Digilent AVR Part Description File Specification

Revision: 2/6/06

DIGILENT®
www.digilentinc.com
215 E Main Suite D | Pullman, WA 99163
(509) 334 6306 Voice and Fax

Overview

The Digilent AVR Programmer™ uses part description files to program an AVR device. Part description files are XML documents that include device-specific information needed to program that AVR device as well as to display device-specific information in the Programmer's user interface. An example of a complete part description file is included in the appendix.

File Structure

All information in an XML file is contained in document elements delimited by tags. A tag has the form <TAGNAME>, where a tag name string is delimited by '<' and '>' with no spaces. A tagged section begins with a tag and ends with an end-tag. An end-tag has the form </TAGNAME>, which matches the beginning tag except that the tagname is preceded by a '/' character. The body of a tagged section may be made up of text or other tagged sections. All tagged sections must be fully nested.

The entire contents of the AVR part description file is contained within a tag called AVRPART.

```
<AVRPART>
    document body
</AVRPART>
```

The body of the AVRPART section is made up of five sections:

```
<FORMAT>
<ADMIN>
<MEMORY>
<FUSE>
<LOCKBIT>
```

These sections do not need to be in any particular order, but the above order is recommended.

FORMAT Section

This section is used to indicate that this is a Digilent format file. The body of this section is made up of the text "Digilent".

```
<FORMAT>Digilent</FORMAT>
```

ADMIN Section

This section is used to specify a data set name for the contents of the file, the device name for the specific AVR device that the file describes, and the signature bytes for the device. It contains the following three tags:

<name>
<device>
<signature>

NAME tag: This tag is used to specify a data set name for the file contents. This is currently not used by the AVR Programmer.

DEVICE tag: This tag is used to specify the name of the AVR device described by the contents of the file.

SIGNATURE tag: This tag is used to specify the signature bytes for the AVR device. The body of this section is made up of three hexadecimal bytes separated by spaces.

Example ADMIN section:

```
<ADMIN>
   <name>Digilent Cerebot</name>
   <device>ATmega64</device>
   <signature>0x1E 0x96 0x02</signature>
</ADMIN>
```

MEMORY Section

This section is used to specify the sizes of the memories in the device. This section is made up of the following tags:

<flash>
<flash_page>
<eeprom>

FLASH tag: This tag is used to specify the size of the program flash memory. The body of this tag contains a single decimal number giving the total size of the device program flash memory in bytes.

FLASH_PAGE tag: This tag is used to specify the size of the program flash page buffer. The body of this tag contains a single decimal number giving the flash page buffer size in 16 bit words. For AVR devices that aren't paged, the value should be 1.

EEPROM tag: This tag is used to specify the size of the data EEPROM memory. The body of this tag contains a single decimal number giving the total size of the data EEPROM memory in bytes.

Example MEMORY section:

```
<MEMORY>
   <flash>65536</flash>
   <flash_page>128</flash_page>
   <eeprom>2048</eeprom>
</MEMORY>
```

FUSE and LOCKBIT Sections

The FUSE section is used to describe the contents of the fuses list box in the Fuses tab of the AVR Programmer. The LOCKBIT section is used to describe the contents of the Lock Bits tab of the AVR programmer. The user interface elements that populate these list boxes are described in these sections. This section also specifies the number of fuse bytes and the default values for the fuse bytes.

The FUSE and LOCKBIT sections are made up of user interface element declarations, user interface element instantiations, and housekeeping values.

The user interface element declarations will declare one or more of the following interface elements:
<toggle>, <enum>, or <cond>

These are described fully in the user interface element declarations section of this document.

Following the user interface element declarations come the user interface element instantiations. These instantiations are made up of one or more <citm> tags. Each <citm> tag specifies an element to be added to the user interface list box and references a user interface element declared using one of the declarations described above.

Following the user interface element instantiations, the FUSE section contains a <count> tag and a <default> tag. These specify the number of fuse bytes in the device and the default value for the fuse bytes. The LOCKBIT section contains only a <default> tag.

COUNT tag: This tag specifies the number of fuse bytes in the device. The contents of this tag is a single decimal number in the range 1-3. For example:

```
<count>3</count>
```

DEFAULT tag: This tag specifies the default values for the fuse bytes or the lock bits. The contents of this tag will be 1 to 3 hexadecimal byte values separated by spaces. For example:

```
<default>0xFF 0xD9 0xE4</default>
```

The default tag for the fuses will specify as many values as the <count> tag indicated. The default value for the lock bits is a single byte value.

