LM75 Device Driver

1.0

Generated by Doxygen 1.8.14

Tue Sep 17 2019 11:38:08

Contents

1 Arduino Library for the LM75A Temperature Sensor **1**

2 GNU GENERAL PUBLIC LICENSE **3**

3 Class Index **13**

3.1 Class List 13

4 File Index **15**

4.1 File List 15

5 Class Documentation **17**

5.1 LM75 Class Reference 17

5.1.1 Detailed Description 18

5.1.2 Member Enumeration Documentation 18

5.1.2.1 faultQueue 18

5.1.2.2 opMode 19

5.1.2.3 outputMode 19

5.1.2.4 outputPolarity 20

5.1.2.5 setPointType 20

5.1.2.6 tempUnit_t 20

5.1.3 Constructor & Destructor Documentation 21

5.1.3.1 LM75() 21

5.1.4 Member Function Documentation 21

5.1.4.1 convCtoF() 22

5.1.4.2 convCtoK() 22

5.1.4.3 convFtoC() . . . 23
5.1.4.4 convKtoC() . . . 24
5.1.4.5 getAddr() . . . 25
5.1.4.6 read16() . . . 25
5.1.4.7 read8() . . . 26
5.1.4.8 readSetPoint() . . . 27
5.1.4.9 readTemp() . . . 28
5.1.4.10 setFaultQueue() . . . 29
5.1.4.11 setOperationMode() . . . 30
5.1.4.12 setOutputMode() . . . 31
5.1.4.13 setOutputPolarity() . . . 32
5.1.4.14 write16() . . . 32
5.1.4.15 write8() . . . 33
5.1.4.16 writeSetPoint() . . . 34
5.1.5 Member Data Documentation . . . 35
5.1.5.1 _addr . . . 35
5.1.5.2 busAddr . . . 35

6 File Documentation **37**

6.1 LM75.cpp File Reference . . . 37
6.1.1 Detailed Description . . . 37
6.2 LM75.cpp . . . 39
6.3 LM75.h File Reference . . . 42
6.3.1 Detailed Description . . . 43
6.3.2 Macro Definition Documentation . . . 44
6.3.2.1 LM75_BROADCASTADDR . . . 44
6.3.2.2 LM75_CONF . . . 45
6.3.2.3 LM75_CONF_DOM_NORMAL . . . 45
6.3.2.4 LM75_CONF_DOM_SHUTDOWN . . . 45
6.3.2.5 LM75_CONF_OSFQUE_1 . . . 45
6.3.2.6 LM75_CONF_OSFQUE_2 . . . 46
6.3.2.7 LM75_CONF_OSFQUE_4 . . . 46
6.3.2.8 LM75_CONF_OSFQUE_6 . . . 46
6.3.2.9 LM75_CONF_OSOM_COMP . . . 46
6.3.2.10 LM75_CONF_OSOM_INT . . . 46
6.3.2.11 LM75_CONF_OSPOL_AH . . . 47
6.3.2.12 LM75_CONF_OSPOL_AL . . . 47
6.3.2.13 LM75_CONF_RES . . . 47
6.3.2.14 LM75_I2CDEFAULTADDR . . . 47
6.3.2.15 LM75_TEMP . . . 47
6.3.2.16 LM75_THYST . . . 48
6.3.2.17 LM75_TOS . . . 48
6.4 LM75.h . . . 48

Index 51

Chapter 1

Arduino Library for the LM75A Temperature Sensor

This library was written to enable remote sensing of the temperature of batteries and motor controllers used in remotely piloted aircraft, for the purpose of real time data logging and air to ground telemetry.

This sensor uses the I2C bus protocol to communicate allowing the Arduino standard Wire library to communicate with the device. 2 pins are required to interface the device to an Arduino - the SDA and SCL lines.

Installing

Download the distribution package and decompress it.
Rename the uncompressed folder ***/lm75***.
Check that the ***/lm75*** folder contains the following files;

LM75.cpp
LM75.h
LM75.chm
LM75.pdf
property.h
doxyfile

Place the ***/lm75*** library folder into your ***arduinosketchfolder/libraries/*** folder.
You may need to create the libraries subfolder if its your first library. Restart the IDE.

Documentation

LM75.chm and *LM75.pdf* contain the documentation for the classes.
A Doxygen script is included to enable generation of documentation. You will need the graph tool, the dot tool, and the help compiler, in addition to editing the paths to these tools in the script to suit your environment.

Author

John Fitter B.E., Eagle Air Australia Pty. Ltd.

License

Chapter 2

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either
  - (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or
  - (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. E↩ XCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLI↩ ED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE P↩ ROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRI↩ GHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CO↩ NSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROG↩ RAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>

[program] Copyright (C) [year] [name of author]

> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see https://www.gnu.org/licenses/.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> [program] Copyright (C) [year] [name of author]

This program comes with ABSOLUTELY NO WARRANTY; for details type ∗"show w"∗. This is free software, and you are welcome to redistribute it under certain conditions; Type ∗"show c"∗ for details.

The hypothetical commands ∗"show w"∗ and ∗"show c"∗ should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read https://www.gnu.org/licenses/why-not-lgpl.html.

