# ZBasic

Application Note

# AN-101 ZX Download and Firmware Update Procedures

## 1.0 Introduction

This application note describes the procedure for downloading code and data to ZX devices (both VM and native mode) via the serial port. The procedure for downloading to a generic AVR containing a ZBasic-compatible bootloader is also discussed. This information is provided for those cases where it is desirable to effect downloading in a different manner than otherwise supported.

Source code that implements the described procedures is included as part of the ZBasic installation process, e.g. in the directory C:\Program Files\ZBasic\src\zload.

In this document, the special names CR, LF and ESC represent the hexadecimal character codes 0D, 0A and 1B respectively.

## 1.1 Background

The `.zxb` file created by the ZBasic compiler contains the code and data resulting from compiling a ZBasic program. For VM mode devices, the file contains records in Intel Hex format with some additional special record types specific to ZX devices. Information about the Intel Hex format is widely available on the Internet (see, for example, http://en.wikipedia.org/wiki/Intel_HEX) and is therefore not reproduced here. For native mode devices, the file contains records in a special native ZX format. The elements of the native mode format necessary for downloading are described in a later section.

## 2.0 Download Procedure – VM Mode Devices

For VM mode devices, the ZBasic compiler outputs records of the types shown in the table below. The first two record types are standard Intel Hex record types. The remaining record types are unique ZX extensions. Unless otherwise indicated, the address field in the record is ignored.

**ZX Record Types**

| Record Type (hexadecimal) | Description |
|---|---|
| `00` | The record's data is for Program Memory at the given address. |
| `01` | The "end record" indicating that no more records follow. |
| `56` | The record's data is for Persistent Memory at the given address. |
| `58` | The record's data indicates the minimum firmware version number. |
| `59` | The record's data indicates the target device name. |

ZX devices are capable of handling only the first four record types. Record type 59 must either be ignored by the downloader or handled specially. See further discussion on this topic below.

Data is downloaded to a VM mode ZX device using the following procedure.

### *Step 1*

Get the device into command mode. If the device has ATN circuitry, this can be accomplished by causing a positive going transition on the DTR line (resulting in a ZX reset pulse) at least 4 times at intervals of between

60mS and 90mS. The minimum pulse width for either phase of the DTR signal is 25uS (assuming that DTR is capacitively coupled to the reset transistor as in the recommended ATN circuitry). After the last reset in the series, the ZX will go into command mode, with its serial port speed set to 115.2K baud, and will output a command prompt (>, hex code 3e) to indicate its readiness to accept commands.

If the device is configured to respond to an "ATN character", it can be put in command mode by sending the selected ATN character. It is important to note that in this case the serial port speed for the device will not be changed. If desired, the serial port speed of the device can be changed by sending the speed change command.

The useful commands for the ZX device are shown in the table below. In each case, the hexadecimal code of the command character is shown in parentheses. The “Devices” column indicates whether the command is available for VM devices, Native mode devices or both.

**ZX Command Mode Commands**

| Command | Name | Devices | Description |
|---|---|---|---|
| `L (4C)` | `Load` | Both | Enter Download mode. |
| `V (56)` | `Verify` | Both | Enter Download Verification mode. |
| `I (49)` | `Identify` | Both | Emit a device identification string. |
| `! (21)` | `Execute` | Both | Execute the last-downloaded program. |
| `< (3D)` | `Low Speed` | Both | Switch to low speed (19.2K baud). |
| `> (3E)` | `High Speed` | Both | Switch to high speed (115.2K baud). |
| `U (55)` | `Update` | VM | Enter firmware update mode (with feedback). |
| `u (75)` | `Update` | VM | Enter firmware update mode (no feedback). |
| `? (3F)` | `Native Query` | Native | Output Y if native mode. |
| `& (38)` | `Read Persistent` | Native | Output 16 bytes from Persistent Memory beginning at the hexadecimal address given. |

Except for the last entry, each of the command characters in the table above must be followed by a CR or an LF. You may also send an ESC to get the ZX to ignore previously sent characters and emit a CRLF and another command prompt. Sending an "empty" command (just a CR or LF) has the same effect.

