DIOGENE
(Digital I/O GENerator Engine)

Project Requirements

Document : SCO-DIOGENE-001-01.doc Revision : 01 Status: draft

APPROVAL

	Name	Signature	Date
Prepared by	Sergio Cigoli		11/12/07
Contributors and revisers			
Approved by			

AMENDEMENT RECORD

Date	Revision	Modified Pages	Description
	AA	All	Initial Release

Table of Contents

DIOGENE is a project proposed by Parvis Systems and Services srl for SCORE Software Engineering Contest

Parvis Systems and Services srl
via Prudenzio, 16
20138 Milano Italy
Phone: +39 02 58019193
Fax: +39 02 58029734
Milan, Italy (**http://www.parvis.it**)

ABSTRACT

In the field of industrial automation it is common to have quite long periods of system integrations on customer's site, especially due to the difficulty to reproduce the system to be controlled in a laboratory. As a result, not only system test, but also debugging could have to be done on field.
In many cases, the quality of a delivered control system and the time spent on testing on customer's site are strongly related to the accuracy and the completeness of system testing done in the software factory. Whenever target system is not available, an effective system test can be done only if a suitable target simulator is available.
The system to be designed is a programmable generator of digital signals that can be used to test a control application when final target system is not available, mainly to reproduce expected behaviour of the target, but also with the capability to simulate error cases (e.g. hardware problems or random jitters) that could actually occur when interacting with real target system.

1. INTRODUCTION

Testing of applications that require interactions with external dedicated electrical devices is often affected by the following problems: it could not be performed in the early steps of software development due to the unavailability of hardware devices and, in some cases, it cannot be fully performed on a test bench using only a subset of the whole system. In particular, testing of software applications with real-time requirements that use digital I/O boards is usually a very critical activity but it is often postponed to the end of the development process, when most of required subsystems are available. The existence of an equipment that can be connected through a digital I/O cable, and then programmed to fully stress even a subset of the software product to be delivered, should lead to more extensive system tests and it could help to deliver more dependable products.

The following sections provide

- Application domain presentation.
- Design requirements.
- List of expected deliverables.
- Guidelines for developing teams that need support.

