

Logger Firmware Functional Specification

Overview

The Logger is a piece of hardware to which can be attached some number of sensor devices of various types. It reads data values from these sensors according to a specified schedule, queues these values, and sends them to a web-attached data repository, also according to a specified schedule. It also regularly reports to a web-attached configuration manager, which monitors status and provides configuration updates as appropriate.

It is intended that the Logger operate without service for an indefinite period of time. However, it is fundamentally NOT a standalone unit. It is intended to be part of a system that includes other similar loggers, an Internet-attached server that manages configuration of all deployed loggers, and web-attached data repositories.

The Logger periodically reports its configuration and status to the web-attached server, and it accepts whatever reconfiguration specifications that the server may provide, including firmware updates.

Hardware

Current implementation of the Logger uses a TINI board, made by Dallas Semiconductor, and running a Java Operating System. The firmware is written in Java.

Attached to the logger will be some number of sensor devices, capable of detecting various physical events.

However, this Firmware specification is independent of the specific hardware implementation. Future versions of the Logger might well use a different hardware platform.

Initialization

When power is applied to the logger, it automatically starts the Logger Firmware.

Device Discovery

Immediately after beginning operation, the Logger Firmware discovers all attached devices. It builds a list of these devices, and configures them according to defaults appropriate to the device type.

Configuration Load

When the Logger is started, a default configuration is loaded. This includes:

- URL of the configuration management server – <http://chuck-wright.com/config.php>
- URL of the data server – <http://chuck-wright.com/config.php>

- Timing specification for contacting the servers – "0,10,20,30,40,50 * * * *". If the Logger is determined to not be attached to an Ethernet, then dialup operation will be assumed, and a string of "0 0 * * *" is used (connect each day at midnight).
- For each device, a Sample Timing and Policy appropriate to that device type.

When the logger starts, it immediately reports to the configuration manager server. This server may supply new configuration to override the defaults.

Supported Devices

The Logger supports the following Dallas Semiconductor devices, attached to a "1-wire" bus:

- DS2405 – Addressable Switch – Can be used to sense digital voltage levels
- DS18S20 – Temperature probe
- DS2450 – Quad Analog to Digital Converter
 - This device contains 4 independent 8-bit A/D converters. The input voltage range is 0-5 volts. When sending and receiving XML, the four channels are assigned device Ids of <deviceid>_0, <deviceid>_1, <deviceid>_2 and <deviceid>_3.
- DS2438 – Smart Battery Monitor (contains temperature probe and 2 A/D converters).
 - The temperature sensor is equivalent to the one used on the DS18S20. The deviceid for reading this sensor is the deviceid of the DS2438 itself.
 - The first A/D converter can be used to read either the supply voltage or an external voltage. Both accept an input voltage range of 0 to 10.23 volts, in 10 mV steps. The device IDs for these devices are <deviceid>_0 and <deviceid>_1, respectively.
 - The second A/D converter can be used to measure input voltages from -0.25 to 0.25 volts, in 0.2441 mV steps. The device ID for this sensor is <deviceid>_2.
- DS2423 – Counter plus 4k RAM (useful for counting switch closures that occur too rapidly to be detected by the main software loop). planned

Support for other devices will be added in time. The architecture allows attachment of devices outside the DS 1-wire family, such as might attach to a serial (RS232) port. However, attachment of such devices is not covered by this specification.

Master Clock

The Logger operates on a master clock that has a period of one minute. Finer time resolution than this is not possible.

Data Sampling Specification

For each attached device, the Logger contains a specification that determines how data will be sampled. This includes a description of when samples should be collected, plus information about how the data should be processed, as described below.

Sample Timing

The times at which the Logger will read data from attached devices is specified, for each device, by a character string. This character string follows the format used by the Unix "cron" function, containing 5 fields, separated by spaces. These fields are

- MINUTE(0-59)
- HOUR(0-23, 0 = 12 AM)
- DAYOFMONTH(1-31)
- MONTHOFYEAR(1-12)
- DAYOFWEEK(0-6, 0 = Sunday)

If a field contains a comma-delimited list, then processing will occur at each of the times specified in the list. If a field is "*", then processing will occur at every legal time for that field.

