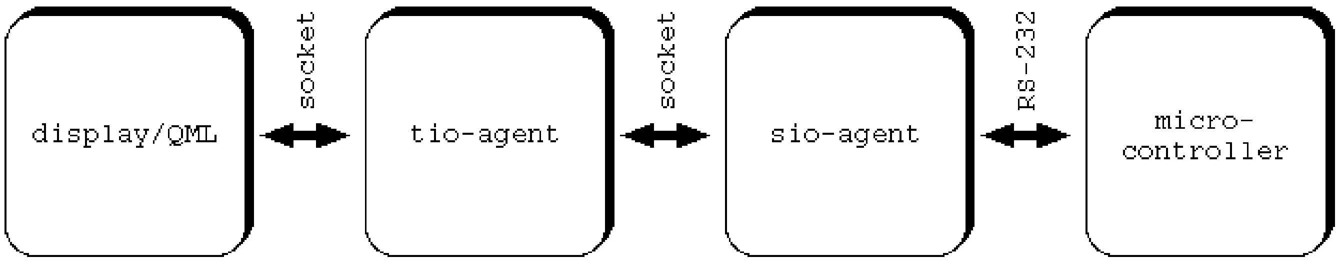
# G2 Series QML I/O Agents

## *Overview*

The G2 Series display modules from Reach are designed to interact with the outside world using one of the modules' many interface options. A common interface is the RS-232 serial port. This typically is connected to a micro-controller that does the work of controlling the low level I/O.

In order to allow the GUI written in QML to interact with the connected micro-controller, two functions are needed.  First, the physical RS-232 port needs to be adapted to internal communications with configurable settings.  Second, a character string level translation needs to be performed to allow the micro-controller to use terse commands and responses while allowing for the richness of the QML object naming conventions.

A pair of programs, known collectively as "I/O agents," performs these two tasks. These agents are background processes running on the display module. The sio-agent exchanges messages between a serial port device and a network socket that connects to the tio-agent. The tio-agent then communicates with the QML viewer program over another socket. See Drawing 1.



*Drawing 1*

The IO agents are designed so that they can run either on the host VM or on the target Display Module. This enables the developer to run the same code on both environments for faster development. Basically, a serial port on the host PC can be used in place of the serial port on the target. The entire GUI can be developed and debugged on the host including interaction with the micro-controller. The application can be run on the target at any time to check for performance or target-specific issues.

The I/O agents are invoked on the host using a script "Start IO Agents Local.sh" or "Start IO Agents USB1". The Local script does not connect to the serial port and allows the developer to see the monitor and send messages. The USB1 script connects to the physical serial port /dev/ttyUSB1.

The I/O agents reside on the display module /application/bin directory and are normally started automatically by a startup script in /etc/init.d

## *Details*

The I/O agents each have command line flags that can be specified when invoking the programs to affect their behavior. Options are separated from one another with space characters. Both short and long options are supported. Short options begin with a single dash and a single letter. Some options have required or optional arguments which must immediately follow the option letter (no space).

Long options have two dashes and a word, typically making this form more memorable. Arguments for the options, if needed, follow the option word and are separated with an equals sign. Some examples of options are shown here:

| | |
|---|---|
| • -a | short option with no argument |
| • -a23 | short option with argument |
| • --opt | long option with no argument |
| • --opt=arg | long option with argument |

## sio-agent

The purpose of this program is to convert RS-232 serial communications into messages to be forwarded to the tio-agent and vice versa. The options that can be specified are:

| short option(s) | long option | arg | description |
|---|---|---|---|
| -b | –baud | R | This option configures the serial port for the specified bit rate given by the argument; if not specified, the speed is 115200 bits per second. The communications channel always uses 8 bits, no parity, one stop bit. |
| -d | --daemon | N | Specifying this option causes the agent to disconnect from the controlling terminal (i.e., not listen for signals such as CTRL-C and not printing anything to the terminal) and becoming a background process. This is typically how the agent should be run in a production environment. |
| -e | --test | N | Debug option, tells the agent to echo characters sent from the serial port back to the attached device and also enables processing of the backspace character for rudimentary line editing. This is useful for debugging the system using a terminal connected instead of the microcontroller. This option is not recommended in production. |
| -p | --pty | N | Use a pseudo terminal device instead of a real serial device. This is useful for debugging various elements including the tio-agent's translation rules. When you specify this option, the program prints the name of the pty device to open with a terminal emulator program, such as minicom, to communicate with the agent. |
| -s | --sio_port | O | Ordinarily, the communications between the two agents takes place over a Unix domain socket. By specifying this option, a TCP socket is used instead. If no argument is specified, a default value of 7880 is used. This option is not typically useful for end products but may be used for debugging purposes. |
| -t | --serial | R | Give this option to use the specified serial device instead of the default of /dev/ttyUSB0. |

