CS 361 Project Description

ChronoTimer 1009

Your team has been tasked with building the software portion of a sports timing product to be sold to sports teams and clubs. A simple hardware platform has been proposed that will be inexpensive to build and be able to withstand the abuse encountered at meets and competitions.

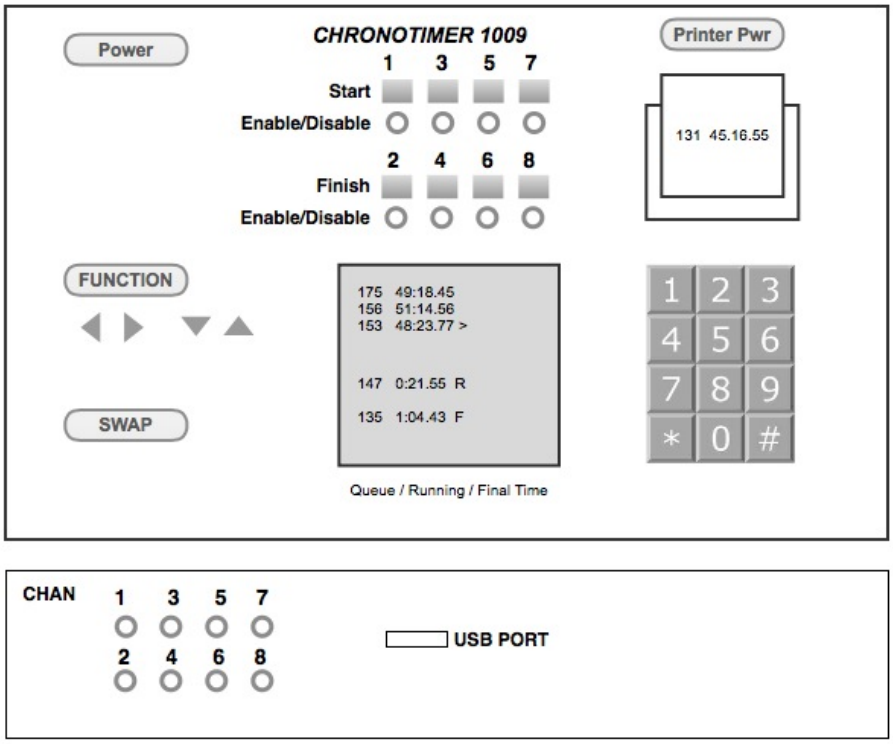
The hardware development is just getting underway, so the actual hardware will not be available to use in testing until much later in the project. However, the interface between the hardware and software has been well defined and you will able to develop your system as long as it is compatible with the interface definition. The hardware team has elected to use a simple processor embedded in a field box with a display, printer, and various keyboard controls. To make it easier and faster to build, your team has also been asked to build a simulator - a system that will exercise the software in a real-world manner. The product management team has written up the following requirements for the system. They will write the user instruction guide.

Requirements

Overview

Timing for many sports events involves recording a start time and end time to yield a time for a run, leg or lap. This can be used in ski races, swim meets, cross country races, bobsled runs, or similar events. In all cases, a start is generated by either a start gate (or electric eye) or a manual start, and the finish is recorded when the competitor trips the sensor at the end. The sensor could be an electric eye or a touch pad. A finish signal can also be generated manually from the timing system.

Physical Layout



Units of Time

Timing is done in hundredths of seconds ranging from 0.00 to 9999.99 seconds. A run time is calculated by subtracting the system time at the start from the system time at the moment the finish trigger was tripped.

For example:

Start: 00034.03 seconds
End: 00073.48 seconds

Time: 00039.45 seconds

When the system is initialized, the system time is set to the current internal clock that is maintained in firmware.

Types of Competitions

There are several types of competitions (events) that occur:

IND Individual timed events (such as ski races, bobsled runs)

In the IND events, racers queue up for single "runs" of the race. Each racer has a start event and end event. By *convention* the start is on channel 1, the finish is on channel 2, but there is nothing particularly special about the events on each channel. When a start event is

received, it is associated with the next racer in the start queue. When a finish event is received, it is associated with the next racer that is to finish. If there is more than one racer active, the finish event is associated with racers in a FIFO (first in, first out) basis.