It is important to note that the ZX will echo back all characters that it receives and that a CR or LF will be echoed as CR followed by LF. This fact needs to be taken into account when processing a response from the device.

## *Step 2*

Put the ZX in download mode by sending the Load command followed by a CR. The ZX will indicate its readiness to receive Intel Hex records by sending an ACK character (A, hex code 41).

## *Step 3*

Send one complete Intel Hex record, beginning with the colon through the checksum byte. Do not send a CR and/or LF - only the characters of the record itself. In general, if the ZX receives the record correctly and processes it successfully, it will respond with an ACK character (A, hex code 41). If not, it will respond with a NAK character (N, hex code 4e). If the ZX does not respond within 250mS after sending it a record it is not likely to do so. The record could be re-sent or the process aborted.

Repeat this Step 3 for each record of the download file (but see comments below regarding the target device record). When the End record is received (`:00000001ff`), the ZX will return to command mode, emitting another prompt character.

For the Minimum Firmware Version record (record type 58), if the firmware version is at least as new as the version specified the ZX will respond with the ACK character. Otherwise, it will send the Firmware Mismatch response (F, hex code 46). If it doesn't recognize the record type, either no response is sent or an Unrecognized Type response (U, hex code 55) is sent depending on the firmware version.

It is important to note that the ZX can handle Intel Hex records containing at most 16 data bytes. The compiler will never output larger records. If records are constructed by an application, they should always be limited to at most 16 data bytes.

Note: the compiler usually outputs a Target Device record (record type 59). This record should not be sent to the ZX device since it may not respond to it. The record can either be skipped over or its content can be compared to device identification data retrieved prior to beginning the download (see further discussion below).

### *Step 4 (optional)*

If the download process is completed successfully, i.e. all records sent were affirmatively acknowledged, an optional verification pass may be run. The verification phase is identical to the download phase except that it is begun by sending a Verify command instead of a Load command. As each Intel Hex record is sent, the device will respond with an ACK character if the record was received correctly, a NAK character if not, or a Verification Error character (V, hex code 56) if there was a verification error.

### *Step 5*

Once the download and optional verification steps are completed, the ZX can be made to run its program by sending the Execute command or by toggling the DTR line once or twice. If the latter method is used it is important to avoid toggling the DTR line in a manner that will be interpreted by the ZX as a request to enter command mode (see the specifications given in Step 1 above).

## *2.1 Retrieving Device Identification Data*

If you plan to check the target data in a download file against the device identification you must retrieve the identification data prior to commencing the download process. With the ZX in command mode, send the Identify command followed by a carriage return. The device will respond with a character sequence like the following:

```
I\r\nZX24a v1.2.3\r\n0100,abcd
```

The first character of this response is the echoed Identify command. The `\r\n` represents a CR and LF. The next sequence of characters indicates the device type and its firmware version number followed by another CR and LF. Following that is information about the last-downloaded program. The first four hexadecimal digits represent the size, in bytes, of the last-downloaded program. The second set of 4 hexadecimal digits represent the 16-bit CRC value of the last-downloaded program. Note that there is no CR/LF following the CRC value.

If the VM mode download file contains a target identification record, it will look something like the following:

```
:060000595a583234610028
```

This record contains 6 data bytes (as indicated by the first 06 value) which immediately follow the record type. The 59 value is the "target identification" record type. In this example, the data bytes are 5A, 58, 32, 34, 61 and 00. The first 5 are the ASCII codes for the string `ZX24a` and the trailing zero value is a null termination byte.

