AN2614
Application note

Serial numbering implementation

Introduction

It is common practice to give appliances a unique serial or identification number.

This serial number is located in the external memory of the device, for example in EEPROM, but it can also be programmed with the MCU software, inside the MCU program memory array.

In some cases, the serial number should remain secret and unreadable to external devices. In other cases, it should be readable from the MCU by a remote PC through any interface specified by the manufacturer (for example, SCI).

This application note provides technical details on the software and tools associated with implementing a serial numbering technique. Some hints have been taken from the on-line help of the STVP7 visual programming software tool.

C language examples are provided which are ready to be added to the source code if compiled with a Metrowerks C compiler (such as old Hiware, Panta or Codewarrior). The examples provide may also be ported to other C compilers.

The examples can be applied to any ST7 microcontrollers which is equipped with non volatile memory.

1 Serial number encoding techniques

A serial number can be composed of any number of letters and/or digits. A common serial number encoding technique is to encode the serial number as a string of ASCII characters, as shown in *Table 1*.

Table 1. Serial numbering

Serial number	**0 1 2 3 4 5 6 7**
Corresponding ASCII codes (hexa)	**30h 31h 32h 33h 34h 35h 36h 37h**

The numbers can also be combined in pairs, to form more concise 2-digit characters as shown in *Table 2*.

Table 2. Serial numbering in pairs

Serial number	**0 1 2 3 4 5 6 7**
Corresponding numbers (hexa)	**01h 23h 45h 67h**

Other encoding techniques are possible.

Such ‘encryption’ can be used so that only authorized people can decode the serial number.

2 Embedding the serial number

The serial number may need to be:

1. Embedded in the MCU, or stored in external memory (such as EEPROM)
2. Read from the MCU by an external device (for example, through an SCI interface)
3. Protected against read-out
4. Programmed at production level or updated just before shipment

Embedding the serial number inside the MCU software meets all the above requirements. This is especially true for the ST7 microcontrollers, whose Flash memory array can be programmed anytime with minimum hardware requirements (by means of ICP or IAP).

Before embedding the serial number, its format (size, encoding etc.) should be known .

For exampe, a serial number encoded over a four-character identification string may look as follows:

```
char Serial_Number[4];
```

2.1 Embedding the serial number inside the MCU software

2.1.1 Fixed serial number

This is the easiest way to implement the serial number, since it can either be programmed with the MCU software or programmed later.

If a common serial number is assigned to all appliances of a production lot, the string can be part of the normal MCU software without any specific requirements, as shown below:

```
#pragma INTO_ROM // in case -Cc option is not used
const char Serial_Number[4] = "0123"; // forget the trailing '\0'
```

2.1.2 Reserving space for unique serial numbers

If a different number needs to be given to each MCU, then a specific area must be reserved inside the software memory array which can be programmed on a chip-by-chip basis.

This is done inside the PRM file. A section indicating the size of the identification string should be created, and a placement name assigned to it.

In the following example, 4 bytes are reserved at the beginning of the Flash memory array:

```
SECTIONS
//...
SER_NUM = READ_ONLY 0x1000 TO 0x1003;/* Serial number area */
USER_ROM  = READ_ONLY 0x1004 TO 0xFFDF;/* FLASH area */
PLACEMENT
//...
SERIAL INTO SER_NUM;
```

In the software, the identification string may then be allocated inside this dedicated memory space as follows:

```
#pragma CONST_SEG SERIAL // map the next string into SERIAL section
const char Serial_Number[4];
#pragma CONST_SEG DEFAULT // restore normal string mapping
```

Once compiled and linked, the MAP file will show a segment called ‘SERIAL’ which is four-bytes wide and contains the serial number as shown below:

```
*****************************************************
SEGMENT-ALLOCATION SECTION
-----------------------------------------------------
Segmentname           Size Type    From       To  Name
SERIAL                  4     R     1000      1003 SER_NUM
```

2.1.3 Protection against read-out

To avoid subsequent reads of the serial number, the entire ST7 Flash memory array can be protected against read-out by enabling the read-out protection bit located in the first option byte. Protection is thus ensured as it can only be removed by erasing the entire Flash memory array (see *Section 3: Automatic serial number generation in the MCU on page 6*).

Note: Such read-out protection is device-dependent and is not a feature of all MCUs.

2.2 Getting the serial number from an external device

The serial number may be programmed in external memory, such as EEPROM. In this way, the MCU software memory array and the EEPROM data array can be programmed in a number of different ways:

- By each device using its own respective programming tool
- By the MCU receiving the serial number later (for example, through an SCI interface) and then programming the EEPROM accordingly

The MCU must read the EEPROM upon reset (if the serial number is to be processed by it) and must store the serial number in RAM, as follows:

```
char Serial_Number[4];
void main(void) {
//...
Read_EEPROM(Serial_Number, SubAddress);
// read string and store it into Serial_Number
```

In this way, the serial number is not permanently stored in the MCU and all MCUs in the production lot can be programmed with exactly the same contents.