User Interface Element Declarations

The fuses and lock bits list boxes are populated with user interface elements declared using the tags described here. Each element to be placed in these list boxes is described by these tags, and then instantiated using <citm> tags in the FUSE or LOCKBIT sections of the part description file.

The following three kinds of user interface elements are provided:

TOGGLE: This is a binary enumeration. The user is allowed to select one of two values. This is used to set or clear a single bit field.

ENUM: This user interface element represents an enumerated value. This represents a multi-bit field where the user is allowed to choose one of N values for the field.

COND: This user interface element represents a conditionally enumerated value. This element uses the value in one multi-bit field to determine which enumeration is used to choose a value for another multi-bit field. This is primarily used with the startup time options, as the set of available choices depends on the clock source selected.

Tags Used to Declare User Interface Elements

The body of a user interface element declaration will contain one or more of the following tags:

<id>
<name>
<msk>
<item>
<citm>

ID tag: The id tag defines an identifier that refers to the element being instantiated by a <citm> tag. The contents of the tag is an arbitrary name string.

NAME tag: The name tag declares the name of the fuse bit or lock bit as described in the Atmel data sheet for the AVR device. This tag is required by the AVR programmer, but the value is not currently used. The contents of this tag is an arbitrary string, but the bit names described in the Atmel documentation should be used.

MSK tag: This tag defines a 32-bit wide bit mask used to isolate a specific bit or set of bits within the fuse bytes or lock bits bytes. The fuse bytes or lock bits bytes are treated as a single 32-bit value and then logically AND-ed with this mask value to isolate the specific bit or bits.

ITEM tag: Item tags specify the set of values that the user interface element can have and the display string associated with each of these values. Toggle elements and enum elements contain items. A toggle contains two items, and an enum contains at least two items.

An item tag is made up of a <val> tag, which specifies the binary value of the appropriate fuse or lock bit or bits, and a <text> item, which specifies the text to be displayed in the fuse or lock bit list box for the item.

CITM tag: Citm tags are used in conditional enumerations (<cond> elements) to list the set value ranges and enumerations that make up the conditional enumeration. A citm is made up of the three tags <min>, <max>, and <eitm>.

The <min> and <max> tags define a value range for the bits specified by the <msk> tag. The body of the <min> and <max> tags contains a single 32-bit hexadecimal number. The <eitm> tag specifies the identifier (<id> tag value) of the enum to display when the bit field value is within that value range. The body of the <eitm> tag contains the <id> of a previously defined enum element.

TOGGLE Element

The body of a toggle element is made up of the following elements:

<id>
<name>
<msk>
<item>
<item>

A toggle contains two items. One item is used to define the state when the fuse bit or lock bit is 0 and the other when the fuse bit or lock bit is 1. When the user clicks on a toggle in the application, its value switches between the two defined values.

Example TOGGLE element:

This example illustrates the watchdog timer fuse bit (WDTON) in the ATmega64.

```
<toggle>
   <id>tglWdton</id>
   <name>WDTON</name>
   <msk>0x00020000</msk>
   <item>
         <val>0x00000000</val>
         <text>[WDTON = 0] Watchdog timer always on enabled</text>
   </item>
   <item>
         <val>0x00020000</val>
         <text>[WDTON = 1] Watchdog timer always on disabled</text>
   </item>
</toggle>
```

ENUM Element

The body of an enum element is made up of the following tags:

<id>
<name>
<msk>
<item>
…
<item>

An enum contains at least two item tags. There is an item tag for each allowed value of the enumeration. If there are disallowed values within the bit field, then there won't be item elements for those values.

When the user clicks on an enum in the application, a drop down list showing the allowed values is displayed, allowing the user to choose a value from the list.