Examples:

- "* * * * *" - sample every minute of every hour of every day of every month (default for all switch devices)
- "0,10,20,30,40,50 * * * * *" - sample every 10 minutes (default for temperature and A/D devices)
- "5,15,25,35,45,55 * * * * *" - sample every 10 minutes, in a different way.
- "0 12 * * * *" - sample each day at 12:00 noon.

In addition, the string "off" is recognized, indicating that no sampling will be performed for that device.

As part of an XML string, sample timing strings will be marked by use of the <CRON></CRON> tags.

Sample Policy

Each attached device will be accessed at its specified times. There is an additional specification for how data is reported, called "Sampling Policy". The choices for this character string are:

- "ALL" – Every sampled value will be queued for transmission to the server

- "POSCHANGE" – Only sampled values that represent a positive change from the previous value will be queued for transmission to the server. (default for switch devices).
- "NEGCHANGE" – Only sampled values that represent a negative change from the previous value will be queued for transmission to the server.
- "CHANGE" – Only sampled values that are different from the previous value are sent to the server. (default for temperature and A/D devices).
- "AVERAGE_ALL" - The device is actually sampled every minute. At the times specified by the sampling specification, the average of all samples taken since the previous specified sample time is queued for transmission to the server.
- "AVERAGE_CHANGE" - The device is actually sampled every minute. At the times specified by the sampling specification, the average of all samples taken since the previous specified sample time is queued for transmission to the server, only if it is different from the previously queued value.

Notes:

- The "CHANGE" modes can be effective at reducing bandwidth.
- "POSCHANGE" is interesting for sensing changes in the state of a switch, such as might be found on a utility meter.

As part of an XML string, sample policy strings will be enclosed within the <POL></POL> tags.

Server Connect Specification

The Logger sends data to the server at times specified by a "cron" string, as defined for Sampling Timing. In the event that a connection cannot be made, the Logger will continue to try, on a short interval.

Data Queueing

Ordinarily, devices will be read at a higher frequency than information is reported to the server(s). This is especially true with dialup connections. It is envisioned that for such cases, server connection would be made only once per day.

So, acquired data must be queued for transmission. In the event of a software restart because of power interruption or other cause, it is likely that some data will be queued. This data must not be lost. So, the data queue is implemented as a "file", which is actually stored in nonvolatile RAM, and not be destroyed on a restart.

Server Communication Protocol

The Logger software exchanges information with the server via XML character streams.

The information, and its XML encoding, are described below.

Uploads

Configuration, status, and data are uploaded to one or more servers via an HTTP POST operation, marked as type "logger_data".

The information is encoded into an XML string, and presented as the body of the POST.