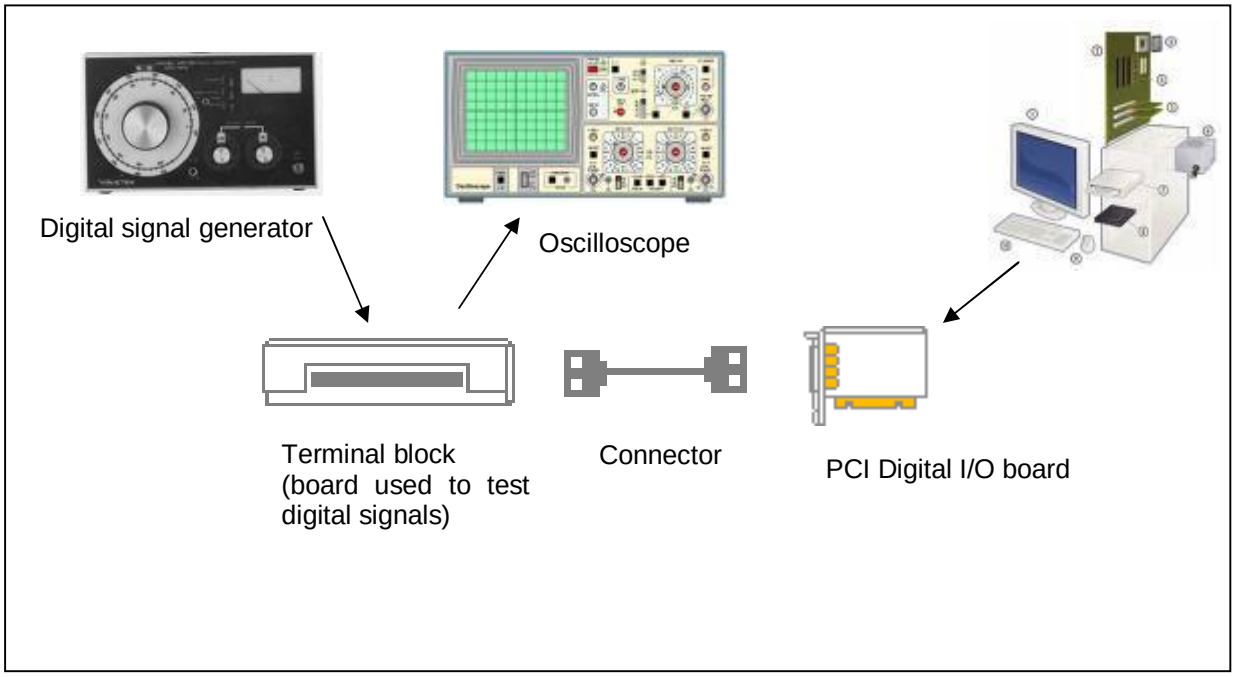
2. APPLICATION DOMAIN AND SCENARIOS

The output of this project is expected to be especially helpful to manufacturer of industrial applications, providing the capability of performing more extensive lab tests through simulation, improving robustness of delivered products and reducing field test duration. Even device driver designers should have great benefits using a programmable generator of digital signals.
A typical scenario in which such digital lines simulator is really useful could be an industrial process that has to be controlled by means of a set of actuators whose activation is driven by a sequential logic based on the status of input sensors. In cases like these, it is hard to test a control application in a lab without a tool that can be programmed to reproduce the interactions that occur during tests on field.
For example, handling of items on a conveyor requires both object tracking and device activation: in this case there is a tight correlation between actions and feedbacks from sensors. Sensors should detect an item only at a particular step of the transport sequence and only if actuators have been driven with a certain action. Following the sequential logic of the process, it is easy to understand that there are sets of I/O states that actually represent the real system behaviour and a programmable simulator should be capable to reproduce them, including simulation of possible sensors failure.
The following sections provide more details about project requirements and expected deliverables.

3. PROJECT GOALS (REQUIREMENTS)

3. PROJECT GOALS (REQUIREMENTS)

This section includes a set of required features for the application divided into functional requirements (required functions or behaviour), non-functional requirements (criteria used to judge required behaviour, e.g. performances) and project requirements (e.g. constraints for hardware/software choices). As specified below in project requirements, target system should be a standard PC with a digital I/O card.



3.1 Functional requirements

This chapter is organised in two subsections, the first providing a set of general requirements, the second including a list of more specific features better defining the general requirements addressing the actual simulation needs. In fact for most of the industrial processes to be simulated there is no need of such flexibilty and specific features can be used instead of the correspondent “generic” ones.

Basic features

3.1- 1. Main goal of the DIOGENE application is the capability to generate programmable digital signals that are sent out by means of the output lines of a digital I/O board mounted on the target system.

3.1- 2. Application should be programmable in order to provide digital output sequences related to other I/O lines by means of a sequential logic.

3.1- 3. In addition to the output lines management, the application should have the capability of monitoring state changes on the input lines of the digital I/O board.

3.1- 4. Application shall be able to monitor state changes of all available input lines.

a. A graphical display should provide a list of input lines and their updated value.

b. The user interface should handle a configurable number of input lines.

3.1- 5. Application should have the capability to activate each single digital output line. The "active" state can be defined either from raising edge to falling edge or viceversa.

a. A configurable setting should define the digital level that is defined as "active".

3.1- 6. It shall be possible to stop and restart the generator of output signals (in case of "stop" all output lines should be deactivated.

3.1- 7. Application should provide a random jitter for each output line. This jitter should be disabled by default and could be enabled for a specified output line by changing the configuration settings. Also maximum jitter should be configurable.

3.1- 8. Programming of output lines settings should be done through a user friendly interface. Any change in output line programming should not require application restart.

3.1- 9. Graphical user interface should display the status of both input and output lines in the main panel.

3.1- 10. (Optional) Application should provide the capability of assigning a symbolic name to each I/O line and display it in user interface.

3.1- 11. (Optional) A debouncing mechanism could be applied while reading input lines (e.g. notify a state change if the state transition is confirmed by three consecutive read on input port).

Features that address simulation of industrial processes

The purpose of this section is to restrict specifications of requirement 3.1- 2 since they could be too general, while only a subset of them is actually useful for simulation of industrial processes. Developing teams are requested to focus their design on the features that follow instead of considering the general-purpose solution specified in 3.1- 2.