Example ENUM element:

```
<enum>
   <id>enmBootSz</id>
   <name>BOOTSZ</name>
   <msk>0x00000600</msk>
   <item>
      <val>0x00000000</val>
      <text>[BOOTSZ = 00] Boot Size = 4096 words, Reset vector =
0x7000</text>
   </item>
   <item>
      <val>0x00000200</val>
      <text>[BOOTSZ = 01] Boot Size = 2048 words, Reset vector =
0x7800</text>
   </item>
   <item>
      <val>0x00000400</val>
      <text>[BOOTSZ = 10] Boot Size = 1024 words, Reset vector =
0x7C00</text>
   </item>
   <item>
      <val>0x00000600</val>
      <text>[BOOTSZ = 11] Boot Size = 512 words, Reset vector =
0x7E00</text>
   </item>
</enum>
```

COND Element

The body of a cond element is made up of the following tags:

<id>
<name>
<msk>
<citm>
…
<citm>

A cond contains at least two citm tags. These specify the different enumerations that are displayed to the user depending on the setting of the bits specified by the <msk> tag.

In the application, a cond looks like an enum. The difference is that different enums are displayed, depending on the current setting of the bits specified by the <msk> tag.

Example COND element:

```
<cond>
   <id>cndSut</id>
   <name>SUT</name>
   <msk>0x0000000F</msk>
   <citm>
         <min>0x00000000</min>
         <max>0x00000004</max>
         <eitm>enmSutInt</eitm>
   </citm>
   <citm>
         <min>0x00000005</min>
         <max>0x00000008</max>
         <eitm>enmSutXrc</eitm>
   </citm>
   <citm>
         <min>0x00000009</min>
         <max>0x00000009</max>
         <eitm>enmSutLfx</eitm>
   </citm>
   <citm>
         <min>0x0000000A</min>
         <max>0x0000000F</max>
         <eitm>enmSutXtl</eitm>
   </citm>
</cond>
```