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

LM75 17

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

LM75.cpp
LM75 Family Device Driver Library - CPP Source file 37
LM75.h
LM75 Family Device Driver Library - CPP Header file 42

Chapter 5

Class Documentation

5.1 LM75 Class Reference

Public Types

- enum tempUnit_t { LM75_TK, LM75_TC, LM75_TF }
- enum setPointType { SPT_OVERTEMP, SPT_HYSTERESIS }
- enum outputPolarity { POL_LOW, POL_HIGH }
- enum outputMode { MODE_COMP, MODE_INT }
- enum opMode { OPM_NORMAL, OPM_SHDN }
- enum faultQueue { FQ_1 = 1, FQ_2 = 2, FQ_4 = 4, FQ_6 = 6 }

Public Member Functions

- LM75 (uint8_t=LM75_I2CDEFAULTADDR)

  LM75 Device class constructor.
- boolean begin ()

  Initialize the device and the i2c interface.
- boolean **isReady** (void)
- double readTemp (tempUnit_t=LM75_TC)

  Return the measured temperature in specified units.
- double readSetPoint (tempUnit_t=LM75_TC, setPointType=SPT_OVERTEMP)

  Return the specified setpoint in specified units.
- void writeSetPoint (double, tempUnit_t=LM75_TC, setPointType=SPT_OVERTEMP)

  Write the specified setpoint supplied in the specified units.
- void setOutputPolarity (outputPolarity=POL_LOW)

  Set the OS polarity.
- void setOutputMode (outputMode=MODE_COMP)

  Set the OS operation mode.
- void setOperationMode (opMode=OPM_NORMAL)

  Set the device operation mode.
- void setFaultQueue (faultQueue)

  Set the value of the OS fault queue.
- double convCtoK (double degC)

  Convert temperature in degrees C to degrees K.
- double convCtoF (double degC)

  Convert temperature in degrees C to degrees F.
- double convKtoC (double degK)

  Convert temperature in degrees K to degrees C.
- double convFtoC (double degF)

  Convert temperature in degrees F to degrees C.

Public Attributes

- Property< uint8_t, LM75 > busAddr

Private Member Functions

- uint8_t getAddr (void)

  Return the device I2C Bus address.
- uint8_t read8 (uint8_t)

  Return an 8 bit register value from the device.
- uint16_t read16 (uint8_t)

  Return a 16 bit register value from the device.
- void write8 (uint8_t, uint8_t)

  Write a value to an 8 bit register in the device.
- void write16 (uint8_t, uint16_t)

  Write a value to a 16 bit register in the device.

Private Attributes

- boolean **_ready**
- uint8_t _addr

5.1.1 Detailed Description

Definition at line 81 of file LM75.h.

5.1.2 Member Enumeration Documentation

5.1.2.1 faultQueue

`enum LM75::faultQueue`

Enumerations for fault queue length.

Enumerator

FQ↩ _1	fault queue value = 1
FQ↩ _2	fault queue value = 2
FQ↩ _4	fault queue value = 4
FQ↩ _6	fault queue value = 6

Definition at line 111 of file LM75.h.

```
00111                     {FQ_1 = 1,                          /**< fault queue value = 1 */
00112                      FQ_2 = 2,                          /**< fault queue value = 2 */
00113                      FQ_4 = 4,                          /**< fault queue value = 4 */
00114                      FQ_6 = 6                           /**< fault queue value = 6 */
00115     };
```

5.1.2.2 opMode

```
enum LM75::opMode
```

Enumerations for operating mode.

Enumerator

OPM_NORMAL	normal operation mode
OPM_SHDN	shutdown mode

Definition at line 107 of file LM75.h.

```
00107                 {OPM_NORMAL,                          /**< normal operation mode */
00108                  OPM_SHDN                             /**< shutdown mode */
00109     };
```

5.1.2.3 outputMode

```
enum LM75::outputMode
```

Enumerations for output mode.

Enumerator

MODE_COMP	OS comparator mode
MODE_INT	OS interrupt mode

Definition at line 103 of file LM75.h.

```
00103                     {MODE_COMP,                         /**< OS comparator mode */
00104                      MODE_INT                           /**< OS interrupt mode */
00105     };
```

5.1.2.4 outputPolarity

```
enum LM75::outputPolarity
```

Enumerations for output polarity.

Enumerator

POL_LOW	OS output active low
POL_HIGH	OS output active high

Definition at line 99 of file LM75.h.

```
00099                         {POL_LOW,                         /**< OS output active low */
00100                          POL_HIGH                         /**< OS output active high */
00101     };
```

5.1.2.5 setPointType

```
enum LM75::setPointType
```

Enumerations for setpoint registers.

Enumerator

SPT_OVERTEMP	overtemp shutdown register
SPT_HYSTERESIS	hysteresis register

Definition at line 95 of file LM75.h.

```
00095                       {SPT_OVERTEMP,                    /**< overtemp shutdown register */
00096                        SPT_HYSTERESIS                   /**< hysteresis register */
00097     };
```

5.1.2.6 tempUnit_t

```
enum LM75::tempUnit_t
```

Enumerations for temperature units.

Enumerator

LM75_TK	degrees Kelvin
LM75_TC	degrees Centigrade
LM75_TF	degrees Fahrenheit

Definition at line 90 of file LM75.h.

```
00090                     {LM75_TK,                                 /**< degrees Kelvin */
00091                      LM75_TC,                                 /**< degrees Centigrade */
00092                      LM75_TF                                  /**< degrees Fahrenheit */
00093     };
```

5.1.3 Constructor & Destructor Documentation

5.1.3.1 LM75()

```
LM75::LM75 (
             uint8_t i2caddr = LM75_I2CDEFAULTADDR )
```

LM75 Device class constructor.

constructor

Parameters

in	*i2caddr*	Device address (default: published value).

Definition at line 50 of file LM75.cpp.

References _addr, busAddr, and getAddr().

```
00050                           {
00051
00052     busAddr.Set_Class(this);
00053     busAddr.Set_Get(&LM75::getAddr);
00054
00055     _addr  = i2caddr;
00056     _ready = false;
00057 }
```

Here is the call graph for this function:

5.1.4 Member Function Documentation

5.1.4.1 convCtoF()

```
double LM75::convCtoF (
            double degC )
```

Convert temperature in degrees C to degrees F.

Parameters

in	*degC*	Temperature in degrees Centigrade.

Returns

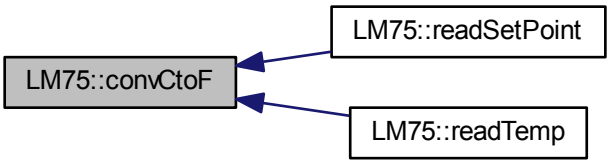
Temperature in degrees Fahrenheit.

