AgilityContest 3.9.X

API del interfaz de comunicaciones serie para cronómetros con puerto RS232

Autor: Juan Antonio Martínez <info@agilitycontest.es>
Versión: 1.3 Julio 2019

El siguiente documento indica el protocolo de comunicaciones que maneja el programa interfaz de comunicaciones serie que permite conectar cronómetros que dispongan de dicho interfaz a la aplicación AgilityContest

Contenido:

Historial de cambios del documento:

1.3 Julio 2019
- Añadido comando “BRIGHT” para control del brillo del display
- Instalación de la aplicación
- Descripción del interfaz web

1.2 Junio 2019
- Añadido comando “DATA” para indicar F:T:R:E:N de una sola vez

1.1 Junio 2019
- Nuevos comandos y correcciones a los existentes
- Informacion sobre consola interactiva
- correcciones a las opciones de linea de comandos

1.0 Mayo 2019
- Versión preliminar

Introducción a la aplicación de gestión de cronómetros por puerto serie de AgilityContest

AgilityContest incluye un API de comunicaciones basado en peticiones AJAX a través de internet. Esto significa que cualquier dispositivo conectado en red local a un servidor ejecutanto AgilityContest, puede convertirse en un nodo de la aplicación

La conexión a ethernet, desgraciadamente no es el método más habitual de conexión al PC de los diversos cronometros que existen en la actualidad: la gran mayoría utilizan un puerto de comunicaciones serie (DB9 o USBSerie) y programas específicos de control

Los objetivos de la aplicación de cronómetro son:

1- Ofrecer un interfaz común de control (basado en una aplicación web) que permita la visualización de tiempos y control de las operaciones de cronómetro, y que pueda ser utilizado como pantalla-cronómetro en una competición

2- Ofrecer un interfaz de comunicaciones entre el puerto serie y el API de eventos por red local que ofrece AgilityContest, de manera que cualquier cronómetro con puerto serie se pueda comunicar con la aplicación

3- Ofrecer un API común de comunicaciones serie basado en un set de comandos ASCII, de manera que cualquier cronómetro que lo utilice pueda integrarse de manera directa en el sistema

4- Ofrecer la posibilidad de añadir módulos específicos (bibliotecas .DLL) para aquellos cronómetros cuyo protocolo de comunicaciones difiera del API propuesto

5- Incluir una consola de comandos de manera que el usuario pueda, en modo texto llevar control del sistema

La aplicación consta de un ejecutable, y una o más bibliotecas DLL, cada una de ellas correspondiente a un modelo determinado de cronómetro. En el momento del arranque el usuario define el ring, el dispositivo y el puerto serie que va a utilizar, y el programa carga la biblioteca correspondiente

La función de las bibliotecas es la de traducir los comandos enviados/recibidos desde/hacia el cronómetro en comandos del API de comunicaciones serie, de manera que el resto de los sub-sistemas utilicen un protocolo común

La aplicación está escrita en C. Se ofrecen ejecutables para Linux, Windows (Win32) y MacOSX (en proyecto)

Instalación:

AgilityContest Serial Chrono se distribuye en formato de fichero zip, cuyo contenido es el siguiente:

AC_SerialProtocol.pdf	- Manual del usuario
LICENSE	- Licencia y CopyRight
serial_chrono.ini	- Fichero de configuracion
html	- Carpeta que contiene los diversos ficheros del interfaz web
SerialChronometer.exe	- Ejecutable para Windows (Win32 Api - windowsw 7 o superior)
generic.dll	- Biblioteca dinámica (Windows DLL) para el cronometro Genérico
digican.dll	- Biblioteca (Windows DLL) para el cronometro "Digican (serie naranja)
SerialChronometer	- Ejecutable para Linux (64 bits)
generic.so	- Biblioteca dinámica (Linux .so) para el cronometro Genérico
digican.so	- Biblioteca (Linux .so) para el cronometro "Digican (serie naranja)