Appendix: Digilent Cerebot™ AVR Part Description File

The Digilent AVR Part Description file for the ATmega64 device on a Digilent Cerebot board is included here as a complete example:

```
<?xml version="1.0" encoding="utf-8" ?>
<AVRPART>
   <FORMAT>Digilent</FORMAT>
   <ADMIN>
      <name>Digilent Cerebot</name>
      <device>ATmega64</device>
      <signature>0x1E 0x96 0x02</signature>
   </ADMIN>

   <MEMORY>
      <flash>65536</flash>
      <flash_page>128</flash_page>
      <eeprom>2048</eeprom>
   </MEMORY>

   <FUSE>
      <!-- Enumeration for the ATmega103 compatibility bit -->
      <toggle>
         <id>tglM103C</id>
         <name>M103C</name>
         <msk>0x00020000</msk>
         <item>
            <val>0x00000000</val>
            <text>[M103C = 0] ATmega103 compatibility mode enabled</text>
         </item>
         <item>
            <val>0x00020000</val>
            <text>[M103C = 1] ATmega103 compatibility mode disabled</text>
         </item>
      </toggle>

      <!-- Enumeration items for the watch dog timer -->
      <toggle>
         <id>tglWdton</id>
         <name>WDTON</name>
         <msk>0x00010000</msk>
         <item>
            <val>0x00000000</val>
            <text>[WDTON = 0] Watchdog timer always on enabled</text>
         </item>
         <item>
            <val>0x00010000</val>
            <text>[WDTON = 1] Watchdog timer always on disabled</text>
         </item>
      </toggle>
```

```
      <!-- Enumeration for the on chip debug system -->
      <toggle>
         <id>tglOcden</id>
         <name>OCDEN</name>
         <msk>0x00008000</msk>
         <item>
            <val>0x00000000</val>
            <text>[OCDEN = 0] On-Chip Debug enabled</text>
         </item>
         <item>
            <val>0x00008000</val>
            <text>[OCDEN = 1] On-Chip Debug disabled</text>
         </item>
      </toggle>

      <!-- Enumeration for the JTAG enable -->
      <toggle>
         <id>tglJtagen</id>
         <name>JTAGEN</name>
         <msk>0x00004000</msk>
         <item>
            <val>0x00000000</val>
            <text>[JTAGEN = 0] JTAG interface enabled</text>
         </item>
         <item>
            <val>0x00004000</val>
            <text>[JTAGEN = 1] JTAG interface disabled</text>
         </item>
      </toggle>

      <!-- Enumeration for the CKOPT field -->
      <enum>
         <id>tglCkopt</id>
         <name>CKOPT</name>
         <msk>0x00001000</msk>
         <item>
            <val>0x00000000</val>
            <text>[CKOPT = 0] Clock oscillator option (see data
sheet)</text>
         </item>
         <item>
            <val>0x00001000</val>
            <text>[CKOPT = 1] Clock oscillator option (see data
sheet)</text>
         </item>
      </enum>

      <!-- Enumeration for the EESAVE field -->
      <toggle>
         <id>tglEesave</id>
         <name>EESAVE</name>
```

```
         <msk>0x00000800</msk>
         <item>
            <val>0x00000000</val>
            <text>[EESAVE = 0] EEPROM memory preserved through chip
erase</text>
         </item>
         <item>
            <val>0x00000800</val>
            <text>[EESAVE = 1] EEPROM memory erased when chip is
erased</text>
         </item>
      </toggle>

      <!-- Enumeration items for the boot size -->
      <enum>
         <id>enmBootSz</id>
         <name>BOOTSZ</name>
         <msk>0x00000600</msk>
         <item>
            <val>0x00000000</val>
            <text>[BOOTSZ = 00] Boot Size = 4096 words, Reset vector =
0x7000</text>
         </item>
         <item>
            <val>0x00000200</val>
            <text>[BOOTSZ = 01] Boot Size = 2048 words, Reset vector =
0x7800</text>
         </item>
         <item>
            <val>0x00000400</val>
            <text>[BOOTSZ = 10] Boot Size = 1024 words, Reset vector =
0x7C00</text>
         </item>
         <item>
            <val>0x00000600</val>
            <text>[BOOTSZ = 11] Boot Size = 512 words, Reset vector =
0x7E00</text>
         </item>
      </enum>

      <!-- Enumeration for the boot reset vector -->
      <toggle>
         <id>tglBootRst</id>
         <name>BOOTRST</name>
         <msk>0x00000100</msk>
         <item>
            <val>0x00000000</val>
            <text>[BOOTRST = 0] Boot block enabled, reset vector at
beginning of boot block</text>
         </item>
         <item>
            <val>0x00000100</val>
```

```
            <text>[BOOTRST = 1] Boot block disabled, reset vector at address
0x0000</text>
         </item>
      </toggle>

      <!-- Enumeration for the brown out detect level -->
      <enum>
         <id>enmBodLevel</id>
         <name>BODLEVEL</name>
         <msk>0x00000080</msk>
         <item>
            <val>0x00000000</val>