Definition at line 265 of file LM75.cpp.

Referenced by readSetPoint(), and readTemp().

```
00265 {return (degC * 1.8) + 32.0;}
```

Here is the caller graph for this function:



5.1.4.2 convCtoK()

```
double LM75::convCtoK (
            double degC )
```

Convert temperature in degrees C to degrees K.

Parameters

in	*degC*	Temperature in degrees Centigrade.

Returns

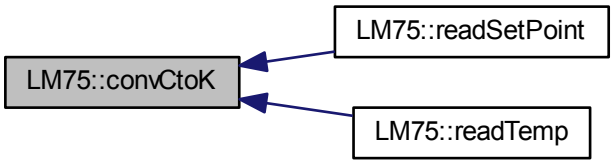
Temperature in degrees Kelvin.

Definition at line 258 of file LM75.cpp.

Referenced by readSetPoint(), and readTemp().

```
00258 {return degC + 273.15;}
```

Here is the caller graph for this function:



5.1.4.3 convFtoC()

```
double LM75::convFtoC (
            double degF )
```

Convert temperature in degrees F to degrees C.

Parameters

in	*degF*	Temperature in degrees Fahrenheit.

Returns

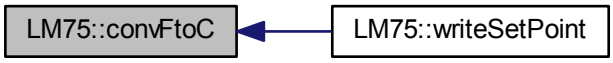
Temperature in degrees Centigrade.

Definition at line 279 of file LM75.cpp.

Referenced by writeSetPoint().

```
00279 {return (degF - 32.0) / 1.8;}
```

Here is the caller graph for this function:



5.1.4.4 convKtoC()

```
double LM75::convKtoC (
            double degK )
```

Convert temperature in degrees K to degrees C.

Parameters

in	*degK*	Temperature in degrees Kelvin.

Returns

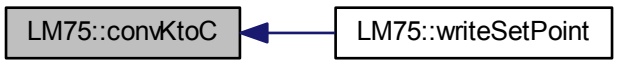
Temperature in degrees Centigrade.

Definition at line 272 of file LM75.cpp.

Referenced by writeSetPoint().

```
00272 {return degK - 273.15;}
```

Here is the caller graph for this function:



5.1.4.5 getAddr()

```
uint8_t LM75::getAddr (
            void  )  [private]
```

Return the device I2C Bus address.

Returns

Device address.

Definition at line 173 of file LM75.cpp.

References _addr.

Referenced by LM75().

```
00173 {return _addr;}
```

Here is the caller graph for this function:

5.1.4.6 read16()

```
uint16_t LM75::read16 (
            uint8_t reg )  [private]
```

Return a 16 bit register value from the device.

Parameters

in	*reg*	16b Register to read from.

Returns

Value read from register.

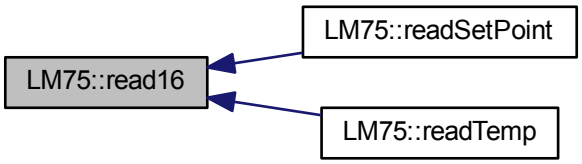
Definition at line 200 of file LM75.cpp.

References _addr.

Referenced by readSetPoint(), and readTemp().

```
00200                                   {
00201     uint16_t val;
00202
00203     // send the device address then the register pointer byte
00204     Wire.beginTransmission(_addr);
00205     Wire.write(reg);
00206 //    Wire.endTransmission(false);
00207     Wire.endTransmission();
00208
00209     // resend device address then get the 2 returned bytes
00210     Wire.requestFrom(_addr, (uint8_t)2);
00211
00212     // data is returned as 2 bytes big endian
00213     val = Wire.read() << 8;
00214     val |= Wire.read();
00215
00216     return val;
00217 }
```

Here is the caller graph for this function:



5.1.4.7 read8()

```
uint8_t LM75::read8 (
            uint8_t reg )  [private]
```

Return an 8 bit register value from the device.

Parameters

in	*reg*	8b Register to read from.

Returns

Value read from register.

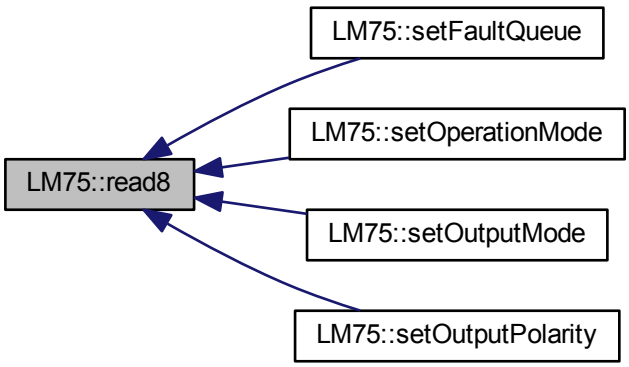
Definition at line 180 of file LM75.cpp.

References _addr.

Referenced by setFaultQueue(), setOperationMode(), setOutputMode(), and setOutputPolarity().

```
00180                                {
00181
00182     // send the device address then the register pointer byte
00183     Wire.beginTransmission(_addr);
00184     Wire.write(reg);
00185 //    Wire.endTransmission(false);
00186     Wire.endTransmission();
00187
00188     // resend device address then get the returned byte
00189     Wire.requestFrom(_addr, (uint8_t)1);
00190
00191     // data is returned as 1 byte
00192     return Wire.read();
00193 }
```

Here is the caller graph for this function:



5.1.4.8 readSetPoint()

```
double LM75::readSetPoint (
             tempUnit_t tunit = LM75_TC,
             setPointType spt = SPT_OVERTEMP )
```

Return the specified setpoint in specified units.

Remarks

Setpoint is returned as a 2's complement value in the most significant 9 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.

