

## **MCA8000A Data Format**

### **Spectral Data Structure**

The counts in each channel (further referred as channel data) are accumulated in a long integer variable, 4 bytes long. The first two bytes form the lower word while the higher two bytes form the upper word of the channel data. The maximum channel number of a single spectrum is 16,384.

### **MCA Status Data Structure**

The Status Data Structure provides information for the current status of the MCA. Every time the MCA8000A sends data to the computer the entire status structure is updated. The status of the MCA8000A is reported in 20 bytes. The format of the status structure is shown below. The first byte of the status structure transmitted is **DataChkSum\_3** while the last transmitted byte of the structure is the **Checksum** byte:

#### **8 bit Status structure**

**DataChkSum\_3...first transmitted**

**DataChkSum\_2**

**DataChkSum\_1**

**DataChkSum\_0**

**PresetTime\_2**

**PresetTime\_1**

**PresetTime\_0**

**Battery**

**RealTime\_2**

**RealTime\_1**

**RealTime\_0**

**RealTime\_75**

**LiveTime\_2**

**LiveTime\_1**

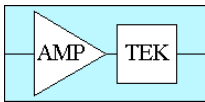
**LiveTime\_0**

**LiveTime\_75**

**Threshold\_1**

**Threshold\_0**

**Flags**

**Checksum...last transmitted**

**DataChkSum\_0** to **DataChkSum\_3** represent a 32 bit unsigned integer variable **DataChkSum**. **DataChkSum\_0** is the least significant byte. **DataChkSum** is either the sum MOD  $2^{16}$  of all bytes of the channel data that were transmitted at the last data exchange preceding the transmission of the status structure, or is the current group number and the serial number of the MCA8000A unit.

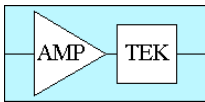
**PresetTime\_0** to **PresetTime\_2** bytes form a 24 bit integer variable that is the preset acquisition time of the MCA8000A in seconds. **PresetTime\_0** is the least significant byte. The value of the **PresetTime** can be modified by the PRESET TIME COMMAND.

**Battery** is a single byte value that represents the MCA8000A battery voltage or indicates the use of an external power source. If **Battery** is equal to 0 then the MCA8000A is powered by an external power source. The table below shows the **Battery** value vs. Remaining battery capacity for Alkaline (3V) and NiCd (2.5V) batteries:

Remaining Capacity	Battery Value	
	Alkaline (3V)	NiCd (2.5V)
[%]		
< 5	< 50	< 68
12.5	63	73
25	68	74
37.5	71	75
50	74	76
62.5	77	77
75	80	78
> 87.5	> 88	> 80

The four bytes from **RealTime\_0** to **RealTime\_75** are used to calculate the elapsed real acquisition time. The real time is calculated as:

$$\text{RealTime} = 2^{16} * \text{RealTime}_2 + 2^8 * \text{RealTime}_1 + \text{RealTime}_0 + (1 - (\text{RealTime}_75/75))$$



The first three terms give the integer part of the elapsed time in seconds. The last term is a fractional part that increments in  $1/75^{\text{th}}$  of second. The elapsed live time is calculated similarly to the **RealTime** using the four bytes from **LiveTime\_0** to **LiveTime\_75**. Both real and live elapsed time cannot be modified by the host computer.

**Threshold\_0** and **Threshold\_1** form a 16 bit unsigned integer that represents the low level threshold of the MCA8000A. The threshold is given as a channel number and should not exceed half of the currently set resolution of the MCA8000A.

The **Flags** byte contains information about various parameters of the MCA. This byte can be modified from the host PC. The **Flags** bit assignments are as follows:

- bit **Flags.2** to bit **Flags.0**: ADC Resolution (Spectrum Length :000b = 16k, 001b = 8k, 010b = 4k, 011b = 2k, 100b = 1k, 101b = 0.5k, 110b = 0.25k )
- bit **Flags.3** : Timer flag ( 1 = Live, 0 = Real )
- bit **Flags.4**: Start/Stop (1 = Start, 0 = Stop )
- bit **Flags.5**: Security Flag (1 = MCA Protected, 0 = MCA Public )
- bit **Flags.6**: Main Battery Type (1 = NiCad , 0 = Alkaline,)
- bit **Flags.7**: Backup Battery Condition (1 = Bad, 0 =OK, )

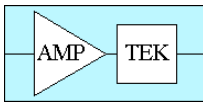
**Checksum** is a byte that contains the sum MOD256 of the 19 status bytes - from **Flags** to the **DataChkSum\_3**.