            <text>[BODLEVEL = 0] Brown out detect level 4.0V</text>
         </item>
         <item>
            <val>0x00000080</val>
            <text>[BODLEVEL = 1] Brown out detect level 2.7V</text>
         </item>
      </enum>

      <!-- Enumeration for the brown out detect enable -->
      <toggle>
         <id>tglBoden</id>
         <name>BODEN</name>
         <msk>0x00000040</msk>
         <item>
            <val>0x00000000</val>
            <text>[BODEN = 0] Brown out detect enabled</text>
         </item>
         <item>
            <val>0x00000040</val>
            <text>[BODEN = 1] Brown out detect disabled</text>
         </item>
      </toggle>

      <!-- Enumeration for the clock select (CKSEL) -->
      <enum>
         <id>enmCksel</id>
         <name>CKSEL</name>
         <msk>0x0000000F</msk>
         <item>
            <val>0x00000000</val>
            <text>[CKSEL = 0000] External Oscillator</text>
         </item>
         <item>
            <val>0x00000001</val>
            <text>[CKSEL = 0001] Calibrated Internal Oscillator,
1.0Mhz</text>
         </item>
         <item>
            <val>0x00000002</val>
```

```
            <text>[CKSEL = 0010] Calibrated Internal Oscillator,
2.0Mhz</text>
         </item>
         <item>
            <val>0x00000003</val>
            <text>[CKSEL = 0011] Calibrated Internal Oscillator,
4.0Mhz</text>
         </item>
         <item>
            <val>0x00000004</val>
            <text>[CKSEL = 0100] Calibrated Internal Oscillator,
8.0Mhz</text>
         </item>
         <item>
            <val>0x00000005</val>
            <text>[CKSEL = 0101] External RC Oscillator, 0.1-0.9Mhz</text>
         </item>
         <item>
            <val>0x00000006</val>
            <text>[CKSEL = 0110] External RC Oscillator, 0.9-3.0Mhz</text>
         </item>
         <item>
            <val>0x00000007</val>
            <text>[CKSEL = 0111] External RC Oscillator, 3.0-8.0Mhz</text>
         </item>
         <item>
            <val>0x00000008</val>
            <text>[CKSEL = 1000] External RC Oscillator, 8.0-12.0Mhz</text>
         </item>
         <item>
            <val>0x00000009</val>
            <text>[CKSEL = 1001] External Low Frequency (32.768Khz)
Crystal</text>
         </item>
         <item>
            <val>0x0000000A</val>
            <text>[CKSEL = 101X] External Crystal/Resonator, Low
Frequency</text>
         </item>
         <item>
            <val>0x0000000C</val>
            <text>[CKSEL = 110X] External Crystal/Resonator, Medium
Frequency</text>
         </item>
         <item>
            <val>0x0000000E</val>
            <text>[CKSEL = 111X] External Crystal/Resonator, High
Frequency</text>
         </item>
      </enum>

      <!-- Start up time enumerations. These are used in the startup time
```

```
    conditional enumeration based on the CKSEL selection in effect.
-->
<!-- Enumeration for external clock and internal oscillator -->
<enum>
   <id>enmSutInt</id>
   <name>SUT</name>
   <msk>0x00000030</msk>
   <item>
      <val>0x00000000</val>
      <text>[SUT = 00] Start-up time, 6 CK</text>
   </item>
   <item>
      <val>0x00000010</val>
      <text>[SUT = 01] Start-up time, 6 CK + 4.1ms</text>
   </item>
   <item>
      <val>0x00000020</val>
      <text>[SUT = 10] Start-up time, 6 CK + 65ms</text>
   </item>
</enum>

<!-- Startup time enumeration for external RC oscillator -->
<enum>
   <id>enmSutXrc</id>
   <name>SUT</name>
   <msk>0x00000030</msk>
   <item>
      <val>0x00000000</val>
      <text>[SUT = 00] Start-up time, 18 CK</text>
   </item>
   <item>
      <val>0x00000010</val>
      <text>[SUT = 01] Start-up time, 18 CK + 4.1ms</text>
   </item>
   <item>
      <val>0x00000020</val>
      <text>[SUT = 10] Start-up time, 18 CK = 65ms</text>
   </item>
</enum>

<!-- Startup time enumeration for low frequency crystal -->
<enum>
   <id>enmSutLfx</id>
   <name>SUT</name>
   <msk>0x00000030</msk>
   <item>
      <val>0x00000000</val>
      <text>[SUT = 00] Start-up time, 1K CK + 4.1ms</text>
   </item>
   <item>
      <val>0x00000010</val>
      <text>[SUT = 01] Start-up time, 1K CK + 65ms</text>
```

```
         </item>
         <item>
            <val>0x00000020</val>
            <text>[SUT = 10] Start-up time, 32K CK + 65ms</text>
         </item>
      </enum>

      <!-- Startup time enumeration for external crystal/resonator -->
      <enum>
         <id>enmSutXtl</id>
         <name>SUT</name>
         <msk>0x00000031</msk>
         <item>
            <val>0x00000000</val>
            <text>[CKSEL = XXX0, SUT = 00] Start-up time, 258 CK +
4.1ms</text>
         </item>
         <item>
            <val>0x00000010</val>
