The AutoBaud function determines the baudrate and format of an asynchronous data stream starting with a known character. This package may be used to implement the standard AT command set or other characters.

In order to use the AutoBaud mode, the Serial Communication Controller (SCC) is initially programmed to BISYNC mode. The SCC receiver then synchronizes on the falling edge of the START bit. Once a start bit is detected, each bit received is processed by the AutoBaud controller. The AutoBaud controller measures the length of the START bit to determine the receive baudrate and compares the length to values in a user supplied lookup table. After the baudrate is determined, the AutoBaud controller assembles the character and compares it against two user-defined characters. If a match is detected, the AutoBaud controller interrupts the host and returns the determined nominal start value from the lookup table. The AutoBaud controller continues to assemble the characters and interrupt the host until the host stops the reception process. The incoming message should contain a mixture of even and odd characters so that the user has enough information to decide on the proper character format (length and parity). The host then uses the returned nominal start value from the lookup table, modifies the SCC Configuration Register (SCON) to generate the correct baudrate, and reprograms the SCC to UART mode.

Many rates are supported including: 150, 300, 600, 1200, 2400, 4800, 9600, 14.4K, 19.2K, 38.4K, 57.6K, 64K, 96K, and 115.2K. To estimate the performance of the AutoBaud microcode package, the performance table in Appendix A of the MC68302 user's manual can be used. The maximum full-duplex rate for a BISYNC channel is one-tenth of the system clock rate. So a 16.67 MHz 68302 can support 115.2K AutoBaudrate with another low-speed channel (<50 kbps) and a

20 MHz MC68302 can support 115.2K AutoBaudrate with 2 low-speed channels. The performance can vary depending on system loading, configuration, and echoing mode.

It is important that the highest priority SCC be used for the AutoBaud function, since it is running at a very high rate. Any SCC that is guaranteed to be idle during the search operation of the AutoBaud process will not impact the performance of Autobaud in an application. Idle is defined as not having any transmit or receive requests to/from the SCC FIFOs.

1.2 AutoBaud Channel Reception Process

The interface between the AutoBaud controller and the host processor is implemented with shared data structures in the SCC Parameter RAM and in external memory and through the use of a special command to the SCC.

The AutoBaud Controller uses Receive Buffer Descriptor number 7 (Rx BD7) for the AutoBaud Command Descriptor. This Rx BD is initialized by the Host to contain a pointer to a lookup table residing in the external RAM (contains the maximum and nominal START bit length for each Baudrate). The Host also prepares two characters against which the AutoBaud Controller will compare the received character (usually these characters are ‘a’ and ‘A’), and the host initializes a pointer to a buffer in external memory where the assembled characters will be stored until the host stops the AutoBaud process. Finally, the host initializes the SCC Data Synchronization Register (DSR) to $7FFF in order to synchronize on the falling edge of the start bit.

Once the data structures are initialized, the host programs the SCON register to provide a sampling clock that is 16X the maximum supported baudrate. The Host then issues the Enter_Baud_Hunt command and enables the SCC in the BISYNC mode.

The AutoBaud Controller reception process starts when the START bit arrives. The AutoBaud Controller then starts to measure the START bit length. With each byte received from the SCC that “belongs” to the START bit, the AutoBaud Controller increments the Start Length Counter and compares it to the current lookup table entry. If the Start Length Counter passed the maximum bit length defined by the current table entry, the AutoBaud Controller switches to the next lookup table entry (the next slower baudrate). This process goes on until the AutoBaud Controller recognizes the end of the Start bit. Then, the AutoBaud controller starts the character assembly process.

The character assembly process uses the nominal bit length, taken from the current lookup table entry, to sample each incoming bit in it’s center. Each bit received is stored to form an 8 bit character. When the assembly process is completed (a Stop bit is received), the character is compared against two user-defined characters.

If the received character does not match any of the two user defined characters, the AutoBaud Controller re-enters the Enter_Baud_Hunt process. The host is not notified until a match is encountered.

If a match is found, the character is written to the Received Control Character Register (RCCR) with the corresponding status bit set in Rx BD7. The channel will generate the Control Character Received (CCR) interrupt (bit 3 in the SCCE), if enabled. If the character matched, but a framing error was detected on the stop bit, the AutoBaud Controller will also set the framing error status bit in Rx BD7.

The AutoBaud Controller then continues to assemble the incoming characters and to store them in the external data buffer. The host receives a CCR interrupt after each character is received.The host is responsible for determining the end of the incoming message (for example, a carriage return), stopping the Autobaud Process, and reprogramming the SCC to UART mode. The AutoBaud Controller returns the nominal start bit length value for the detected baudrate from the lookup table and a pointer to the last character received that was written to the external data buffer. The host must be able to handle each character interrupt in order to determine parity and character length (this information may be overwritten when the next character interrupt is presented to the host). The host uses the two received characters to determine 1) whether a properly formed “at” or “AT” was received, and 2) the proper character format (character length, parity).

Once this is decided, three possible actions can result. First, the host may decide that the data received was not a proper “at” or “AT”, and issue the Enter_Baud_Hunt command to cause the AutoBaud controller to resume the search process. Second, the host may decide the “at” or “AT” is proper and simply continue to received characters in BISYNC mode. Third, the M68000 core may decide that the “at” or “AT” is proper, but a change in character length or parity is required.