## *2.2 Sending Configuration Data – VM Mode Devices*

The ZX reserves the first 32 bytes of Persistent Memory for system configuration information. One of the types of information stored in this area is the EEPROM configuration data as follows:

**System Configuration Data**

| Address | Description |
|---|---|
| `0006` | EEPROM page size in units of 16 bytes. |
| `0007` | LSB indicates if the EEPROM requires full page writes. |

The ZX can be configured for different EEPROMs by sending it a properly constructed type 56 record containing the hexadecimal address 0006 and the two correct byte values as described above. Of course, the EEPROM configuration data must be properly set before downloading program data. An example of a type 56 record for setting the EEPROM configuration for an AT25HP512 EEPROM is shown below.

```
:02000656080199
```

## 2.3 Retrieving Persistent Memory Data – VM Mode Devices

It may be useful in some cases to read the values contained in portions of the ZX's Persistent Memory. An extension of the Intel Hex format allows you to request the content of 16 contiguous bytes beginning at a specified address. To accomplish this, put the ZX in download mode and send it a type 5A record containing the starting address and a zero byte count. The first example record below shows a request to read Persistent Memory starting at address 6. The ZX will respond to this record by sending back two hexadecimal characters representing the value of each of 16 bytes beginning at the requested address. This group of 32 characters will be followed by the usual ACK character (A, hex code 41).

You may send as many of these type 5A records as you wish, reading back the Persistent Memory data for each one. When finished, you should send an End record (type 01) to return the ZX to command mode. An End record is shown below as the second example record.

```
:0000065aa0
:00000001ff
```

## 2.4 Firmware Update Procedure – VM Mode Devices

The process for performing a firmware update is similar in concept to downloading a user program but the details are different.

### Step 1

Get the device into Firmware Update mode. This can be done in two different ways. The first way is to get the device into command mode (as in Step 1 of the Download Procedure above) and then send the device an Update command (U or u). On ZX devices with built-in LEDs, the U command performs the firmware update while providing visual feedback by flashing the LEDs. The u command performs the firmware update without the visual feedback. On ZX devices without built-in LEDs, the two commands are equivalent.

The second way to get the ZX device into Firmware Update mode is to make the appropriate hardware connections to request Emergency Update mode as described in the ZBasic Reference Manual. When the device starts up with these special connections it immediately goes into Firmware Update mode and no prompt is issued.

In either case, when Firmware Update mode has been entered, the ZX will be prepared for communicating at 115.2K baud on its Com1 serial port.

### Step 2

Send the device a complete firmware update record. A firmware update record can be up to 300 bytes in length and it must be created from data stored in the specially formatted `.zvm` firmware update file. The firmware update file is a text file that contains lines composed of hexadecimal characters and comment lines. Any line that does not begin with a hexadecimal character is considered to be a comment line. The text below is the first few and last few lines of a firmware update file with the ellipsis representing additional lines elided for brevity.

```
; ZX-24a VM v1.5.0.
; Install using the IDE or zload.exe.
; See the documentation for more details.
011ADE243AEBE1FB32442967872EDE2F2D48110CDFD65E5515CFF614CD71874A
F88EEE98C2196C7886F1C31D1239E584B69AA5D20A50D5FA89BA5945CC5E11BC
...
CE8FEDC30F249D7F234ECEEB16B978A20FF3A025B289D58C3E6151FD87AFB1D0
DCED16209815052BF83DC6A5D64BDB96000022ED
```

The hexadecimal portion of the firmware update file can be viewed as a stream of bytes comprising records that are broken up into lines for readability, i.e., the end-of-line has no significance. This textual form must be converted to binary form before transmitting to the ZX, i.e. strip out the comments and end-of-line characters and convert pairs of hexadecimal digits to the corresponding byte values. All 16-bit values in the records are represented in “big endian” form meaning that the most significant byte precedes the least significant byte.

A record is extracted from the update file by first determining the length of the next record and then processing that number of bytes as a record along with the record length. The record length is determined by reading four hexadecimal digits and converting them to the equivalent numeric value. In the example above, the value of the first four hexadecimal digits on the first data line is `011a` in hexadecimal or 282 in decimal. This value gives the number of data bytes in the record, which bytes immediately follow the record length. Again referring to the example above, the first few data bytes of the record are `de 24 3a eb`.