Parameters

in	*tunit*	Temperature units to convert raw data to.
in	*spt*	Setpoint type (overtemp or hysteresis)

Returns

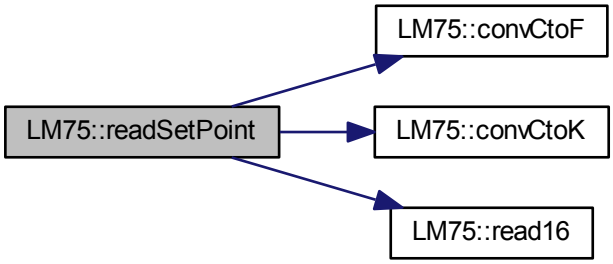
Temperature.

Definition at line 93 of file LM75.cpp.

References convCtoF(), convCtoK(), LM75_TF, LM75_THYST, LM75_TK, LM75_TOS, read16(), and SPT_OVERTEMP.

```
00093                                                                {
00094     double val;
00095
00096     val = read16(spt == SPT_OVERTEMP ? LM75_TOS :
      LM75_THYST) * 0.125/128.0;
00097     switch(tunit) {
00098         case    LM75_TK : return convCtoK(val);
00099         case    LM75_TF : return convCtoF(val);
00100         default : return val;
00101     }
00102 }
```

Here is the call graph for this function:



5.1.4.9 readTemp()

```
double LM75::readTemp (
              tempUnit_t tunit = LM75_TC )
```

Return the measured temperature in specified units.

Remarks

Temperature is returned as a 2's complement value in the most significant 11 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.

Parameters

in	*tunit*	Temperature units to convert raw data to.

Returns

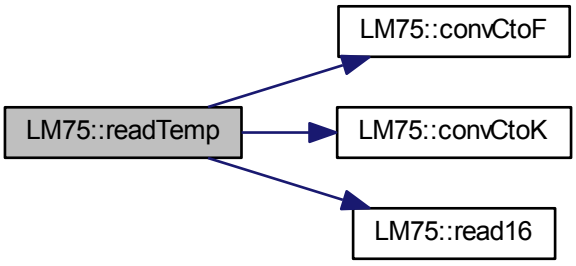
Temperature.

Definition at line 74 of file LM75.cpp.

References convCtoF(), convCtoK(), LM75_TEMP, LM75_TF, LM75_TK, and read16().

```
00074                                       {
00075     double temp;
00076
00077     temp = read16(LM75_TEMP) * 0.125/32.0;
00078     switch(tunit) {
00079         case LM75_TK : return convCtoK(temp);
00080         case LM75_TF : return convCtoF(temp);
00081     }
00082     return temp;
00083 }
```

Here is the call graph for this function:



5.1.4.10 setFaultQueue()

```
void LM75::setFaultQueue (
             faultQueue fqueue )
```

Set the value of the OS fault queue.

Parameters

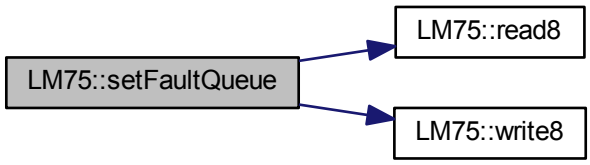
in	*fqueue*	Enumerated OS fault queue programming value.

Definition at line 157 of file LM75.cpp.

References FQ_1, FQ_2, FQ_4, FQ_6, LM75_CONF, LM75_CONF_OSFQUE_1, read8(), and write8().

```
00157                                           {
00158     uint8_t fq;
00159
00160     switch(fqueue) {
00161         case    FQ_1 : fq = LM75_CONF_OSFQUE_1; break;
00162         case    FQ_2 : fq = LM75_CONF_OSFQUE_1; break;
00163         case    FQ_4 : fq = LM75_CONF_OSFQUE_1; break;
00164         case    FQ_6 : fq = LM75_CONF_OSFQUE_1;
00165     }
00166     write8(LM75_CONF, read8(LM75_CONF) & fq);
00167 }
```

Here is the call graph for this function:



5.1.4.11 setOperationMode()

```
void LM75::setOperationMode (
              opMode mode = OPM_NORMAL )
```

Set the device operation mode.

Parameters

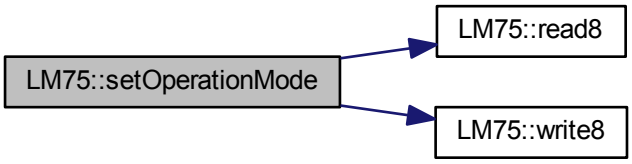
in	*mode*	Enumerated operation mode.

Definition at line 147 of file LM75.cpp.

References LM75_CONF, LM75_CONF_DOM_NORMAL, LM75_CONF_DOM_SHUTDOWN, OPM_NORMAL, read8(), and write8().

```
00147                                       {
00148
00149     write8(LM75_CONF, read8(LM75_CONF) &
00150         (mode == OPM_NORMAL ? LM75_CONF_DOM_NORMAL :
      LM75_CONF_DOM_SHUTDOWN));
00151 }
```

Here is the call graph for this function:



5.1.4.12 setOutputMode()

```
void LM75::setOutputMode (
            outputMode mode = MODE_COMP )
```

Set the OS operation mode.

Parameters

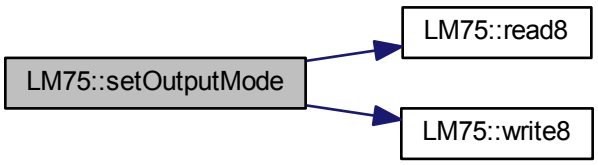
in	*mode*	Enumerated OS operation mode.

Definition at line 137 of file LM75.cpp.

References LM75_CONF, LM75_CONF_OSOM_COMP, LM75_CONF_OSOM_INT, MODE_COMP, read8(), and write8().

```
00137                                       {
00138
00139     write8(LM75_CONF, read8(LM75_CONF) &
00140         (mode == MODE_COMP ? LM75_CONF_OSOM_COMP :
      LM75_CONF_OSOM_INT));
00141 }
```

Here is the call graph for this function:



5.1.4.13 setOutputPolarity()

```
void LM75::setOutputPolarity (
            outputPolarity pol = POL_LOW )
```

Set the OS polarity.