1.3 AutoBaud Channel Transmit Process

The AutoBaud microcode package supports two methods for transmission. The first method is automatic echo which is supported directly in the SCC hardware, and the second method is a smart echo or software transmit which is supported with an additional clock and software.

The automatic echo requires Revision C or later of the MC68302. Automatic echo is enabled by setting the DIAG bits in the SCC Mode Register (SCM) to ‘10’ and asserting the $\overline{\text{CD}}$ pin (externally on SCC1 and on SCC2 and SCC3, either externally or by leaving the pin as a general purpose input). The ENT bit of the SCC should remain cleared. The transmitter is not used, so this echoing method does not impact performance.

The smart echo or software transmit requires use of an additional clock and the transmitter, so the overall performance could be affected if other SCCs are running. This method requires an additional clock for sampling the incoming bit stream since the baudrate generator (BRG) must be

used to provide the correct frequency for transmission. The user needs to provide the sampling clock that will be used for the AutoBaud function on the RCLK pin (for example, a 1.8432 MHz clock for 115.2K). The clock that will be used for the SCC transmission can be provided to the BRG from the system clock or on TIN1.The TIN1 and RCLK1 pins can be tied together externally. After the first two characters have been received and character length and parity determined, the host programs the DSR to $FFFF, enables the transmitter (by setting ENT), and programs the transmit character descriptor (overlays CONTROL Character 8). The host is interrupted after each character is transmitted.

1.4 Autobaud Parameter RAM

When configured to operate in the AutoBaud mode, the MC68302 overlays some entries of the UART-Specific Parameter RAM as illustrated in Table 1 on page 6.

Table 1. AutoBaud Specific Parameter

Address	Name	Width	Description
SCC Base + 9C *	MAX_IDL	Word	Maximum IDLE Characters
SCC Base + 9E	MAX_BIT	Word	Current Maximum Start Bit Length
SCC Base + A0	NOM_START	Word	Current Nom. Start Bit (used to determine baudrate)
SCC Base + A2 *	PAREC	Word	Receive Parity Error Counter
SCC Base + A4 *	FRMEC	Word	Receive Framing Error Counter
SCC Base + A6 *	NOSEC	Word	Receive Noise Counter
SCC Base + A8 *	BRKEC	Word	Receive Break Error Counter
SCC Base + AA *	ABCHR1	Word	User Defined Character1
SCC Base + AC *	ABCHR2	Word	User Defined Character2
SCC Base + AE	RCCR	Word	Receive Control Character Register
SCC Base + B0 *	CHARACTER1	Word	CONTROL Character1
SCC Base + B2 *	CHARACTER2	Word	CONTROL Character2
SCC Base + B4 *	CHARACTER3	Word	CONTROL Character3
SCC Base + B6 *	CHARACTER4	Word	CONTROL Character4
SCC Base + B8 *	CHARACTER5	Word	CONTROL Character5
SCC Base + BA *	CHR6/RxPTR	Word	CONTRChar6/MSW of pointer to external Rx Buffer
SCC Base + BC *	CHR7RxPTR	Word	CONTRChar7/LSW of pointer to external Rx Buffer
SCC Base + BE *	CHR8/TxBD	Word	CONTROL Character8/Transmit BD

*– These values should be initialized by the user (M68000 core).

Note the new parameters that have been added to the table. They are MAX_BIT, NOM_START, ABCHR1, ABCHR2, RxPTR (2 words), and TxBD. These parameters are of special importance to the AutoBaud controller. They must be written prior to issuing the Enter_Baud_Hunt command.

When the channel is operating in the AutoBaud hunt mode, the MAX_BIT parameter is used to hold the current maximum start bit length. The NOM_START location contains the current nominal start from the LookUp Table. After the AutoBaud is successful and the first character is matched, the user should use the NOM_START value from the Autobaud Specific Parameter RAM to determine which baudrate from the LookUp table was detected. Also the Tx Internal Data Pointer (at offset SCC Base + 94) will point to the last character received into external data buffer.

NOTE

When the channel is operating in the UART mode, the NOM_START_/BRKCR is used as the Break Count Register and must be initialized before a STOP_TRANSMIT command is issued.

The characters ABCHR1 and ABCHR2 are the AutoBaud characters that should be searched for by the AutoBaud controller. Typically these are ‘a’ and ‘A’ (i.e. $0061 and $0041) if using the Hayes command set. These characters must be odd in order for the AutoBaud controller to correctly determine the length of the start bit. Characters are transmitted and received least significant bit first, so the AutoBaud controller detects the end of the Start Bit by the least significant bit of the character being a ‘1’.

The RxPTR is a 2 word location that contains a 32-bit pointer to a buffer in external memory used for assembling the received characters and must be initialized before the Enter_Baud_Hunt command is issued. WARNING: Since a length for this external buffer is not given, the user must provide enough space in memory for characters to be assembled and written until the Autobaud process is to avoid overwriting other data in memory. This location is not used as the CHARACTER7 value in the control character table until the channel operates in normal UART mode. After reception begins in normal UART mode (i.e. the “a” or “A” is found), this entry is available again as a Control Character Table entry.