### **MCA8000A Start Stamp Structure**

The Start Stamp Structure provides the acquisition start time and start date of the current MCA group. Every time the acquisition of the MCA8000A is activated, the computer must set the start time and the start date. The start stamp of the MCA8000A is reported in 8 bytes. Each byte contains a packed BCD number. For more details see the START TIME COMMAND, START DATE COMMAND and GET START STAMP COMMAND.

#### ***8 bit Start Stamp structure***

**Century** (packed BCD 19 or 20) last transmitted byte.  
**Year** (packed BCD 0 to 99)



**Month** (packed BCD 1 to 12)  
**Day** (packed BCD 1 to 31)  
**NotUsed**  
**Hours** (packed BCD 0 to 23)  
**Minutes** (packed BCD 0 to 59)  
**Seconds** (packed BCD 0 to 59) `first transmitted byte.

## MCA8000A Commands

### Command Format

The host computer (PC) controls the actions of the MCA8000A and the data flow from/to MCA8000A using a number of commands. Each command is sent as a packet of 5 bytes. The command bytes are defined as follows:

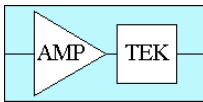
**Byte 0** Command code, this byte indicates the specific action that MCA8000A has to take.  
**Byte 1-3** These 3 bytes contain data that support the command code and/or set various parameters of the MCA8000A (e.g. time, threshold)  
**Byte 4** Check sum byte, the sum of bytes 0 to 3, MOD 256.

**Byte0** is sent first while **Byte4** is the last byte in the command queue.

### SEND DATA AND CHECK SUM COMMAND (Command Code = 0)

This is a command sent to the MCA that causes the MCA to respond by sending data back to the PC.

**Byte 0** 0  
**Byte 1,2** These two bytes form an integer (Byte 1 lower byte) that represent the relative address of the first byte of either the lower word or the upper word of the first channel of the current group to be transmitted from MCA.  
Byte 1,2 word is calculated as  $(\text{channel\#} * 4 + n)$ , where n is variable that is 0 for transfers of the lower word of the channel data, and n is 2 for transfers of the upper word.  
**Byte 3** Must be zero and included in the check sum.



**Byte 4** Check sum byte.

When this command is issued the received **DataChkSum** is the sum MOD  $2^{16}$  of all bytes of the channel data that were transmitted from the MCA to the computer at the last data exchange preceding the transmission of the status structure. This variable may be used as a check sum of the transmitted channel data.

### **SEND DATA, GROUP AND S/N COMMAND (Command Code = 16)**

This is a command sent to the MCA that causes the MCA to respond by sending data back to the PC.

- Byte 0:** 16
- Byte 1,2** These two bytes form an integer (Byte 1 lower byte) that represent the relative address of the first byte of either the lower word or the upper word of the first channel of the current group to be transmitted from MCA. Byte 1,2 word is calculated as  $(\text{channel\#} * 4 + n)$ , where n is a variable that is 0 for transfers of the lower word of the channel data, and n is 2 for transfers of the upper word.
- Byte 3** Do not care, but must be non-zero and included in the check sum.
- Byte 4** Check sum byte.

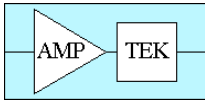
When this command is issued, the received **DataChkSum\_0** is the current group number (byte). **DataChkSum\_2** and **DataChkSum\_3** form a unsigned integer that represent the serial number of the MCA devices connected to the computer.

### **GET START STAMP COMMAND (Command Code = 48)**

This is a command sent to the MCA that causes the MCA to send back the eight bytes of the **Start Stamp Structure**.

- Byte 0:** 48
- Byte 1,2,3** do not care, but must be non-zero and included in the check sum.
- Byte 4** check sum byte.

The MCA responds to this command by sending the **Start Stamp Structure**



first transmitting **Seconds. Century** is transmitted last. For details on synchronizing the data flow from MCA to the computer refer to **Data Exchange Sequence** section of this document.

### **SET START DATE COMMAND (Command Code = 19HEX or 20HEX)**

This is a command sent to the MCA that causes the MCA to store the start date in the **Start Stamp Structure**. . MCA will accept this command only if the MCA is not acquiring data, otherwise the command will be ignored.