Parameters

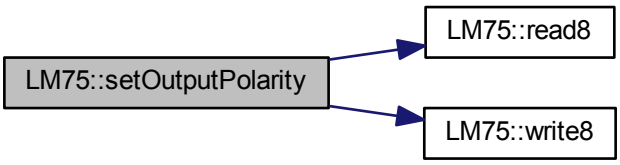
in	*pol*	Enumerated OS polarity selection.

Definition at line 127 of file LM75.cpp.

References LM75_CONF, LM75_CONF_OSPOL_AH, LM75_CONF_OSPOL_AL, POL_LOW, read8(), and write8().

```
00127                                                {
00128
00129     write8(LM75_CONF, read8(LM75_CONF) &
00130         (pol == POL_LOW ? LM75_CONF_OSPOL_AL :
      LM75_CONF_OSPOL_AH));
00131 }
```

Here is the call graph for this function:



5.1.4.14 write16()

```
void LM75::write16 (
            uint8_t reg,
            uint16_t data )  [private]
```

Write a value to a 16 bit register in the device.

Parameters

in	*reg*	Register to write to.
in	*data*	Value to write.

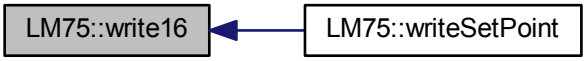
Definition at line 241 of file LM75.cpp.

References _addr.

Referenced by writeSetPoint().

```
00241                                           {
00242
00243     // send the device address then the register pointer byte
00244     Wire.beginTransmission(_addr);
00245     Wire.write(reg);
00246
00247     // write the data high byte first
00248     Wire.write(highByte(data));
00249     Wire.write(lowByte(data));
00250     Wire.endTransmission(true);
00251 }
```

Here is the caller graph for this function:



5.1.4.15 write8()

```
void LM75::write8 (
            uint8_t reg,
            uint8_t data )  [private]
```

Write a value to an 8 bit register in the device.

Parameters

in	*reg*	Register to write to.
in	*data*	Value to write.

Definition at line 224 of file LM75.cpp.

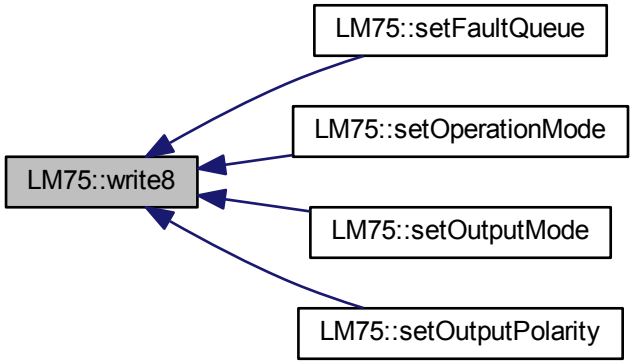
References _addr.

Referenced by setFaultQueue(), setOperationMode(), setOutputMode(), and setOutputPolarity().

```
00224                                           {
00225
00226     // send the device address then the register pointer byte
00227     Wire.beginTransmission(_addr);
00228     Wire.write(reg);
```

```
00229
00230     // write the data
00231     Wire.write(data);
00232 //    Wire.endTransmission(true);
00233     Wire.endTransmission();
00234 }
```

Here is the caller graph for this function:



5.1.4.16 writeSetPoint()

```
void LM75::writeSetPoint (
            double val,
            tempUnit_t tunit = LM75_TC,
            setPointType spt = SPT_OVERTEMP )
```

Write the specified setpoint supplied in the specified units.

Remarks

Setpoint is written as a 2's complement value in the most significant 9 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.

Parameters

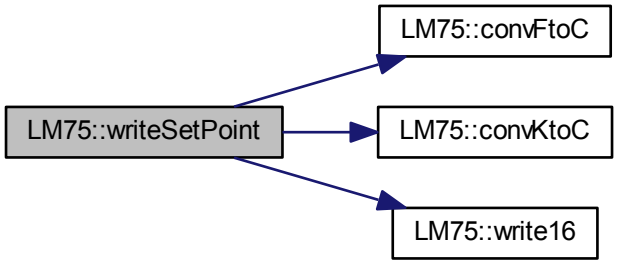
in	*val*	Setpoint temperature.
in	*tunit*	Temperature units of the setpoint.
in	*spt*	Setpoint type (overtemp or hysteresis)

Definition at line 112 of file LM75.cpp.

References convFtoC(), convKtoC(), LM75_TF, LM75_THYST, LM75_TK, LM75_TOS, SPT_OVERTEMP, and write16().

```
00112                                                                   {
00113     uint16_t data;
00114
00115     switch(tunit) {
00116         case    LM75_TK : val = convKtoC(val); break;
00117         case    LM75_TF : val = convFtoC(val);
00118     }
00119     data = (uint16_t)(val * 128.0/0.125 + 0.5);
00120     write16(spt == SPT_OVERTEMP ? LM75_TOS :
      LM75_THYST, data);
00121 }
```

Here is the call graph for this function:



5.1.5 Member Data Documentation

5.1.5.1 _addr

```
uint8_t LM75::_addr  [private]
```

Slave address

Definition at line 133 of file LM75.h.

Referenced by getAddr(), LM75(), read16(), read8(), write16(), and write8().

5.1.5.2 busAddr

```
Property<uint8_t, LM75> LM75::busAddr
```

I2C Bus address property

Definition at line 87 of file LM75.h.

Referenced by LM75().