            <text>[CKSEL = XXX0, SUT = 01] Start-up time, 258 CK +
65ms</text>
         </item>
         <item>
            <val>0x00000020</val>
            <text>[CKSEL = XXX0, SUT = 10] Start-up time, 1K CK</text>
         </item>
         <item>
            <val>0x00000030</val>
            <text>[CKSEL = XXX0, SUT = 11] Start-up time, 1K CK +
4.1ms</text>
         </item>
         <item>
            <val>0x00000001</val>
            <text>[CKSEL = XXX1, SUT = 00] Start-up time, 1K CK +
65ms</text>
         </item>
         <item>
            <val>0x00000011</val>
            <text>[CKSEL = XXX1, SUT = 01] Start-up time, 16K CK</text>
         </item>
         <item>
            <val>0x00000021</val>
            <text>[CKSEL = XXX1, SUT = 10] Start-up time, 16K CK +
4.1ms</text>
         </item>
         <item>
            <val>0x00000031</val>
            <text>[CKSEL = XXX1, SUT = 11] Start-up time, 16K CK +
65ms</text>
         </item>
      </enum>
```

```
    <!--- Conditional enumeration set for the startup times. The enum
         chosen is based on the CKSEL setting in effect.
    -->
    <cond>
       <id>cndSut</id>
       <name>SUT</name>
       <msk>0x0000000F</msk>
       <citm>
          <min>0x00000000</min>
          <max>0x00000004</max>
          <eitm>enmSutInt</eitm>
       </citm>
       <citm>
          <min>0x00000005</min>
          <max>0x00000008</max>
          <eitm>enmSutXrc</eitm>
       </citm>
       <citm>
          <min>0x00000009</min>
          <max>0x00000009</max>
          <eitm>enmSutLfx</eitm>
       </citm>
       <citm>
          <min>0x0000000A</min>
          <max>0x0000000F</max>
          <eitm>enmSutXtl</eitm>
       </citm>
    </cond>

    <!-- Define the actual fuse window contents -->
    <citm>tglM103C</citm>
    <citm>tglWdton</citm>
    <citm>tglOcden</citm>
    <citm>tglJtagen</citm>
    <citm>tglCkopt</citm>
    <citm>tglEesave</citm>
    <citm>enmBootSz</citm>
    <citm>tglBootRst</citm>
    <citm>enmBodLevel</citm>
    <citm>tglBoden</citm>
    <citm>enmCksel</citm>
    <citm>cndSut</citm>

    <!-- Specify the number of fuse bytes in the device. -->
    <count>3</count>

    <!-- Define the default fuse setting for this device -->
    <!-- Note: this is the Digilent Cerebot default fuse setting and
         not the ATmega64 factory default -->
    <default>0xFF 0xD9 0xE4</default>
  </FUSE>
```

```
   <LOCKBIT>
      <!-- Define the enumeration values used in the window. -->
      <!-- Enumeration for the basic lock bits -->
      <enum>
         <id>enmLock</id>
         <name>LB</name>
         <msk>0x03</msk>
         <item>
            <val>0x03</val>
            <text>Lock Mode 1: No memory lock features enabled.</text>
         </item>
         <item>
            <val>0x02</val>
            <text>Lock Mode 2: Programming of Flash and EEPROM
disabled.</text>
         </item>
         <item>
            <val>0x00</val>
            <text>Lock Mode 3: Programming and verification of Flash and
EEPROM disabled.</text>
         </item>
      </enum>

      <!-- Enumeration for the application section lock bits -->
      <enum>
         <id>enmApp</id>
         <name>BLB0</name>
         <msk>0x0C</msk>
         <item>
            <val>0x0C</val>
            <text>Application Section Lock Mode 1: No restriction on LPM or
SPM</text>
         </item>
         <item>
            <val>0x08</val>
            <text>Application Section Lock Mode 2: SPM not allowed</text>
         </item>
         <item>
            <val>0x00</val>
            <text>Application Section Lock Mode 3: LPM and SPM not
allowed</text>
         </item>
         <item>
            <val>0x04</val>
            <text>Application Section Lock Mode 4: LPM not allowed</text>
         </item>
      </enum>

      <!-- Enumeration for the boot section lock bits -->
      <enum>
         <id>enmBoot</id>
         <name>BLB1</name>
```

```
        <msk>0x30</msk>
        <item>
           <val>0x30</val>
           <text>Boot Section Lock Mode 1: No restriction on LPM or
SPM</text>
        </item>
        <item>
           <val>0x20</val>
           <text>Boot Section Lock Mode 2: SPM not allowed</text>
        </item>
        <item>
           <val>0x00</val>
           <text>Boot Section Lock Mode 3: LPM and SPM not allowed</text>
        </item>
        <item>
           <val>0x10</val>
           <text>Boot Section Lock Mode 4: LPM not allowed</text>
        </item>
     </enum>

     <!-- Define the actual window content -->
     <citm>enmLock</citm>
     <citm>enmApp</citm>
     <citm>enmBoot</citm>

     <!-- Define the default setting for the lock bits -->
     <default>0xFF</default>
  </LOCKBIT>
</AVRPART>
```