The TxBD entry is used as the transmit character descriptor for smart echo or software transmit. This location is not used as the CHARACTER8 value in the control character table until the channel operates in normal UART mode. After reception begins in normal UART mode (i.e. the “a” or “A” is found), this entry is available again as a Control Character Table entry.

1.5 AutoBaud Programming Model

The following sections describe the details of initializing the Autobaud microcode, preparing for the AutoBaud process, and the memory structures used.

1.5.1 Initialization of the AutoBaud Microcode Package

After a power up or hardware reset, the following steps should be followed. Step 2 is not required after a hardware reset unless the dual-port RAM has been modified by the user after the reset.

1. Write $0700 to the BAR register. The base address of the internal ram after this action is $700000. If a different base address is used, then the S record file should be modified to the selected address.
2. Load the S-records file into the internal ram.
3. Write $0001 to address $0F8 in the supervisory space to enable the Autobaud package. The user will no longer be able to read or write the dual-port RAM.
4. Write a software reset command to the CR register.
5. Continue with the normal initialization sequence.

NOTE: This initialization sequence is for REV B or later parts (mask 2B14M or later).

1.5.2 Preparing for the AutoBaud Process

The Host begins preparation for the Autobaud process with the following steps. Steps 1 and 2 are required if the SCC has been used after reset or after UART mode in order to re-enable the process.

1. Disable the SCC by clearing the ENR and ENT bits. (The host may wish to precede this action with the STOP_TRANSMIT commands to abort transmission in an orderly way).

2. Issue the ENTER_HUNT_MODE comand to the SCC (This ensures that an open buffer descriptor is closed).

3. Set up all the AutoBaud parameters in the AutoBaud specific parameter RAM shown in Table 1 on page 6, the AutoBaud command descriptor shown in Table 2 on page 10, and the LookUp Table shown in Table 3 on page 12. Of these three areas, the AutoBaud controller only modifies the AutoBaud specific parameter RAM and the first word of the AutoBaud command descriptor during its operation.

4. Write the SCON to configure the SCC to use the baudrate generator clock of 16x the maximum supported baudrate. A typical value is $4000 assuming a 1.8432 MHz clock rate on TIN1 and a maximum baudrate of 115.2K, but this can change depending on the maximum baudrate and the EXTAL frequency.

6. Write the DSR of the SCC with the value $7FFF in order to detect the start bit.

7. The Host initiates the AutoBaud search process by issuing the Enter_Baud_Hunt command

8. Write the SCM of the SCC with $1133 to configure it for BISYNC mode, with the REVD and RBCS bits set, software operation mode, and the transmitter disabled. After a few characters have been received, the transmitter can be enabled, and the software echo function may be performed after issuing the RESTART TRANSMIT command.

In general, the AutoBaud controller uses the same data structure as that of the UART controller. The first character (if matched) is stored in the Receiver Control Character Register and the external data buffer, and the status of that character is reported in the AutoBaud Command Descriptor. After the first character, each incoming character is then stored in the buffer pointed to by RxPTR, and the status is updated in the Autobaud Command Descriptor. The Tx Internal Data Pointer (at offset SCC Base + 94) is updated to point to the last character stored in the external data buffer.

1.5.3 Enter_Baud_Hunt Command

This command instructs the AutoBaud Controller to begin searching for the baudrate of a user predefined character. Prior to issuing the command the M68000 prepares the AutoBaud Command Descriptor to contain the lookup table size and pointer.

The Enter_Baud_Hunt uses the GCI command with opcode = 11, and the channel number set for the corresponding SCC. For example, with SCC1, the value written to the Command Register would be $71.

1.5.4 AutoBaud Command Descriptor

The AutoBaud Controller uses the Receive Buffer Descriptor number 7 (Rx BD7) as an AutoBaud Command Descriptor. The AutoBaud Command Descriptor is used by the M68000 core to transfer command parameters to the AutoBaud Controller, and by the Autobaud Controller to report information concerning the received character.

The structure of the AutoBaud Command descriptor for the AutoBaud Process is shown in Table 2. The first word of the descriptor or the status word is updated after every character is received.

Table 2. Autobaud Command Descriptor

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2						FE	M2	M1					EOT		OV	CD
4	**LookUp Table Size**															
6	**Function Code**															
8	**LookUp Table Pointer**															

FE – Framing Error (Bit 10)

If this bit is set, a character with a framing error was received. A framing error is detected by the AutoBaud Controller when no stop bit is detected in the received data. FE will be set for a 9 bit character (8 bits + parity) if the parity bit is ‘0’. **NOTE**: the user must clear this bit when it is set.

M2 - Match Character2 (Bit 9)

When this bit is set, the character received matched the User Defined Character 2. The received character is written into the Receive Control Character Register (RCCR).

M1 - Match Character1 (Bit 8)

When this bit is set, the character received matched the User Defined Character 1. The received character is written into the Receive Control Character Register (RCCR).

