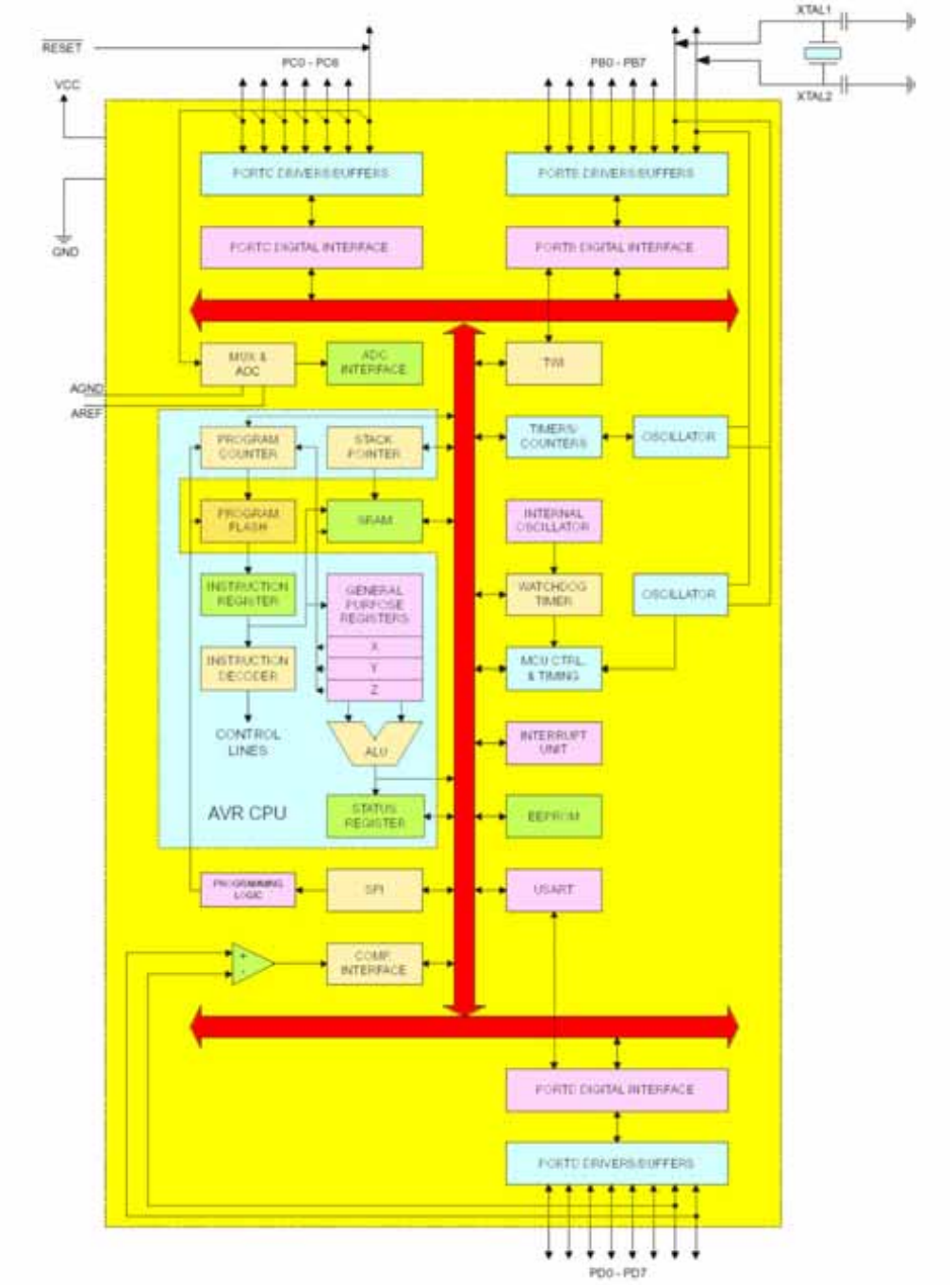
Theoretic/Practical course on BASCOM AVR Programming.
Author: DAMINO Salvatore.