XML Definition

The following DTD defines this XML:

```
<!DOCTYPE LOGGER_UPLOAD [  
    <!-- Packet sent from logger to server -->  
    <!ELEMENT LOGGER_UPLOAD (CONFIG?, SAMPLE_DATA?)>  
        <!-- Information related to overall logger function-->  
        <!ELEMENT LOGGER (ID, UPTIME, CFGURL, LOGURL,  
            FWVER, OSVER, CRON, CRON1, DNS1, DNS2,  
            PRXSRV, PRXPT,  
            USINGCRON, LOGGERTIME, STARTTIME, PPP_PHONE?,  
            PPP_ID?, PPP_DNS1?, PPP_DNS2?, MSG*, REMMON_IP?,  
            MAC, IP, SUB, GATE, EXT)>  
        <!-- Unique id of logger (in HW) -->  
        <!ELEMENT ID (#PCDATA)>  
        <!-- Informational, number of seconds since  
            startup -->  
        <!ELEMENT UPTIME (#PCDATA)>  
        <!-- URL to which config/status will be sent-->  
        <!ELEMENT CFGURL (#PCDATA)>  
        <!-- URL to which data will be sent-->  
        <!ELEMENT LOGURL (#PCDATA)>  
  
        <!-- String representing logger OS level -->  
        <!ELEMENT OSVER (#PCDATA)>  
        <!-- String specifying times at which server  
            will be contacted. -->  
        <!ELEMENT CRON (#PCDATA)>  
        <!-- Upload time spec after an upload failure -->
```

```
<!ELEMENT CRON1 (#PCDATA)>
<!-- Upload spec currently in effect -->
<!ELEMENT USINGCRON (#PCDATA)>
<!-- Current logger time (secs since epoch) -->
<!ELEMENT LOGGERTIME (#PCDATA)>
<!-- IP configuration - used for local
      configuration -->
<!ELEMENT DNS1 (#PCDATA)>
<!ELEMENT DNS2 (#PCDATA)>
<!ELEMENT PRXSV (#PCDATA)>
<!ELEMENT PRXPT (#PCDATA)>
<!-- Phone Number for PPP Connections -->
<!ELEMENT PPP_PHONE (#PCDATA)>
<!-- User ID for PPP Connections -->
<!ELEMENT PPP_ID (#PCDATA)>
<!-- Password for PPP Connections -->
<!ELEMENT PPP_PWD (#PCDATA)>
<!-- Primary nameserver for PPP Connections -->
<!ELEMENT PPP_DNS1 (#PCDATA)>
<!-- Secondary Nameserver for PPP Connections -->
<!ELEMENT PPP_DNS2 (#PCDATA)>
<!-- Message - informational only -->
<!ELEMENT MSG (#PCDATA)>
<!-- IP Address of remote monitor -->
<!ELEMENT REMMON_IP (#PCDATA)>
<!-- Logger's MAC Address -->
<!ELEMENT MAC (#PCDATA)>
<!-- Logger's IP Address -->
<!ELEMENT IP (#PCDATA)>
<!-- Logger's subnet mask -->
<!ELEMENT SUB (#PCDATA)>
<!-- Logger's gateway IP -->
<!ELEMENT GATE (#PCDATA)>
<!-- Current Extension configuration -->
<!ELEMENT EXT (#PCDATA)>
```

```
<!-- Information related to attached devices-->
<!ELEMENT RDRS (RDR+)>
  <!-- Information related to a specific device -->
  <!ELEMENT RDR (ID, CH, BR, CRON?, POL?)>
    <!-- Unique ID, from hardware. -->
    <!ELEMENT ID (#PCDATA)>
    <!-- Channel number, if
         multi-channel device -->
    <!ELEMENT CH (#PCDATA)>
    <!-- Specifies branch number,
         if any, of device -->
    <!ELEMENT BR (#PCDATA)>
    <!-- Sample timing for device -->
    <!ELEMENT CRON (#PCDATA)>
    <!-- Sample policy for device -->
    <!ELEMENT POL (#PCDATA)>

<!-- Queued data for server -->
<!ELEMENT SAMPLE_DATA (TIME+, TDELTA+, DEV+, DATA+,
                       CHECKSUM)>
  <!-- Current time (GMT), seconds since 1/1/70 -->
  <!ELEMENT TIME (#PCDATA)>
  <!-- Time delta (seconds) since last
         reported sample -->
  <!ELEMENT TDELTA (#PCDATA)>
  <!-- Unique device ID -->
  <!ELEMENT DEV (#PCDATA)>
  <!-- Channel ID (should probably removed..
         carries no unique information) -->
  <!ELEMENT CH (#PCDATA)>
  <!-- data -->
  <!ELEMENT DATA (#PCDATA)>

  <!-- checksum, reported at end of sequence. It is
         calculated by the logger, based on the
```

```

        values of the <TIME>, <DEV>, <TDELTA>,
        <CH>, and <DATA> tags. -->
<!ELEMENT CHECKSUM (#PCDATA)>
]>

```

Data Format

Uploaded data will be in the form of a character string representation of a decimal number, with or without a decimal point.

- Temperature data – xxx.xx, representing degrees celsius.
- Switch closure – assuming policy is set to POSCHANGE, data will always be 1.0. If policy is ALL or CHANGE, data will be 1.0 or 0.0.
- A/D Converter – xxx.xx
- Counter - xxxx, leading zeroes suppressed, integer format

Downloads

A server may, in response to any contact made by the logger, supply configuration information to be used in place of that currently being used by the Logger:

- Sampling timing and policy for each attached device.
- Timing for connection to server.
- A URL to which status and configuration information will be reported.
- A URL to which data samples will be reported (possibly the same URL).
- Current time (GMT), specified as an integer representing the number of seconds since Midnight, January 1, 1970.