3.1- 12. Application should have an internal clock with variable frequency: state changes on output lines should be synchronized with it

a. Clock frequency should be changed through runtime adjustments done with user interface. It should be possible to decrease the clock speed to zero.

3.1- 13. The state of each available output line On shall be programmable in one of the following ways:

a. Counter signals, activated for CW_n clock pulses every CN_n clock pulses.

b. Fixed value signal, always active or not active, with the possibility to apply manual changes though graphical user interface.

c. Delayed signal that is related to a reference signal (i.e. a specified input or output line) and it is activated with a programmable delay from the reference signal. In particular, delayed signals are activated with a delay of DD_n clock pulses and with a duration of DW_n pulses from each activation of a specified reference signal. As an example, Figure 1 shows a delayed signal related to a counter signal, please note that each pulse of counter signal programs a delayed impulse.

d. Conditioned delayed signal, as delayed signal, but it is activated only if a specified input or output line is also active at evaluation time (i.e. when related counter signal is activated). NOTE: programming of a delayed signal shall be done whenever a counter signal is activated and only if another specified line is active at evaluation time. Logical AND operator is applied at evaluation time.

e. **Boolean operations** (OR, AND, NOT) applied to input or output lines (max. 2 lines).

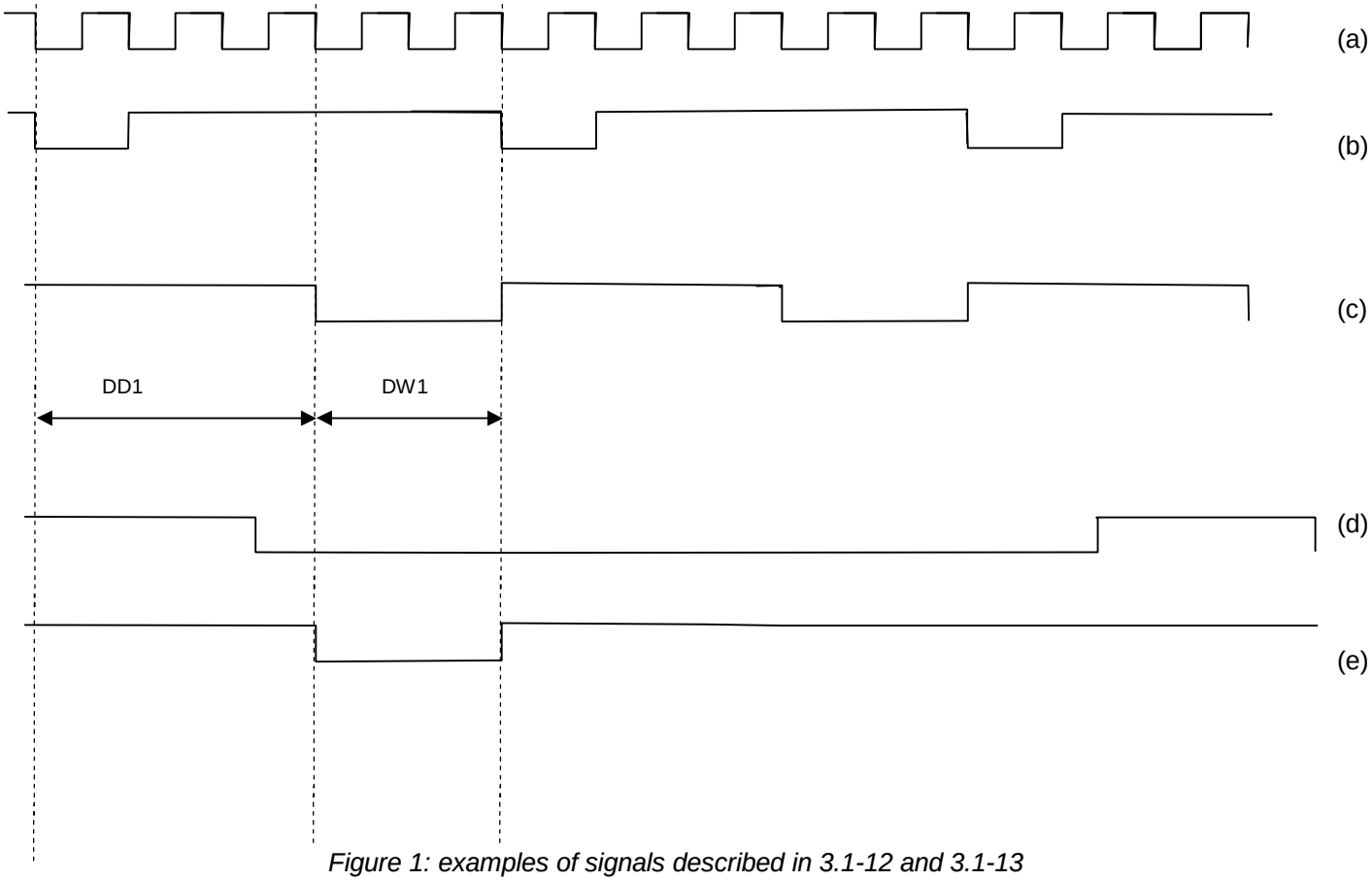


Figure 1: examples of signals described in 3.1-12 and 3.1-13

(a) internal clock
(b) counter signal programmed to be active for CW1=1 clock pulses and every CN1=4 clock pulses
(c) delayed signal related to (b): state changes with a delay of DD1 =3 pulses from (b) , new state is kept for DW1=2 pulses
(d) generic input signal
(e) conditioned delay ed signal: as (c) but state change depends on (e) value

3.2 Non-functional requirements (Performances)

System performances are strongly related on chosen hardware, in this section minimum and desired performances are listed. Regardless of the chosen hardware platform, software solutions that maximize system performances will be valued as a plus.

3.2- 1. Application has soft-real time constraints: no blocking error is raised if a maximum response time is exceeded, but the whole system has to be designed in order to respect in most of the cases the response times specified in the following requirements.

3.2- 2. Application should be capable to handle input signals with a frequency of at least 500Hz (desired goal 1 KHz).

a. Delay in displaying changes in input lines is not relevant but it should be lower than 0.1s

3.2- 3. Application should be capable of providing at least a clock frequency range of 0-500Hz (desired goal 0-1 KHz).

