

AVR Data Recorder / Logger

Version 1.02

The AVR Data Recorder is a versatile programmable device that can be used to capture and store analog data for retrieval at a later date.



The intent of this writing is to present the reader with various degrees of details on the *AVR Data Recorder*; from a quick list of features for the impatient qualifying engineer to the creative programmer with the desire to extend beyond simplistic data recoding functionality.

AVR Data Recorder / Logger

Version 1.02

Contents

FEATURES AT A GLANCE	4
1 GETTING STARTED	5
2 DATA ACQUISITION AND CONTROL MODULES.....	5
2.1 ANALOG INPUT CHANNELS.....	5
2.2 TRUE-RMS READING	5
2.3 CURRENT SENSE INPUT	5
2.4 ANALOG COMPARATOR INPUT.....	5
2.5 DIGITAL OUTPUT.....	6
3 INTERNAL HARDWARE	6
3.1 THE AVR PROGRAMMABLE CONTROLLER.....	6
3.2 PROGRAMMABILITY - THE ATTOBASIC INTERPRETER	6
3.3 ON-CHIP PROGRAM STORAGE.....	7
3.4 ANALOG-TO-DIGITAL CONVERSION	7
3.5 ANALOG GAIN AND ATTENUATION STAGE.....	7
3.6 CURRENT SENSE	8
3.7 ANALOG COMPARATOR.....	8
3.8 PULSE WIDTH MODULATOR.....	8
3.9 VIRTUAL COMMUNICATIONS PORT USING USB.....	9
3.10 INTERNAL LI-ION RECHARGEABLE BATTERY	9
3.11 SELF-START SWITCH.....	9
3.12 ON/OFF SWITCH	9
3.13 DTR-RST SWITCH	9
4 PROGRAMMING THE AVR DATA RECORDER	10
4.1 GENERAL USE	10
4.2 ATTOBASIC SYNTAX.....	10
4.3 ENTERING COMMANDS AND PROGRAMS	11
4.4 BASIC DATA CAPTURE AND DISPLAY.....	12
4.5 BASIC DATA CAPTURE AND STORAGE	12
4.6 DATA RETRIEVAL.....	12
4.7 PROGRAM MEMORY AND NON-VOLATILE PROGRAM STORAGE	13
4.8 NON-VOLATILE DATA FILE STORAGE.....	13
4.9 RENUMBERING PROGRAM LINE NUMBERS	14
4.10 RECOVERING THE DEFAULT CAPTURE PROGRAM	14
4.11 SETTING THE ANALOG CHANNEL GAINS.....	14
4.12 SETTING THE TRUE-RMS CHANNEL	14
4.13 USING THE ANALOG COMPARATOR.....	14
4.14 DIGITAL OUTPUT.....	16
4.15 REAL-TIME COUNTER	16
4.16 LOW-POWER SLEEP UNTIL EVENT	17
4.17 EXTENDING THE TESTING CAPABILITY OF THE IF-THEN STATEMENT	18
5 EXAMPLE PROGRAMS	19
5.1 EXAMPLE 1 - BASIC 2-CHANNEL WITH REAL-TIME COUNTER.....	19
5.2 EXAMPLE 2 - ANALOG COMPARATOR TRIGGERING.....	20
5.3 EXAMPLE 3 - SWEEPING VREF TO DETECT AN INPUT VOLTAGE	20
5.4 EXAMPLE 4 - CHARGE TIME OF A CAPACITOR - METHOD 1	20
5.5 EXAMPLE 5 - CHARGE TIME OF A CAPACITOR - METHOD 2.....	21
6 SPECIFICATIONS	21

AVR Data Recorder / Logger

Version 1.02

6.1	ANALOG TO DIGITAL CONVERTER	21
6.2	ADC REFERENCE VOLTAGE GENERATOR	22
6.3	ANALOG INPUTS	22
6.4	ANALOG COMPARATOR AND PROGRAMMABLE REFERENCE GENERATOR	22
6.5	TRUE RMS TO DC CONVERTER	22
6.6	CURRENT SENSE INPUT	22
6.7	DIGITAL OUTPUT.....	23
6.8	NON-VOLATILE MEMORIES	23
6.9	INTERNAL TIME BASE	23
6.10	CALIBRATION OF THE INTERNAL TIME BASE	23
7	SUMMARY	23

AVR Data Recorder / Logger

Version 1.02

Features at a glance

- Processing core is an ATMEL ATmega328p running at 8MHz.
- Versatile; can be used as a stand-alone data capture and recording device or connected to a USB serial console for real-time display and capture of data.
- Portable; Internal Li-Ion rechargeable battery runs up to 14 days on a single charge.
- Two (2) channels of analog input;
 - ✓ 8-bit analog resolution (9.8 mV / bit @ 1x gain, post-attenuator).
 - ✓ Each channel is bipolar (+/-) with respect to ground, reading is positive only.
 - ✓ Software selectable input attenuator for a range of 2.5volts, 25 volts and 250 volts.
 - ✓ Software selectable input amplifier extends sensitivity to 250 millivolts per channel.
- One (1) channel of True-RMS reading, software selectable between either analog channel (post-attenuator).
- Current sensing input with a range of 0 to 250ma, 0.5Ω series resistance, externally extensible.
- Analog comparator input;
 - ✓ Comparator source is software selectable between dedicated input with fixed 0 to 25 volt input range or either analog channel (post-attenuator).
 - ✓ 8-bit programmable reference.
- Digital output for control of external devices;
 - ✓ Open-drain "low-side" N-MOSFET solid-state switch with 175 mΩ "on" resistance can switch up to 60 volt loads with 1.5 amps of continuous current.
 - ✓ DDS (direct digital synthesis) for toggling at a specific frequency (1 Hz resolution, 1 Hz minimum frequency).
- 64K bytes of internal data storage capability in non-volatile memory.
- Preprogrammed "out of the box" to record a 24-bit time stamp and both analog channels every 60 seconds for data recording up to 218 hours.
- "Data" LED indicates data file access.
- Programmable;
 - ✓ Uses AttoBASIC programming language, a Tiny-BASIC with an enhanced command set.
 - ✓ Program can be saved in internal non-volatile memory for execution.
 - ✓ Switch-selectable "self-start" feature enables immediate user program execution upon power-up.
 - ✓ Programmability creates a stand-alone intelligent capture device with decision-making and data manipulation capabilities.
- AttoBASIC features;
 - ✓ User programmable using BASIC-like command syntax.
 - ✓ 8-bit data handling with 26 variables.
 - ✓ Arithmetic, logical and relational operators allow manipulation of data before storage.
 - ✓ 32-bit Real-time counter for time-stamping of data in resolutions of 1 ms, 10 ms and 100 mS.
 - ✓ Extensions to language for direct control of the internal hardware, including the on-chip analog-to-digital converter, gain settings for each analog channel, analog comparator, analog comparator's DAC reference generator, MOSFET digital output and the internal non-volatile "data file" storage memory.
 - ✓ Direct data file access allows data dump in ASCII text CSV format for importation into EXCEL™ and other data-processing applications. Number of data fields per record is user programmable from 1 to 255 fields.

AVR Data Recorder / Logger

Version 1.02

- USB interface via *Virtual Communications Port* supported by all major operating systems (WINDOWS™, MAC OS-X™, and Linux).
- Internal boot-loader allows AttoBASIC updates or user-custom application software to be uploaded to the AVR's internal FLASH memory.

1 Getting started

The *AVR Data Recorder* comes pre-loaded with a program in non-volatile memory. Upon power-up, if the "Self-Start" switch is set to the "Self-Start" position, it will begin executing the program and recording data. The program is designed to record a 24-bit time stamp and data from both analog channels every sixty (60) seconds, which will save data for up to 218 hours.

Both channels are set to attenuate by 10x accommodating an input range of 0 to 25.0 volts. The time-stamp has a resolution of 0.1 seconds. The data is stored 5 bytes per record, where bytes *A*, *B* and *C* are the 24-bits of the RTC, byte *D* is Analog Channel 0 data byte and *E* is Analog Channel 1 data byte. The RTC result will be in tenths of seconds. Reference section 4.15 for more information on using the real-time counter and calculating its actual value from the stored data.