EOT - End Of Table (bit 3)

When this bit is set, the AutoBaud Controller measured start length exceeded the maximum start length of the last entry in the LookUp table (lowest baudrate). **NOTE**: the user must clear this bit when it is set.

OV - Overrun (bit 1)

If this bit is set, a receiver overrun occurred during AutoBaud reception. **NOTE**: the user must clear this bit when it is set.

CD - Carrier Detect Lost (bit 0)

If this bit is set, the carrier detect signal was negated during AutoBaud reception. **NOTE**: the user must clear this bit when it is set.

Lookup Table Size -

LookUp Table Size is the number of baudrate entries in the external lookup table.

LookUp Table Pointer -

The LookUp Table Pointer is the address in the external RAM where the lookup table begins. WARNING: The LookUp Table cannot cross a 64k memory block boundary.

1.5.5 AutoBaud LookUp Table

The AutoBaud Controller uses an external LookUp table to determine the baudrate while in the process of receiving a character. The LookUp table contains two entries for each supported baudrate. The first entry is the maximum START length for the particular baudrate, and the second entry is the nominal length for a 1/2 START bit.

To determine the two values for each table entry, first calculate the AutoBaud sampling rate (EQ 2). To do this EQ 1 must be used until EQ 2 is satisfied. The sampling rate is the lowest speed baudrate that can be generated by the SCC baudrate generator that is over a threshold defined in EQ 2.

BRG Clk Rate = System Clock or TIN1 / ((Clock Divider bits in SCON) + 1) (EQ 1)

assuming that the DIV bit in SCON is set to 0, (otherwise an additional “divide-by-4” must be included).

Sampling Rate = BRG Clk Rate, where BRG Clk rate >= (Max Desired UART Baud Rate) x 16 (EQ 2)

For instance, if a 115.2K baudrate is desired, with a 16.67 MHz system clock, the minimum sampling rate possible is 1.843 MHz = 115.2K x 16. This exact frequency can be input to RCLK1 or TIN1 as the sample clock. If the system clock is to be used, a 16.67 MHz system clock cannot produce an exact baudrate clock of 1.843 MHz. The lowest one that can be used is Baud Rate = 16.67 MHz / (7+1) = 2.083 MHz. Thus, 2.083 MHz is the sampling rate, and the SCON should be set to $000e to produce this.

Once the sampling rate is known, the other two equations follow easily. The maximum START bit length is calculated by the following equation:

Maximum START length = (Sampling Rate/Recognized baudrate) x 1.05 (EQ 3)

Thus, for the first entry in the table, the Maximum Start Length is 1.8432 Mhz/115200 x 1.05 = 17 for an external sample clock. The value 1.05 is a suggested margin that allows characters 5% larger than the nominal character rate to be accepted. In effect, the margin determines the “split point” between what is considered to be a 56.7K character rate and what is a 38.4K character rate. The margin should not normally be less than 1.03 due to clocking differences between UARTs.

The nominal START bit length is calculated by:

Nominal START length = (Sampling Rate/Recognized baudrate) / 2 (EQ 4)

For the 115.2K example, in the first entry of the table, this would be 1.8432 MHz/115.2K/2 = 8.

The structure of the LookUp table is shown in Table 3 on page 12. The table starts with the maximum UART baudrate supported and ends with the minimum UART baudrate supported.

Table 3. AutoBaud LookUp Table Format

OFFSET from LookUp Table Pointer	DESCRIPTION
0	Maximum START Length
2	Nominal START Length
4	Maximum START Length
6	Nominal START Length
•	Maximum START Length
•	Nominal START Length
(LookUp Table Size - 1) * 4	Maximum START Length
[(LookUp Table Size - 1) * 4] + 2	Nominal START Length

NOTE

If less margin is used in the calculation of the Maximum START length above, it is possible to distinguish between close UART rates such as 64K and 57.6K. However variations in RS232 drivers of up to 4%, plus nominal clocking rate variations of 3%, plus the fact that the sampling rate may not perfectly divide into the desired UART rate, can make this distinction difficult to achieve in some scenarios.

1.5.5.1 LookUp Table Example

Table 4 is an example AutoBaud Lookup Table. The Maximum Start and Nominal Start values are derived assuming a 1.8432 MHz sampling clock on TIN1 or RCLK and a shift factor of 5%.

Table 4. LookUp Table Example

Desired Baudrate	Maximum Start	Nominal Start
115200	17	8
57600	34	16
38400	50	24
28800	67	32
19200	101	48
14400	134	64
12000	161	77
9600	202	96

Table 4. LookUp Table Example

Desired Baudrate	Maximum Start	Nominal Start
7200	269	128
4800	403	192
2400	806	384
1200	1613	768
600	3226	1536
300	6451	3072
110	17594	8378

1.5.6 Determining Character Length and Parity

Table 5 on page 13 shows the different possible character lengths and parity that will be discussed. The following paragraphs will discuss for each case how to determine the parity.