| short option(s) | long option | arg | description |
|---|---|---|---|
| -v | --verbose | N | With this option, information is printed to the terminal when certain activities occur, such as connecting to the tio-agent and receiving or sending a completed message. This is useful for debugging but should not be used in production. |
| -h -? | --help | N | When specified, a brief usage summary is printed to the terminal. |

key to arg column: N = none, R = required, O = optional

## tio-agent

This agent performs a translation between two different message formats. Each end of this translation is in the form of newline terminated character sequences carried over network streams. The program accepts these command line options:

| short option(s) | long option | arg | description |
|---|---|---|---|
| -d | --daemon | N | Specifying this option causes the agent to disconnect from the controlling terminal (i.e., not listen for signals such as CTRL-C and not printing anything to the terminal) and becoming a background process. This is typically how the agent should be run in a production environment. |
| -f | --file | R | Use this option to override the default file path from which the translation rules are read. If not specified, /application/bin/translate.txt is used. The format for the contents of the file is explained below. |
| -r | --refresh | O | This option enables an automatic refresh of translation rules if the file has changed. The check for whether a change has occurred is done when a message is received from either the QML viewer or from the sio-agent but no more frequently than the the number of seconds specified by the optional argument. By default, this is every second. This option is useful while developing the translations so that the agent does not need to be restarted but should not be used in production because of the unnecessary overhead incurred. |
| -s | --sio-port | O | Ordinarily, the communications between the two agents takes place over a Unix domain socket. By specifying this option, a TCP socket is used instead. If no argument is specified, a default value of 7880 is used. This option is not typically useful for end products but may be used for debugging purposes. |
| -t | --tio-port | O | Like the previous option, this one says to use a TCP socket for communication with the sio-agent. If the optional argument is not specified, port number 7885 is used. Use of this option is not recommended for production. |
| -v | --verbose | N | With this option, information is printed to the terminal when certain activities occur, such as connecting to the tio-agent and receiving or sending a completed message. This is useful for debugging but |

| short option(s) | | long option | arg | description |
|---|---|---|---|---|
| | | | | should not be used in production. |
| -h | -? | --help | N | When specified, a brief usage summary is printed to the terminal. |

## translation rules

The tio-agent uses a set of rules to translate messages received from the sio-agent before sending to the QML viewer and a separate set of rules for messages going the other direction. Both sets are in the same file, which by default is /application/bin/translate.txt but it can be changed through a command line option. The translations file is composed like so:

- each translation rule occupies a line in the translations file
- empty lines and lines that contain a # character in the first column are ignored
- otherwise, the first character should be a G or an M
    - G means the translation is for messages received from the QML viewer (GUI) destined for the microcontroller
    - M is for messages received from the microcontroller to be sent to the GUI
- the next character is a colon
- next is a % to indicate a default translation or a sequence of characters which will be matched against messages from the GUI or microcontroller
    - a combination of letters and numbers comprising a word which will be matched verbatim to the same sequence received from the entity
    - an equals sign
    - the characters %d
- a comma and T separate the matching string from the translated string
- the remainder of the line is the translated string
    - a combination of letters and numbers comprising a word that will be substituted for the original string
    - an equals sign
    - the characters %d
- if the matching string is a % to indicate a default translation, the translated string becomes the default message to be passed as-is to the other party; this is used whenever a message is received from the originating party which does not have a matching translation defined
- if no translation is found for a message and no default translation has been defined for the originating party, the message is forwarded to the other party untranslated

Putting this all together, a normal (non-default) translation, looks like **M:v=%d,T:dial.value=%d**. This rule means when a message received from the microcontroller begins with **v=**, it will be turned into a message starting with **dial.value=**. The %d in the matching string represents a number whose value is provided in the translated message. Thus the whole original message from the microcontroller **v=32** becomes **dial.value=32**.