Para instalar la aplicación procederemos de la siguiente manera:

1- Descomprimir el fichero zip en el disco duro local en una carpeta en que el usuario tenga permiso de escritura. Se recomienda la carpeta de usuario o bien c:\AgilityContest

2- Editar con un editor de textos (vim, edit, Wordpad) el fichero de configuración conforme al equipo y al manual de usuario

3- Conectar el cronómetro al puerto serie seleccionado

4- En su caso, arrancar la aplicación AgilityContest. Se recuerda que para su uso combinado se precisa de una licencia con soporte para cronómetro electrónico

5- Arrancar la aplicación lanzando desde un interprete de comandos (cmd.exe en Windows) el ejecutable correspondiente. En caso de haber habilitado en el fichero de configuración el servidor web local, es posible que el firewall de windows presente un aviso de alerta, que podremos aceptar e ignorar

6.1 - Si se utiliza AgilitContest, desde el menú de control de sesiones comprobar que el cronómetro es reconocido, y que se puede controlar desde el tablet

6.2 - Si se utiliza como aplicación autónoma, desplegar un navegador en la dirección http://localhost:XXXX+Ring (siendo X el valor escogido en el fichero de configuracion para “web_port” , esto es: el puerto TCP en el que el servidor web interno de la aplicación atenderá a la consola web)

Por ejemplo: si web_port vale 8000 y el crono se va a utilizar en el ring 2, en el navegador utilizaremos la URL: http://localhost:8002

6.3 - Si se activa la consola de comandos, la pantalla presentará en la ventana del interprete de comandos un prompt "cmd>" pudiendo interactuar con la aplicación introduciendo directamente los comandos definidos en la documentacion ("SerialAPI")

Configuración:

Mediante un editor de texto (tipo Vim/Edit/WordPad o similar) podemos acceder al fichero de configuración y en su caso ajustar los parámetros a los valores deseados:

[Global]	
local_port = 8880	Puerto UDP para comunicaciones internas entre módulos
console = 1	Activar la consola de comandos (1:si 0:no)
ring = 1	Ring que va a contolar el cronómetro (1..4)
[Debug]	
#logfile = c:\\windows\\temp\\serial_chrono.log	Fichero donde se guarda el registro de eventos. La carpeta donde reside
logfile = /home/user/tmp/serial_chrono.log	debe existir y el usuario tener permiso para crear ficheros en ella
loglevel = 8	Nivel de depuración (0 .. 8) (Ver la sección “SerialAPI”)
verbose = 0	Enviar también la salida de depuración hacia la consola de comandos
[Server]	
ajax_server = localhost	Direccion IP donde se ejecuta el programa AgilityContest
client_name = serial_chrono	Nombre con que se identifica el programa ante AgilityContest
[Serial]	
module = digican	Módulo (.DLL) de comunicaciones a utilizar
#comm_port = COM1	Puerto de comunicaciones (Windows)
comm_port = /dev/ttyUSB0	Puerto de comunicaciones(Linux)
baud_rate = 115200	Velocidad de transferencia en el puerto serie
[Web]	
web_port = 8080	Puerto (HTTP) donde arrancar el servidor web autónomo. 0: desactivar

La aplicación utiliza el puerto UDP “local_port” + ”ring” para establecer comunicaciones internas entre los diversos componentes. De esta manera se pueden iniciar varias instancias de la aplicación, cada una escuchando en un ring distinto

Se recomienda revisar el firewall para abrir los puertos UDP correspondientes

Los valores para “web_port” y “local_port” no deben ser menores de 1024, pues dicho rango de puertos está reservado por el sistema, y requiere de permisos de Administrador para su uso. No hay ningún inconveniente en usar el mismo valor para “local_port” y “web_port”, pues ambas conexiones trabajan con protocolos distintos (UDP y TCP, respectivamente)

En Linux/Mac, el usuario debe tener permiso de acceso en lectura/escritura al dispositivo serie especificado. Normalmente esto, dependiendo de la distribución concreta de Linux, implica pertenecer al grupo “dialout” o “uucp”

Ejecución:

AC_SerialChrono puede ser invocada de diversas formas
- A través de la aplicación AgilityContest, desde el menú de control de sesiones.