The documentation for this class was generated from the following files:

- LM75.h
- LM75.cpp

Chapter 6

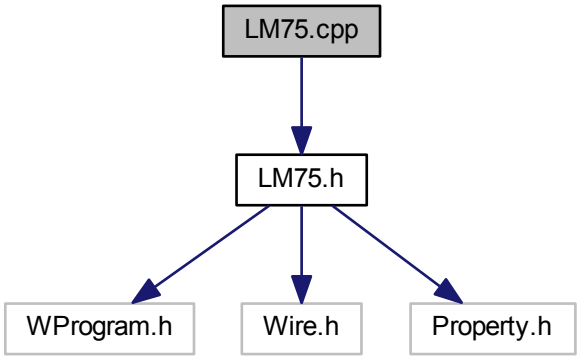
File Documentation

6.1 LM75.cpp File Reference

LM75 Family Device Driver Library - CPP Source file.

```
#include "LM75.h"
```

Include dependency graph for LM75.cpp:



6.1.1 Detailed Description

LM75 Family Device Driver Library - CPP Source file.

Details

Based on the LM75 Family Data Sheet 3901090614 Rev 004 09jun2008.

- The current implementation does not manage PWM (only digital data by I2C).
- Sleep mode is not implemented yet.

Note

THIS IS ONLY A PARTIAL RELEASE. THIS DEVICE CLASS IS CURRENTLY UNDERGOING ACTIVE D↩ EVELOPMENT AND IS STILL MISSING SOME IMPORTANT FEATURES. PLEASE KEEP THIS IN MIND IF YOU DECIDE TO USE THIS PARTICULAR CODE FOR ANYTHING.

Author

J. F. Fitter jfitter@eagleairaust.com.au

Version

1.0

Date

jan2016

Copyright

License

Definition in file LM75.cpp.