3 Automatic serial number generation in the MCU

The STVP7 software tool provided with each EPB or ST7 STICK is well suited for automatic serial number generation in the MCU.

The step-by-step procedure described below is also available from the on-line help in the ‘help/index’ menu under STVP7.

3.1 Creating a project

The first step is to create a ‘project’ under STVP7.

A project is a way of automatically configuring STVP7 each time it is run. A project exists in a file format that contains STVP7 commands. This allows the user to set up a programming environment and to define programming procedures by opening the project file.

Such procedures may include the following actions:

- Erase everything before programming (Flash memory devices only)
- Blank check before programming
- Verify after programming
- Enable read-out protection
- Prompt for new programming cycle
- Generate and write serial numbers

To create a project:

- Click on ‘new’ in the project menu. The ‘create new project’ dialog box opens.
- In the drive list, click the drive on which you want to create the new project file.
- In the box beneath the drive list, double-click the name of the folder in which you want to create the new project file. Continue double-clicking subfolders until you open the subfolder in which you want to create the file.
- Type the project file name in the file name box.
- Click on ‘save’ (the project file name extension is .stp).
- The name of the project appears in the window title bar.

The project is now created and contains the configuration settings (under the configuration menu) ‘hardware’, ‘port’, ‘programming mode’ and ‘device’.

3.2 Project set-up

Under the ‘project/edit’ menu, five configuration windows (or tabs) need to be set up.

3.2.1 Configuration window

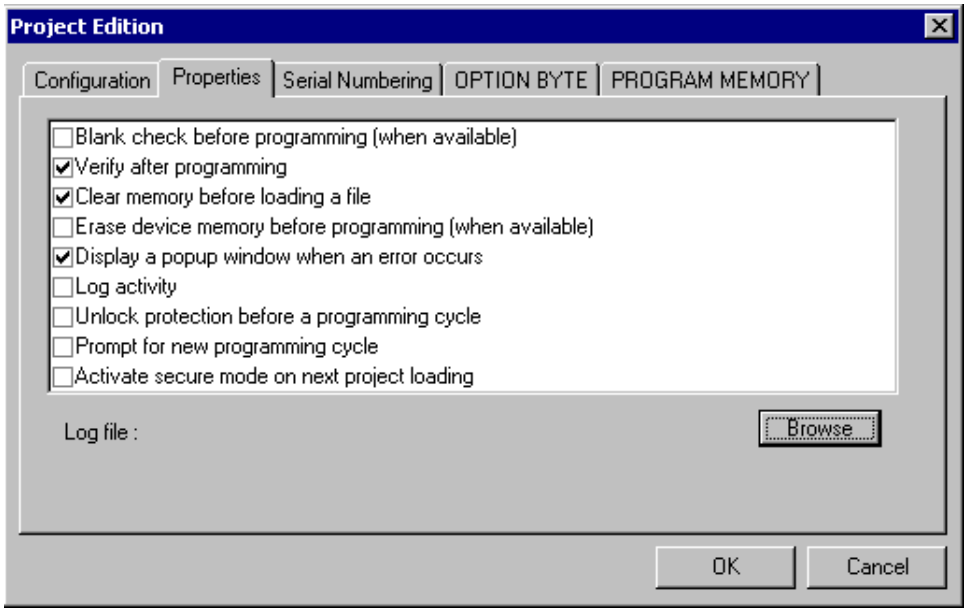
The programmer’s hardware and the MCU type are described in the configuration window. The configuration at the time the project was created can be modified here.

3.2.2 Properties window

The list of sequential actions to be performed for each chip are defined in the properties window.

Figure 1 shows the various programming options.

Figure 1. Programming options



The main options include:

- Blank check before programming: The device is blank checked first
- Verify after programming
- Clear memory before loading a file: This clears (= fill with FFh) the existing memory values in the STVP7 session when loading a new file. If it is not checked each time a new file is loaded, only those memory values in the newly loaded file are overwritten.
- Erase device memory before programming
- Unlock protection before a programming cycle: If the read-out protection needs to be removed at the beginning of a cycle because the device was protected prior to exectuting the project, STVP7 must reprogram the device in a specific way

- Prompt for new programming cycle: If the same programming procedure needs to be repeated for several microcontrollers, a dialog box appears at the end of each programming cycle and asks if the same programming cycle is to be restarted.