2 Data Acquisition and Control Modules

2.1 *Analog Input Channels*

The *AVR Data Recorder* has two (2) analog input channels, "CH0" and "CH1", which are each D.C. coupled to separate input attenuators. The maximum input to the analog converter is 2.5 volts D.C. Thus the input attenuators are used to insure that the input voltage to the AVR is maintained within that range. The input attenuators are programmable to unity (1x) gain, 10x gain, 0.1x gain and 0.01x gain, which yields input ranges of 0 to 2.50 volts, 0 to 250 millivolts, 0 to 25.0 volts and 0 to 250 volts, respectively. Upon power-up, the attenuators are set to 0.01x (250 volt) for safety.

Although the analog input channels can accept bipolar voltages, the analog converter can only read positive voltages. Therefore, any negative voltages will be clamped and the analog converter will read "0" on that particular channel.

2.2 *True-RMS Reading*

The *AVR Data Recorder* has a True-RMS to DC converter that can be connected to analog input channel 0 or 1. The connection is post-attenuator so the input can be up to 250 volts. Since the True-RMS converter is always connected to one of the two input channels, a reading will always be available. Also note that the True-RMS converter will read the D.C. voltage on the selected input channel.

2.3 *Current Sense Input*

The *AVR Data Recorder* has a current sense input. The input range is 0 to 250ma D.C. through a 0.5Ω series resistance. The maximum voltage across the series resistance is ±18 volts with respect to the Data Recorder's signal ground.

2.4 *Analog Comparator Input*

The *AVR Data Recorder* has an analog comparator input that can be used to sense whether the

AVR Data Recorder / Logger

Version 1.02

input is higher or lower than the internal programmable reference. The input range is 0 to 25 volts D.C. on the dedicated input but either analog channel can be selected as the source allowing the use of that channel's attenuator/gain stage.

The analog comparator's reference input is programmable between 0 and 3 volts in 255 steps, yielding an effective resolution of 118 millivolts per step when used with the dedicated input, more or less when used with the analog inputs.

Note that if one needs to monitor an A.C. voltage, a simple full or half-wave rectifier stage can be implemented and its output can be connected to the analog comparator's input, maximum voltage being respected, of course.

2.5 *Digital Output*

The *AVR Data Recorder* has a digital output that consists of an N-channel MOSFET transistor. The maximum drain-to-source voltage is 60 volts with a continuous drain current of 1.5 amps. The source is internally connected to the *AVR Data Recorder's* ground.

The digital output pin can also be toggled at a 50% duty cycle using a DDS (direct digital synthesis) signal. The DDS signal has a resolution of 1 Hz with the lowest frequency being 1 Hz. The maximum frequency is load dependent but is typically limited to approximately 200 Hz.

3 Internal Hardware

3.1 *The AVR Programmable Controller*

The heart of the *AVR Data Recorder* is an ATMEL ATmega328P microcontroller running at 8 MHz. The ATmega328P has 32K bytes of internal program FLASH memory, 2K bytes of internal RAM and 1K bytes of internal EEPROM. It has several built-in peripherals, including a 10-bit analog-to-digital converter, an 8-bit PWM (pulse width modulator), an analog comparator and multiple 8 and 16-bit timers. They are implemented as described below.

3.2 *Programmability - The AttoBASIC Interpreter*

The ATmega328P is running an internal application firmware called "AttoBASIC". AttoBASIC is a "byte-wide" (8-bits) Tiny-BASIC-like interpreter originally written by Dick Cappels (<http://cappels.org>) for the ATMEL AT90S2313 back in the year of 2002. It was subsequently ported to an ATMEL AT90S8515 and then to an ATMEL ATmega163 within the following year. In the year of 2011, K. Scott Vitale (kvs_prj@gmx.com) came across AttoBASIC and realized its value as a tool for quick testing and qualifying of circuits and peripherals. Mr. Vitale ported AttoBASIC to the ATMEL ATmega88/168 and enhanced it with capabilities including DDS, input capture, I2C and SPI control. Shortly thereafter, AttoBASIC was ported to the ATMEL ATmega32U4 and USB serial I/O was added. In the summer of 2012, Mr. Vitale, inspired by conversations with Mr. Cappels of a portable data recorder project, implemented new commands to support an external serial EEPROM for use as a data file, a real-time counter for time-stamping and lower-power operation for battery conservation.

In its current incarnation, AttoBASIC is the underlying framework upon which a user of the *AVR Data Recorder* has complete control over the associated hardware. Within a few minutes and with a handful of commands, a simple data recorder program can be written and executed. With twenty-six (26) variables, arithmetic, logical and relational operators, data can be

AVR Data Recorder / Logger

Version 1.02

compared, manipulated and stored or used to control the digital output. With looping and control structures, more complex data capture and manipulation can be performed.

Once data is captured and stored, it is normally retrieved for manipulation and post-processing, typically on a higher-performance computing platform. The *AVR Data Recorder*, via an AttoBASIC command, can dump the contents of its data file through the USB port to a host. The data is dumped in CSV text format with a selectable number of data fields per record for importation into EXCEL™ or some other data-manipulation application.

Once written, programs can be saved in the AVR's internal EEPROM. The selectable "self-start" feature allows the program to be loaded from the EEPROM and executed upon power-up without the need for further user intervention, thus making the *AVR Data Recorder* a stand-alone intelligent capture device.

Included with AttoBASIC is an "ARDUINO compatible" boot-loader. When invoked, updates to AttoBASIC firmware can be accomplished. The boot-loader also allows one to completely replace the *AVR Data Recorder's* firmware with a customized application, if so desired.

3.3 *On-chip program storage*

The AVR has 2K of internal RAM (volatile memory) and 1K of internal EEPROM (non-volatile memory). The RAM is used for program storage, variable storage and other internal storage. The EEPROM is only used to permanently save a program for later execution when the *AVR Data Recorder* is performing some task, such as data capture and store. There are 1012 bytes of user-program memory available.

3.4 *Analog-to-Digital Conversion*

The AVR's analog-to-digital converter, hereinafter referred to as "ADC", is a 10-bit successive approximation converter. Although it can convert with 10-bit accuracy, the *AVR Data Recorder* averages sixteen (16) 10-bit readings and rounds them to a single 8-bit value, which can then be manipulated and stored by the AttoBASIC interpreter. The AVR's ADC is driven by one of four (4) selectable input channels. Input 0 and 1 are driven by the analog input channels "CH 0" and "CH 1" respectively. Input 2 is driven by the internal True-RMS to DC converter and input 3 is driven by the current sense input.

3.5 *Analog Gain and Attenuation Stage*

The *AVR Data Recorder's* two analog input channels are each directly connected to a gain and attenuator stage before being routed to the AVR's ADC. Each channel has its own separate gain/attenuator stage, which is user programmable for a gain of 1x or 10x or an attenuation of 10x or 100x (0.1x gain and 0.01x gain respectively).

Since the ADC uses a 2.50 volts reference, the "bit weight" is 9.8 millivolts per bit, post-attenuator. Therefore, the value obtained from the ADC must be multiplied by the appropriate bit weight to determine the actual value of the input signal based upon the gain or attenuation selected for the measurement. The following table shows the actual bit value based upon the gain/attenuator settings.

AVR Data Recorder / Logger

Version 1.02

<u>Input Source</u>	<u>per bit weight</u>
Ch 0/1 @ 0.01x	980 mV
Ch 0/1 @ 0.1x	98 mV
Ch 0/1 @ 1x	9.8 mV
Ch 0/1 @ 10x	0.98 mV

TABLE 1

To calculate the actual value of the reading obtained, multiply the value obtained by the bit weight. For example, a reading of 125 from analog channel "0" set for a gain of 1x would mean that 1.23 volts was applied to the input.

If accuracy is not as important, one may use the value obtained directly from the ADC as a one-to-one ratio of the input voltage, which will bear a +2% error. In other words, if a reading on the 250 volt scale is "125" then it can be considered as "125 volts" even though it is actually 123 volts.

3.6 Current Sense

The *AVR Data Recorder* has an analog input that is specifically designed to sense D.C. current in the range of 0 to 250 mA. The internal sense resistance is 0.5Ω , which produces a maximum voltage drop of 125 millivolts at 250 mA. Although the current sense input measures a single polarity, it can be used with an A.C. signal, which will have the effect of being rectified but not filtered. The bit weight for the current sense input is 0.98 ma per bit.

3.7 Analog Comparator

The AVR's analog comparator produces a logical "1" or "0" depending on the relative state of the two inputs. The analog comparator's "compare" input can be driven through the dedicated input, which has a 10x voltage divider with an input range of 0 to 25 volts D.C. or either analog channel, having selectable gain or attenuation.

The analog comparator's "reference" input is driven by an adjustable analog voltage produced by the Pulse Width Modulator (PWM) as further described.

The bit weight for the analog comparator's dedicated input is 98 mV per bit. If using analog channel 0 or 1 then the bit weight is the same as illustrated in "TABLE 1" above.

3.8 Pulse Width Modulator

One of the AVR's timers is used as a pulse width modulator, hereinafter referred to as "PWM". The PWM output drives a simple R/C filter network creating a rudimentary digital-to-analog converter. The analog voltage derived is relative to the duty cycle as it is set between 0 and 255 (8 bits) and the internal power supply rail of 3 volts. Thus the effective resolution is 118 millivolts per bit. This analog voltage is connected to the analog comparator's "reference" input, which provides a programmable reference voltage for the analog comparator's "compare" input to be compared to.

Like the ADC, the reference voltage generator has a bit weight, which is dependant on the input selected and the gain or attenuator settings as defined in the following table:

AVR Data Recorder / Logger

Version 1.02

<u>Input Source</u>	<u>per bit weight</u>
Dedicated ACO	118 mV
Ch 0/1 @ 0.01x	1.18 V
Ch 0/1 @ 0.1x	118 mV
Ch 0/1 @ 1x	11.8 mV
Ch 0/1 @ 10x	1.18 mV

TABLE 2

To calculate the actual value to be passed to the PWM (using the *ACR* command), divide the desired reference voltage by the bit weight from "TABLE 2". For example, if a 1.50 volt reference is desired for the dedicated ACO input, then $(1.20\text{V} / 0.118\text{V}) = 10.1$, rounding down to 10. Therefore, the actual voltage produced for the reference will be $(10 \times 0.118\text{V}) = 1.18\text{V}$.

3.9 Virtual Communications Port using USB

The *AVR Data Recorder* uses a USB mini-B connector to communicate to a host computer through. The USB port is recognized by the host computer as a "*Virtual Communication Port*", thus it is supported on most major computing platforms. The USB port is also used to charge the internal Li-Ion battery, which can be either a computer's USB port or a standalone USB charger.

3.10 Internal Li-Ion Rechargeable Battery

The *AVR Data Recorder* uses a Li-Ion rechargeable battery as a power source. The Li-Ion battery has a capacity of 2200 maH and is charged through the USB port (mini-B type) from a computer USB port or USB charger.

3.11 Self-Start Switch

The *AVR Data Recorder* can execute a saved program upon power-up. When this switch is deselected, AttoBASIC does not execute the saved program and immediately responds with a greeting and prompt.

3.12 On/Off Switch

The *AVR Data Recorder's* On/Off switch disconnects power from the majority of the circuitry while maintaining the connection of the Li-Ion battery to the USB power source for charging. Keep in mind that if the Self-Start switch is active, then upon power-up, any previously saved program will execute. If this is not the intent then any previously recorded data may potentially be over-written when the program executes.

3.13 DTR-RST Switch

The *AVR Data Recorder* can execute a boot-loader to update or replace the firmware. The boot-loader is invoked when the DTR signal is toggled by the host computer's boot-loader application. This switch should be selected when using the boot-loader. However, when this switch is selected, a connection by a host communication application may toggle the DTR line, which initiates a hardware reset of the *AVR Data Recorder*. If the Self-Start switch is also selected then any saved program will be executed without warning, which may result in existing data being over-written without warning. **Caveat: Always deselect the Self-Start switch when connecting to a host computer.**

AVR Data Recorder / Logger

Version 1.02

4 Programming the AVR Data Recorder

While some knowledge of the ATMEL AVR ATmega328P microcontroller is helpful in programming the *AVR Data Recorder*, it is not necessary or required.

4.1 General use

Programming the *AVR Data Recorder* is as simple as;

- plugging the USB port into the computer's USB port,
- installing the drivers (if necessary),
- invoking a terminal emulator (i.e. *HyperTerm* or *RealTerm* under WINDOWS® or *picocom* or *minicom* under Linux) and connecting to the Virtual Communications Port.
- writing and editing a program using either the AttoBASIC resident text editor or a text editor on the host computer,
- downloading the program to the recorder,
- testing for proper operation (and debugging) on the *AVR Data Recorder*.

When you are satisfied that the *AVR Data Recorder* is executing its program to satisfaction, save it to the recorder's internal non-volatile memory. By typing the "*RUN*" command, the *AVR Data Recorder* will execute the user's program. If the "*SELF-START*" switch is set to the "Self-Start" position, the program will automatically execute every time power is applied.

The following subsections detail programming methods and examples. Since AttoBASIC is a Tiny-BASIC, it can either execute commands in immediate mode or run commands embedded in a program.

A list of the available commands (in three-letter format) can always be referenced by issuing the "*HELP*" command at the AttoBASIC prompt.

4.2 AttoBASIC Syntax

The following programming and syntax rules apply:

- All commands and variable names can be either upper case or lower case.
- When embedded in a program, all program lines must start with a number in the range of 1 to 255. Any other values are illegal and will be reported as an error.
- All commands can be shortened to the first three (3) letters of the command. Using only the first three letters is an efficient programming technique that also conserves program memory when larger and more complex programs are required.
- In all cases, no space between the program line number and the command is acceptable.
- In most cases, spaces between commands and their first numerical parameter can be removed as well as spaces between commands and the command separator (";").
- All commands are executed and program lines are stored once the ENTER key is pressed.
- All data values are 8-bit, yielding 256 possible combinations, including zero (0)
- FOR-TO-NEXT loops are limited to one (1) level deep
- GOSUB calls are limited to one (1) level deep
- The "#" (hash or pound symbol) can be used as a comment separator. This is primarily useful when composing programs on a host computer's text editor and will be stripped when uploading to the *AVR Data Recorder*.

For example:

```
10 PRI ADC 0          # SPACES ARE OKAY BUT TAKE UP PROGRAM SPACE
10PRI ADC0            # NO SPACE BETWEEN LINE #, ADC COMMAND AND CHANNEL 0 IS OKAY
```

AVR Data Recorder / Logger

Version 1.02

```
20 PRI ADC 1 ; PRI ADC 3      # SPACES ARE OKAY BUT TAKE UP PROGRAM SPACE
20PRI ADC1;PRI ADC3          # NO SPACES BETWEEN COMMANDS AND COMMAND SEPARATOR ARE OKAY
```

Refer to the AttoBASIC command reference for details of each command and their parameters, if any.

4.3 Entering Commands and Programs

To further fulfill the usefulness of the *AVR Data Recorder*, one needs to have a method to communicate with its on-board central processor. This is accomplished by plugging the *AVR Data Recorder* into a host computer's USB port. Review the following before continuing.

Requirements:

- a host computer with a USB communications port having the ability to support the *Virtual Communications Port* (VCP) protocol. This could be a PC running WINDOWS®, Linux or MAC OS-X®.
- appropriate *Virtual Communications Port* (VCP) drivers, if necessary for your host's OS.
- a terminal emulator program (i.e. *HyperTerm* or *RealTerm* for WINDOWS® or *picocom* or *minicom* for Linux) with file send and capture abilities. Refer to the documentation for your specific terminal emulator program as the use of a terminal emulator program is beyond the scope of this writing.
- a USB Type A to USB Mini-B cable.

