AUDIBLE COMPUTING, Part 3 (Forth Day Special) --- Masa Kasahara

Masa will discuss the other reason behind his project and how Forth might play an entirely different role in computer language history.

My First Encounter with Forth

- In 1984, I was involved in building a telephone message system. There were three of us: One for overall system design, one for software implementation in Forth, and one for hardware implementation, which was me.
- In 1986, I took Forth class through UC Berkeley Extension. I used FIG Forth on my Apple II computer.

The Last Two Decades of Computer Improvements:

- Hardware: Computation power has increased enormously while the price of hardware has decreased significantly.
- Internet: WWW, e-mail
- Software: Do we really have a significant change?

Why not much progress in Software Area?

- We exhausted ideas 20 years ago?
- JAVA was so hot at one point, but it hasn't saved SUN. Also, it did not create anything new. VM had been already there with p-PASCAL and JAVA syntax came largely from C++.
- Newcomers: Ruby on Rails – Any comment on these new languages?

My View on Programming Languages:

- Most of them, if not all, are visually oriented. Prof: Why don’t you read it off to your neighbors to see if he/she understands it?
- They are like mathematical equations since they are primarily designed by engineers or scientists, but **linguist**. – Why this is a problem? All humans are equipped with natural language skills which are more essential to our survival. Computer languages are not leveraging our natural ability in general.

Why can’t we achieve the same kind of mastery in computer programming?

- Look at what trained musicians can do: They can play complex music without noticeable mistakes. Some of them can create music instantaneously, called improvisations – the highest form of musicianship.
- Is it possible to achieve this in programming? If so, how? - We know that Forth programming style encourages far less bugs in programming.

How is this possible in music?

- Internalization (memorization): Musicians can do everything in his/her head – Is this possible in our computer languages?
- Historically, we had aural tradition where everything was transferred to the next generation aurally.
- There is a traditional Japanese singer designated as a national treasure by the government. He can sing off any song from his memory, over thousands of songs. Once he dies, this mastery disappears.

If you think this is only given to talented musicians, check this:

- You can strike a conversation with a stranger about any subject. Isn’t this an improvisation? This is why we can draw such an analogy between music and human languages.
- In order for this to work effectively, a certain quality should be observed, which is, a subject at any moment in your conversation is relevant to close proximity.

Now, What is Audible Computing?

- My Definition: To perform any kind of computing tasks based on audible interface rather than visual or other physical means.

How this project started for those who don’t know:

1. I acquired an old notebook computer with a broken LCD panel. It functions fine with an external monitor. But, I realized that this might be still a nice machine for data logging and/or control applications since it runs on 12V DC. I have friends who live in woods. They have solar systems and need some monitoring and controls.
2. I picked up SPARCstation 4 from the neighborhood. I didn’t know anything about Open Firmware or Open Boot, but I thought having Forth in firmware was very practical for trouble shooting.

The status of SPARCstation 4 (SS4)

- The target machine doesn’t have to be SS4, but it contains Open Boot, a sister of Open Firmware. The source code for a similar UltraSPARC Station is available, which could be helpful if I ever want to play with it.
- The booting issue resolved, which means both Fedora 3 and Solaris 8 will be operational shortly.
- Type 5 keyboard is acquired.
- I just need to verify if the mouse is operational. I also need to acquire a reflective mouse pad.

How Morse Code is Constructed: Brief Explanation

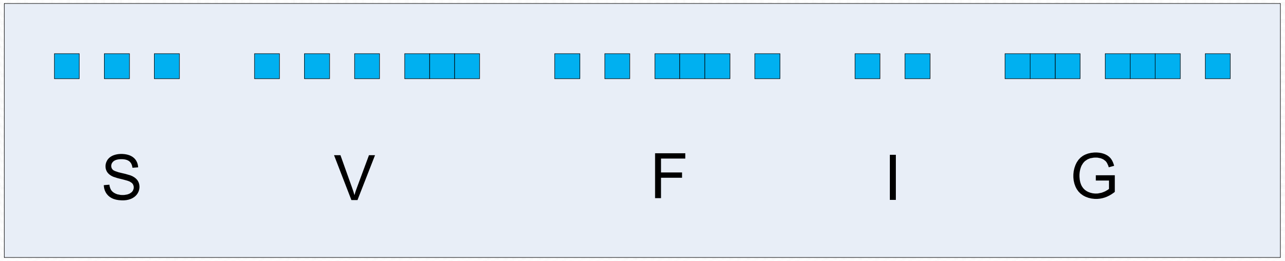
- Timing is everything in Morse Code:

Dot: 1 [Unit length, which changes according to the speed of keying.]