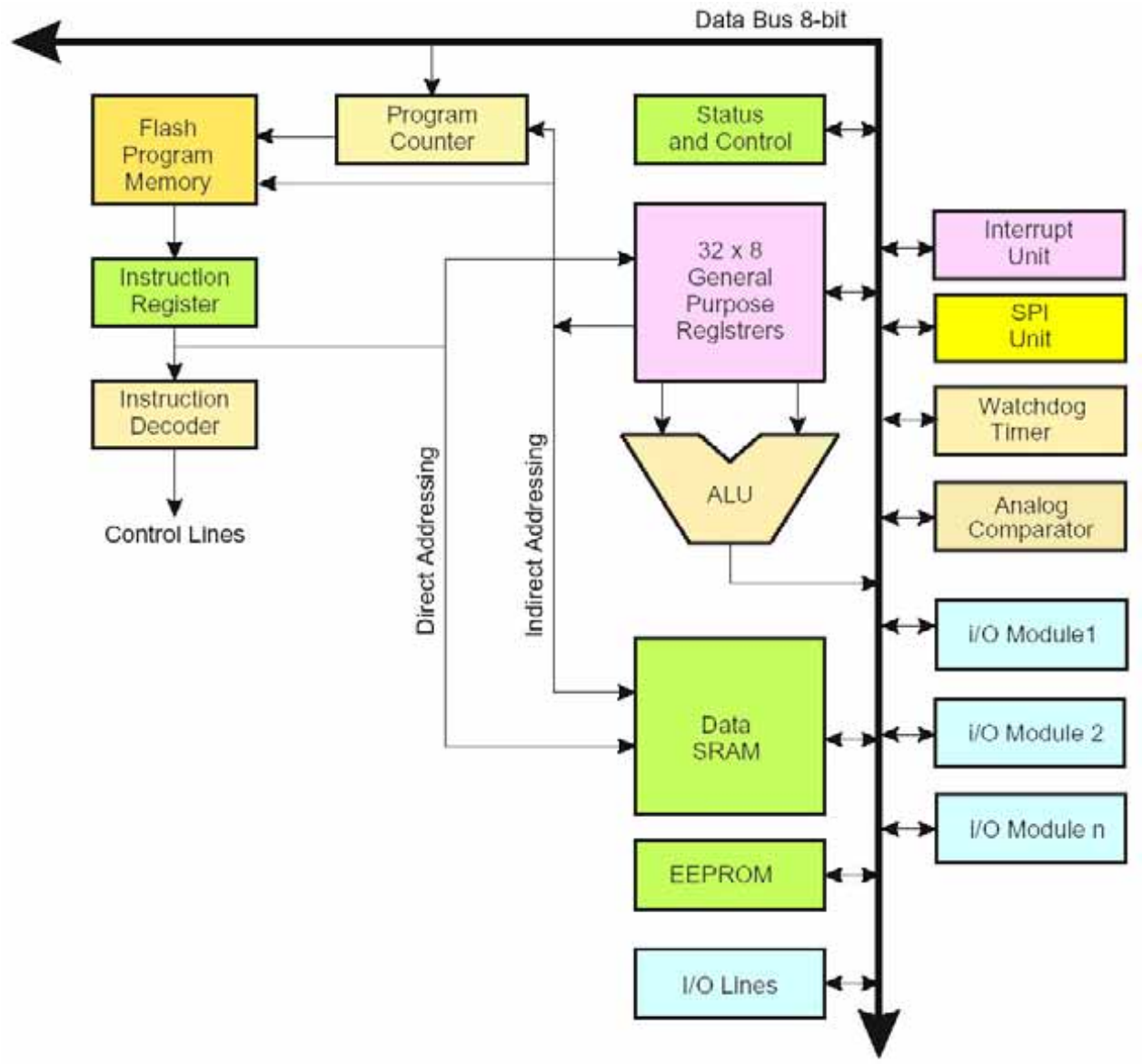
MORSE CODE (3).

As previously stated in order to efficiently use the features of **Mini Modules** it is necessary a deep knowledge of the used **CPU** structure. For this purpose it is really suggested a complete reading of **CPU Data-Sheet**. Below there are some brief but important part of this documentation.



Blocks Diagram of Atmel ATmega08 CPU.

In this lesson we take advantage of the functionalities offered by **Example.014** and we enlarge them by adding an important feature offered by standard **GMM AM08 Mini Module** that is the internal **EEPROM**.

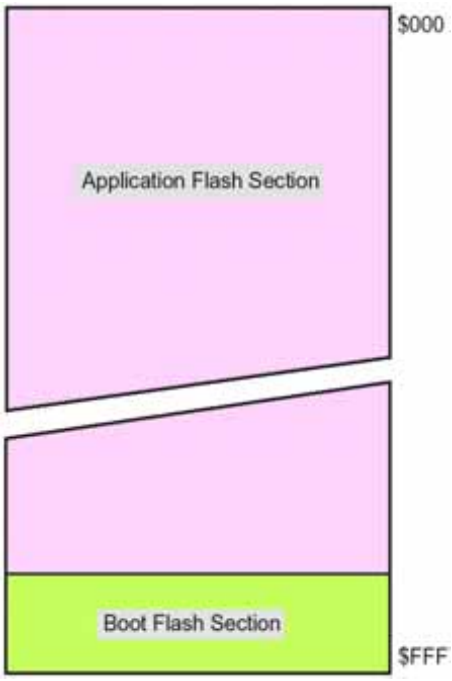


CPU Structure.

About memories, the **GMM AM08** has the following resources:

- ***8K*** *bytes of* ***FLASH*** *for Code.*
- ***1K*** *bytes of internal* ***RAM*** *for data.*
- ***512*** *Bytes of* ***EEPROM*** *for data.*
- ***32*** *"****General Working Registers****"*

Data Memory Map.



Program Memory Map.

In this example we take into account the possibilities offered by the **512 Bytes** of internal **EEPROM**.