6.2 LM75.cpp

```
00001 /**************************************************************************************************//**
00002  *  \brief      LM75 Family Device Driver Library - CPP Source file
00003  *  \par
00004  *  \par        Details
00005  *              Based on the LM75 Family Data Sheet 3901090614 Rev 004 09jun2008.
00006  *  \li         The current implementation does not manage PWM (only digital data by I2C).
00007  *  \li         Sleep mode is not implemented yet.
00008  *
00009  *  \note       THIS IS ONLY A PARTIAL RELEASE. THIS DEVICE CLASS IS CURRENTLY UNDERGOING
00010  *              ACTIVE DEVELOPMENT AND IS STILL MISSING SOME IMPORTANT FEATURES. PLEASE KEEP
00011  *              THIS IN MIND IF YOU DECIDE TO USE THIS PARTICULAR CODE FOR ANYTHING.
00012  *
00013  *  \file       LM75.CPP
00014  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00015  *  \version    1.0
00016  *  \date       jan2016
00017  *  \copyright  Copyright (c) 2016-2017 John Fitter.  All right reserved.
00018  *
00019  *  \par License
00020  *              GNU Public License. Permission is hereby granted, free of charge, to any
00021  *              person obtaining a copy of this software and associated documentation files
00022  *              (the "Software"), to deal in the Software without restriction, including
00023  *              without limitation the rights to use, copy, modify, merge, publish, distribute,
00024  *              sublicense, and/or sell copies of the Software, and to permit persons to whom
00025  *              the Software is furnished to do so, subject to the following conditions:
00026  *  \par
00027  *              The above copyright notice and this permission notice shall be included in
00028  *              all copies or substantial portions of the Software.
00029  *  \par
00030  *              THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00031  *              IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00032  *              FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00033  *              AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00034  *              LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00035  *              OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
00036  *              THE SOFTWARE.
00037  *
00038  **************************************************************************************************/
00039
00040 #include "LM75.h"
00041
00042 /**************************************************************************************************/
00043 /*  LM75 Device class functions.                                                                  */
00044 /**************************************************************************************************/
00045
00046 /**
00047  *  \brief              LM75 Device class constructor.
00048  *  \param [in] i2caddr  Device address (default: published value).
00049  */
00050 LM75::LM75(uint8_t i2caddr) {
00051
00052     busAddr.Set_Class(this);
00053     busAddr.Set_Get(&LM75::getAddr);
00054
00055     _addr  = i2caddr;
00056     _ready = false;
00057 }
00058
00059 /**
00060  *  \brief  Initialize the device and the i2c interface.
00061  */
00062 boolean LM75::begin(void) {
00063
00064     return _ready = true;
00065 }
00066
00067 /**
00068  *  \brief             Return the measured temperature in specified units.
00069  *  \remarks           Temperature is returned as a 2's complement value in the most significant
00070  *                     11 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.
00071  *  \param [in] tunit  Temperature units to convert raw data to.
00072  *  \return            Temperature.
00073  */
00074 double LM75::readTemp(tempUnit_t tunit) {
00075     double temp;
00076
00077     temp = read16(LM75_TEMP) * 0.125/32.0;
00078     switch(tunit) {
00079         case LM75_TK : return convCtoK(temp);
00080         case LM75_TF : return convCtoF(temp);
00081     }
00082     return temp;
00083 }
00084
```

```
00085 /**
00086  *  \brief              Return the specified setpoint in specified units.
00087  *  \remarks            Setpoint is returned as a 2's complement value in the most significant
00088  *                      9 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.
00089  *  \param [in] tunit   Temperature units to convert raw data to.
00090  *  \param [in] spt     Setpoint type (overtemp or hysteresis)
00091  *  \return             Temperature.
00092  */
00093 double LM75::readSetPoint(tempUnit_t tunit,
      setPointType spt) {
00094     double val;
00095
00096     val = read16(spt == SPT_OVERTEMP ? LM75_TOS :
      LM75_THYST) * 0.125/128.0;
00097     switch(tunit) {
00098         case    LM75_TK : return convCtoK(val);
00099         case    LM75_TF : return convCtoF(val);
00100         default : return val;
00101     }
00102 }
00103
00104 /**
00105  *  \brief              Write the specified setpoint supplied in the specified units.
00106  *  \remarks            Setpoint is written as a 2's complement value in the most significant
00107  *                      9 bits of a 16 bit field expressed in degrees C to a resolution of 0.125C.
00108  *  \param [in] val     Setpoint temperature.
00109  *  \param [in] tunit   Temperature units of the setpoint.
00110  *  \param [in] spt     Setpoint type (overtemp or hysteresis)
00111  */
00112 void LM75::writeSetPoint(double val, tempUnit_t tunit,
      setPointType spt) {
00113     uint16_t data;
00114
00115     switch(tunit) {
00116         case    LM75_TK : val = convKtoC(val); break;
00117         case    LM75_TF : val = convFtoC(val);
00118     }
00119     data = (uint16_t)(val * 128.0/0.125 + 0.5);
00120     write16(spt == SPT_OVERTEMP ? LM75_TOS :
      LM75_THYST, data);
00121 }
00122
00123 /**
00124  *  \brief              Set the OS polarity.
00125  *  \param [in] pol     Enumerated OS polarity selection.
00126  */
00127 void LM75::setOutputPolarity(outputPolarity pol) {
00128
00129     write8(LM75_CONF, read8(LM75_CONF) &
00130         (pol == POL_LOW ? LM75_CONF_OSPOL_AL :
      LM75_CONF_OSPOL_AH));
00131 }
00132
00133 /**
00134  *  \brief              Set the OS operation mode.
00135  *  \param [in] mode    Enumerated OS operation mode.
00136  */
00137 void LM75::setOutputMode(outputMode mode) {
00138
00139     write8(LM75_CONF, read8(LM75_CONF) &
00140         (mode == MODE_COMP ? LM75_CONF_OSOM_COMP :
      LM75_CONF_OSOM_INT));
00141 }
00142
00143 /**
00144  *  \brief              Set the device operation mode.
00145  *  \param [in] mode    Enumerated operation mode.
00146  */
00147 void LM75::setOperationMode(opMode mode) {
00148
00149     write8(LM75_CONF, read8(LM75_CONF) &
00150         (mode == OPM_NORMAL ? LM75_CONF_DOM_NORMAL :
      LM75_CONF_DOM_SHUTDOWN));
00151 }
00152
00153 /**
00154  *  \brief              Set the value of the OS fault queue.
00155  *  \param [in] fqueue  Enumerated OS fault queue programming value.
00156  */
00157 void LM75::setFaultQueue(faultQueue fqueue) {
00158     uint8_t fq;
00159
00160     switch(fqueue) {
00161         case    FQ_1 : fq = LM75_CONF_OSFQUE_1; break;
00162         case    FQ_2 : fq = LM75_CONF_OSFQUE_1; break;
00163         case    FQ_4 : fq = LM75_CONF_OSFQUE_1; break;
00164         case    FQ_6 : fq = LM75_CONF_OSFQUE_1;
```

```
00165     }
00166     write8(LM75_CONF, read8(LM75_CONF) & fq);
00167 }
00168
00169 /**
00170  *  \brief            Return the device I2C Bus address.
00171  *  \return           Device address.
00172  */
00173 uint8_t LM75::getAddr(void) {return _addr;}
00174
00175 /**
00176  *  \brief            Return an 8 bit register value from the device.
00177  *  \param [in] reg   8b Register to read from.
00178  *  \return           Value read from register.
00179  */
00180 uint8_t LM75::read8(uint8_t reg) {
00181
00182     // send the device address then the register pointer byte
00183     Wire.beginTransmission(_addr);
00184     Wire.write(reg);
00185 //    Wire.endTransmission(false);
00186     Wire.endTransmission();
00187
00188     // resend device address then get the returned byte
00189     Wire.requestFrom(_addr, (uint8_t)1);
00190
00191     // data is returned as 1 byte
00192     return Wire.read();
00193 }
00194
00195 /**
00196  *  \brief            Return a 16 bit register value from the device.
00197  *  \param [in] reg   16b Register to read from.
00198  *  \return           Value read from register.
00199  */
00200 uint16_t LM75::read16(uint8_t reg) {
00201     uint16_t val;
00202
00203     // send the device address then the register pointer byte
00204     Wire.beginTransmission(_addr);
00205     Wire.write(reg);
00206 //    Wire.endTransmission(false);
00207     Wire.endTransmission();
00208
00209     // resend device address then get the 2 returned bytes
00210     Wire.requestFrom(_addr, (uint8_t)2);
00211
00212     // data is returned as 2 bytes big endian
00213     val = Wire.read() << 8;
00214     val |= Wire.read();
00215
00216     return val;
00217 }
00218
00219 /**
00220  *  \brief            Write a value to an 8 bit register in the device.
00221  *  \param [in] reg   Register to write to.
00222  *  \param [in] data  Value to write.
00223  */
00224 void LM75::write8(uint8_t reg, uint8_t data) {
00225
00226     // send the device address then the register pointer byte
00227     Wire.beginTransmission(_addr);
00228     Wire.write(reg);
00229
00230     // write the data
00231     Wire.write(data);
00232 //    Wire.endTransmission(true);
00233     Wire.endTransmission();
00234 }
00235
00236 /**
00237  *  \brief            Write a value to a 16 bit register in the device.
00238  *  \param [in] reg   Register to write to.
00239  *  \param [in] data  Value to write.
00240  */
00241 void LM75::write16(uint8_t reg, uint16_t data) {
00242
00243     // send the device address then the register pointer byte
00244     Wire.beginTransmission(_addr);
00245     Wire.write(reg);
00246
00247     // write the data high byte first
00248     Wire.write(highByte(data));
00249     Wire.write(lowByte(data));
00250     Wire.endTransmission(true);
00251 }
```