AVR Data Recorder settings:

- power switch "on" or "off". If still set to "on", the current program may still be running.
- powered or not, insure the "SELF-START" switch is not set to "Self-Start". Otherwise, the current program stored in non-volatile memory (if any) will be run if when the power is (re-)applied, potentially over-writing some of all of the previously recorded data.
- insure the "DTR-RST" switch is set to "DTR-RST".

Once connected and the host computer detects the *AVR Data Recorder* as a VCP, the terminal host computer's emulator program is invoked and connected to the VCP at 19.2K baud. The *AVR Data Recorder* will likely be reset by the host's terminal emulator upon connection to the VCP. This will cause AttoBASIC to print its sign-on message on the screen, the command prompt character ">" and enter immediate mode awaiting a command.

At this point, the user may begin entering commands or writing programs, following the syntax described in section 4.2. Refer to the AttoBASIC programming reference for the available commands and their syntax (or type *HELP* at the AttoBASIC command prompt).

Programs may be *RUN* and debugged then *SAVE*d to non-volatile memory when their execution is satisfactory. Note that previously *SAVE*d programs may be retrieved from non-volatile memory using the *LOAD* command. Also, if starting a program from scratch, enter the *NEW* command to insure there are no unwanted program lines in program memory.

Alternatively, a user may write a program using any text editor application that saves its files in ASCII data format. The host's terminal emulator can then be used to upload the program into the *AVR Data Recorder* for further testing and evaluation.

When using the terminal emulator to upload a program file, be sure to set the "character delay" to 15 ms and the "line delay" to 50 ms, otherwise AttoBASIC will not accept your program properly.

AVR Data Recorder / Logger

Version 1.02

4.4 Basic Data Capture and Display

AttoBASIC supports real-time direct access and readout of the four (4) analog channels using the "ADC [x]" command, where "x" is the channel number, 0 to 3. Data can be captured and printed to the serial console and/or stored in the data file for retrieval at a later.

The ADC command can be used with the *PRINT* command, the value returned can be assigned to a variable or the value can be used in a conditional statement. For example:

```
PRI ADC 0           # PRINT THE VALUE OF ANALOG CHANNEL 0
A:= ADC 3           # ASSIGN THE VALUE OF ANALOG CHANNEL 3 TO THE
                    # VARIABLE A
10IF ADC0 > 128 THEN PRI "Too HIGH!" # IF THE READING ON CHANNEL 0 IS GREATER THAN
                    # 128 THEN PRINT THE MESSAGE
```

In addition, the value of the analog comparator can be captured using the "ACO" command. For example:

```
PRINT ACO           # PRINT THE VALUE OF THE ANALOG COMPARATOR ("0" OR "1")
A:= ACO             # ASSIGN THE VALUE OF THE ANALOG COMPARATOR TO THE
                    # VARIABLE A
10 IF ACO THEN PRINT ACO # IF ACO = 1 THEN PRINT THE VALUE OF ACO
```

4.5 Basic Data Capture and Storage

With the use of the Data File commands, up to 65,536 bytes (8-bit) of data can be stored in the data file's non-volatile memory. The data is stored in pages of 256 bytes and there are 256 pages available. The data file logging command, "DFL" uses an internal address pointer, which is incremented each time the DFL command is executed. If the maximum address of 65,536 is exceeded, the address counter rolls over to "0". The address pointer cannot be read but can be reset to "0" by executing the "DFX" command. For example, the following program is the basis for a general purpose two-channel analog data recorder that takes samples one (1) second apart:

```
10 DFX; ADG 0 2; ADG 1 1 # RESET POINTER TO 0, SET CH0 GAIN TO 1X AND CH1 TO 0.1X
20 DFL ADC 0; DFL ADC 1 # STORE CH0 AND CH1 TO DATA FILE, AUTO-INCREMENT POINTER
30 SLP 6; GOTO 20        # SLEEP 1 SECOND AND LOOP BACK TO LINE 20
```

4.6 Data Retrieval

Once the data has been stored into the AVR Data Recorder's non-volatile memory, at some point, it needs to be retrieved for further processing. This is accomplished by plugging the *AVR Data Recorder* into a host computer's USB port. Review the requirements and instructions in section 4.3 before continuing.

Once connected and the host computer detects the *AVR Data Recorder* as a VCP, the terminal host computer's emulator program is invoked and connected to the VCP at 19.2K baud. The *AVR Data Recorder* will likely be reset by the host's terminal emulator upon connection to the VCP. This will cause AttoBASIC to print its sign-on message on the screen, the command prompt character ">" and enter immediate mode awaiting a command.

With the use of the Data File commands, data stored in the non-volatile memory can be "dumped" to a host computer through the USB VCP in ASCII CSV format. The "DFD [p] [n]" command prints the data, "n" pages starting at page "p", where 0 ≤ N ≤ 255 and 0 < P ≤ 255. The "DFF [x]" command sets the DFD command's printing format to the number of fields per record defined by "x". The DFD command is usually issued directly to the

AVR Data Recorder / Logger

Version 1.02

AttoBASIC interpreter in immediate mode after the host computer's terminal emulator is set to capture mode. For example:

```
DFI 4                                # 4 DATA VALUES WERE STORED PER SAMPLE SO WE WANT 4
                                      FIELDS PER RECORD
DFD 0 8                              # WE TOOK 2000 SAMPLES SO DUMP 2048 (8 X 256)
```

The default program stores the data as 5 bytes per record; where A is the RTC[2¹⁶] byte, B is the RTC[2⁸] byte, C is the RTC[2⁰] byte, D is Analog Channel 0 data byte and E is Analog Channel 1 data byte. The time-stamp has a resolution of 0.1 seconds and therefore must be converted to an actual time value using the formula $((A * 2^{16}) + (B * 2^8) + (C * 2^0)) * 0.1$. The RTC result will be in tenths of seconds. Reference section 4.15 for more information on using the real-time counter.

4.7 Program Memory and Non-Volatile Program Storage

The *AVR Data Recorder* executes programs from program memory. The program memory is limited to 1012 characters. The ENTER key is stored as one character. Strings normally take up more memory than commands.

When beginning a new program, it is suggested to execute the *NEW* command to clear program memory.

Under most circumstances, a user program will fit into program memory with room to spare. If the user program is more complex and memory conservation is needed, then it is suggested to consult section 4.2 for methods to reduce the program's memory size.

The *AVR Data Recorder* stores programs to and loads programs from non-volatile memory using the *SAVE* and *LOAD* commands.

4.8 Non-Volatile Data File Storage

The *AVR Data Recorder* stores data in a "Data File memory", which is non-volatile. The capacity of the Data File memory is 65,536 data bytes, which means it can hold 65,536 pieces of data.

AttoBASIC accesses the data file using the "Data File" commands as described in the command reference. With these commands, operations such as reading and writing individual locations in the data file memory and initializing the selected pages (or the entire memory) can be performed.

Normally, a program would need to keep track of an address pointer that points to the current (or next) address in the data file to read from or write to. This would require two 8-bit variables to keep track of the "page" and "offset".

In the case of simply capturing and storing data in sequential locations, a most useful command, the "DFL" command, can be implemented, which allows a piece of data to be written to the data file and an internal address counter incremented to point to the next available location.

In some cases, the user program may need to access the internal address pointer, which is stored in RAM at the location defined by the symbol named "DFLaddr", which points to the next available location to be written to. The actual address "DFLaddr" is \$567 ("page") and \$568 ("offset"). The address can be read or written using the *PEEK* and *POKE* statements as illustrated in the following example:

AVR Data Recorder / Logger

Version 1.02

```
PRINT PEEK $5 $67      # PRINT THE CURRENT PAGE #
PRINT PEEK $5 $68      # PRINT THE CURRENT OFFSET WITHIN THE PAGE

POKE $0A $5 $67        # SET THE ADDRESS POINTER PAGE TO $0A (PAGE 10)
POKE 0 $5 $68          # SET THE ADDRESS POINTER OFFSET TO 0
```

Writing to the address pointer can be useful if one wants to segregate the collected data from perhaps a preloaded data lookup table (residing in page "0"). The data in the lookup table might be used to perform calculations upon the data collected before being stored or used as a comparison constant.