XML Definition

The following DTD defines the XML used for download:

```

<!DOCTYPE LOGGER_DOWNLOAD [
    <!-- Configuration downloaded to logger -->
    <!ELEMENT LOGGER_DOWNLOAD (LOGGER?, RDRS?)>
        <!-- Related to overall logger operation. -->
        <!ELEMENT LOGGER (CFGURL?, LOGURL?, CRON?, CRON1?,
USINGCRON?, TIME?, PPP_PHONE?, PPP_ID?, PPP_PWD?, PPP_DNS1?, PPP_DNS2?,
IP?, SUB?, GATE?, DNS1?, DNS2?, DHCP?, PRXSRV?, PRXPT?, SENDCFG,
REMMON_IP, DOWNLOAD?, REBOOT?)>
        <!-- URL to which config/status is reported. -->

```

```
<!ELEMENT CFGURL (#PCDATA)>
<!-- URL to which data is reported. -->
<!ELEMENT LOGURL (#PCDATA)>
<!-- Time spec for contacting server. -->
<!ELEMENT CRON (#PCDATA)>
<!-- Secondary time spec for contacting server. -->
<!ELEMENT CRON1 (#PCDATA)>
<!-- Specification of the CRON that should be
      in use. Compared against the one currently
      used, and index switched if necessary. -->
<!ELEMENT USINGCRON (#PCDATA)>
<!-- Phone number to use for dialup->
<!ELEMENT PPP_PHONE (#PCDATA)>
<!-- userid to use for dialup-->
<!ELEMENT PPP_USER (#PCDATA)>
<!-- password to use for dialup -->
<!ELEMENT PPP_PWD (#PCDATA)>
<!-- DNS to use for dialup -->
<!ELEMENT PPP_DNS1 (#PCDATA)>
<!ELEMENT PPP_DNS2 (#PCDATA)>
<!-- IP Setup. Used for local configuration -->
<!ELEMENT IP (#PCDATA)>
<!ELEMENT SUB (#PCDATA)>
<!ELEMENT GATE (#PCDATA)>
<!ELEMENT DNS1 (#PCDATA)>
<!ELEMENT DNS2 (#PCDATA)>
<!ELEMENT DHCP (#PCDATA)>
<!-- Proxy server-->
<!ELEMENT proxy port (#PCDATA)>
<!-- Force the logger to upload its device
configuration -->
<!ELEMENT SENDCFG (#PCDATA)>
<!-- IP address of a remote UDP listener (for debug
status) -->
<!ELEMENT REMMON_IP (#PCDATA)>
<!-- -->
```

```
<!ELEMENT (#PCDATA)>
<!-- Current time, seconds since 1/1/70
      logger will set its clock to this value -->
<!ELEMENT TIME (#PCDATA)>

<!-- URL of new firmware to be downloaded -->
<!ELEMENT DOWNLOAD (#PCDATA)>
<!-- Causes a reboot. Can be "all", "heap" or "yes"
      -->
<!ELEMENT REBOOT (#PCDATA)>
<!-- configuration of attached devices. -->
<!ELEMENT RDRS (RDR+)>
  <!-- configuration of one attached device -->
  <!ELEMENT RDR (ID, CH, CRON, POL)>
    <!-- unique ID (for matching only) -->
    <!ELEMENT ID (#PCDATA)>
    <!-- Channel, in case multi-channel device -->
    <!ELEMENT CH (#PCDATA)>
    <!-- Sampling times -->
    <!ELEMENT CRON (#PCDATA)>
    <!-- Alternate Sampling times -->
    <!ELEMENT CRON1 (#PCDATA)>

    <!-- Sampling policy -->
    <!ELEMENT POL (#PCDATA)>
  <!-- Return Status. A string of "OK" indicates that the
        server received the uploaded XML and properly
        processed it. -->
  <!ELEMENT STATUS (#PCDATA)>
]>
```

Device Ids

The devices that can be accessed from the logger belong to the "1-wire" family of devices made by Dallas Semiconductor. Each of these devices contains a unique 64-bit serial number. This serial number is used in all XML communications to identify the specific device.

The processor itself contains a device with a unique serial number. This serial number is used to identify each specific logger.

In some cases, a physical package on the 1-wire bus contains multiple physical sensors, although the device itself contains only a single serial number. For devices like this, an extended device ID is created by appending "_0", "_1", etc.

Internet Connection

Communication resources will vary with the site at which the logger is installed. The following possibilities are accommodated:

Web-attached Ethernet - DHCP

For this configuration, the logger is entirely self-configuring. It automatically leases an IP address from the DHCP server, which provided all necessary network configuration.