To reiterate, the data from the records must be sent to the device in binary form, i.e. each pair of hexadecimal digits must be converted to the equivalent value and then sent to the device as a byte value. Using the example data above, one would first send the value `01`, then the value `1a` (together comprising the record length), then the data bytes `de`, `24`, `3a`, etc. through the end of the record for a total of (record length + 2) bytes.

When the complete record has been sent, the next step is to await a response from the device. It is recommended to wait at least 500mS for the device’s response which will be one of those in the table below.

**Firmware Update Record Response Codes**

| Response | Name | Meaning |
|---|---|---|
| `A (41)` | `ACK` | Record received and processed successfully. |
| `R (52)` | `Reset` | The Reset record was received; the device will now reset. |
| `E (45)` | `End Of File` | The end-of-file record was received. |
| `C (43)` | `CRC Error` | The record was received but a CRC error was detected. |
| `S (53)` | `Signature Error` | The record was received but the signature was incorrect. |
| `T (54)` | `Time Out` | The device timed out waiting for the next record byte. |

The first three responses should all be considered to be “success” responses even though the second and third responses should occur at most once during an update session. The last three response codes indicate a problem of some type. The record could be re-transmitted in case of a CRC error or a Time Out error. A signature error most likely indicates that the update file is not the correct one for the device being updated.

This Step 2 is repeated until a record length of zero is encountered, indicating the end of the update file. In the update file excerpt shown above, the zero record length is the sequence of four zeros near the end of the last line. The last four hexadecimal digits in the last line, immediately following the zero record length, represent the CRC value computed over all of the preceding data. Neither of these data values should be sent to the device; they are there solely for use by the downloader program.

The specifications of the CRC computation are described by the RockSoft model parameters shown below (see http://www.repairfaq.org/filipg/LINK/F_crc_v34.html#CRCV_005).

**RockSoft CRC Model Parameters**

| Parameter | Value |
|---|---|
| Name | `CRC-16` |
| Width | `16` |
| Poly | `8005` |
| Init | `0000` |
| RefIn | `false` |
| RefOut | `false` |
| XorOut | `0000` |
| Check | `fee8` |

If you wish to implement some sort of progress indicator, you must first scan the update file and count the number of records that it contains. Then, as the records are sent to the device a percentage of completion can be computed.

If the update file is scanned to obtain the record count, you may also compute the CRC over all of the data records and compare it to the file CRC given as the last four hexadecimal digits. Performing this check verifies the integrity of the update file. A further test of each record length, ensuring that it is not longer than the maximum record size (300), may be useful as well.