Table 5. Character Lengths and Parity Cases

Case #	Character Length	Parity	Notes
1	7-bit	no parity, 1 stop bit	Not Supported
2	7-bit	even parity odd parity parity=1 parity=0	Parity is indicated by the most significant bit of the byte
3	8-bit	no parity	same as 7-bit, parity=0
4	8-bit	even parity odd parity parity=0	Parity is indicated by which characters generate a FE interrupt
5	8-bit	parity=1	Not Supported

- Case 1– This case can not be supported because the Autobaud can not separate the first character from the second character.
- Case 2– As each character is assembled, it is stored into a complete byte. Assuming that the characters are ASCII characters with 7-bit codes, the 8th bit of the byte will contain the parity bit. If the parity is either even or odd, then after receiving an odd character and an even character, the 8th bit should be different for the odd and even characters. The parity can be determined by the setting of the parity bit for one of the two characters. If the 8th bit is always a 1, this is the same as a 7-bit character, no parity and at least 2 stop bits or a 7-bit character with force 1 parity. If the 8th bit is always a zero, then either the character is a 7-bit character with force 0 parity, or the character is a 8-bit character with no parity.
- Case 3– This case is the same as 7-bit character with force 0 parity. The 8th bit of the byte will always be zero.

- Case 4–This case assumes a 8-bit character with the 8th bit of the character equal to a 0 (ASCII character codes define the 8th bit as zero). If the parity is either even or odd, then after receiving an odd and an even character, a framing error (FE) interrupt should have been generated for one of them (the interrupt is generated when the parity bit is zero). The user can determine the parity by which character generated a FE interrupt (if the odd character did, then the parity is odd). If a framing error occurs on every character, then the character is 8-bits with force 0 parity. If no framing error occurs, than this is the same as Case 5.
- Case 5– This case is not supported, because it can not be differentiated from 7-bit force 0 parity and 8-bit no parity. If the 9th bit is a 1, then it will be interpreted as a stop bit.

1.5.7 AutoBaud Reception Error Handling Procedure

The AutoBaud controller reports reception error conditions using the AutoBaud Command Descriptor. Three types of errors are supported:

- Carrier Detect Lost during reception

  When this error occurs and the channel is not programmed to control this line with software, the channel terminates reception, sets the Carrier Detect lost (CD) bit in the Command Descriptor, and generates the CCR interrupt, if enabled. CCR is bit 3 of the SCCE register.

- Overrun Error

  When this error occurs, the channel terminates reception, sets the overrun (OV) bit in the Command Descriptor, and generates the CCR interrupt, if enabled.

- End Of Table Error

  When this error occurs, the channel terminates reception, sets the end of table (EOT) bit in the Command Descriptor, and generates the CCR interrupt, if enabled.

Any of these errors will cause the channel to abort reception. In order to resume AutoBaud operation after an error condition, the M68000 should clear the status bits and issue the Enter_Baud_Hunt command again.

1.5.8 AutoBaud Transmission

The autobaud package supports two methods for echoing characters or transmitting characters. The two methods are automatic echo and smart echo.

1.5.8.1 Automatic Echo

This method uses the SCC hardware to automatically echo the characters back on the TxD pin. The automatic echo is enabled by setting the DIAG bits in the SCM to ‘10’. On Revision C and later of the MC68302, the transmitter should not be enabled. The hardware echo is done automatically. The $\overline{CD}$ pin needs to be asserted in order for the characters to be transmitted back. On SCC1, the external $\overline{CD}$ pin must be tied low. On SCC2 and SCC3, either the external $\overline{CD}$ pin must be tied low or the $\overline{CD}$ pins should be left configured as general purpose input pins (the $\overline{CD}$ signal to the SCC is then connected to ground internally).

Using the automatic echo, the receiver still autobauds correctly and performance is not affected. The SCC echos the received data with a few nanoseconds delay.

1.5.8.2 Smart Echo

This method requires addition hardware and software to implement. The user must provide two clock sources. One clock source is the sample clock which is input on RCLK and cannot be divided down. The BRG is used to divide the second clock down to provide the clock used for transmit. The second clock can be either the system clock or a clock connected to TIN1. The TIN1 and RCLK pins can be connected to each other externally.

After the first character is received, the user must take the following steps:

1. Determine the baudrate from the returned NOM_START value and program SCON to (input frequency/baudrate)-1, where the input frequency is either the system clock or the clock on TIN1.
2. Program the DSR to $FFFF. The DSR will need to be programmed back to $7FFF before the Enter_Baud_Hunt command is issued again.
3. Set the ENT bit in the mode register.
4. Program the transmit character BD as show in Table 6 on page 15.

Table 6. Transmit Character BD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		CL		PE	PM			CHAR							

R (ready bit)

0 = Character is not ready
1 = Character is ready to transmit

CL (character len)

0 = 7 bits + parity or 8 bits with no parity
1 = 8 bits + parity

PE (parity enable)

0 = no parity
1 = parity

PM (parity mode)

0 = even parity
1= odd parity

The AutoBaud controller issues a Tx interrupt after each character is transmitted.

1.5.9 Reprogramming to UART Mode or another protocol

The following steps should be followed in order to switch the SCC from AutoBaud to UART mode or to another protocol.