PARIND Parallel events (e.g. skiing) with individual starts

In the PARIND events, racers queue up for parallel timed "runs" of the race. Each racer has a start event and end event. By *convention* the start is on channel 1 and channel 3, the finish is on channel 2 and channel 4, but there is nothing particularly special about the events on each channel. When a start event is received, it is associated with the next racer in the start queue. When a finish event is received, it is associated with the next racer that is to finish. If there is more than one racer active, the finish event is associated with racers in a FIFO basis.

GRP Group events (cross country skiing or running) with individual finishes

In the GRP events, racers all start at the same time (such as a gun or horn) so all have the same start time. By *convention* the start is on channel 1 and there are multiple finishes on channel 2, but there is nothing particularly special about the events on each channel. When a start event is received, it is associated with the start of the event. When a finish event is received, it is associated with the next "place" – for example, the first finish is entered in the list of finishers with the first time, the next time with 2nd place, etc. There may be up to 9,999 places. Bib numbers may be associated with places by entering the bib numbers in the order of finishing (such as a cross country race).

PARGRP Group parallel events (swimming) with one start and individual finishes.

In the PARGRP events, racers all start at the same time (such as a gun or horn) so all have the same start time. By *convention* the start is on channel 1 and but there are multiple finishes on channels 1 through 8. When a start event is received, it is associated with the start of the event. Each racer is associated with a lane, numbered 1 through 8. As a racer number is entered it is associated with the next lane number (first is lane 1, second lane 2, etc). The start is conducted by doing a manual trigger on channel 1, and finishes are recorded for each lane. After all racers have finished, a new set may be entered, and a new start initiated. If more than 8 numbers are entered, they are ignored.

Input Sensors

Any sensor connected to the system generates a single "trigger" event that indicates that the sensor has been activated. All sensors can be armed or disarmed. Input from a sensor can be blocked in the timing system, regardless of its armed state. For the first version of our system, we can have two channels, but this may increase up to 12. By convention, the odd number channels are start and the even numbered are finish sensors. For example, channel 1 would be the start and channel 2 would be the finish. This allows the sensors to be used for different purposes in different types of competition. Each channel has a button on it that can be used to force an event (as if the sensor had been triggered) for a start or finish.

There are several types of sensors:

1) Push button on the console that corresponds to the channel
2) An "electric eye" that triggers when the light beam is interrupted
3) A "gate" or micro-switch that triggers when the wand is pushed open.
4) A pad that can be hung over the edge of a pool that triggers when pressure is applied.

Note: Each of these should have emulator that actually generates the triggers for the event so that the main system can be tested as if the event actually occurred.

Operation of Unit

The unit can be turned on and off through an ON/OFF switch. Upon power up, the timing system is ready to be enabled. Initially all of the input channels are disabled (meaning they will ignore any signal coming from a connected device). Enabling the channel will allow the corresponding signals to trigger events on the timing system. Disabling the channel will cause all inputs to the channel to be ignored.

The printer can be turned on or off with a separate switch. If the printer is off, events are still logged in the event log. Turning the printer on at any point allows printing from that point on, but no previous events are printed.

Display of Time

The time of a run is displayed at the timer console, and can also be displayed on a graphical display. The content of the display depends upon the type of event that is being timed.

Note: For the first version of our system, the time only needs to be displayed on the console and paper tape.

The display should show the racers queued for start, the racers' times that are actually running, and the finish time of the last racer.

<NUMBER 5> <SYSTEM TIME>
<NUMBER 4> <SYSTEM TIME>
<NUMBER 3> <RUNNING TIME>

<NUMBER 2> <RUNNING TIME>
<NUMBER 1> <FINISH TIME>

Commands

POWER	(if off) Turn system on, enter quiescent state
POWER	(if on) Turn system off (but stay in simulator)
EXIT	Exit the simulator
RESET	Resets the System to initial state
TIME <hour>:<min>:<sec>	Set the current time
TOG <channel>	Toggle the state of the channel <CHANNEL>
CONN <sensor> <NUM>	Connect a type of sensor to channel <NUM> <sensor> = {EYE, GATE, PAD}
DISC <NUM>	Disconnect a sensor from channel <NUM>
EVENT <TYPE>	IND | PARIND | GRP | PARGRP
NEWRUN	Create a new Run (must end a run first)
ENDRUN	Done with a Run
PRINT <RUN>	Print the run on stdout
EXPORT <RUN>	Export run in XML to file "RUN<RUN>"
NUM <NUMBER>	Set <NUMBER> as the next competitor to start.
CLR <NUMBER>	Clear <NUMBER> the competitor from queue
SWAP	Exchange next two competitors to finish in IND
DNF	The next competitor to finish will not finish
TRIG <NUM>	Trigger channel <NUM>
START	Start trigger channel 1 (shorthand for TRIG 1)
FINISH	Finish trigger channel 2 (shorthand for TRIG 2)