## *3.0 Sample CRC Code*

The following sample code illustrates the implementation of a CRC computation helper routine written in ZBasic and in C. The ZBasic subroutine takes the previous CRC value (passed in as the first parameter) and a new data byte and computes a new CRC value which is returned indirectly via the first parameter. The C function performs the same operation but returns the new CRC value directly. In both cases, when beginning the computation for a new stream of data, the CRC value should be initialized to zero before calling the CRC computation routine the first time. After all of the bytes of a stream have been passed to the CRC computation routine, call it twice more with a data value of zero to compute the final CRC for the stream.

```
Sub crc16(ByRef crcVal as UnsignedInteger, ByVal dataVal as Byte)
   ' the CRC-16 polynomial value
   Const crcPoly as UnsignedInteger = &H8005

   Dim crcData(1 to 4) as Byte
   Dim crcLow as Byte Alias crcData(1)
   Dim crcMid as UnsignedInteger Alias crcData(2)
   Dim crcHigh as Byte Alias crcData(4)
   Dim crcFull as UnsignedLong Alias crcData(1)

   ' put the previous CRC value in the middle 16 bits of the CRC accumulator
   crcMid = crcVal

  ' put the data value in the low 8 bits of the CRC accumulator
   crcLow = dataVal

   ' permute the CRC accumulator using the data byte
   Dim i as Byte
   For i = 1 to 8
       ' shift the accumulator left one bit
       crcFull = Shl(crcFull, 1)

       ' See if a bit shifted out of the middle 16 bits.  If so, permute
       ' the accumulator with the polynomial value.
       If (CBool(crcHigh And &H01)) Then
```

```
           crcMid = crcMid Xor crcPoly
       End if
   Next i

   ' extract the new CRC value from the middle 16 bits of the CRC accumulator
   crcVal = crcMid
End Sub

unsigned short
crc16(unsigned char ch, unsigned short prevCRC)
{
#define POLYNOMIAL      0x8005L
      unsigned char i;
      unsigned long crcVal;

      // concatentate the input byte and the previous CRC value making a 24-bit value
      crcVal = ((unsigned long)prevCRC << 8) | ch;

      for (i = 0; i < 8; i++)
      {
            // shift the 24-bit value left one place
            crcVal <<= 1;

            // XOR the polynomial if bit 23 was a 1
            if (crcVal & 0x1000000L)
                  crcVal ^= (POLYNOMIAL << 8);
      }

      // return bits 8-23 of the result as the new CRC value
      return((unsigned short)(crcVal >> 8));
#undef POLYNOMIAL
}
```

## 4.0 Download File Format – Native Mode ZX Devices

The .zxb file is a plain text file generated by the ZBasic compiler for native mode devices that contains a header followed by record data. The header consists of one or more lines that begin with two slashes. The first line of the header will always be as shown below.

```
// ZXC
```

This fact can be used to distinguish between a .zxb file for a native mode device and a .zxb file for a VM mode device. (Recall that every line of a VM mode .zxb file begins with a colon.) The remaining lines of the header give additional textual information about the .zxb file; they can be ignored for downloading purposes.

The records of a native mode .zxb file have the structure shown below, the elements of which are described subsequently.

*<type> <address> <byte count> <data> <CRC>*

***<type>***

This is a one-byte field indicating the type of the record. The set of record types are described in the table below.

**Native Mode Record Types**

| Type | Meaning |
|---|---|
| S | The record contains a “signature” indicating the target device for which the file is intended. |
| F or f | The record contains data to be written to the device’s Flash memory, i.e. Program Memory. |
| E | The record contains data to be written to the device’s EEPROM memory, i.e. Persistent Memory. |
| X | The record marks the end of the data records. This will always be the last record in the file. |

***<address>***

This is a six-byte field containing an address associated with the data of the record expressed as a hexadecimal value, i.e. each character will be in the set { 0-9, A-F }. For record types other than F and E, this field is ignored and will generally contain all zeroes.

***<byte count>***

This is a four-byte field containing the number of bytes contained in the <*data*> section expressed as a hexadecimal value, i.e. each character will be in the set { 0-9, A-F }. Note that the byte count may be zero.

***<data>***

This field contains the data values for the record where each data byte is expressed using two hexadecimal digits.

***<CRC>***

This field contains the CRC value computed using the data values of all of the preceding fields of the record. Note that the CRC is computed using the character value of the <*type*> field and, for all other fields, the byte value of successive hexadecimal characters taken two at a time. The CRC algorithm is identical to that described earlier in this document.

It is important to note that the record data may contain embedded end-of-line sequences. These are cosmetic and may be ignored.