To program an MCU repeatedly, the following boxes should be ticked:

- Verify after programming
- Display a popup window when an error occurs
- Prompt for new programming cycle (optional)

The above assumes that the MCU is blank. If it has to be erased first, the following two boxes should also be checked:

- Blank check before programming
- Erase device memory before programming

If the read-out protection was previously activated, the following box should also be checked:

- Unlock protection before a programming cycle

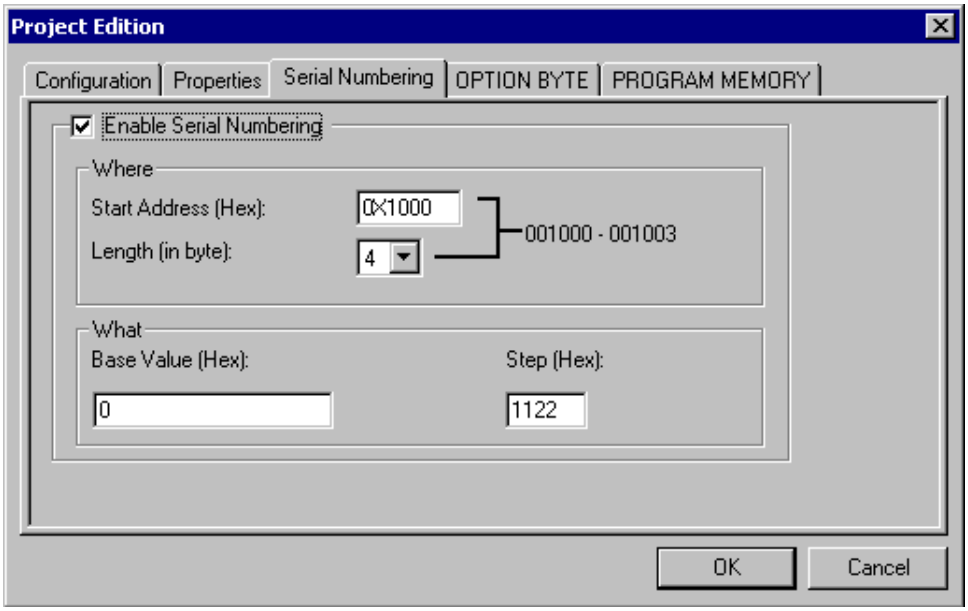
3.2.3 Serial number definition

The serial numbering window can automatically generate and write serial numbers into the memory of the MCU. Automatic serial numbering is defined by specifying:

- The memory address where it is to be programmed, for example 1000h
- The length (in bytes), for example 4
- The starting value: Any value specified by the customer, for example 00000000h
  - This is the serial number to be programmed in the MCU
- The step increment value: Any value specified by the customer, for example 1122h

Serial number settings are shown in *Figure 2: Serial number settings on page 9*.

Figure 2. Serial number settings



Each time a new chip is programmed, the current 'base value' is programmed at the 'start address' location, over 'length' bytes, and then the 'step' is added to the 'base value', forming the new 'base value' for the next programming cycle.

A unique serial number is therefore assigned to each newly programmed MCU.

Note: The current serial number value is saved each time a project is closed. It is loaded again, the next time the project (file) is opened.

3.2.4 Option byte configuration

The option byte configuration window selects the .S19 file corresponding to the MCU option byte definition. This file is created in the 'option byte' tag of the main window by selecting the desired option byte configuration for ST7 microcontrollers with Flash memory and read-out protection. The file is then saved under a distinct name (different from the main project .S19 file).

This .S19 file is automatically loaded each time the project is opened.

3.2.5 Program memory configuration

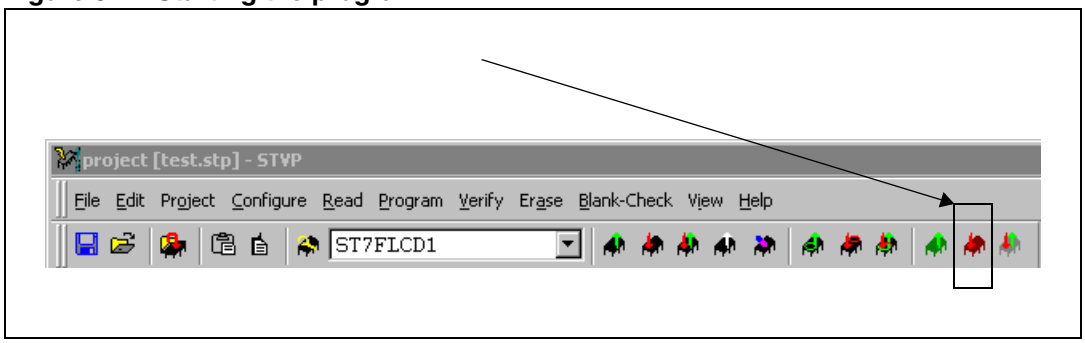
The program memory configuration window selects the .S19 file corresponding to the MCU software binary file, as created by the C-complier or assembly toolchain. This file does not contain any information regarding the option byte (refer to *Section 3.2.4*).

This .S19 file is automatically loaded each time the project is opened.

3.3 Automatic programming

Once the project is fully configured, the programming can be started by clicking on ‘all tabs’ in the ‘program’ menu or on the icon in *Figure 3* below.

Figure 3. Starting the program



The list of actions enabled in the ‘properties’ window is then executed by STVP7, including the serial numbering.

After each programming sequence, the serial number is updated according to the ‘serial numbering’ window settings.

If the ‘prompt for new programming cycle’ box was checked under the ‘properties’ window, a dialog box automatically appears to start the next programming cycle.

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
23-Aug-2007	1	Initial release