Web-attached Ethernet - Fixed IP Addresses

This configuration requires manual setup of the network configuration. This will be done via a serial (RS232) connection.

Phone Connection (PPP)

If the Ethernet is determined to be unavailable at startup, Logger Firmware will attempt to make a dialup connection. Point to Point Protocol (PPP) is used to establish a dialup connection to an Internet Service Provider, via a dedicated (probably low-speed) modem.

Logger Firmware contains a phone number (probably a toll-free number) for a server that will always be available. This phone number will be dialed when the Logger is initialized. Given the connection to the Internet, the Logger reports in to the default server, which provides additional configuration information. This information includes:

- A second phone number, usually to a local Independent Service Provider (ISP), to be used as the main dialup connection
- A user ID
- A user password
- Additional network programming information, as required by the local ISP, such as primary and secondary Domain Name Server (DNS) Addresses.

If a PPP connection is used, communication with the server is necessarily slower and carries more overhead than with a direct connection. It is also likely that the phone line will be shared, not dedicated. In these instances, the programming of server connection timing will be much less frequent, and timed to occur at a time that the line is otherwise unused.

The PPP connection will not dial out if the phone line is in use.

If a transfer fails while in PPP mode, a secondary upload timing spec is switched in until a successful transfer is accomplished. This spec is marked by the XML tags `<CRON1></CRON1>`. Whereas the default upload timing spec is presumably relatively infrequent for a dialup link, use of this secondary spec provides an opportunity to attempt contact more frequently after a transfer has failed.

Reliability and Data Integrity

It is essential that the logger be capable of operation without service. To this end, the following reliability features are implemented:

Survival Mode

A "survival mode" minimal functionality programmed into Flash ROM.

Configuration

Operational parameters are saved in nonvolatile RAM, so that after a system restart due to a watchdog timeout or loss of power, the system resumes normal operation.

There are two exceptions:

- To avoid certain lockup conditions, two specific operational parameters are not restored. These are the phone number and the configuration URL. These are restored to their default values. This means that after a restart, the logger will be on the original schedule, but will contact the default configuration URL, over the default (usually 800) phone number.
- If the restart blasted the heap, then the configuration server will be contacted immediately.

Queueing

Data to be transmitted is stored in the "file system" (implemented in nonvolatile RAM) instead of in internal data structures, so that data loss will not occur if a system crash it. Queueing of data to be transmitted, and retry in the event of a checksum mismatch.

Error Restart

Certain conditions will cause a restart of the processor.

- The logger may from time to time experience a power outage that causes it to restart. This is NOT an error restart, and no data should be lost as a result of it.
- If the main loop is unable to dispatch the data upload thread n consecutive times, the process is killed, which will lead to a watchdog restart.

- If the transmit process fails more than M consecutive times, the main process is killed, which will also lead to a watchdog restart.
- The main loop operates on a 1-minute cycle. The watchdog timer is serviced once during each cycle. Any condition that prevents this will lead to a timeout of the watchdog timer, followed by a processor restart.
- Reboots are tracked. If more than a certain number of these occurs within a particular time, the system will attempt another reboot, blasting the heap. This means that all data stored in RAM is lost, including the data queue, and any new version of the firmware. This is a last-ditch recovery effort.

Data Transfers

All data transfers are made via HTTP POST operations initiated by the logger. HTTP is built upon TCP/IP, which is in itself an exceptionally reliable transmission medium. Still, the design of the logger must and does consider the limitations of the medium.

Checksums are presented for all upload XML. The checksum represents a sum of all bytes bracketed by XML tags. It does not include the XML tags themselves. In reality, because of the error checking inherent in TCP/IP, it is not expected that this will add much to data integrity.

When a POST has been initiated by the logger, HTTP protocol dictates that a response data stream is returned to the sender. The header of this stream contains a response code from the web server that indicates relative success or failure. For our purposes, this is of little value, because it only indicates whether the request was received properly, and the requested CGI script was executed. It does not tell whether the CGI script was able to complete its mission properly.

For the logger, this mission ordinarily includes a number of database accesses, so quite a bit of work is being done. For this reason the response XML is required to include a sequence,

```
<STATUS>OK</STATUS>
```

that indicates successful processing. If the characters inside the XML tags are anything other than "OK", the logger will assume that the transaction was not carried out properly, and reattempt it at a later time.