An example of (a portion of) the content of a native mode .zxb file is given below. The ellipsis indicates additional lines that have been removed for brevity. This example shows six complete records: an S record, two F records, two E records and an X record.

```
// ZXC
// Target Device:    ZX24n
// Flash Page Size:  256
// Compiler Version: 2.8.5
// Generated:        Tue Jun 02 13:26:55 2009
// Code size:        1932 (words), 078c (hex)
// Code CRC:         f8bf (hex)
S00000000055a5832346e63f9
F0000000100
0c95480308918707049d8b0b00998f0f1c85931318819717148d9b1b10899f1f
2cb5a32328b1a72724bdab2b20b9af2f3ca5b33338a1b73734adbb3b30a9bf3f
4cd5e74148d1c74744ddcb4b40d9cf4f5cc5a85058c1eb5454cdeb5850c9df5f
6cf5e36368f1e76764fdeb6b60f9ef6f7ce5f37378e1f77774edfb7b7e7a7c7f
8081a2f5b6abb1a9ba84808bd6cfeffcf9f2929385b1892957764a7a42225320
c041d04324453647a63dccaf32122310a1511253055550564759b87bb92db32d
```

```
62f173c41d32d7276a297b2bcd0dd35d7be363d43522d8430bdcb33aac3d553d
7101ec77d1e6840498096e0f7d0de07bc5f27d166415f86371f9f46f78fdf26b61eb
...
F000f000018
b70a8a1c9d1a5c92a1fe8a9e9c98b20edd101a86ebda1606a82e
E0000010001ffd151
E0000080004c603bff81476
X0000000000405c
```

## *4.1 Download Procedure  – Native Mode ZX Devices*

Data is downloaded to a native mode ZX device using the following procedure.

### *Step 1*

Get the device into command mode.  This is accomplished in the same manner as for VM mode devices; see the discussion of this procedure in step 1 of the VM mode download procedure earlier in this document.  Once in command mode, the set of available commands is the same as for VM mode devices (see the table earlier in this document) except that the firmware upgrade commands (`U` and `u`) are not supported.

Each of the command characters sent must be followed by a CR or an LF.  You may also send an ESC to get the ZX to ignore previously sent characters and then emit a CRLF and another command prompt.  Sending an "empty" command (just a CR or LF) has the same effect.

It is important to note that the ZX will echo back all characters that it receives and that a CR or LF will be echoed as CR followed by LF.  This fact needs to be taken into account when processing a response from the device.

### *Step 2*

Put the ZX in download mode by sending the Load command followed by a CR.  The ZX will indicate its readiness to receive native mode records by sending an ACK character (`A`, hex code 41).

### *Step 3*

Send one complete native mode record, beginning with the record type through the CRC value.  Note that, except for the record type, the fields should be sent as byte values representing pairs of hexadecimal characters.  Do not send a CR and/or LF - only the record itself.  In general, if the ZX receives the record correctly and processes it successfully, it will respond with an ACK character (`A`, hex code 41).  If not, it will respond with a NAK character (`N`, hex code 4E). For the signature record (type `S`), the device will respond with an ACK if the device matches the signature and it will respond with a SIG character (`S`, hex code 53) if the signature does not match the device.  If the ZX does not respond within 250mS after sending it a record it is not likely to do so.  The record could be re-sent or the process aborted.

Repeat this Step 3 for each record of the download file.  When the Exit record is sent (type `X`), the ZX will exit download mode and return to command mode, emitting another prompt character.

### *Step 4 (optional)*

If the download process is completed successfully, i.e. all records sent were affirmatively acknowledged, an optional verification pass may be run.  The verification phase is identical to the download phase except that it is begun by sending a Verify command instead of a Load command.  As the records are sent, the device will respond with an ACK character if the record was received correctly, a NAK character if not, or a Verification Error character (`V`, hex code 56) if there was a verification error.

### *Step 5*

Once the download and optional verification steps are completed, the ZX can be made to run its program by sending the Execute command or by toggling the DTR line once or twice. If the latter method is used it is important to avoid toggling the DTR line in a manner that will be interpreted by the ZX as a request to enter command mode (see the specifications earlier in this document).

## *4.2 Retrieving Persistent Memory Data – Native Mode ZX Devices*

It may be useful in some cases to read the values contained in portions of the ZX's Persistent Memory. For this purpose, a special native mode command is provided. To begin, put the ZX in download mode and send it a "Read Persistent" command (&, hex code 38) followed by one or more hexadecimal digits representing a Persistent Memory address, followed by a CR. The device will respond by outputting 32 hexadecimal digits representing the value of the 16 bytes of Persistent Memory beginning at the specified address.

