AVR100: Accessing the EEPROM

Features
• Random Read/Write
• Sequential Read/Write
• Runnable Test/Example Program

Introduction
This application note contains routines for access of the EEPROM memory in the AVR Microcontroller. Two types of Read/Write access has been implemented:
• Random Read/Write: The user must set up both data and address before calling the Read or Write routine.
• Sequential Read/Write: The user needs only to set up the data to be Read/Written. The current EEPROM address is automatically incremented prior to access. The address has to be set prior to writing the first byte in a sequence.

The application note contains four routines which are described in detail in the following sections. This application note contains routines for accessing the EEPROM in all AVR devices.

Note: In the latest devices the EEWE bit in EECR is called EEPE, and the EEMWE is called EEMPE. Also in the latest devices the EECR consist of two extra bits to set the Programming Mode, EEPM0 and EEPM1. These two bits needs to be initialized before EEPE is set.

Random Write – Subroutine “EEWrite”
Three register variables must be set up prior to calling this routine:
• EEdwr – Data to be written
• EEawr – Address low byte to write
• EEawrh – Address high byte to write

The subroutine waits until the EEPROM is ready to be programmed by polling the EEPROM Write Enable – EEWE bit in the EEPROM Control Register – EECR. When EEWE is zero, the contents of EEdwr is transferred to the EEPROM Data Register – EEDR, and the contents of EEawrh:EEawr is transferred to the EEPROM Address Register – EEARH:EEARL. First the EEPROM Master Write Enable – EEMWE is set, followed by the EEPROM write strobe EEWE in EECR. See Figure 1.
Random Read – Subroutine “EERead”

Prior to calling this routine, two register variables must be set up:

EEard – Address of low byte to read from
EEardh – Address of high byte to read from

The subroutine waits until the EEPROM is ready to be accessed by polling the EEWE bit in the EEPROM Control Register – EECR. When EEWE is zero, the subroutine and transfers the contents of EEardh:EEard to the EEPROM Address Register – EEARH:EEARL.

It then sets the EEPROM Read Strobe – EEERE.

In the next instruction the content of the EEDR Register is transferred to the register variable EEdrd. See Figure 2.
Sequential Write – Subroutine “EEWrite_seq”

Prior to calling this routine, one register variable must be set up:

EEdwr_s – Data to write

The subroutine waits until the EEPROM is ready to be programmed by polling the EEWE bit in the EEPROM Control Register – EECR. When EEWE is zero and the contents of the EEPROM Address Register – EEARH:EEARL are read into the register variable EEWTMPH:EEWTMP. EEwtmp is incremented and written back to EEARH:EEARL. This increments the current EEPROM address by one. The contents of EEdwr is then transferred to the EEPROM Data Register – EEDR, before EEWE in EECR is set, and then EEMWE is set. See Figure 3.
Figure 3. “EEWrite_seq” Flow Chart

Sequential Read – Subroutine “EERead_seq”

The subroutine waits until the EEPROM is ready to be accessed by polling the EEWE bit in the EEPROM Control Register – EECR. The subroutine then increments the current EEPROM address by performing the following operation: Transfer EEAR to the register variable EERTMPH:EERTMP, increments this register and writes the new address back to EEARH:EEARL. The routine then sets the EEPROM Read Strobe – EERE twice. Finally, the EEPROM data is transferred from EEDR to the register variable EErd_s. See Figure 4.
Optimization for different devices

Not all the instructions are necessary for all devices. If the device has an EEPROM of 256 bytes or less, the high address of the EEPROM Address Register doesn’t need to be changed. On the AT90S1200, the EEMWE bit in the EEGR doesn’t have to be set. See the section EEPROM Read/Write in the datasheet for further information.

Test Program

The application note assembly file contains a complete program which calls the four subroutines as a test of operation, and also as an example of usage. The test program is suitable for running in AVR Studio®.

The test programs contains comments on how to port the code to work on any AVR-part.

Note: If the code initiates a write to EEPROM shortly after Reset, keep in mind the following: If EEPROM contents are programmed during the manufacturing process, the MCU might change the code shortly after programming. When the programmer then verifies the EEPROM contents, this might fail because the EEPROM contents have already been modified by the MCU. Also notice that some In-System Programmers will allow the MCU to execute a short time between each step in the programming and verification process.

Table 1. CPU and Memory Usage

<table>
<thead>
<tr>
<th>Function</th>
<th>Code Size</th>
<th>Cycles</th>
<th>Example Register Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEWrte</td>
<td>10 words</td>
<td>15</td>
<td>R16, R17, R18</td>
<td>EEPROM Random Location Write</td>
</tr>
<tr>
<td>EERead</td>
<td>7 words</td>
<td>11</td>
<td>R0, R17, R18</td>
<td>EEPROM Random Location Read</td>
</tr>
<tr>
<td>EEWrite_seq</td>
<td>13 words</td>
<td>19</td>
<td>R24, R25, R18</td>
<td>EEPROM Sequential Location Write</td>
</tr>
<tr>
<td>EERead_seq</td>
<td>10 words</td>
<td>17</td>
<td>R0, R24, R25</td>
<td>EEPROM Sequential Location Read</td>
</tr>
</tbody>
</table>
### Table 1. CPU and Memory Usage (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Code Size</th>
<th>Cycles</th>
<th>Example Register Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>8 words</td>
<td>8</td>
<td>R16</td>
<td>Example Initialisation</td>
</tr>
<tr>
<td>Main</td>
<td>39 words</td>
<td>–</td>
<td>R16, R19, R20</td>
<td>Example Program</td>
</tr>
<tr>
<td>TOTAL</td>
<td>87 words</td>
<td>–</td>
<td>R0, R16, R17, R18, R19, R20, R24, R25</td>
<td>–</td>
</tr>
</tbody>
</table>

### Table 2. Peripheral Usage

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Description</th>
<th>Interrupts Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 I/O Pins</td>
<td>LEDs (example only)</td>
<td>–</td>
</tr>
<tr>
<td>1 I/O Pin</td>
<td>Button (example only)</td>
<td>–</td>
</tr>
<tr>
<td>10 bytes EEPROM</td>
<td>Target EEPROM Locations (example only)</td>
<td>–</td>
</tr>
</tbody>
</table>