4.9 Renumbering Program Line Numbers

There is a command line "renumber" utility for WINDOWS® and Linux provided with the *AVR Data Recorder*. The Linux utility may function on MAC OS-X. The format of the renumber command is the same for both operating systems and takes the form of:

```
RENUMBER [SOURCE FILE] [DESTINATION FILE] [START NUMBER] [INCREMENT]
```

The *source* and *destinations* are mandatory but the *start* and *increment* numbers are optional. If start and increment are left out, the program line number defaults to begin at "5" and increments by "5". If an increment of "5" exceeds program line "255" then the proper increment is automatically determined. If there are more than 255 lines then the program abort with an error message.

4.10 Recovering the Default Capture Program

When the *AVR Data Recorder* is shipped, a data capture and storage program is pre-loaded as a "default". When new programs are written and saved, the default program is over-written. If for any reason, the user wishes to re-load the default program, simply entering the AttoBASIC "LDD" command at the command prompt will clear program memory and insert a copy within. The content of the non-volatile program memory remains intact until the newly loaded default program is saved to it.

4.11 Setting the Analog Channel Gains

As previously described, each analog channel has its own gain/attenuator stage. The "ADG [c] [g]" command is used to set the channel gain. For safety, at power-up the attenuators are set to maximum (250 volt range). For example:

```
10 ADG 0 2; ADG 1 1      # SET CH 0 TO 1X AND CH 1 TO 0.1X
20 DFL ADC 0; DFL ADC 1  # STORE CH 0 AND 1 TO DATA FILE
30 SLP 9; SLP 7; GOTO 20  # SLEEP FOR 10 SECONDS AND LOOP FOR ANOTHER READING
```

4.12 Setting the True-RMS Channel

As previously described, the *AVR Data Recorder* has a True-RMS to DC converter that can read from either analog channel 0 or 1. The "ADS [n]" command accomplishes this task. For example:

```
10 ADG 1 1; ADS 1        # SET CH 1 TO 0.1X, ROUTE CH 1 TO THE TRMS-DC CONVERTER
20 PRINT ADC 2            # PRINT THE TRMS VALUE ON THE CONSOLE
```

4.13 Using the Analog Comparator

As previously described, the *AVR Data Recorder* has an analog comparator with a programmable reference voltage. The input range of the analog comparator is 0 to 25.0 volts

AVR Data Recorder / Logger

Version 1.02

when used with the dedicated input. However, use of the "*ACS [n]*" command allows one to use analog channel 0 or 1 as the source to the analog comparator, providing the flexibility of the channel gain/attenuator settings.

The "*ACO*" command returns a value of "1" or "0" only, where "0" indicates that the input is lower than the reference, and "1" indicates that the input is higher than the reference. The result of the *ACO* command can be printed, stored, used in a conditional statement or assigned to a variable. Assuming input is on the dedicated input, for example:

```
A:= ACO          # ASSIGN THE VALUE OF THE COMPARATOR TO VARIABLE A
PRI ACO          # PRINT THE VALUE OF THE COMPARATOR TO THE CONSOLE
DFL ACO          # STORE THE VALUE OF THE COMPARATOR TO THE DATA FILE

10 IF ACO THEN GOTO 30  # TEST FOR ACO = 1, IF SO, GOTO LINE 30
20 SLP 7; GOTO 10      # SLEEP 2 SECONDS THEN CHECK THE ACO AGAIN
30 PRINT "THRESHOLD REACHED!"  # THRESHOLD REACHED, NOTIFY ON THE CONSOLE
```

The analog comparator's programmable reference can be programmed using the "*ACR [x]*" command, where "x" is between 0 and 255. Since the step size of "x" is dependant upon which input source is used, refer to "TABLE 2" to derive the appropriate value of "x".

It is important to keep in mind that the useful range of "x" is actually from 1 to 211 as any value of "x" above 211 will exceed the range of the input signal.

If one wanted to set a threshold of 12.0 volts on the analog comparator input, then a setting of 12.0 / 0.118 would yield a value of 102. Likewise, to set a threshold of 2.0 volts on analog channel 0 with a gain of 1x would yield a value of (2.0 / 0.0118) = 170. Assuming input is on Channel 0 @ 1x gain, for example:

```
10 ACS 0; ACR 110      # SELECT CH 0, SET REF@1.3V THRESHOLD,
15 DIG 1               # TURN ON THE CHARGER
20 IF ACO THEN GOTO 50  # TEST FOR ACO = 1, IF SO, GOTO LINE 50
30 PRINT ADC 3          # PRINT THE CHARGE CURRENT (MEASURED THROUGH CURRENT
                        # SENSE)
40 SLP 8; GOTO 20       # SLEEP 4 SECONDS THEN CHECK THE ACO AGAIN
50 DIG 0               # THRESHOLD REACHED, TURN OFF CHARGER
60 PRINT "THRESHOLD REACHED!"  # AND NOTIFY ON THE CONSOLE
```

This same function could have also been implemented using one of the analog input channels set to 0.1x gain and comparing the value to "13" (1.3 volts) by replacing line 20 as shown below:

```
20 IF ADC 0 > 12 THEN GOTO 50 # TEST FOR ADC 0 > 12, IF SO, GOTO LINE 50
```

The analog comparator can also be used as an interrupt event source when used with the "*SLP 0*" command. When used in this manner, any change of the analog comparator's state will trigger an interrupt event. The command to enable the analog comparator to be used as an interrupt source is "*ACI 1*" ("*ACI 0*" disables the interrupt). Assuming the signal to compare is on the dedicated input, for example:

```
10 ACS 2; ACR 35; ACI 1  # INSURE ANALOG COMP. INPUT, SET REF@4.1 VOLT THRESHOLD
20 ACI 1; DIG 1; DEL 50  # ENABLE THE ACO INTERRUPT AND CHARGER, DELAY 500MS
30 PRINT ADC 3          # PRINT THE CHARGE CURRENT (MEASURED THROUGH CURRENT
                        # SENSE)
40 RTR; SLP 0           # RESET RTC COUNTER AWAIT FOR ACO TO TRIGGER EXIT FROM
                        # SLEEP
50 RTP; DIG 0; ACI 0     # PRINT THE RTC, TURN OFF CHARGER, DISABLE ACO
```

AVR Data Recorder / Logger

Version 1.02

```
                                INTERRUPTS
60 ACI 0; SLP 1; SLP 1          # CLEAR ANY SPURIOUS INTERRUPTS GENERATED BY ACO
70 PRINT "THRESHOLD REACHED!"    # NOTIFY ON THE CONSOLE
```

4.14 Digital Output

The *AVR Data Recorder* has a single digital output. It consists of an N-MOSFET transistor with a <200 mΩ drain-to-source ON resistance. The drain-to-source breakdown voltage is 60 volts and the continuous drain-to-source current is about 1.5 amps with a peak instantaneous surge of 5 amps. The N-MOSFET's source is connected to the *AVR Data Recorder's* power supply/signal ground. A typical use for this output would be to drive a relay or high-side switch to supply or disconnect power to/from a circuit upon a given event occurring. The "*DIG [x]*" command directly controls the state of the digital output. For example:

```
10 DIG 1; SLP 7; DIG 0          # ENABLE OUTPUT, SLEEP 2 SECONDS THEN DISABLE OUTPUT
```