## *5.0 Download File Format – Generic AVR Devices*

When you specify the device parameters for a generic AVR device indicating that it has a ZBasic-compatible bootloader, the compiler generates a .zxb file that is very similar to the one generated for a native mode ZX device. The primary difference is the information in the comment block preceding the download data records. An example of (a portion of) the content of a generic device .zxb file is given below. The ellipsis indicates additional lines that have been removed for brevity. This example shows five complete records: an S record, one `f` record, two E records and an X record.

```
// ZXC
// Target Device:    atmega644P
// Signature:        0x1e960a
// Flash Page Size:  256
// Compiler Version: 4.0.1
// Generated:        Thu Apr 19 10:17:26 2012
// Code size:        818 (words), 0332 (hex)
// Code CRC:         a28f (hex)
S00000000031e960a3d0b
f0000000100
4fc0000082c0000080c000007ec000007cc000007ac0000078c0000076c00000
74c0000072c0000070c000006ec000006cc000006ac0000068c0000066c00000
fbc0000062c0000060c000005ec000005cc000005ac0000058c0000056c00000
54c0000052c0000050c000004ec000004cc000004ac0000048c000000d48656c
6c6f2c20776f726c6421e200010101010200c000023501040801c800c100c900
11241fbecfefd0e1debfcdbf80e090e060e072e082d19ebf8dbf11e0a0e0b1e0
e2e6f6e002c005900d92a230b107d9f711e0a2e0b1e001c01d92a533b107e1f7
12e014bd84e085bd8ce764d1882331f015bb10936e008fef8093340188e48093ac25
...
E0000010001ffd151
E000008000432038fa2a581
X0000000000405c
```

Note, particularly that the "Target Device:" line indicates the AVR device type. Also, an additional line is present in the comment block indicating the AVR device signature. This same data is present in the type S record as well.

## 5.1 Download Procedure – Generic AVR Devices

Although the .zxb file format for generic AVR devices is nearly identical to that for a native mode ZX device, the downloading procedure is very different. Data is downloaded to a generic AVR device using the following procedure.

### Step 1

Get the device into command mode. This is accomplished in the same manner as for native mode ZX and VM devices; see the discussion of this procedure in step 1 of the VM mode download procedure earlier in this document. Note, however, that the command mode prompt emitted by a generic AVR device is a percent sign (%). This fact can be used to distinguish between a generic AVR device and a ZX device. Once in command mode, the set of available commands is as shown in the table below.

### Step 2

Send one or more commands to the device to effect the desired operations. Each of the commands sent must be terminated by an LF (optionally preceded by a CR). You may also send an ESC to get the device to ignore previously sent characters and then emit a CRLF and another command prompt. Sending an "empty" command (just an LF) has the same effect. For some commands, additional data values must be provided following the command character. Unless otherwise specified, the data elements must be in hexadecimal format and may be preceded by one or more spaces.

It is important to note that the device will echo back all characters that it receives and that an LF will be echoed as CR followed by LF. This fact needs to be taken into account when processing a response from the device.

**Generic AVR Bootloader Command Mode Commands**

| Command | Name | Data | Description |
|---|---|---|---|
| `S (53)` | Get device signature | none | Return the device signature. |
| `V (56)` | Get bootloader version | none | Return the bootloader version number. |
| `B (42)` | Set baud rate | baud selector | Set the serial port baud rate. |
| `A (41)` | Set memory address | memory address | Set the address for the next memory operation |
| `f (66)` | Read Flash memory | byte count | Read data from Flash memory. |
| `F (46)` | Write Flash memory | byte count, data | Write data to Flash memory. |
| `e (65)` | Read EEPROM | byte count | Read data from EEPROM. |
| `E (45)` | Write EEPROM | byte count, data | Write data to EEPROM. |
| `X (58)` | Execute | none | Execute the last-downloaded program. |

Generally, the records of the .zxb file will be read and the appropriate commands will be sent to the AVR device to handle each record. For example, when the S record is seen, a command would be sent to the device to retrieve the device signature, the response to which would be compared to the content of the record. Similarly, for an record type `f` or `E`, a "set address" command would be sent followed by an `F` or `E` command, respectively, followed by the byte count, LF and memory data.