- Byte 0:** 19HEX for the 20<sup>th</sup> century date. 20HEX for the 21<sup>st</sup> century date.  
**Byte 1,2,3** **Year, Month, and Day**, these three bytes represent packed BCD numbers. See **Start Stamp Structure**.  
**Byte 4** check sum byte.

### **SET START TIME COMMAND (Command Code = 37)**

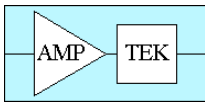
This is a command sent to the MCA that causes the MCA to store the start time in the **Start Stamp Structure**. MCA will accept this command only if the MCA is not acquiring data, otherwise the command will be ignored.

- Byte 0:** 37  
**Byte 1,2,3** **Hours, Minutes, and Seconds**, these three bytes represent packed BCD numbers. See **Start Stamp Structure**.  
**Byte 4** check sum byte.

### **SET GROUP COMMAND (Command Code = 17)**

This is a command that sets the MCA memory group. will accept this command only if the MCA is not acquiring data, otherwise the command will be ignored.

- Byte 0:** 17  
**Byte 1** 0  
**Byte 2** Group Number  
**Byte 3** do not care, but must be non-zero and included in the check sum.  
**Byte 4** check sum byte.



The Group Number depends on the ADC resolution and can be: 0 or 1 for 16k, 0 to 3 for 8k, 0 to 7 for 4k, 0 to 15 for 2k, 0 to 31 for 1k, 0 to 63 for 0.5k, 0 to 127 for 0.25k.

### **SET MCA LOCK COMMAND (Command Code = 117)**

This is a command that sets the MCA security lock number. The two byte (int) lock number is used to unlock the MCA after power up or after long periods of taking data in local mode. Refer to the users manual for details about locking and unlocking the MCA.

<b>Byte 0:</b>	117
<b>Byte 1,2</b>	Lock number.
<b>Byte 3</b>	do not care, but must be non-zero and included in the check sum.
<b>Byte 4</b>	check sum byte.

### **CONTROL COMMAND (Command Code = 1)**

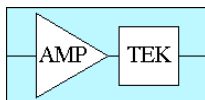
This command configures the MCA8000A according to the bits in the **Flags** byte. This command also sets the threshold. To take a specific action the Flags byte and the Threshold setting must first be retrieved from MCA8000A using the SEND DATA COMMAND. Once Flags and Threshold values are obtained they can be modified and send back to the MCA8000A. For example, if the threshold does not have to be changed, then send as byte 2 and byte 3 the threshold word obtained from MCA8000A Status structure..

<b>Byte 0:</b>	1
<b>Byte 1</b>	Control flags (Flags byte)
<b>Byte 2</b>	lower byte of the threshold value.
<b>Byte 3</b>	upper byte of the threshold value.
<b>Byte 4</b>	check sum byte.

### **PRESET TIME COMMAND (Command Code = 2)**

This command sends the preset time value (acquisition time) to the MCA8000A. This is either live time or real depending on timer flag (bit Flags.3 of the Flags byte).

<b>Byte 0</b>	2
<b>Byte 1</b>	First (lowest) byte of the preset time
<b>Byte 2</b>	Second byte of the preset time



**Byte 3** Third byte of the preset time

**Byte 4** check sum byte.

NOTE: This command does not affect the timer type (live or real) that controls the acquisition.

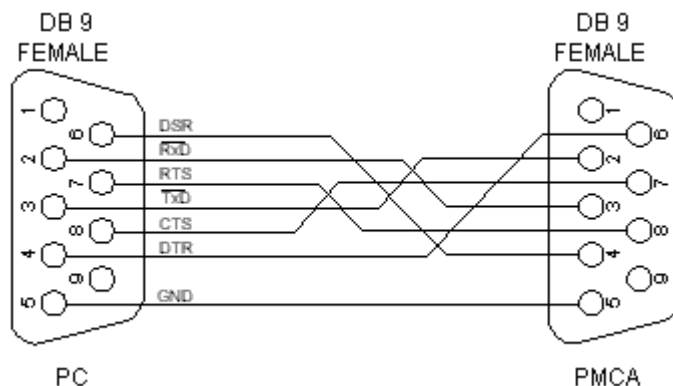
**DELETE COMMAND (Command Code = 5)**

Deletes either data, time, or both time and data.