Another feature of the digital output is the ability to toggle it at a specific frequency using the "*DDS*" command. The frequency resolution is 1 Hz with a minimum frequency of 1 Hz. The maximum frequency is load-dependant and is typically limited to approximately 250 Hz. The *DDS* command uses the X, Y and Z variables as its accumulator set registers. Once the *DDS* frequency is set with these variables, they can be reused for some other purpose. For all practical purposes, if variable X remains "0" then variable Y and Z will contain the BCD equivalent of the frequency covering the range of 0 to 255 Hz. For example:

```
10 X:=0; Y:= $01; Z:= $50      # SET THE DDS REGISTERS TO 150 HZ
15 DDS 1                        # AND ENABLE THE FREQUENCY ON THE DIGITAL OUTPUT
20 SLP 9; Z:=10; DDS 1          # SLEEP 8 SECONDS, SET REGISTERS TO 10 HZ AND RE-ENABLE
```

4.15 Real-time Counter

AttoBASIC supports a 32-bit Real-time Counter (RTC) implementation. The "*RTP*" command can be used to print relative time-stamp information to the console or the data file when logging data. The "*RTR*" command can be used to reset the RTC counter to "0" so as to set a "base-line time reference". For example:

```
10 RTR; RTP                     # RESET RTC COUNT TO 0 AND PRINT IT
20 SLP 6; RTP                   # SLEEP 1 SECOND THEN PRINT IT AGAIN TO SEE WHAT THE ERROR IS
```

The internal resolution of the RTC is 1mS at power-up thus the RTC increments 1000 times per second yielding a duration of 49.71 days before a counter roll-over occurs. The "*RTI [x]*" command allows one to set the increment rate to 1mS, 10mS or 100mS thus the RTC increments 1000, 100 or 10 times per second, yielding a duration of 49.71, 497.1 or 4971 days respectively.

The following example program illustrates simple real-time data-logging to the serial console using the RTC to timestamp the data:

```
10 ADG 0 1; RTI 2               # CH 0 GAIN TO 0.1X, RTC INCREMENT TO 100MS
20 FOR I= 1 TO 10               # 10 ITERATIONS OF LOOP FOR 10 SECONDS DELAY
21 DEL 100                      # 1 SECOND DELAY
22 NEXT
25 RTP                          # PRINT THE RTC AS A TIME-STAMP
30 PRI "CH 0: "; PRI ADC 0      # PRINT STRING THEN VALUE OF CH 0
35 PRI "ACO IS "; PRB ACO       # PRINT STRING THEN STATE OF THE ACO (ON SEPARATE LINES)
40 GOTO 20                      # LOOP FOREVER
```

The lower 8-bits of the RTC can be directly assigned to a variable or if one needs access to

AVR Data Recorder / Logger

Version 1.02

more digits of the RTC, using the PEEK statement will accomplish this as the following example program illustrates:

```
10 A:=RTP                      # "RTP" IS THE SAME AS "PEEK $5 $65"
15 B:=PEEK $5 $64              # RTC REGISTERS AT 0X0562 {MSB} TO 0X0565 {LSB}
20 C:=PEEK $5 $63              # "$" MEANS HEXADECIMAL VALUE FOLLOWS
25 D:=PEEK $5 $62
30 DFL D; DFL C                # LOG ALL 32 BITS TO DATA FILE
35 DFL B; DFL A
40 DFL ADC 0                   # LOG CH 0, 1 AND CURRENT SENSE TO DATA FILE
41 DFL ADC 1
42 DFL ADC 3
50 FOR N= 1 TO 60              # LOOP FOR 60 ITERATIONS
55 SLP 6                       # SLEEP 1 SECOND PER LOOP ITERATIONS
60 NEXT; GOT 10                # EXIT LOOP TO SAMPLE AND RECORD MORE DATA
```

When importing the stored data from the above program into a spreadsheet, for example, the record width would be "7" (using the "DFF 7" command). The 4 byte values comprising the 32-bits of the RTC would be imported in columns A to D. To calculate the RTC's value, a formula such as $= (A1^1 * 2^{24}) + (B1 * 2^{16}) + (C1 * 2^8) + (D1 * 2^0) * ["RTI \text{ value}" \text{ in milliseconds}]$ would calculate the actual time.

If one needs to directly read or modify the RTC registers, the PEEK and POKE command can be used to accomplish this. The following table lists the specific memory locations in which the RTC counter registers are stored (MSB to LSB):

```
$562 - RTC3 (MSB)
$563 - RTC2
$564 - RTC1
$565 - RTC0 (LSB)
```

4.16 Low-Power SLEEP until event

The AVR microcontroller supports a low-power "idle" mode, which lowers the overall power consumption of the AVR microcontroller. *Idle* mode is used when AttoBASIC is awaiting console input in the immediate mode. While running a program, the *idle* mode can be entered using AttoBASIC's "*SLP [x]*" command. The value of "x" ranges between "0" and "9". Upon execution of this command, AttoBASIC enters the low-power sleep mode and awaits a hardware interrupt event. Once the event occurs, sleep mode is exited and program execution continues with the next command (or next program line).

When "*SLP 0*" is executed, the AVR's peripheral clocks are still running. In this way, any supported interrupt source may be used as the event trigger. Proper use of the "*SLP 0*" command requires that the user enable the desired interrupt source(s) before execution of the command. The **POKE** statement allows one to program the desired Interrupt Control and Interrupt Mask Registers to setup the desired interrupt source.

The "*ACI 1*" command is intended to be used with "*SLP 0*" as it initializes and enables the analog comparator's interrupt as an event source. Any source selected by the "*ACS [n]*" command is valid.

In addition to having the ability to use a hardware interrupt as an event source, the *AVR Data Recorder* may also use the AVR's built-in *WATCHDOG timer* as the interrupt source. When

¹ The appropriate row and column are dependant on the location of the data import. The example assumes the data was imported starting at row 1, column A.

AVR Data Recorder / Logger

Version 1.02

invoked in this manner, the WATCHDOG timer's delay is a preset value corresponding to the value of "x" passed with the SLP command (excluding "0"). This allows a pre-determined delay to be executed before exiting the SLP command. The value of "x" is between "1" and "9" as shown in TABLE 3 below.

<u>x =</u>	<u>Approximate Delay</u>
1	0.032 S
2	0.064 S
3	0.125 S
4	0.25 S
5	0.5 S
6	1 S
7	2 S
8	4 S
9	8 S

TABLE 3

Keep in mind that the AVR's watchdog timer is driven by an internal RC oscillator and has an inherent error associated with it, typically +10%. Therefore the delay duration may be inaccurate. For shorter, more accurate delays, use the "*DELAY N*" command, which also takes advantage of the low-power *idle* mode but is limited to durations between 10 milliseconds and 2.55 seconds.

The following sample program illustrates use of the SLP instruction to sample analog channel 0 approximately every 10 seconds and print the results to the console:

```
10 ADR 1           # SELECT EXTERNAL VREF SOURCE
20 SLP 9; SLP 2     # SLEEP FOR 8 + 2 SECONDS (SEQUENTIAL SLP N ARE OKAY)
30 PRINT ADC 0      # PRINT THE VALUE ON ADC CHANNEL 0
40 GOTO 20          # LOOP FOREVER
```

4.17 Extending the testing capability of the IF-THEN statement

The AttoBASIC interpreter is not capable of testing multiple comparisons within a single *IF-THEN* statement because only the value of the 1st comparison is returned to the *IF-THEN* function. Therefore a work-around method is required.

Before the *IF-THEN* test can be performed, the result of a comparison must be assigned to a variable. Since the result of any comparison is "1" for true or "0" for false then the bitwise logical operators will operate on bit position 0 only (bits 7:1 will be "0") and return a value of "1" or "0" to the *IF-THEN* statement to test. For example:

```
10 A:= ADC0 > 128      # ASSIGN THE RESULT OF TEST TO VARIABLE A
20 B:= ADC1 < 128      # ASSIGN THE RESULT OF TEST TO VARIABLE B
30 C:= ACO              # ASSIGN ANALOG COMP TO VARIABLE C
30 IF A & B & C THEN PRINT "1" # PRINTS "1" IF VARIABLE A "AND" B "AND" C ARE "1"
40 PRINT "0"           # OTHERWISE PRINTS "0" IF ANY VARIABLE A THROUGH C IS "0"
```

The same *IF-THEN* tests can be performed using the logical OR operator but not for the logical XOR operator. For example:

```
PRINT A | B | C | D | E # PRINTS "1" IF ANY VARIABLE A THROUGH E ARE "1", PRINTS "0" IF ALL VARIABLES A THROUGH E ARE "0"
```

As another example, refer to *Example 1* in section 5.1, where the assignments of variables M and N in line #40 accomplish this task so line #45 can perform a proper comparison on the

AVR Data Recorder / Logger

Version 1.02

result of the logical AND function ("&"), which always returns a single value to the operation preceding it. As a matter of fact, this method illustrates how one can test for multiple "true" or "false" conditions in a single *IF-THEN* statement. The depth of the *IF-THEN* test can be up to six (6) "true" or "false" conditions.