This section is a **Memory** that is capable to maintain the information even without power supply. It is different from **FLASH** memory in fact it can be written, erased, rewritten, etc. at **Byte** level, for a very high number of times. In the specific case of this device the **Manufacturer Company** ensures **100K Cycles**, that aren't really few.

The application fields are numerous and they are limited only by fantasy of the developer. For example, you can set a defined number of parameters dedicated to specialize each produced devices or, as it happens in our example, it can memorizes some messages that are then generated by **Morse** code.

Once the message has been edited, it is memorized inside a **list** of messages on **EEPROM** of **GMM AM08**. At each message it is assigned an identification number that is used to recall and generate it.

All the messages will remain on the **Mini Module** even when the device is turned off. Then when it is turned back on, the messages can be recalled and regenerated without problems.

Telegraph Key.

Example.015. Training Program for MORSE Code With EEPROM.

Added Definitions:
None

Added Declarations:
None

Added Instructions:
READEEPROM ; WRITEEEPROM ; CHR ; SPACE.

Added Operators:
None

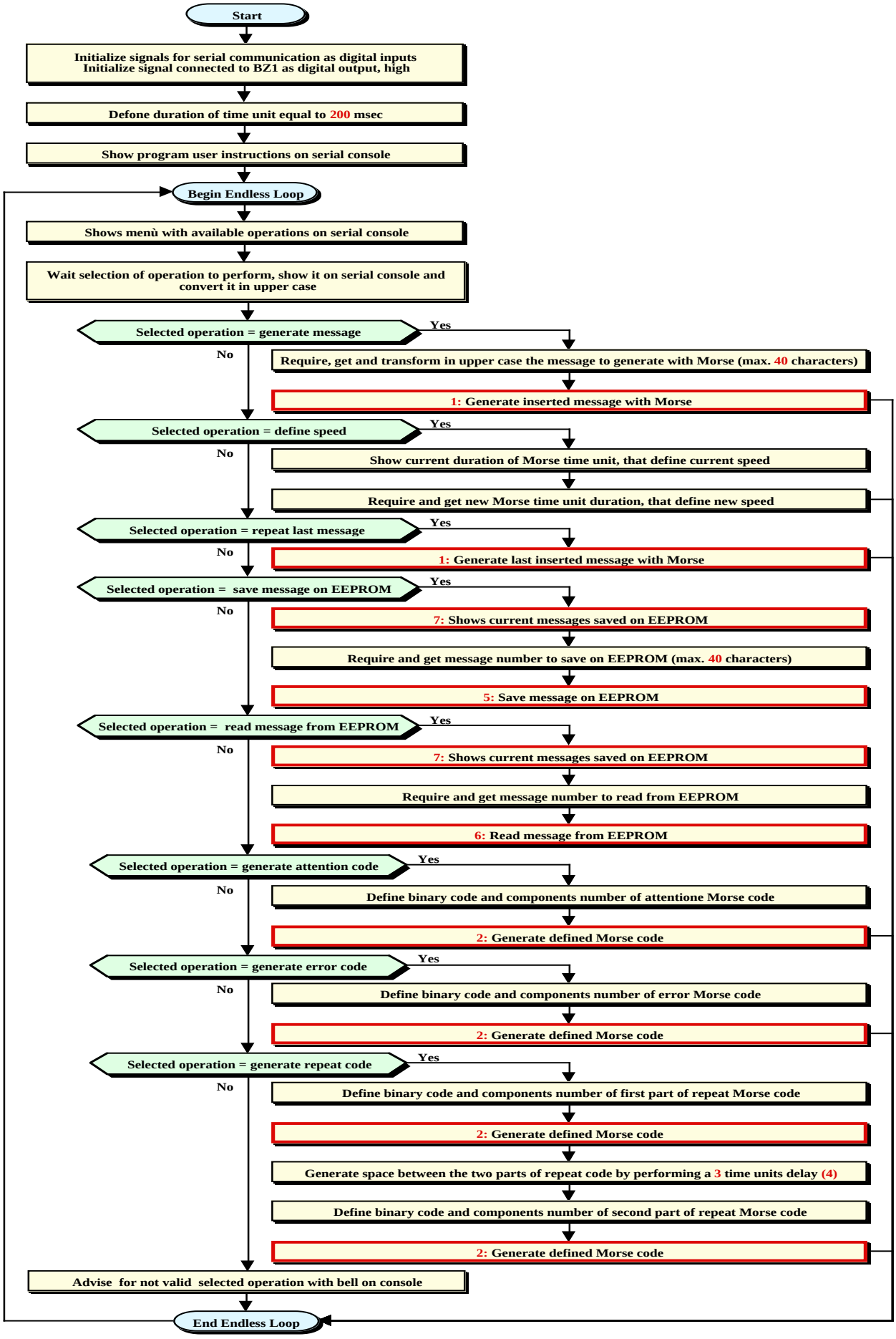
Example program **15** of **BASCOM AVR** course.