- Disable the SCC by clearing ENR and ENT.
- Issue the Enter_Hunt_Mode command.
- Initialize the SCC parameter RAM (specifically, the Rx and Tx internal states and the words containing the Rx and Tx BD#s) to the state they were at after reset and initialize the Protocol Specific Parameter Area for the new protocol.
- Re-enable the SCC with the new mode.

NOTE

This initialization sequence is for REV B or later parts (mask 2B14M or later).

1.6 AutoBaud Programming Example

The following program is a simple working example that shows how to initialize the autobaud controller and handle the interrupts. This was run on a 68302 ADS board with a 16.67 MHz system clock and a 1.8432 MHz clock on TIN1 for the sampling clock.

```
****************************** EQU TABLE ******************************
* The following three values are application dependent
BASE      EQU       $0700000          ; This is set according to the value in BAR
INIT      EQU       $0030300          ; Initialization Routine
INT_VEC   EQU       $0031000          ; Interrupt Vector for SCC1
LOOKUP    EQU       $0030200          ; Look up table for baud rate
LK_SZ     EQU       15                ; Number of entries in the lookup table
RXCHARS   EQU       $4000             ; Pointer to receive buffer for characters
NOMSP     EQU       BASE+$4A0         ; Nominal speed reported for SCC1
RXINTPT   EQU       BASE+$488         ; Pointer to internal receive pointer
TXINTPT   EQU       BASE+$494         ; Pointer to internal transmit pointer
```

```
CARRT     EQU       $0F              ; ASCII for Carriage Return

* Commands
RSTCP     EQU       $81              ; Reset CP
STPTX1    EQU       $01              ; STOP TRANSMIT SCC1
RSTX1     EQU       $11              ; RESTART TRANSMIT SCC1
ENHM1     EQU       $21              ; ENTER HUNT MODE SCC1
ENBHM1    EQU       $071             ; ENTER BAUD HUNT MODE SCC1

CR        EQU       BASE+$0860       ; Command Register
BAR       EQU       $0F2             ; Base Address Register
MER       EQU       $0F8             ; UCODE Option Enabling Register
SCR       EQU       $0F4             ; System Control Register
GIMR      EQU       BASE+$0812       ; Global Interrupt Mode Register
IPR       EQU       BASE+$0814       ; Interrupt Pending Register
IMR       EQU       BASE+$0816       ; Interrupt Mask Register
ISR       EQU       BASE+$0818       ; In-Service Register
PACNT     EQU       BASE+$081E       ; PORT A CONTROL
PBCNT     EQU       BASE+$0824
SIMODE    EQU       BASE+$08B4       ; Serial Interface Mode Register

SCON1     EQU       BASE+$0882       ; SCC1 Configuration Register
SCM1      EQU       BASE+$0884       ; SCC1 Mode Register
DSR1      EQU       BASE+$0886       ; SCC1 DSR Register
SCCE1     EQU       BASE+$0888       ; SCC1 Event Register
SCCM1     EQU       BASE+$088A       ; SCC1 Mask Register

EN_SCC    EQU       $0C              ; ENR and ENT bits in SCM
ENRON     EQU       $08              ; For Setting ENR
ENTON     EQU       $04              ; For Setting ENT
ENTOFF    EQU       $0FFFB           ; For Clearing ENT

*** SCC Parameter Table ***
ST_BD     EQU       0                ; Status and Control in BD
SS_BD     EQU       1                ; Status in BD
LN_BD     EQU       2                ; Data Length in BD
PT_BD     EQU       4                ; Buffer pointer in BD
SZ_BD     EQU       $08              ; Size of BD = 8 bytes

READY     EQU       $07              ; Ready bit in the 1st byte of TX BD
EMPTY     EQU       $07              ; Empty bit in the 1st byte of RX BD
WRAP      EQU       $05              ; Wrap bit in the 1st byte of BD

* SCC1 Parameter *
FCR_1     EQU       BASE+$0480       ; RFCR and TFCR for SCC1
MRBLR_1   EQU       BASE+$0482       ; Max Rx Buffer Length
MIDL_1    EQU       BASE+$049C       ; Max IDLE char (received)
BRKCR_1   EQU       BASE+$04A0       ; Break counter (transmitted)
PAREC_1   EQU       BASE+$04A2       ; RX Parity error
FRMEC_1   EQU       BASE+$04A4       ; RX Framing error
NOSEC_1   EQU       BASE+$04A6       ; RX Noise counter
BRKEC_1   EQU       BASE+$04A8       ; RX Break condition counter
UADR1_1   EQU       BASE+$04AA       ; UART Address char 1
UADR2_1   EQU       BASE+$04AC       ; UART Address char 2
RCCR_1    EQU       BASE+$04AE       ; RX Control Char
RCH1_1    EQU       BASE+$04B0       ; CONTROL char 1
RCH6_1    EQU       BASE+$04BA       ; L word to hold ptr to char buffer
RCH8_1    EQU       BASE+$04BE       ; CONTROL char 8
```

```
* Autobaud special
ABCR1_1   EQU     BASE+$04AA          ; User defined Char 1 (a/A)
ABCR2_1   EQU     BASE+$04AC          ; User defined Char 2 (a/A)
RxPTR     EQU     BASE+$04BA          ; Pointer to external Rx data buffer
TxBD      EQU     BASE+$04BE          ; Transmit character descriptor
RXBD_01   EQU     BASE+$0400          ; 1st RXBD of SCC1
TXBD_01   EQU     BASE+$0440          ; 1st TXBD of SCC1
RXBD_71   EQU     BASE+$0438          ; 7th RXBD of SCC1 for Autobaud command


* The followings are application dependent, for this example