5 Example Programs

The following are sample programs that illustrate the versatility of the *AVR Data Recorder*. While the programs were written to illustrate its capabilities, the reader is encouraged to use them as a guide to writing their own, perhaps more complex programs.

5.1 **Example 1 - Basic 2-channel with real-time counter**

This example is the program pre-loaded into the *AVR Data Recorder's* internal non-volatile memory when shipped. It is printed here for reference in the case that the original program is over-written and the user wishes to re-load it.

```
5 DFW 0; DFI 0 255; DFX      # CLEAR DATA FILE AND RESET POINTER
10 RTI 2                     # 100MS RTC INTERVAL
15 ADG 0 1; ADG 1 1          # SET CH0 AND CH1 GAIN TO 0.1X
20 RTR                       # CLEAR RTC
25 A:=PEEK $5 $67            # CURRENT DATA FILE PAGE @$567
30 B:=PEEK $5 $68            # CURRENT DATA FILE OFFSET @$568
35 M:= A= $FF; N:= B> $FE    # CHECK FOR NEAR ROLL-OVER
40 IF M & N THEN GOTO 110     # INHIBIT ROLLOVER
45 DFL PEEK $5 $63           # LOG 24 BITS RTC TO DATA FILE
50 DFL PEEK $5 $64
55 DFL PEEK $5 $65
60 DFL ADC 0                 # LOG CH 0 AND 1 TO DATA FILE
65 DFL ADC 1
70 RTP; PRI ":~"             # PRINT THE VALUE OF THE RTC
75 PRI "ADC0: "; PRI ADC 0   # PRINT THE VALUE OF ADC0
80 PRI "ADC1: "; PRI ADC 1   # PRINT THE VALUE OF ADC1
85 PRI "~"                   # PRINT A CR/LF SEPARATOR
90 FOR I= 1 60               # LOOP COUNTER FOR 60 SECONDS
95 DEL 100                   # DELAY 1000MS
100 NEXT                     # INCREMENT LOOP COUNTER
105 GOTO 25                  # LOOP FOR MORE
110 DFF 5                    # SET DFD FORMAT TO 5 FIELDS PER RECORD
115 END
```

If the program is allowed to run to completion, it will end when the internal data file address pointer reaches location 65,535 (approximately 217 hours and 10 minutes of recording), which is the end of the data file. In other words, the program will exit before the data file address pointer wraps around to zero (0) and starts over-writing previously captured data.

If the user wishes to end program execution, plug the *AVR Data Recorder* into the host computer's USB port and run the host's terminal emulator program. Pressing the *BREAK* key two or three times quickly should stop the program. Once AttoBASIC has returned to immediate mode, press the *ENTER* key once or twice to clear any garbage characters from the input buffer. At this point, the user can start the terminal emulator's capture function and enter the AttoBASIC "*DFD 0 255*" command, which will print the entire 65,536 bytes of data in CSV format.

AVR Data Recorder / Logger

Version 1.02

This program can be modified for other needs if used as the basis for a data capture and record loop.

5.2 Example 2 - Analog comparator triggering

This example uses a square-wave generator connected to analog channel 1 set to 1x gain. The square wave is 5 Hz at 2 volts P-P. The analog comparator is triggered each time the rising and falling edge pass 0.95 volts (the reference).

```
5  N:=0                # CLEAR COUNTER
10 ADG 1 2             # SET ADC1 AT GAIN @1X
15 ACR 80              # SET 0.95V REF ((3/255)*N)
20 ACS 1 ; ACI 1       # SELECT ACO TO ADC1 AND ENABLE ACO IRQ
25 SLP; DEL 5; N:= N+1 # WAIT FOR ACO, DELAY 50MS, INCREMENT COUNTER
30 PRI "IRQ # "; PRI N # INFORM USER
35 GOTO 25             # LOOP FOR MORE
```

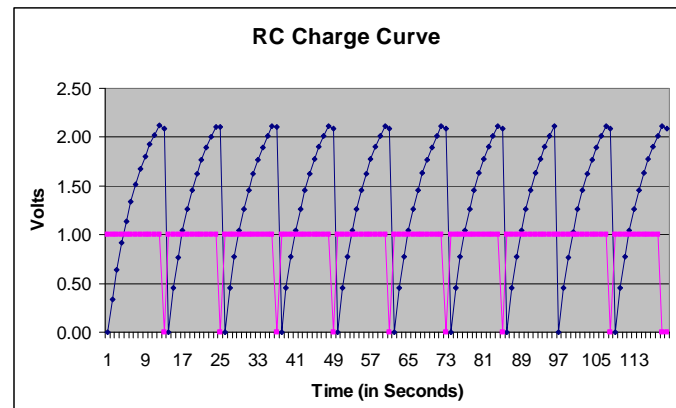
5.3 Example 3 - Sweeping Vref to detect an input voltage

This example uses a 1.00 volt reference connected to both analog channels and a 1.5 volt battery connected to the analog comparator's dedicated input. It selects the channel then increments the reference voltage until the ACO is triggered. Note the reverse logic in determining this (IF ACO = 0 THEN ...). This is just an example of a way to measure the voltage applied to the analog comparator's dedicated input.

```
5  ADG 0 2; ADG 1 2    # ADC0 GAIN @1X, ADC1 GAIN @1X
10 ACS 2; GOSUB 30      # SELECT AC INPUT, RUN ROUTINE
15 ACS 0; GOSUB 30      # SELECT ADC0 INPUT, RUN ROUTINE
20 ACS 1; GOSUB 30      # SELECT ADC1 INPUT, RUN ROUTINE
25 END                 # BYE-BYE
30 FOR N= 1 200         # VREF VOLTAGE INCREMENT LOOP
35 ACR N; DEL 1         # REF TO N, DELAY 10MS FOR SETTLING
40 IF ACO = 0 THEN GOTO 50 # IF AC LESS THAN VREF GOTO LINE 50
45 NEX                 # INCREMENT LOOP COUNTER
50 PRI "ACO REF= "; PRI N # INFORM USER WHAT THE VREF WAS
55 RET                 # RETURN TO CALLER
```

5.4 Example 4 - Charge time of a capacitor - Method 1

This example samples the charge time of a capacitor. It uses the digital output to clamp the capacitor to ground then releases the clamp, allowing the capacitor to charge. Samples are captured every 500 mS, storing the voltage measured across the capacitor to the data file for later retrieval. The voltage reference is set to 2 volts, which is 66% of 3 volts, the charging voltage. It runs the test 10 times before stopping. The chart above is the data collected then imported into EXCEL™. A chart was then created from the data.



```
5  EMI $0C            #CLEAR SCREEN
```