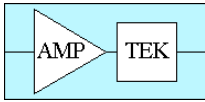
- Byte 0:** 5
- Byte 1** 1 = delete data, 0 = do not delete data
- Byte 2** 1 = delete time, 0 = do not delete time
- Byte 3** do not care, but must be non-zero and included in the check sum.
- Byte 4** check sum byte.

**Data Exchange Sequence**

The data transfer between the MCA8000A and the host PC is accomplished via asynchronous serial connection (RS 232). The serial controller data format is: 8 data bits, one stop bit, one parity bit. MCA8000A sends a parity bit set to zero. It is recommended to set the parity mode of the serial controller to "Space Parity". Even or Odd Parity settings may be used. In this case, however, the parity error generated by the serial controller must be ignored. MCA8000A baud rate is set by the host software (See BAUD RATE SETTING MODE). The wiring diagram of the serial cable connecting PC and MCA8000A is shown in this figure.



Further in the text the signal nomenclature of the RS-232 PC port is used. The names of the signal lines are referred to the PC connector. Normally the serial port Receive Data (RxD)



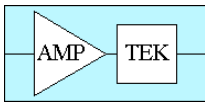
and Transmit Data (TxD) signals of the IBM compatible PC are inverted. Similarly the RxD and TxD lines of the MCA8000A are inverted.

The PC acts always as a master while the MCA responds always as a slave. The PC port output signal RTS is used by the PC to indicate the direction of the data transfer. RTS = HIGH indicates data transfer from PC to MCA (mode SEND), while RTS = LOW sets the data transfer direction from MCA to the PC (mode RECEIVE). The MCA constantly monitors the PC RTS line and responds accordingly. The action of the PC depends on the selected mode: SEND or RECEIVE. The DTR and DSR signals are used to synchronize the data streams from/to MCA8000A.

### **SEND MODE**

In this mode PC sends commands to the MCA. Each command sent to MCA consists of five bytes (see earlier section). The sequence to send a command is as follows:

1. Read and save the state of the DSR signal (DSR bit of the modem status register MSR of the UART). Assume DSR is stored as variable OLDDSR.
2. Clear DTR signal of MCR
3. Set RTS signal high (RTS bit of the modem control register MCR of the serial port) It is recommended that 2 and 3 are done simultaneously by a single output to the MCR register..
4. Point to the first byte of the command sequence.
5. Reset and start a Time-Out Timer (Watch Dog). Allow 110 to 165 ms (2 to 3 clock ticks on IBM compatible PC) for sending a single byte.
6. Check if Transmit Holding Register is empty. If not wait until it becomes empty or exit with time-out error (10).
7. Check if the DSR signal has been complemented. If DSR is different than OLDDSR. Then store DSR in OLDDSR and send the current byte from the command queue. Point to the next command byte in the command sequence and go to point 5 above. If DSR does not change for the time-out period exit with time-out error (10).
8. After the last byte of the command has been sent wait until DSR changes its state indicating that the MCA has received the last byte, the check sum of the command bytes is OK, and the command code is recognizable. Check the time-out timer. If time out expired exit with time-out error (10).
9. If all of the command bytes have been sent successfully read (dummy) the Receive Buffer of the UART and then clear RTS. If the last command sent is a command that prepares MCA for