BD_CNT    EQU     $08                 ; Number of BDs used
SZ_BF     EQU     $10                 ; Size of buffer = 16 bytes
N_DATA    EQU     $0A                 ; Number of data to be sent in a buffer

******************************** Data Used ******************************

          ORG     LOOKUP
          DC.W    17
          DC.W    8                   ;115200
          DC.W    34
          DC.W    16                  ;57600
          DC.W    50
          DC.W    24                  ;38400
          DC.W    67
          DC.W    32                  ;28800
          DC.W    101
          DC.W    48                  ;19200
          DC.W    134
          DC.W    64                  ;14400
          DC.W    161
          DC.W    77                  ;12000
          DC.W    202
          DC.W    96                  ;9600
          DC.W    269
          DC.W    128                 ;7200
          DC.W    403
          DC.W    192                 ;4800
          DC.W    806
          DC.W    384                 ;2400
          DC.W    1613
          DC.W    768                 ;1200
          DC.W    3226
          DC.W    1536                ;600
          DC.W    6451
          DC.W    3072                ;300
          DC.W    17594
          DC.W    8378                ;110

RXFLAG    DC.W    0                   ; Semaphore for character reception

****************** M68302 Initialization Code *****************
* Register Initialized values in ADS board before execution
* USP=00080000 ISP=000040000 (Stack pointer not used)
* This program assumes that the AutoBaud microcode package has
* already been downloaded into the internally dual-port RAM.

          ORG     INIT                ; PC=00030300
          MOVE.W  #$2700,SR           ; SR=2700, mask off interrupts
```

```
                MOVE.W    #$0700,BAR          ; BAR=0700
                LEA.L     $40000,A7           ; Load initial stack pointer

                MOVE.W    #$0001,MER          ; Enable Ucode for autobaud
                MOVE.B    #RSTCP,CR           ; Reset CR of communication processor
LP00            BTST.B    #0,CR
                BNE.B     LP00

                MOVE.L    #0,SCR              ; Nothing special for this example

*** Setups for interrupt ***
                MOVE.W    #$0A0,GIMR          ; Normal mode, v7-v5=5
                MOVE.W    #0,IMR              ; Mask off all for now
                MOVE.W    #$FFFF,IPR          ; Clear IPR

*** Set up Serial Interface Connection ***
                MOVE.W    #$0008,PBCNT        ; FOR TIN1 INSTEAD OF PB3

* Set up SIMODE = NMSI Normal operation
                MOVE.W    #$0,SIMODE

********** SCC1 Initialization for Autobaud **********
* Interrupt Vector: SCC1 interrupt handler is at INTSCC1
* v7-v5=5, v4-v0=$0d => vector=$ad => Exception vector = ($ad<<2) = $2b4
                MOVE.L    #INTSCC1,$02B4

********** begin autobaud **********
                MOVE.W    #$4000,SCON1        ; Set initial divider
                MOVE.W    #$7FFF,DSR1         ; Initialize DSR to look for START bit

* Set SCC1 Mode for autobaud - BISYNC mode, s/w operation, REVD and RBCS set.
                MOVE.W    #$1133,SCM1
********** end autobaud **********

* Set up Parameter RAM
                MOVE.W    #0,FCR_1            ; Clear RFCR and TFCR
                MOVE.W    #$0A,MRBLR_1        ; MAX BUFFER length
                MOVE.W    #$4,MIDL_1          ; MAX_IDL
* The following must be executed before the STOP Transmit command is issued
*               MOVE.W    #$1,BRKCR_1         ; BRKCR = 1.Only one break char. sent if
*                                               STOP TRANSMIT command is executed.
* Init counters
                MOVE.W    #$0,PAREC_1
                MOVE.W    #$0,FRMEC_1
                MOVE.W    #$0,NOSEC_1
                MOVE.W    #$0,BRKEC_1
* Put 'a' and 'A' into user defined characters.
                MOVE.W    #$61,ABCR1_1        ; ASCII for 'a'
                MOVE.W    #$41,ABCR2_1        ; ASCII for 'A'
* Set default SCC mode: Uart, 8-char, no parity, 1 stop, s/w operation
                MOVE.W    #$013D,D_SCM_1
*               UART Address characters not used in normal operation.
                MOVE.W    #$8000,RCH1_1
                MOVE.W    #$0000,RCH8_1       ; no software echoing in this example
* No control characters are initialized to cause special interrupts.
* The flow-control facility (XON-XOFF) is not used.
                MOVE.B    #$FF,SCCE1          ; clear SCCE1
                MOVE.B    #$08,SCCM1          ; only interrupt for RX control character
```

```
**********************************************************************
*** Prepare Buffer Descriptors ***

* Clear M68000 data registers
                CLR.L       D0
                CLR.L       D1
                CLR.L       D2
                CLR.L       D3
                CLR.L       D4
                CLR.L       D5

* SCC1 Rx Buffer Descriptors Initialization values before execution.

* SCC1 Autobaud Descriptors Initialization values before execution:
* 00700430 0000 0008 0002 0000

                LEA.L       RXBD_71,A0          ; A0 points to autobaud command descriptor
                LEA.L       RCH6_1,A1           ; A1 points to the LOOKUP table for autobaud
                MOVE.W      #$0,ST_BD(A0)       ; Clear all Status Bits
                MOVE.W      #LK_SZ,LN_BD(A0)    ; Initialize Lookup Table Size
                MOVE.L      #LOOKUP,PT_BD(A0)   ; Load pointer to Lookup table
                MOVE.L      #RXCHARS,(A1)       ;Load the pointer to rx characters


*       Special function for Auto baud Mode
* At this time the SCCs are still disabled.
* If autobaud is entered dynamically, one should first do the followings
* before issuing ENTER BAUD HUNT for the SCC that is going to run autobaud:
* (1) STOP TRANSMIT, (2) disable ENT and ENR, and (3) ENTER HUNT MODE.

                MOVE.B      #ENBHM1,CR          ; Enter BAUD HUNT mode
LP1             BTST.B      #0,CR               ; Wait for the command to complete
                BNE.B       LP1
                ORI.W       #ENRON,SCM1         ; For SCC1 ENR=1, the transmitter will be
*               automatically enabled by the microcode once
*               it sets up the UART-autobaud-mode.

* Ready to accept interrupt*** Set IMR ***
                MOVE.W      #$2000,IMR          ; Allow SCC1 interrupts only.
                MOVE.W      #$2000,SR           ; Unmask interrupt
                JMP         MAIN                ; Go to Main routine for Tx and Rx

**************************** Main Routine ********************************

* Set up BD pointers
MAIN            CLR.L       D5                  ; COUNTER FOR OVER-RUN
* The following clears out the external data buffer for received characters
RxReady         LEA.L       RXCHARS,A5          ; Load address of receive buffer
                MOVE.W      #$20,D4             ; Load counter for clear loop
CLRLP           MOVE.L      #0,(A5)+            ; Clear out the memory
                DBNE        D4,CLRLP
                LEA.L       RXCHARS,A5          ; Load address of character buffer

*Wait for character to be received.
WAITF           CMPI.W      #0,RXFLAG           ;COMPARE COUNTER
                BNE         RX_REC
                BTST.B      #1,RXBD_71+1        ; CHECK OV
                BEQ         WAITF
                ADDI.W      #1,D5               ; INCREMENT OVERRUN COUNTER
```

```
* This just returns to wait for the next character.
* At this point or in the interrupt handler, the user should put code
* that checks for character length and parity as well as the end of
* message.
*
RX_REC    BRA         WAITF

* When the end of message is detected, the user is responsible for
* the following:
* (1) STOP TRANSMIT, (2) disable ENT and ENR, (3) ENTER HUNT MODE,
* (4) write the Rx and Tx internal states back to their reset values,
* (5) write the Rx and Tx buffer numbers back to their rest values,
* (6) program the SCON register for the correct baudrate (returned at  base + $4A0)
* (7) program the SCC mode to UART mode with the correct character length and parity
* (8) issue the RESTART Transmit and Enter Hunt Mode commands
* and (9) and enable the receiver and transmitter.

********** SCC1 Interrupt handler **********
* This routine only handle the case when RX=1
*
            ORG         INT_VEC             ; Interrupt Vector for SCC1
INTSCC1     CLR.L       D1                  ; Clear D1
            MOVE.B      SCCE1,D1            ; SCCE1 => D1
            ANDI.W      #8,D1               ; Is CCR set?
            CMPI.W      #0,D1               ; If set
            BNE.B       RXINT1              ; Handle receiver's interrupt
*
* The handling of other events, e.g., CTS, CD IDL, BSY, is left to
* the users as desired.
*
EXINT1      MOVE.W      #$2000,ISR          ; Clear SCC1 bit in ISR
            RTE

********** Receiver portion of SCC1 interrupt routine **********
* This routine handles received (non-empty) BD: set data length = 0,
* clear status bits, set empty = 1, and update PRD
*
* Clear the identified events as soon as possible, so as not to lost
* events occur during the interrupt handling
RXINT1      MOVE.B      #8,SCCE1            ; Clear RX in SCCE1
*
* While Not-Empty continue to process the next Character, Else Exit.
*
NXPRD1      MOVE.L      A5,D1               ; Load current pointer
            CMP.L       TXINTPT.L,D1        ; Check to see if a char was rx
            BEQ         EXINT1              ; If no, then exit
            MOVE.B      (A5)+,D1            ; Load Character
            CMP.B       #CARRT,D1           ; Is the char a carriage return
            BEQ         EXRX1               ; If so, then set flag and exit
            BRA         NXPRD1              ; Otherwise check for another char

EXRX1       MOVE.W      #1,RXFLAG           ; SET SEMAPHORE
            JMP         EXINT1              ; Exit receiver portion of the handler

*****************************************************************

            END
```