a. Frequency adjustments can be discrete.

3.2- 4. Maximum allowed delay while generating output signals is 2ms (desired goal: 1ms).

3.2- 5. User interactions through GUI should be minimized in order to reduce any possible overload on generation of output digital signals.

3.2- 6. Handling of interrupts is not required.

3.2- 7. Application should be capable to handle several types of I/O cards with different numbers of I/O lines: software design shall be done in order to minimize changes when a different I/O card is used. Applications should support up to:

a. 128 input lines.
b. 128 output lines.

3.3 Design requirements

The design requirements are grouped in two subsets. The first one including a set of mandatory requirements, the second one including a set of non mandatory requirements that nevertheless will be part of the evaluation criteria of the submitted projects.

Mandatory requirements

3.3- 1. Target system should be a standard PC with a digital I/O card.

3.3- 2. Minimum requirements for digital I/O board are the following:

a. At least 16 input lines.
b. At least 16 output lines.

3.3- 3. No specific requirements are provided for the workstation, a standard desktop computer (better if with dual-core processor) could be suitable for this application.

3.3- 4. No specific requirements are provided for the operating system (OS) and for the software development environment, developing teams are requested to assess available OS and tools and to explain the reasons of the chosen solution.

3.3- 5. Application should have a graphical user interface that can be used either to monitor or to program output lines. The graphical interface should require a monitor with maximum resolution 1280x1024 pixels.

a. Resizable windows interface is not required.

3.3- 6. In order to test the application, an oscilloscope and a signal generator could be used.

3.3- 7. (Optional) a simulation environment could be implemented to run application without digital I/O board either to ease application debugging or for demo purposes.

Non Mandatory requirements

3.3- 8. Ease of use: application interface should be user friendly and it should be easily programmable.

3.3- 9. Stability: in a lab test the application could run for a whole day.

3.3- 10. High configurability: user should be able to program a signal for each available output line.

3.3- 11. Hardware flexibility: it should be easy to replace a digital I/O board with a similar one. It is not necessary to have a “plug & play” application and some code changes when replacing I/O card are allowed, but they should affect only one software module.

3.3- 12. Good product documentation for end users: user manual should be clear enough to allow a fast learning of output signals programming, while reference manual should provide enough details on how to replace an I/O board with a different one.