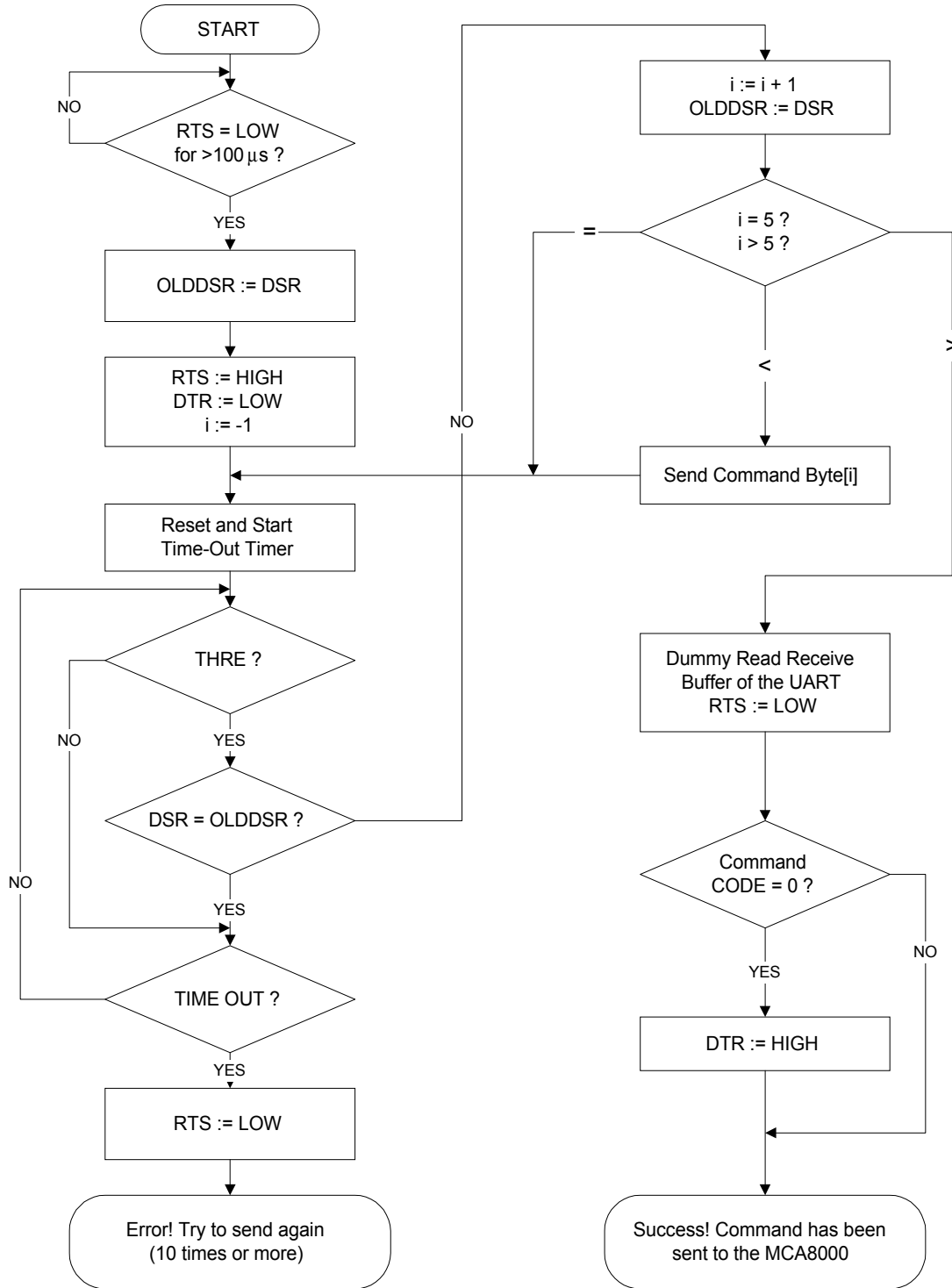
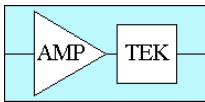


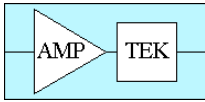
sending data (SEND DATA COMMAND, SEND DATA, GROUP AND S/N COMMAND, GET START STAMP COMMAND), set high the DTR bit in MCR; otherwise keep DTR = LOW. Allow 100 to 200  $\mu$ s or more before sending next command (see **NOTE**).

10. If time-out expires set RTS = LOW. Allow 100 to 200  $\mu$ s before sending next command.

**NOTE** A command may not be sent successfully due to various reasons. The MCA8000A might be physically disconnected (missing, bad or wrong serial cable), turned off or improperly configured (baud rate). In all these cases multiple attempts to send the same command will fail through. Under some circumstances, particularly after the delete command has been sent, the MCA may need more time to respond (up to 2 seconds) and few subsequent attempts to send the command may be unsuccessful. It is, therefore, desirable to try at least 10 times sending a single command, before deciding that something is wrong with the MCA connection. It is important to clear RTS for at least 100  $\mu$ s between the consequent attempts to send the same command.

The algorithm for sending a command to the MCA8000A is illustrated in this flowchart.