- Desde línea de comandos, utilizando las siguientes opciones de arranque del programa:

```
Serial parameters:
       -m module  || --module=module_name Serial comm module to be used. Default "generic"
       -d comport || --device=com_port    Communication port to attach to (required)
       -b baud    || --baud=baudrate      Set baudrate for comm port. Defaults 9600

Web interface:
        -w webport || --port=web_port     Where to listen for web interface. 0:disable . Default 8080

AgilityContest  interface:
       -s ipaddr  || --server=ip_address  Location (IPv4) of AgilityContest server
                                          Values: "none":disable - "find":search - Default: "localhost"
       -n name    || --client_name=name   chrono name sent to AgilityContest. Default: module_name@ring
       -r ring    || --ring=ring_number   Tell server which ring to attach chrono. Default "1"

Debug options:
       -D level   || --debuglog=level     Set debug/logging level 0:none thru 8:all. Defaults to 3:error
       -L file    || --logfile=filename   Set log file. Defaults to "stderr"
       -c         || --console            open cmdline console and enter in interactive (no-daemon) mode
       -v         || --verbose            Send debug to stderr console
       -q         || --quiet              Do not send debug log to console

Additional options:
       -f         || --find-ports         Show available , non-busy comm ports and exit
       -h  || -?  || --help               Display this help and exit
```

Las opciones indicadas mediante línea de comandos tienen prevalencia sobre las indicadas en el fichero de configuración

Cuando el usuario indica la opción de uso de consola de comandos, puede monitorizar el estado, obtener información extra, e incluso comandar directamente el cronómetro y los demás módulos del programa

Si hay varios cronómetros, cada uno en un ring, se deberá iniciar una instancia de la aplicación por cada cronómetro, indicando en casda caso el ring correspondiente a cada uno

La consola web:

Si el usuario inicia la aplicación indicando el despliegue de la consola web, y la licencia instalada lo permite, desde un navegador puede desplegar la consola web en la dirección http://localhost:xxxx; siendo xxxx el valor correspondiente a “web_port” + “ring”.

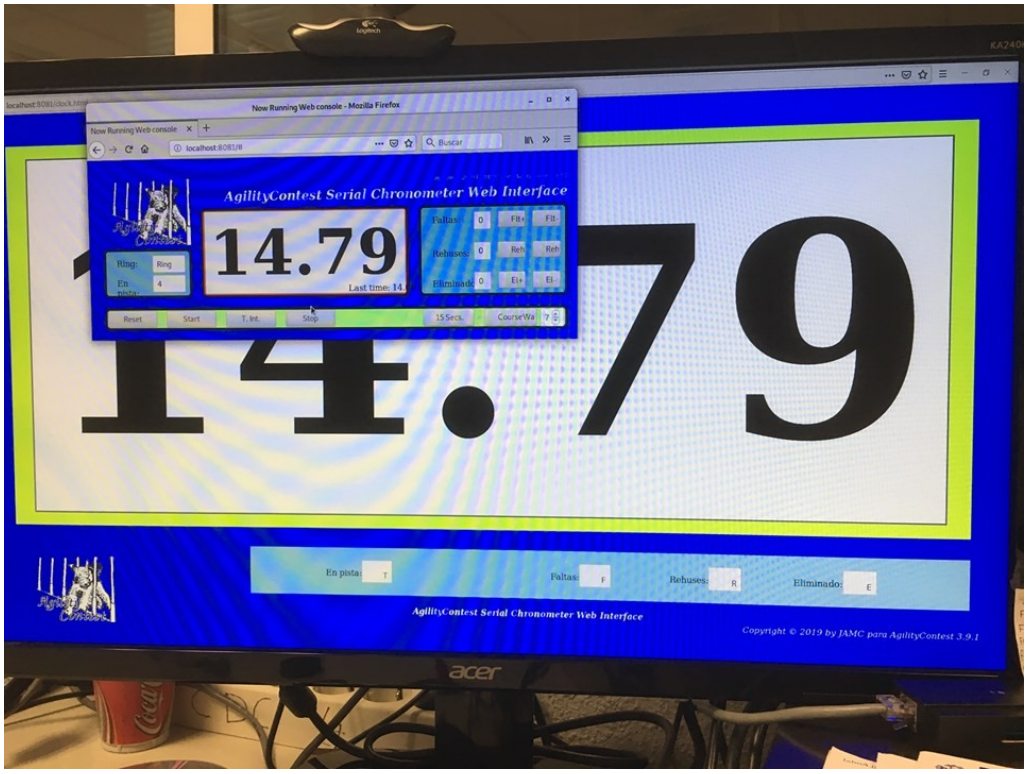
Desde la consola se tiene acceso a la mayor parte de los controles del cronometro.