It can be seen that there could exist a case in which a data event could be recorded multiple times by the server: if a transmission that invokes multiple database writes is interrupted, the server will give the logger a response indicating failure. If this is an undesirable artifact, the server should take steps to remove the duplicate data (because each data sample contains a time stamp, there should never be two samples with the same one).

Reporting In – Server Monitoring

The Logger contacts a server at regular intervals, and reports its configuration and status. The Logger stores information about error conditions that it detects, and reports these. It is expected that the server will track this information, and issue appropriate notification.

One of the most common failures is likely to be loss of the communication channel, particularly if a dialup connection is being used. The configuration server is expected to have complete configuration information for all loggers that it manages, and be able to note failures to report. This failure is likely to require human intervention.

Error Detection and Reporting

Errors can be detected at numerous points in the program. In general, continued operation is attempted. The error is logged and reported to the host using the regular data reporting mechanism.

Each code module is assigned a number. For each code module, each possible error condition is also assigned a code number. If an error should occur, it is logged as if it were regular data. When sent to the server, the following data is sent:

- ID – the unique ID of the logger itself
- TIME – time of occurrence
- DATA – $100 * \text{code module number} + \text{error number}$

Some error conditions are expected to occur at low frequency. These are typically those that involve transmitting data over the internet. The software waits for successful completion of all transfers before clearing data queues. All such occurrences are logged using the same mechanism. It is expected that the server will monitor the rate of such problems, and notify humans whenever it is too high.

The following table lists all of the error codes, and their meanings.

(see [showloggerlog.php](#)).Remote Firmware Updates

When the <DOWNLOAD> tag is sent to the logger, it will retrieve the specified file, store it, and restart, using that file as the code source.

(some work needed here....flash updates for example).

Debug Facilities

HTTP Server

The logger implements an HTTP server that can be used to serve up various XML sequences relative to internal operation. It can also be used to set internal parameters.

The "file" portion of the URL that addresses this logger is a single integer, and it is

interpreted as follows:

0,1- return XML describing the logger configuration, including the device configuration. This also does a read of all devices, and returns their current values, in the <VAL> tag of the device configuration.

2 - return XML summarizing data queue

3 - return XML containing the data queue

4 - return XML summarizing the status queue

5 - return XML containing the status queue

6 - return XML describing current network parameters

Example:

<http://192.168.1.5/0> - query the logger at IP address 192.168.1.5, and ask it to return XML describing its configuration.

Command String

A command string can be appended to the URL, and used to set logger configuration, as if it came from the server. The same tags as defined for XML download are recognized by the logger, but in URL Query format, instead of XML format. NOTE: these strings MUST be in URL Encoded format!

Examples:

http://192.168.1.5/0?LOGGER&CFGURL=*+2+*+*+* - set the config URL parameter to "* 2 * * *"

http://192.168.1.5/0?RDR&ID=A000000004B5471D_1&POL=CHANGE - Set the sampling policy of device [A000000004B5471D_1](#) to "CHANGE".

Console Log Server

If a telnet session is opened to port 58324 of the logger, it will receive a dump of all console messages. Typing "q" closes the connection. Only one session at a time is supported.

UDP Monitor

The logger listens for UDP packets on port 58324. If it receives one that has "WCS?" as the first 4 characters, it sends a responding datagram packet on the same port, containing its MAC address.

If it receives a datagram packet that begins with its MAC address, it sends in response a datagram packet that contains the logger configuration (not the device configuration, though).

If the datagram packet that begins with the logger's MAC address contains additional characters, these are treated as a command string to be processed as would be a query string sent to the http server.

Format of Command String

The comand string of the UDP server is processed by the same routines as for the HTTP server. The same format applies.

Extensions

The logger may utilize various code extensions to provide additional functionality. The implementation of this feature is not defined here. However, we do define a mechanism for loading and starting extensions, and for reporting which extensions are currently active.

This is done using the EXT XML tag. It marks a string that defines one of more extensions, each as a string containing one or more fields, delimited by spaces. The first field is the name of the extension, and the following fields are arguments that are passed to the extension when it is started. The specifications for the different extensions are separated by semicolons.

Example:

`<EXT>sbComm serial0;DavisExtension serial4<EXT>` - start 2 extensions, one named "sbComm", and the other "DavisExtension", passing them the arguments "serial0" and "serial4", respectively.