## **RECEIVE MODE**

PC enters this mode after SEND DATA COMMAND. The sequence to receive a byte is as follows:

1. Check if there is a byte received by the serial port or if interrupts are used an interrupt is generated when a byte has been received.
2. Get and store the byte from the serial port.
3. If the byte received is not the last expected byte from the MCA8000A alternate DTR indicating that the PC is ready to receive data.
4. If the byte received is the last expected byte do not alternate DTR and disable interrupts in case of interrupt transfer.

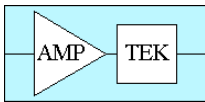
The MCA8000A always sends first the 20 bytes of the MCA8000A status structure. If the check sum of the status bytes is OK the PC may store the status information. After the status information has been received the MCA8000A starts sending the channel data in packets of two bytes - the lower or the upper word (see Channel Data and Send Command). First is send the lower byte of the channel data word. The PC can interrupt the data transfer from MCA8000A to the PC at any time by setting the RTS line HIGH.

**NOTE:** MCA keeps sending data as long as PC enables the data transfer from MCA to the PC by keeping RTS LOW and alternating the DTR signal. Therefore, it is the PC responsibility to monitor and control the data transfer between the MCA8000A and the PC.

## **BAUD RATE SETTING MODE**

MCA8000A sets its default baud rate upon power on to 4800 bps. To set the desired communication baud rate (greater than 19.2 kbps is recommended) a procedure for setting the baud rate must be performed. The maximum baud rate is 115.2 kbps, all other baud rates that are obtained by dividing 115.2 by an integer number (divisor) can be used with MCA8000A. The divisor is a byte that is equal to 1 for the 115.2 kbps. To set the baud rate either of SEND DATA COMMAND or SEND DATA, GROUP AND S/N COMMAND must be issued. In this case Byte 3 of the command bytes must contain the divisor value corresponding to the desired baud rate. The serial port must be configured for 4800 bps. The sequence to set the baud rate is as follows:

1. Set the UART for 4800 bps and store the desired baud rate divisor in Byte 3 of the command sequence bytes.

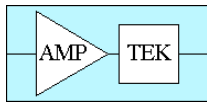


2. Read and save the state of the DSR signal (DSR bit of the modem status register MSR of the UART). Assume DSR is stored as variable OLDDSR.
3. Simultaneously set RTS and DTR signals high (RTS and DTR bits of the modem control register MCR of the serial port)
4. Point to the first byte of the command sequence.
5. Reset and start a Time-Out Timer (Watch Dog). Allow 110 to 165 ms (2 to 3 clock ticks on IBM compatible PC) for sending a single byte.
6. Check if Transmit Holding Register is empty. If not wait until it becomes empty or exit with time-out error (10).
7. Check if the DSR signal has been complemented. If DSR is different than OLDDSR. Then store DSR in OLDDSR and send the current byte from the command queue. Point to the next command byte in the command sequence and go to point 5 above. If DSR does not change for the time-out period exit with time-out error (10).
8. After the last byte of the command has been sent wait until DSR changes its state indicating that the MCA has received the last byte, the check sum of the command bytes is OK, and the command code is valid. Check the time-out timer. If time out expired exit with time-out error (10).
9. If all of the command bytes have been sent successfully read (dummy) the Receive Buffer of the UART, set the UART for the desired baud rate and then clear RTS. Set high the DTR bit in MCR. Allow 100 to 200  $\mu$ s or more before sending next command
10. If time-out expires clear RTS and DTR. Allow 100 to 200  $\mu$ s before new attempt to set the baud rate.

### **MCA8000A REMOTE POWER ON**

The MCA8000A may be powered on remotely by a host computer through the RS-232 interface. The Power-On interface utilizes the host computer's "Data Terminal Read" (DTR) and "Request to Send" (RTS) outputs to generate a control signal in the MCA. To power the MCA on:

1. Hold RTS high (>4V).
2. While RTS is high, drive DTR with an approximate square wave:
  - Frequency: 1kHz to 200 kHz
  - Duration: >50ms
  - Amplitude: >4Vpp



This can be accomplished in software by toggling the DTR for 50 ms while RTS is high.

**Power Down:**

In order to save battery life, the MCA8000A has an automatic Power-Down feature. It will automatically Power-Down if it is not acquiring data and is not communicating with the host computer for a period of approximately 3 minutes.

When in battery mode and not acquiring data, in order to avoid automatic Power-Down, the customer written software should communicate with the MCA8000A at least once every 2 minutes.