Pulsando con el ratón sobre el logo, se despliega una cuadro de diálogo con información del sistema

Pulsando sobre el cronómetro se despliega éste en una nueva ventana. Dicha ventana, puede ser utlizada como pantalla de visualización externa usando las salidas de video adicionales de que el ordenador disponga. De esta manera se puede tener en una pantalla auxiliar conectada a un monitor de gran tamaño el cronómetro, y en la pantalla principal del ordenador la consola web

Hay que indicar que el logotipo que se presenta en ambas pantallas es el que viene indicado en la licencia que recibe el club; no se puede cambiar. En la versión de evaluación este logotipo corresponde al de la marca AgilityContest(tm)

Si la aplicación se lanza en modo conectado a AgilityContest, la consola web no es realmente necesaria: las funciones de consola y pantalla grande son realizadas respectivamente por el tablet y la ventana de cronómetro de AgilityContest, permitiendo de este modo acciones e información adicional



El interfaz de comandos:

Cuando se lanza AgilityContest con la opción “console” activada, la interfaz de comandos presenta el prompt “CMD>” en el termina.

Desde éste podemos teclear directamente los diversos comandos del API de comunicaciones serie (ver anexo) como si estuvieramos trabajando desde AgilityContest o desde la consola web

Adicionalmente se dispone de comandos adicionales, que permiten obtener información sobre el sistema, ver el estado, activar la traza y depuración, reconectarse con AgilityContest, o finalizar la aplicación, entre otros

Tambien existe la opción de solicitar ayuda, bien sobre los comandos disponibles, bien sobre un comando en concreto

Serial Communications Data Protocol (API)

API Command description:

- Is up to the user select serial port baudrate. No parity and 1 stop bit is assumed elsewhere
- Upper/Lower case is ignored
- Every command are ASCII text based, and ends with 0x0D (Carriage Return) + 0x0A (Line Feed) sequence ("windows/DOS newline")
- Extra whitespaces and non-ASCII characters are ignored
- Unknown commands or extra keywords should also be ignored
- Time stamps are given in miliseconds; but if dot/comma is present in timestamp, seconds are assumed
- Important: time deltas are *trunk'd* to lowest cents of second; NOT round'd, as KCC directives says

Messages from Chronometer to Computer:

- **START** [timestamp] < newline > (required)
  Chronometer starts. Timestamp mark is optional. When ommited zero (0) is assumed
- **INT** timestamp < newline > (optional)
  Intermediate course run timestamp. Time shown is "timestamp - start", trunk'd to cents of seconds
- **STOP** timestamp < newline > (required)
  End of course run. Time shown is "timestamp - start", trunk'd to cents of seconds
- **FAIL** < newline > (optional)
  Sensor failure. Should be sent every second while error remains
- **OK** < newline > (optional - required whenever fail is implemented)
  Chronometer is ready. Sensor error is over

Messages from Computer to Chronometer:

- **MSG** <seconds> <message> < newline > (optional)
  Show message on chronometer display during requested seconds
- **CLOCK** [hh:mm[:ss]] (optional)
  Tell chronometer to run in clock mode (that is, show HH:MM time). If time mark is ommited, chronometer should use their internal clock data. Receiving RESET will return to chronometer mode
- **DORSAL** + <newline>
- **DORSAL** - <newline>
- **DORSAL** <number> <newline>
  Tell chronometer to set up current running dog order information. Default is increase dog count
- **BRIGHT** + <newline>
- **BRIGHT** - <newline>
- **BRIGHT** <number> <newline>
  Set up display bright level. Value range is 0 (off) to 9 (maximum bright)

Bi-Directional messages:

Can be sent either for the chronometer or the Computer.
Chronometer can ignore these commands, but honoring RESET is recommended

- **WALK** [seconds] <newline>
  Start course walk coundown. Seconds is optional, defaults to 420 (7minutes)
  Setting seconds to zero means stop course walk coundown
- **DOWN** [seconds] < newline >
  Start CountDown when competitor receives ack to run. Seconds is optional. defaults to 15 secs
- **FAULT** + < newline >
- **FAULT** - < newline >
- **FAULT** number < newline >
  Increase / Decrease / Set fault counter
- **REFUSAL** + < newline >
- **REFUSAL** - < newline >
- **REFUSAL** numero < newline >

Increase / Decrease / Set refusal counter. If up to the user set ELIM flag after 3 refusals

- **ELIM** < newline >
- **ELIM** + < newline >
  Set eliminated mark
- **ELIM** - < newline >
  Clear eliminated mark
- **DATA** <F:R:E> < newline >
  Indicate competitor penalization data in one line with colon separated absolute values for faults, refusals and eliminated (0/1). ie: DATA 1:2:0 <newline>
- **RESET** < newline >
  Clears chronometer status, setting zero fault/refusal/countdown counters. Stop and clears chronometer

Messages from command console:

Aditionally there are some extra commands that can be used from command console. These messages are not parsed for other modules:

-**VERSION**
Show program and module(s) information
- **PORTS** <newline>
Show available serial ports
- **HELP** [command] <newline>
Show command list. Enter “help <command> shows extra info about requested command
-**STATUS** <newline>
Show chronometer status: dorsal, faults, refusals, and so
- **EXIT** <newline>
Ask application to finish and exit
- **SERVER** <address> <newline>
Tell application where to connect to AgilityContest server via Event API bus. Possible values are:
“none” to disable connection,
“find” to start auto-search server
“x.y.z.t” IPv4 AgilityContest server address.
“localhost” AC server is running in local machine.
You can also provide FQDN host names, but to minimize problems on timeouts when DNS is not available (i.e: isolated networks) is recommended not to use at all
- **DEBUG** [level] <newline>
Set/get debug level info. When level is ommited shows current debug level

Debug level is provided as integer value:
0: *none* → ignore every debug information
1: *panic* → Application is unusable
2: *alert* → Error non recuperable by program. Requires user action
3: *error* → Runtime errors due to normal program flow
4: *notice* → Abnormal runtime result that may require further user attention
5: *info* → Normal runtime messages
6: *debug* → additional user info provided for locating software errors
7: *trace* → in-deep debugging traces indicating program flow

Apéndice 1: Descripción física del conector. Protocolos de bajo nivel

Líneas utilizadas (vistas desde el conector RS232 del PC)

- Requeridas
(5) GND masa común
(2) RxD (entrada) Recepción de datos en el PC
(3) TxD (salida) Envío de datos desde el PC

- Opcionales
(4) DTR (salida) Indica que el programa está activo y listo para enviar/recibir
(6) DSR (entrada) Indica que el cronómetro está activo y listo para enviar/recibir
(9) Ri (entrada) Indica activación de alguno de los sensores