AVR Data Recorder / Logger

Version 1.02

```
10 DFI 0 8; DFF 2; DFX      # INIT 8 PAGES, 2 FIELDS PER RECORD, RESET ADDRESS
                             POINTER
15 DIG 1; ADG 0 2          # INSURE CAP IS SHORTED AND SET CH 0 GAIN TO 1X
20 R:= 16                  # 2V ACO REF (0.1X ON ACO)
25 ACR R; SLP 5            # SET ACO REF AND DELAY 500MS
30 PRI "TESTING ACO TRIGGER LEVEL...~"
35 FOR N= 1 10             # LOOP TEST 10 TIMES
40 PRI "RUNNING TEST # "; PRI N; PRI "~" # PRINT TEST #
45 DFL ADC 0; DFL ACO      # LOG 1ST ADC AND ACO READING TO DATA FILE
50 DIG 0                   # RELESE CLAMP ON CAPACITOR
55 SLP 5; DFL ADC 0; DFL ACO      # SLEEP 500MS, TAKE AND STORE READING
60 PRI "SAMPLING ...~"
65 IF ACO = 0 THE GOTO 55 # KEEP TAKING READINGS TILL ACO TRIGGERED
70 PRI "~ACO TRIGGERED AT "; PRI ADC 0 # PRINT THE VALUE
75 DIG 1                   # CLAMP THE CAPACITOR
80 NEX                     # CONTINUE LOOPING
```

5.5 Example 5 - Charge time of a capacitor - Method 2

This example uses the same circuit connections as *Example 4* (§ 5.4) but illustrates using the analog comparator as an event trigger source. In this case, the charge time of the capacitor is measured using relative measurement based on the value of the RTC. No data is stored, just printed on the console. Line 70's multiple "*SLP 1*" commands are needed because the discharge of the capacitor also triggers an ACO interrupt event on the falling edge, which is latched and pending until execution of the next "*SLP*" command.

```
10 EMI $0C                # CLEAR SCREEN
15 DIG 1; R:= 16          # INSURE CAP IS SHORTED AND SET 2V ACO REF (0.1X ON ACO)
20 ACS 2; ACR R; SLP 5    # SET ACO AS INPUT, R=ACO REF AND DELAY 500MS
23 RTI 2                  # SET RTC INTERVAL TO 100MS
25 PRI "TESTING ACO TRIGGER LEVEL...~"      # INFORM USER
30 FOR N= 1 10            # LOOP TEST 10 TIMES
35 PRI "RUNNING TEST # "; PRI N      # PRINT TEST #
40 PRI "RELEASING CLAMP~" ;RTR      # INFORM USER AND RESET RTC
45 ACI 1; DIG 0           # ENABLE ACO IRQ'S AND RELEASE CLAMP ON CAPACITOR
50 SLP; DEL 5             # WAIT FOR ACO IRQ THEN DELAY 50 MS
55 PRI "ACO TRIGGERED @ " ; RTP      # INFORM USER OF EVENT AND PRINT RTC (CHARGE TIME)
60 PRI "APPLYING CLAMP~~" # INFORM USER OF RE-CLAMPING
65 DIG 1; DEL 50          # CLAMP THE CAP AND DELAY 500MS
70 ACI 0 ; SLP 1; SLP 1    # DISABLE ACO IRQ, CLEAR SPURIOUS IRQ'S
75 NEX                    # INCREMENT LOOP CONNTER
80 ACI 0 # DISABLE ACO IRQ'S
```

6 Specifications

6.1 Analog to Digital Converter

The AVR microcontroller's data sheet lists the following specifications for the on-chip analog to digital converter:

Resolution	10 bits
Absolute Accuracy (Including INL, DNL, quantization error, gain and offset error)	2 LSB (at 10 bits)
Integral Non-Linearity (INL)	0.5 LSB (at 10 bits)
Differential Non-Linearity (DNL)	0.25 LSB (at 10 bits)
Gain Error	2 LSB (at 10 bits)

AVR Data Recorder / Logger

Version 1.02

Offset Error

2 LSB (at 10 bits)

Although the ADC has 10 bits of resolution, the *AVR Data Recorder* uses only the most significant 8 of those 10 bits. Thus the absolute accuracy at 8 bits resolution is $\pm 0.4\% \pm 1$ LSB at the input to the ADC, which is post-gain/attenuator. The maximum input voltage at the input to the ADC is 2.5 volts, also post-gain/attenuator.

6.2 ADC Reference Voltage Generator

The ADC uses a MAXIM MAX6002 2.50 volt precision reference generator, which specifies an initial accuracy of $\pm 1\%$ at 25°C with a temperature coefficient of 20 to 100 ppm/°C.

6.3 Analog Inputs

The analog input stage provides a gain of 1x or 10x and an attenuation of 10x or 100x. At all ranges, the absolute error is $\pm 2\%$ at 25°C. The maximum input voltage to either analog input channel is as follows:

Gain or Attenuator setting	Maximum Input Voltage
100x Attenuation	250 volts
10x Attenuation	25 volts
1x Gain	2.5 volts
10x Gain	250 millivolts

6.4 Analog Comparator and Programmable Reference Generator

The analog comparator's input stage uses a 10:1 divider with an absolute error of $\pm 5\%$ at 25°C. This translates to a maximum input offset voltage error of $\pm 1.6\%$. The maximum input voltage is 25 volts D.C.

The analog comparator's programmable reference generator is driven by a Pulse-Width Modulated signal, which is converted to an analog voltage. The maximum error is $\pm 0.5\%$.

6.5 TRUE RMS to DC Converter

The *AVR Data Recorder* uses an ANALOG DEVICES AD536AJ True-RMS to DC converter. The data sheet specifies the following:

Conversion accuracy, total error	± 5 mv $\pm 0.5\%$ of reading
Error vs. Crest Factor, worst case of crest factor = 7	-1.0% of reading
Frequency Response, additional $\pm 1\%$ error	45 KHz at $V_{IN} = 100$ mV

These specifications apply post-gain/attenuator.

6.6 Current Sense Input

The *AVR Data Recorder's* current sense input uses the TEXAS INSTRUMENTS INA197 Current Shunt Monitor, which monitors the voltage drop across a 0.5Ω shunt resistor yielding a D.C. current range of 0 to 250 ma. The data sheet specifies the following for the INA197:

Full-scale input voltage	150 millivolts typical
Gain	20V/V typical
Total Output Error	$\pm 2.2\%$ at 25°C
Gain Error	$\pm 1.0\%$ at 25°C
Non-linearity Error	$\pm 0.1\%$ at 25°C
Frequency Response	200 KHz typical

AVR Data Recorder / Logger

Version 1.02

6.7 Digital Output

The *AVR Data Recorder* has a digital output that consists of an N-channel MOSFET transistor. The maximum drain-to-source voltage is 60 volts with a continuous drain current of 1.5 amps. The MOSFET's source is internally connected to the *AVR Data Recorder's* ground.

6.8 Non-volatile Memories

There are two separate devices internal to the *AVR Data Recorder*. The EEPROM memory used to store the user program and the EEPROM memory used to store the data. The pertinent specifications are:

EEPROM, Program Memory	
Write/Erase Cycles	100,000
Data Retention	100 years at 25°C
EEPROM, Data Memory	
Write/Erase Cycles	1,000,000
Data Retention	>200 years at 25°C

6.9 Internal Time Base

The internal time base uses the AVR microcontroller's on-chip oscillator, which is factory calibrated to be within $\pm 10\%$ at 25°C. However, the factory calibrating is typically within $\pm 1\%$. If the internal time base frequency error is found to be unacceptable, the on-chip oscillator can be calibrated using AttoBASIC's "OSC" command.

The accuracy of the Real-time Counter and "DELAY" command are affected by any frequency errors in the internal time-base.

6.10 Calibration of the Internal Time Base

It is possible to use the digital output driven by a frequency emitted using the DDS command to calibrate the AVR Data Recorder. A resistor is connected between the drain and a positive voltage potential (3 to 5 volts is sufficient). The negative voltage potential is connected to the source. A frequency counter is connected across the source and drain. Using a ten (10) second gate time will allow calibration to better than $\pm 0.1\%$ if a DDS frequency of 200Hz is used. The OSC command can be used to adjust the calibration byte in small increments until the emitted frequency is within the desired tolerance.

7 Summary

As illustrated in the aforementioned description, the *AVR Data Recorder* is a versatile programmable device offering portable low-power battery operation for capturing and storing data "in the field" or as a real-time data capture and display unit for bench-top testing and measurement. It's intuitive and straightforward programming language make it flexible and quick to learn.