**Command Details**

**S -** No additional data is required. The device will respond by sending the signature length, in bytes, followed by device signature bytes (all in hexadecimal form) followed by a CR and LF. For a three byte signature, the response would be eight hexadecimal characters.

**V -** No additional data is required. The device will respond by sending the bootloader version number as two bytes, in hexadecimal form followed by a CR and LF. The first byte represents the major version number and the second represents the minor version number.

**B -** The desired baud rate selector must be specified as a hexadecimal value with as many as 6 hexadecimal digits. The sixteen least significant bits of the value are written directly to the UART baud rate selector register. The seventeenth bit selects "double speed" mode if asserted.  The device will wait for all pending serial data to be sent before changing the baud rate and then sending an ACK (A, hex 41).

**A -** The desired address must be specified as a hexadecimal value with as many as 6 hexadecimal digits. The device will respond by sending an ACK (A, hex 41).

**f -** The number of bytes to read must be specified as a hexadecimal value with as many as 4 hexadecimal digits. The device will respond by sending the requested number of Flash memory bytes (two hexadecimal digits per byte) beginning at the last-specified address followed by a CR and LF.

**F -** The number of bytes to written to Flash memory must be specified as a hexadecimal value with as many as 4 hexadecimal digits.  (Note, however, that the number of bytes to write should always be even and never be larger than the page size for the device.)  If the address is not on a page boundary for the device, a NAK (N, hex 4e) will be sent.  Otherwise, the device expects to receive the data to be written in hexadecimal form, two hexadecimal digits per byte.  If the write is successful, the device will respond by sending an ACK (A, hex 41).

**e -** The number of bytes to read must be specified as a hexadecimal value with as many as 4 hexadecimal digits. The device will respond by sending the requested number of EEPROM bytes (two hexadecimal digits per byte) beginning at the last-specified address followed by a CR and LF.

**E -** The number of bytes to written to EEPROM must be specified as a hexadecimal value with as many as 4 hexadecimal digits.  The device expects to receive the data to be written in hexadecimal form, two hexadecimal digits per byte.  If the write is successful, the device will respond by sending an ACK (A, hex 41).

**X -** No additional data is required. The device will respond by waiting for all remaining serial data to be sent and then invoking a watchdog reset.

### *Step 3 (optional)*

If the download process is completed successfully, i.e. all records sent were affirmatively acknowledged, an optional verification pass may be run.  The verification phase would be implemented by sending commands to read Flash memory or EEPROM and comparing it to the record data.

### *Step 4*

Once the download and optional verification steps are completed, the device can be made to run its program by sending the Execute command or by toggling the DTR line once or twice.  If the latter method is used it is important to avoid toggling the DTR line in a manner that will be interpreted by the device as a request to enter command mode (see the specifications earlier in this document).

## *Author*

Don Kinzer is the founder and CEO of Elba Corporation. He has extensive experience in both hardware and software aspects of microprocessors, microcontrollers and general-purpose computers.  Don can be contacted via email at dkinzer@zbasic.net.

## Revision History

April 2012 - Added information related to generic AVR devices.
June 2009 - Added information related to native mode ZX devices.

**e-mail:** support@zbasic.net **Web Site:** http://www.zbasic.net

**Disclaimer:** Elba Corp. makes no warranty regarding the accuracy of or the fitness for any particular purpose of the information in this document or the techniques described herein. The reader assumes the entire responsibility for the evaluation of and use of the information presented. The Company reserves the right to change the information described herein at any time without notice and does not make any commitment to update the information contained herein. No license to use proprietary information belonging to the Company or other parties is expressed or implied.