It generates messages with **Morse** codes and save/load them into/from internal **EEPROM**.

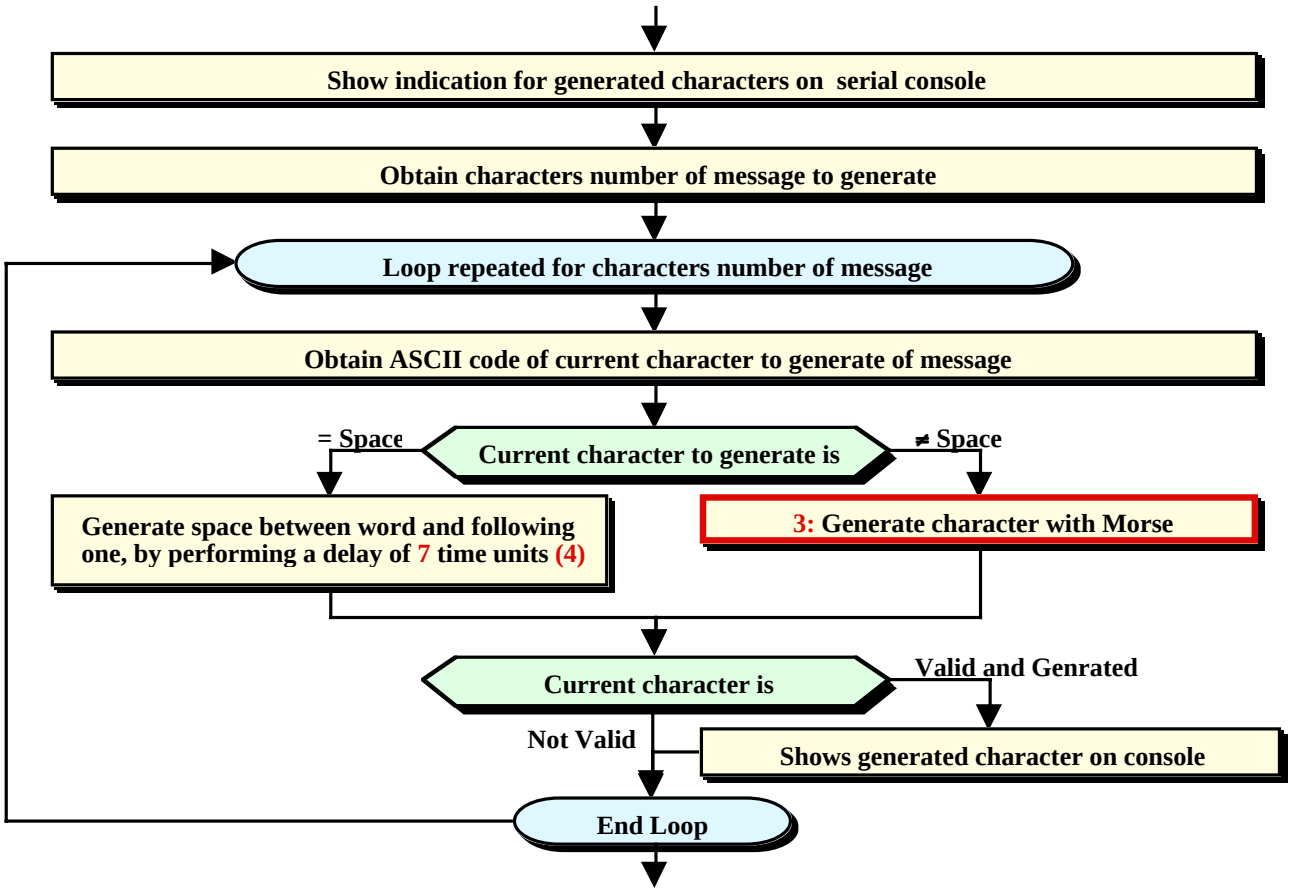
The program can perform the following operations: ask for a message and then generates it with Morse codes on buzzer, define the generation speed of the **Morse** codes, repeat the generation of the last inserted message, save up to **5** messages on **EEPROM**, load a message from **EEPROM** and finally it generates the **3** special **Morse** codes dedicated to attention, error and repeat.

The user interactions happen through a serial console provided of keyboard and monitor and it must communicate with a fixed physical protocol at **19.200 Baud**, **8 Bit x chr**, **1 Stop bit**, **No Parity**.

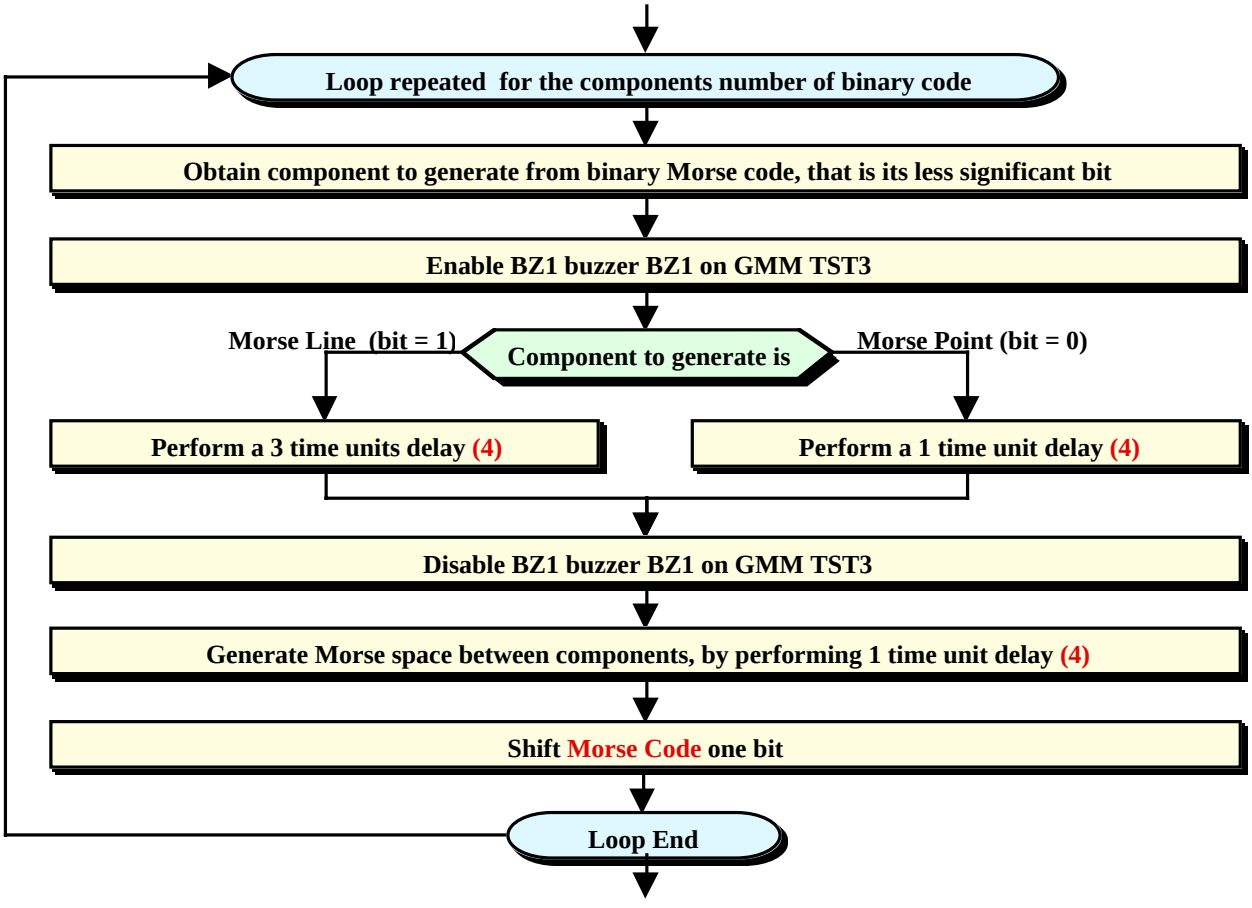
This console can be another system capable to support a serial **RS 232** communication. In order to simplify the use it can be used a **PC** provided of one **COMx** line, that execute a terminal emulation program as **HYPERTERMINAL** or the homonym modality provided by **BASCOM AVR** (see **IDE** Configuration).



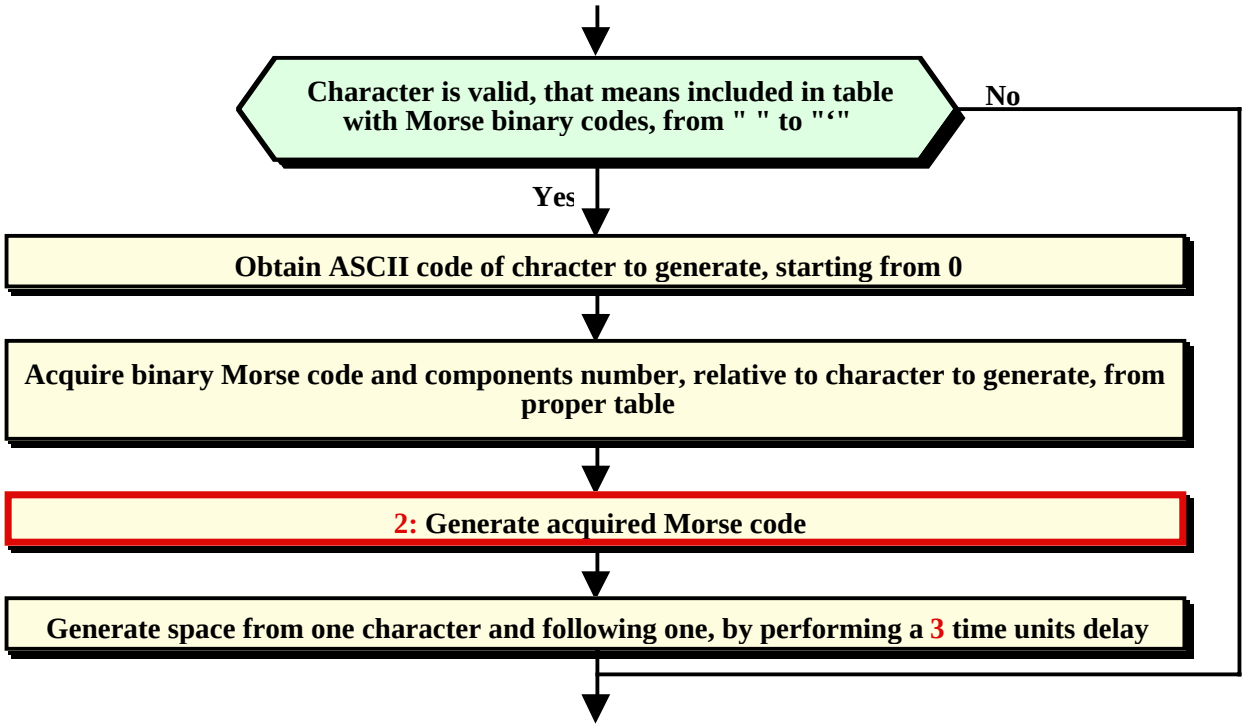
Flow Chart Diagram of the Program.