Approach

The use of an event loop will be valuable in creating this application. It allows us to simulate (to great accuracy) the input of random events and organizing the processing of the events. Assume that the events can easily be processed within each cycle of the loop. For example, if the granularity of the loop is seconds, each cycle of the loop represents a tick of a second hand. If it is hundredths, each cycle of the loop is a hundredth tick. It's your way of not letting everything happen at once. Each pair of start/stop can be identified by a competitor number that can range from 0 to 99999. A competitor may have more than one time in the system, but not in one heat or run.

If the OFF command is encountered, your timer will stop (gracefully). If EXIT is encountered, the simulator will end and exit.

Storage

A full record of events is stored in the timer until the system is reset.

Times are stored for the duration of the race. The times may be exported using the following format:

<TIMESTAMP> <EVENT>
<NUMBER> <EVENT> <EVENT TIME>

Event Codes

Sometimes a competitor start must be cancelled or otherwise marked (Did Not Finish for example). Event codes are:

<competitor> TRIG <n>
<competitor> ELAPSED <time>
<competitor> CANCEL Competitor started, but the start was not valid. The competitor is still in the queue to start.
<competitor> DNF Competitor started, but did not finish and never crossed the finish line.

Default States

The default type of competition is IND.
The default run number is 1.
Channels default to "disarmed" on power up.

Format of Test Input

The system shall be able to read commands from either a test file or the console to drive the system as if it were being driven by actual events.
The format of the input is:

<TIMESTAMP> <CMD> <ARGUMENT LIST> <EOL>

Note: If the input is coming from the console (real time typing), simply use the time the line was entered and prepend it to the command (see below).

Running the Test Environment

You should have three main elements to your executable environment: the ChronoTimer unit, sensors that generate events to the ChronoTimer, and a shell that grabs commands and drives the event generation (you will see a common pattern emerge in the design of this). You may use the standard output as your "paper tape" for your printer.

This project will be done in four parts in four sprints. This will allow the team to develop the system in an iterative manner.

You will want to develop the system to be testable. For testing purposes, you should be able to read the commands from both the console (standard input) and from a

file. If they are read from a file, each command will have a time stamp in front of it in the form <MIN>:<SEC>.<HUNDRETHS>. For example:

1:23.47 START
1:25.23 FINISH

At each release a test file will be supplied for you to test your system against. The test output will be part of your deliverables for the release.

Release 1.0

The first release of the timing system allows one stream of racers (IND) that can start, finish, cancel and not finish (DNF). These are individual times (not parallel or group). Events will be generated by reading in from the command file or console. No GUI required.

Release 2.0

New features for second release include multiple channels and export of data to the file (connected on USB) and display on console. No GUI.

Release 3.0

The ability to handle group races such as cross-country races where there is a single start and a series of single finishes (first, second, third, etc); additional features and maintenance including various displays. Integrate with GUI.

Release 4.0

Ability to handle parallel group races (such as swimming) and send updated results to supplied web server.

Deliverables

For each "release" - sprint, you should submit the following items to D2L for your team:

Use case descriptions

A set of use case descriptions and diagrams for each system use case including the actors using the format discussed in class.

Domain model

An updated UML diagram of the key abstractions in the system and their relationships.

Sprint Retrospective

How did things go? What would you keeping doing? What would you stop doing? What would you change? What were the contributions of each team member (detailed).

Test plan

A test plan for each iteration, following the style discussed in class.

Commented code

Code should be well organized, well documented (JavaDoc compatible) and reasonably formatted submitted in .ZIP format.

Output from Simulation

Using the supplied input file, provide a file with the output from running your simulation. The output should be able to be reproduced on demand (i.e. no editing the test output!).