4. INTENDED OUTPUT OF THE PROCESS (PROCESS FOCUS)

4. INTENDED OUTPUT OF THE PROCESS (PROCESS FOCUS)

The following deliverables are required:

- System design documentation (a system design description and a software system design description).
- Manuals for end user (user manual and reference manual).
- A demonstration of system main system features (for demo purposes, a simulation environment without digital I/O board can be set up).
- A brief report about used test bed (i.e. chosen hw/sw solutions) and measured system performances.
- For the project finalists a full demonstration with hardware components could be required.

System design documentation

- System Design Description (SDD) starts from requirements and provides a high level description of the whole application. This document should include at least:
    - assessment of possible HW/SW solutions and an explanation of the chosen solution,
    - any further constraints (e. g. related to the chosen solution),
    - a block decomposition (the use a graphical notation, preferably UML, is appreciated),
    - a data flow analysis,
    - data structures,
    - test plan (identify a list of tests that can be used to demonstrate that requirements are met),
- Software System Design Description (SSDD) includes low level implementation choices. This document has to be a reference for any developer that could be committed to make some changes to the source code. Use of UML diagrams and flow chart for complex algorithms is encouraged, while the use of and code portions inside this document should be avoided. This document should also include a list of known limitations and implementation constraints, if any.

Manuals for end user

- User manual explains how to use the program, including a detailed guide of how to program output signals; this document should not contain implementation details.
- Reference manual explains what has to be done if digital I/O board is changed (e.g. how to adapt source code). Readers of this document should have programming skills but they could not be aware of implementation details.

Details about demonstrations and final report on performances

To be defined with the contact person of Parvis Systems and Services srl.

5. COMPONENTS, TOOLS AND STANDARDS

5. COMPONENTS, TOOLS AND STANDARDS

The only recommended standard in project documentation is UML.
About the tools this document gives no recommendation about hw/sw choices, but maybe some of the considerations listed below could be helpful.

Workstation
Use of a desktop computer with high performances (e.g. dual-processor) is recommended.

Digital I/O board
Parvis Systems and Services srl is currently using the following I/O boards:

- National Instruments PCI 6509 Digital I/O board equipped with NI terminal block,
- Advantech PCI 1751.

Both the boards widely meet previous requirements.
Advantech board costs about 200 Euros and it is probably one of the cheapest commercial boards that are compliant with requirements of Parvis projects. NI board and also many other similar boards available on the market cost about 700-800 Euros. For the purposes of DIOGENE project, the features that can be found only in top level boards (like NI PCI 6509) are not relevant, any board that is compliant with requirement 3.3- 2 can be used. Lowest prices for suitable I/O boards, including cable and terminal block, could be around 150 Euros (example: 156 Euros for Omega OMG-PCI-DIO32 - 32 Channel Digital I/O Board).
In addition to price and to the number of I/O lines, developing teams should also assess availability of device drivers for chosen operating system.

Operating systems
Use of real-time operating systems is appreciated but it should not be mandatory, the soft real time requirements of DIOGENE application allow the use of a standard OS on a workstation with high performances. Teams should assess the OS choice on the report of system performances.

Programming language
Parvis Systems and Services srl is currently using C/C++ for application engines and C# for user interfaces; no recommendation is given for developing teams.

6. INTERACTION BETWEEN STAKEHOLDER AND DEVELOPING TEAMS

Teams are encouraged to interact with the contact person via email (Giovanni De Toni, detoni@parvis.it). In particular, the student teams will be required to submit a project plan or statement of work before beginning.
The preferred communication media is email, but a Skype teleconference can be set up by a participating team once throughout their development time.
Information to the student teams (e.g., for answering requirements questions) will come primarily from the point-of-contact person via e-mail.

Teams are also given the possibility to perform tests in Parvis laboratory in Milan, Italy (only once during their development time) using one of the digital I/O boards listed in the previous chapter. Because of resources constraints a restricted number of teams can be hosted at Parvis facilities. Teams are required to contact the Parvis reference person to schedule their activities in the labs.
Testing in Parvis lab is of course a good opportunity for teams but this shall not be an advantage in project evaluation criteria.

The expected duration for the project is about 4-6 months and it shall be completed within Jan 09, in case of changes in project deadline teams will be informed as soon as possible.