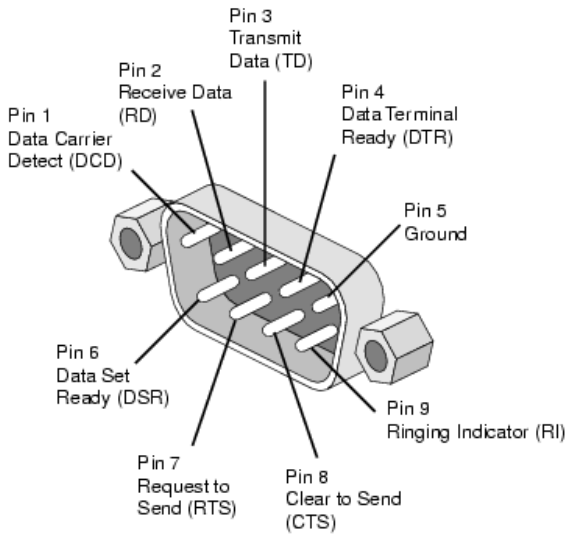
* Si el cronómetro no soporta el protocolo DTR/DSR, se recomienda que el cable tenga cortocircuitados los pines 4 y 6, o bien deshabilitar dicho protocolo en la aplicación de interfaz (opción por defecto)
* El uso de la línea Ring (pin 9) para indicar el paso por sensores NO debe usarse de manera simultánea con los comandos START/INT/STOP del protocolo

Conector DB-9

Descripción y pinout del conector DB9, usado en los PC's para el interfaz serie (RS-232).

Pin	SIG.	Signal Name	DTE (PC)
1	DCD	Data Carrier Detect	in
2	RXD	Receive Data	in
3	TXD	Transmit Data	out
4	DTR	Data Terminal Ready	out
5	GND	Signal Ground	-
6	DSR	Data Set Ready	in
7	RTS	Request to Send	out
8	CTS	Clear to Send	in
9	RI	Ring Indicator	in

El DTE (Ordenador) tiene el conector macho (mostrado abajo), y el DCE (Cronómetro) el conector hembra



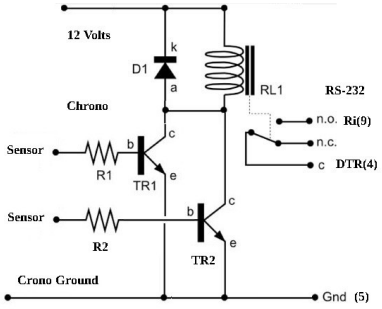
Protocolo de comunicaciones utilizado:

Data transmission: Asynchronous Serial
Speed: configurable (default 9600bps)
Partity: NONE
StopBits: 1
Synchronization: Full duplex
Handshaking: None (DTR/DSR opcional)
Signal levels: MARK = -3 to -15 V ... logic '1'
SPACE = +3 to +15 V ... logic '0'

Sugerencia de conexionado de los sensores de paso (opcional)

En el caso de que se desee usar el pin 9 (Ring) como indicador de paso, será necesario activar éste (ponerlo a nivel bajo) durante el tiempo en que el sensor detecta el corte del haz

El método recomendado es el uso de un relé u opto-acoplador que ante la activación del sensor conecte la señal DTR (pin 4) a la señal Ring(pin 9) durante el tiempo que el sensor está activado, tal y como se muestra en el esquema:



D1: 1N4004
R1 / R2: 1KΩ
TR1 / TR2: 2N2222
RL1: Relé 12 Voltios

En el esquema se asume que cuando el sensor está activado, en la línea “sensor input” aparece una tensión de 12 voltios

Recordar de nuevo que el uso de la linea Ring para el control de paso es opcional, y que en caso de implementarse NO DEBEN usarse los comandos START / INT / STOP del API de comunicaciones simultáneamente con dicha señal

Apéndice 2: Notas para la elaboración de bibliotecas de traslación del CommAPI a cada cronómetro concreto

(Pendiente. Consultar código fuente de ejemplos en GitHub)