VisLang: A graphical programming language

Bryant Eisenbach

June 9, 2015

1 Introduction

VisLang is a block diagram language designed to allow fast and easy prototyping of programs for embedded processors. The language is created with a graphical editor in mind, and the core elements of the language are extensible so that any graphical editor can add additional elements for graphical display or other features.

2 Key Language Features

The language itself is based on the idea of blocks: small parts that can be grouped together into ever larger blocks and re-used across a program or programs. A small group of fundamental (or atomic) blocks will be defined and understood by the compiler for this language. Other blocks will be constructed as configurations of these atomic blocks. A standard library of useful functions will be constructed from these atomic blocks containing common parts such as timers, latches, etc.

Side effects in the produced code will be minimized by the combination of a strong type system and bounded code execution. The type system of VisLang supports common datatypes such as bool, single, double, signed and unsigned integers, as well as static arrays and structures of these simple datatypes. Bounded code execution is guarenteed through the restriction of for loops to a static size. This works well as embedded programs should only need to parse large buffers for digital busses, which are usually defined as being a static size. The language will contain methods for defining digital message structures and how to parse those structures into usable variables. Methods for parsing both packet-based (e.g. Ethernet) and word- based (e.g. RS232) will be provided by the standard library.

Lastly, time variance will be something provided fundamentally by the language. The time-step between subsequent iterations will be maintained in every program and provided to the user as a fundamental part. Most of the fundamental parts will be time-invariant, but this language feature will provide users with the ability to create dynamic parts that will care about time as a measured quantity.

3 Syntax

4 Example Program

```
\lstinputlisting{../example/timed-blinking-light/timed-blinking-light.vs}
```

```
\lstinputlisting{../example/timed-blinking-light/timer.vs}
```