Dash: 3 [Dot length=Unit length]

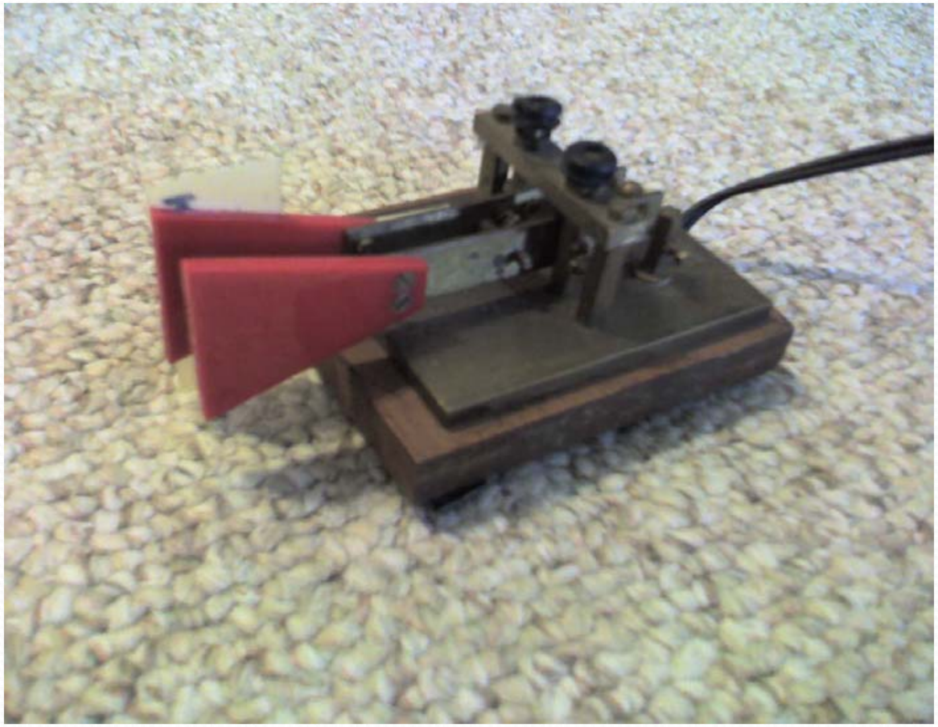
Character Space: 3

Word Space: 7



Keying Devices:

- Modern Squeeze Key – The separate input for Dash and Dot, which makes programming and keying easier.



K7SRA Key

Extended Morse Code:

- Special characters, such as space and carriage return need to be added to the standard Morse Code. I call it "Extended".
- I am in the process of finding out if the Extended Morse Code used in Darci USB is copyrighted. They call it Morse Code Plus. If not, I use the same definition. If yes, I will just come up with my own definitions. This is not an engineering issue, but I need to be careful

Why Forth?

- Most of the languages are not suited for audible computing, simply too complicated.
- Forth architecture makes it a present subject only relevant to close proximity: Think of what you can do with a stack versus what you can do with a group of registers. This could be the single most important reason why Forth programs are less prone to bugs. Of-course, you can defy this feature/disadvantage, but it makes your program very inefficient. For example, C to Forth Compiler.

Prototype 1:

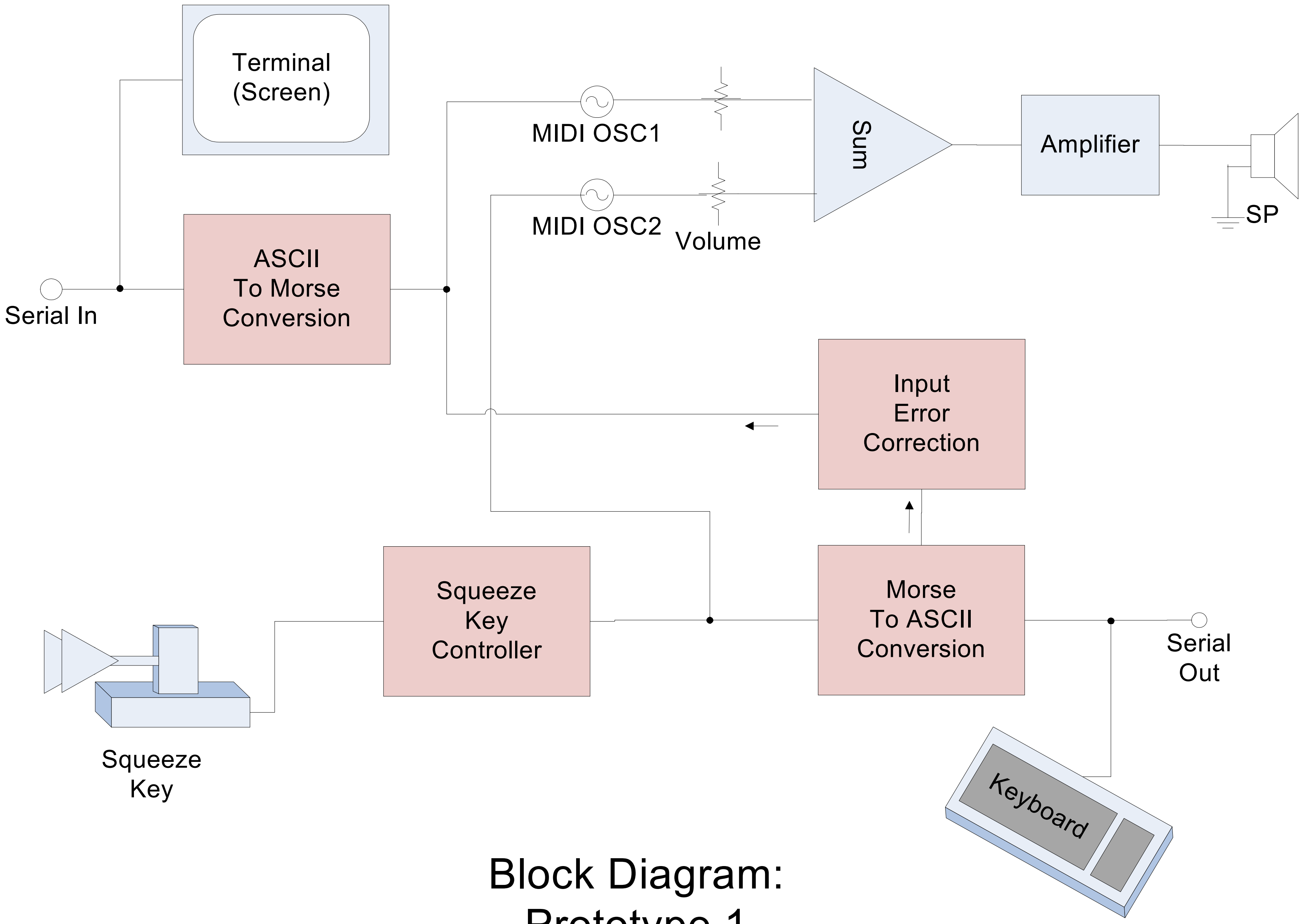
- Prototype 1 is focused on the feasibility of the concept .
- A terminal program under Windows with Morse Code Interface. This terminal program can be connected to another Forth system.
- I found a sample TTY program for Windows at Microsoft web site:

  http://msdn2.microsoft.com/en-us/library/ms810467(d=printer).aspx

  It is a fairly complete program with source code and detailed explanation of the code.

Requirements:

- There are two tones for Morse Code: one for incoming, the other for outgoing for clarity. They should be adjustable for personal preferences.
- Any unrecognized characters in input stream will be simply discarded. – No complicated terminal emulations, but a simple teletype emulation, abbreviated as TTY.
- Any unrecognized character in keying stream will be interrupted with a correction symbol for an immediate re-transmission, which makes it easier to use.



Block Diagram:
Prototype 1

Prototype 2: Porting Prototype 1 to Forth System Running under Windows:

- Prototype 2 is focused on porting Prototype 1 to Forth System running under Windows.
- This is a great opportunity to learn how Windows API is implemented in different Forth Systems.
- I am also looking for Forth System, which can be easily ported to Windows CE environment. The above study will be directly used here.

Prototype 3: Integrate Prototype 2 to Forth System.

- All the Forth routines except hardware related routines will be completed in Prototype 2, which makes the integration easier.
- Prototype 3 is conditional upon the result of the Prototype 1. If Prototype 1 is promising, Prototype 3 is continued. Prototype 2 is actually a learning opportunity for Forth System running under Windows. It will be continued regardless of the result of the Prototype 1.

Prototype 3 (Additions):

- Prototype 3 is the final step of the project and focused on actual implementation of the concept.
- The target Forth system should be modified for Audible Computing, especially in editing (or human interface) area. Preferably, implement a switch to change the mode of operation so that the original interface is kept intact. Or, if a keyboard and/or video interface does not present, it automatically switches into Morse Code mode.

Disclaimer:

- This is created best to my knowledge at the time of writing. It is, however, not guaranteed to be 100% accurate. I shall not be held liable for either direct or indirect damages by your applying the information contained here. You are at your own.

Copyright Notice: