C-035

A Study of a Hardware Control Language

Ivan A. Mantchovski† Shigeyuki Ohara‡

ABSTRACT

In the computer field, hardware and software are often the two faces of the same "coin" called a device. Writing software is intrinsically dependant on the hardware specifications. However, these specifications are often incomplete and written in non-formal languages. It is difficult for software programmers to write code to control this hardware based on such specifications. This paper discusses a language that is used for defining how hardware is controlled.

Keywords

Hardware restrictions, Device drivers, Specifications language

1. Introduction

Many different kind of devices have penetrated our lives and the trend is expected only to escalate. The new devices need some software to control it. The software to control a piece of hardware is generally known as a device driver. Writing device drivers involves much knowledge and it is very error prone. Most of the time, device drivers are designed by system programmers who rely on the specifications given to them by a hardware development team. These specifications are often written in a natural language and are not standardized. This allows for mistakes and misunderstanding. Moreover, these specifications mainly concentrate on describing some registers but say nothing about hardware control restrictions. Without a clear understanding of these hardware restrictions, programmers may design device drivers that are correct from the point of view of the programming language but that are incorrect from the point of view of the hardware restrictions. These kinds of mistakes cannot be checked by the compiler and they can lead to very serious consequences, even to the destruction of the device. This paper discusses a Hardware Control Language (HCL). This language is a specification language that describes the hardware restrictions

2. Hardware Restrictions

2.1 Definition of Hardware Restrictions

We define hardware restrictions as any kind of physical or time limitations that influence the proper function of a device.

2.2 Examples of Hardware Restrictions

Here, we introduce some hardware restrictions both for actuators and for sensors as examples of basic hardware restrictions. The same examples may be used for compound devices using a combination of actuators and sensors.

Let us imagine that we have an application that sends different commands to a motor via a device driver. If this application requests an increase in the motor speed the driver will translate this request into appropriate signals and the motor will revolve faster. This works perfectly till we reach a certain limit after which the motor will be damaged. This is an example of a physical restriction.

Another example is when the above program requests from the motor that it turns left but after that changes its request for a right turn. This is also a very common situation but it will fail if the interval between these two commands is too short. This is because the motor is influenced by the laws of inertia and it cannot respond to commands given in a shorter than a given limit time. This is an example of a time restriction.

Motors are examples of actuators but we can see similar hazards for sensors. For example, let us consider a heat sensor. If the temperature rises above some threshold value, the sensor will give incorrect values and may cause damage to the whole system if its driver does not react to this situation. In this case, the driver should recognize this hardware restriction and when the temperature becomes close to the device limits, it should take appropriate actions.

3. HCL

3.1 Overview of HCL

The HCL is used to state these hardware restrictions in a clear and unambiguous way. It should be used as an important source of information before beginning to write any device driver. Its usage is especially important for

† Department of Electronics, School of Engineering, Tokai University
‡ Professor, Department of Information Media Technology, School of Information Technology and Electronics, Tokai University

double checking of safety critical systems, as car navigation, air-plane steering system, etc. By checking for hardware restrictions abuses, the HCL can guard devices against damages and thus can save money.

3.2 Usage of HCL

To be useful, the HCL needs to have the appropriate tools that will integrate it in the device driver development cycle. Here, we explain the general model for the use of the HCL.

Because the makers of a device should know its limits better than any one else, they are supposed to write the device specification in the HCL. Once the hardware specification in the HCL is ready it is passed to the system programmers who use it during device driver development process.

System programmers use the HCL as illustrated in Figure 1. The HCL source is first passed to the Documentation and Label Generator (DLG). The DLG creates a documentation paper in easy to read and understand form. It also creates Check Labels (CL).

System programmers then code the device driver source. Because there is no standard naming scheme for device drivers it can be difficult for the Device Driver Consistency Check Tool (DDCCT) to check the device driver source without some hints. The role of such hints is played by the CLs that system programmers must use during the device driver coding stage.

After device driver source is coded, it is passed to the DDCCT for a test of the consistency with the specification. If the source fails, DDCCT creates a warning message, and returns the source for another revision by the programmer. In the returned source, the failed sections are specially marked and commented, so the programmer can find the mistakes easier. If the test is successful, DDCCT returns the device driver source with a success message. All the labels are changed to comments that can be used for future reference. From now on, the device driver source can be compiled by the appropriate compiler for the language it was written in, e.g. C language compiler.

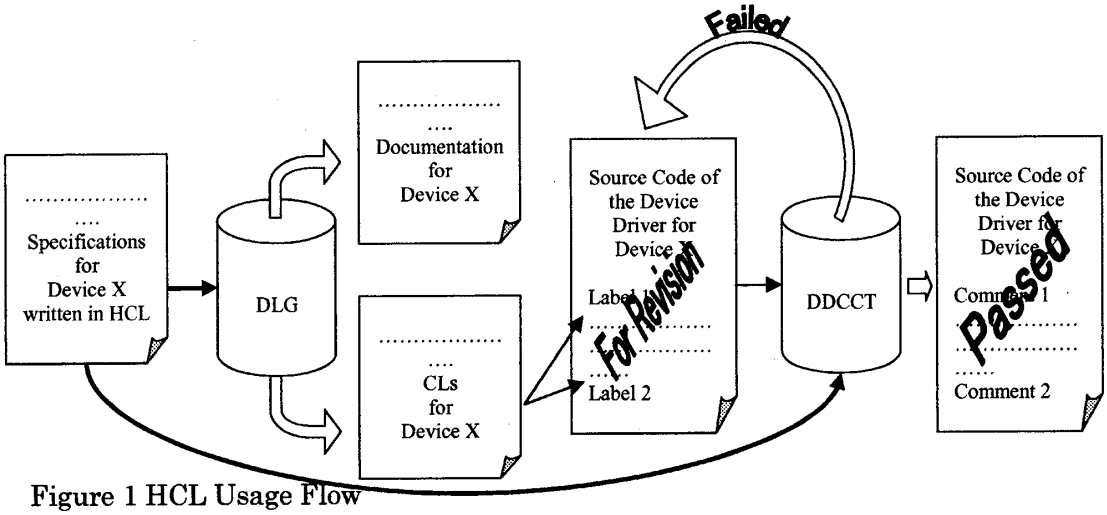

Figure 1 HCL Usage Flow

4. Conclusion and Future Work

This paper presented an introduction to the hardware control language. The next step will be to implement this language and make some tests on how much it reduces the risk for creating device drivers that may damage the device.

5. References

[1] J. Rushby, "Critical System Properties: Survey and Taxonomy", Technical Report CSL-93-01, SRI International, Feb. 1994.

[2] Che-Fn and Virgil, "A Specification and Verification Method for the Prevention of Denial of Service", IEEE Trnsactionson Software Engineering, June 1990

[3] Jan Jürjens, "Abstracting from Failure Probabilities", 2001