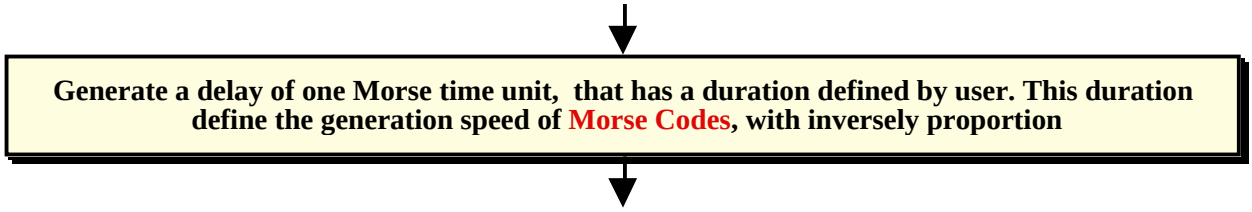
1: Generate Message in Morse.



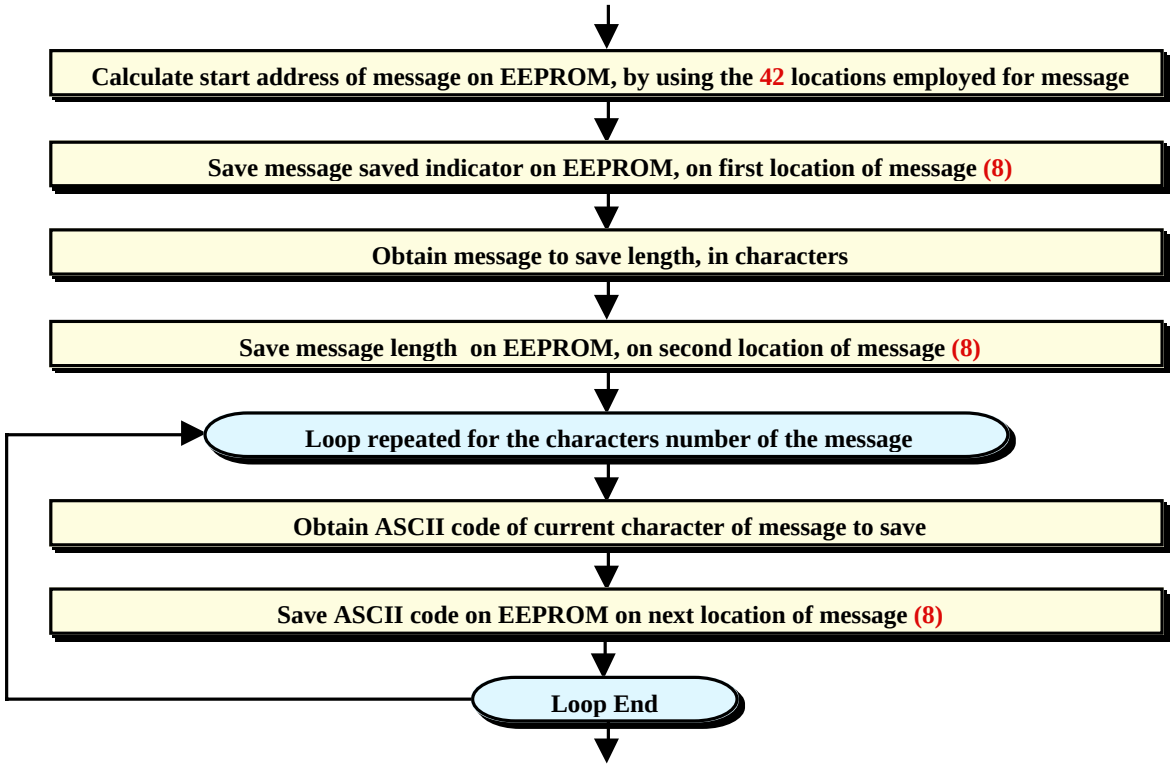
2: Generate Morse Code.



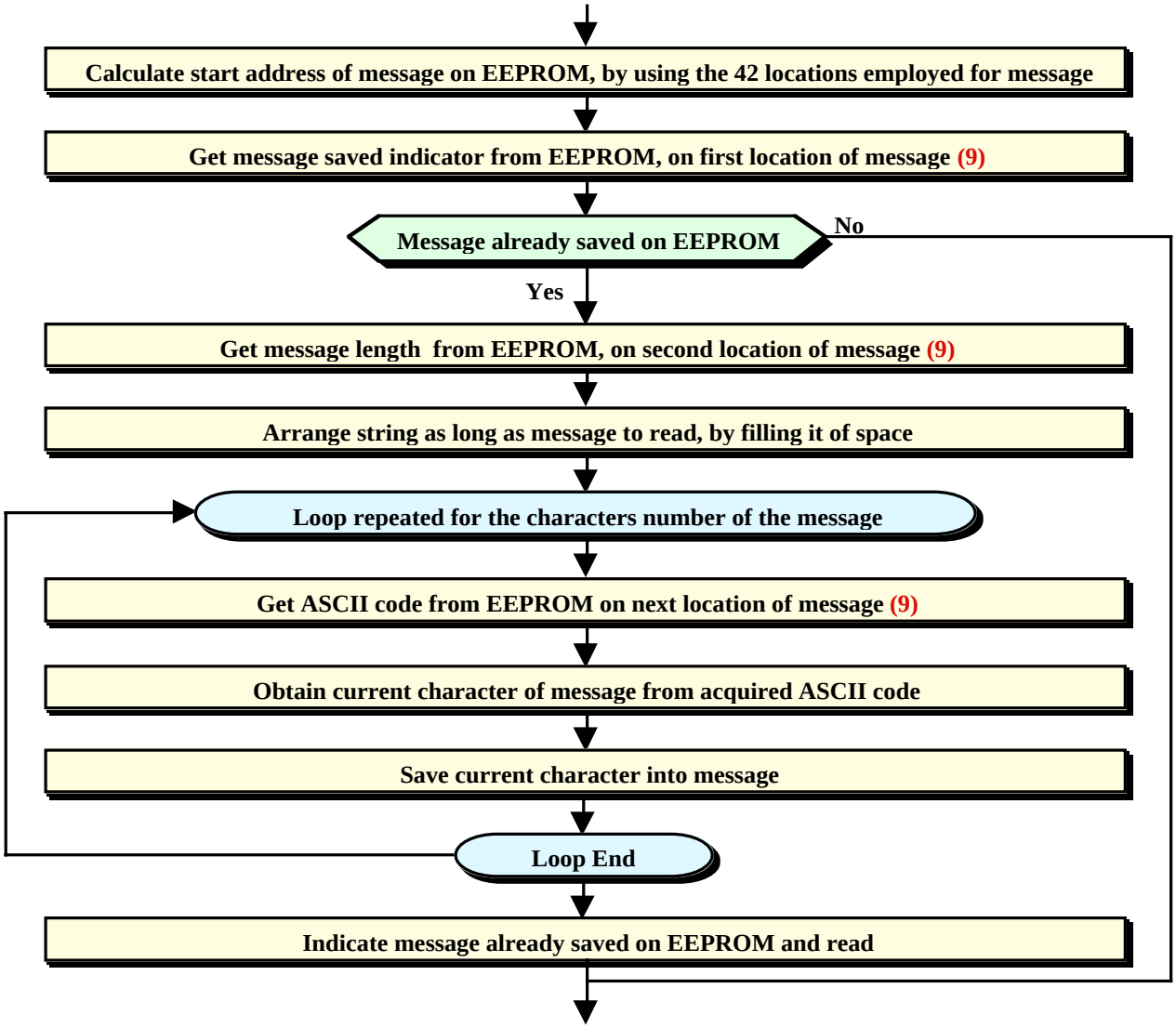
3: Generate Character in Morse.



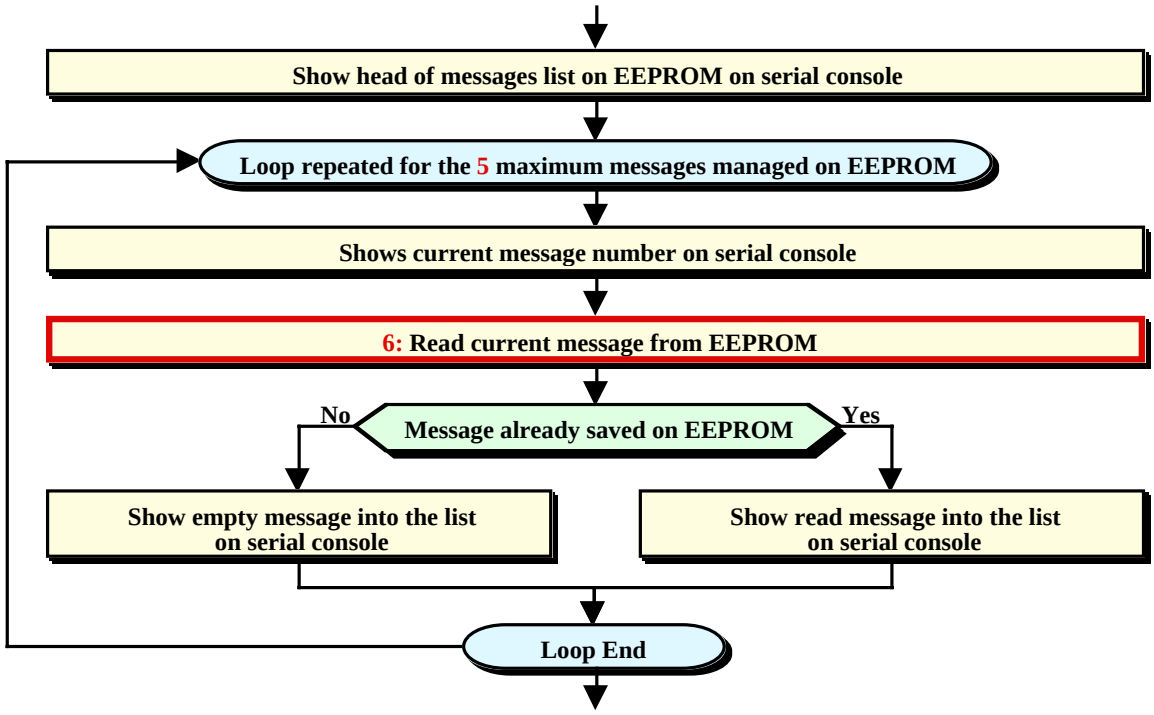
4: Perform Delay of One Time Unit.



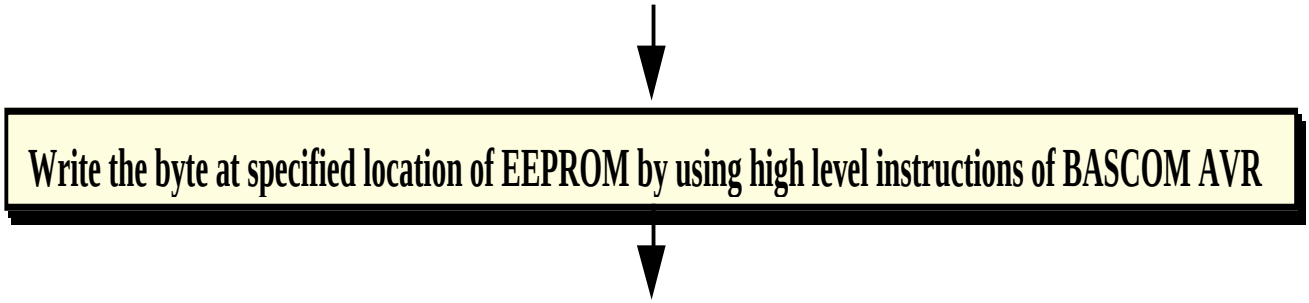
5: Save Message on EEPROM.



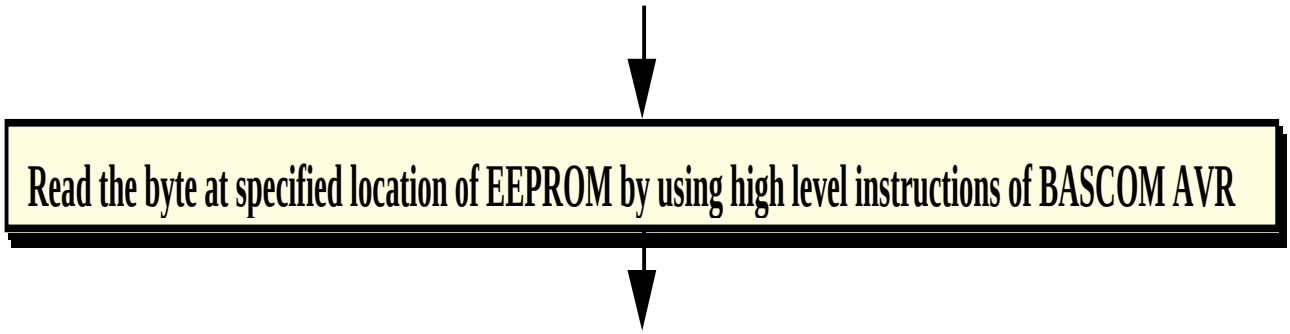
6:Read Message from EEPROM.



7: Show Current Messages Saved on EEPROM.



8: Write a Byte into EEPROM Location.



9: Read a Byte from EEPROM Location.