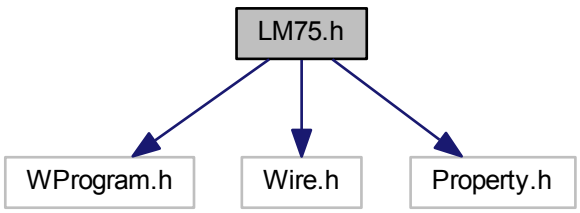
```
00252
00253 /**
00254  *  \brief            Convert temperature in degrees C to degrees K.
00255  *  \param [in] degC  Temperature in degrees Centigrade.
00256  *  \return           Temperature in degrees Kelvin.
00257  */
00258 double LM75::convCtoK(double degC) {return degC + 273.15;}
00259
00260 /**
00261  *  \brief            Convert temperature in degrees C to degrees F.
00262  *  \param [in] degC  Temperature in degrees Centigrade.
00263  *  \return           Temperature in degrees Fahrenheit.
00264  */
00265 double LM75::convCtoF(double degC) {return (degC * 1.8) + 32.0;}
00266
00267 /**
00268  *  \brief            Convert temperature in degrees K to degrees C.
00269  *  \param [in] degK  Temperature in degrees Kelvin.
00270  *  \return           Temperature in degrees Centigrade.
00271  */
00272 double LM75::convKtoC(double degK) {return degK - 273.15;}
00273
00274 /**
00275  *  \brief            Convert temperature in degrees F to degrees C.
00276  *  \param [in] degF  Temperature in degrees Fahrenheit.
00277  *  \return           Temperature in degrees Centigrade.
00278  */
00279 double LM75::convFtoC(double degF) {return (degF - 32.0) / 1.8;}
00280
00281
```

6.3 LM75.h File Reference

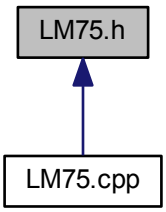
LM75 Family Device Driver Library - CPP Header file.

```
#include "WProgram.h"
#include <Wire.h>
#include "Property.h"
```

Include dependency graph for LM75.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LM75

Macros

- #define LM75_I2CDEFAULTADDR 0x48
- #define LM75_BROADCASTADDR 0
- #define LM75_CONF 0x01
- #define LM75_TEMP 0x00
- #define LM75_TOS 0x03
- #define LM75_THYST 0x02
- #define LM75_CONF_RES 0x00
- #define LM75_CONF_OSFQUE_1 0x00
- #define LM75_CONF_OSFQUE_2 0x08
- #define LM75_CONF_OSFQUE_4 0x10
- #define LM75_CONF_OSFQUE_6 0x18
- #define LM75_CONF_OSPOL_AL 0x00
- #define LM75_CONF_OSPOL_AH 0x04
- #define LM75_CONF_OSOM_COMP 0x00
- #define LM75_CONF_OSOM_INT 0x02
- #define LM75_CONF_DOM_NORMAL 0x00
- #define LM75_CONF_DOM_SHUTDOWN 0x01

6.3.1 Detailed Description

LM75 Family Device Driver Library - CPP Header file.

Details

Based on the Melexis LM75 Family Data Sheet 3901090614 Rev 004 09jun2008.

- The current implementation does not manage PWM (only digital data by I2C).
- Sleep mode is not implemented yet.

Note

THIS IS ONLY A PARTIAL RELEASE. THIS DEVICE CLASS IS CURRENTLY UNDERGOING ACTIVE D↩ EVELOPMENT AND IS STILL MISSING SOME IMPORTANT FEATURES. PLEASE KEEP THIS IN MIND IF YOU DECIDE TO USE THIS PARTICULAR CODE FOR ANYTHING.

Author

J. F. Fitter jfitter@eagleairaust.com.au

Version

1.0

Date

jan2016

Copyright

License

Definition in file LM75.h.

6.3.2 Macro Definition Documentation

6.3.2.1 LM75_BROADCASTADDR

```
#define LM75_BROADCASTADDR 0
```

Device broadcast slave address

Definition at line 56 of file LM75.h.

6.3.2.2 LM75_CONF

```
#define LM75_CONF 0x01
```

REGISTER addresses. RAM reg - Configuration

Definition at line 59 of file LM75.h.

Referenced by LM75::setFaultQueue(), LM75::setOperationMode(), LM75::setOutputMode(), and LM75::setOutputPolarity().

6.3.2.3 LM75_CONF_DOM_NORMAL

```
#define LM75_CONF_DOM_NORMAL 0x00
```

Device operation mode - normal

Definition at line 74 of file LM75.h.

Referenced by LM75::setOperationMode().

6.3.2.4 LM75_CONF_DOM_SHUTDOWN

```
#define LM75_CONF_DOM_SHUTDOWN 0x01
```

Device operation mode - shutdown

Definition at line 75 of file LM75.h.

Referenced by LM75::setOperationMode().

6.3.2.5 LM75_CONF_OSFQUE_1

```
#define LM75_CONF_OSFQUE_1 0x00
```

OS fault queue programming value = 1

Definition at line 66 of file LM75.h.

Referenced by LM75::setFaultQueue().

6.3.2.6 LM75_CONF_OSFQUE_2

```
#define LM75_CONF_OSFQUE_2 0x08
```

OS fault queue programming value = 2

Definition at line 67 of file LM75.h.

6.3.2.7 LM75_CONF_OSFQUE_4

```
#define LM75_CONF_OSFQUE_4 0x10
```

OS fault queue programming value = 4

Definition at line 68 of file LM75.h.

6.3.2.8 LM75_CONF_OSFQUE_6

```
#define LM75_CONF_OSFQUE_6 0x18
```

OS fault queue programming value = 6

Definition at line 69 of file LM75.h.

6.3.2.9 LM75_CONF_OSOM_COMP

```
#define LM75_CONF_OSOM_COMP 0x00
```

OS operation mode - comparator

Definition at line 72 of file LM75.h.

Referenced by LM75::setOutputMode().

6.3.2.10 LM75_CONF_OSOM_INT

```
#define LM75_CONF_OSOM_INT 0x02
```

OS operation mode - interrupt

Definition at line 73 of file LM75.h.

Referenced by LM75::setOutputMode().

6.3.2.11 LM75_CONF_OSPOL_AH

```
#define LM75_CONF_OSPOL_AH 0x04
```

OS polarity selection active HIGH

Definition at line 71 of file LM75.h.

Referenced by LM75::setOutputPolarity().

6.3.2.12 LM75_CONF_OSPOL_AL

```
#define LM75_CONF_OSPOL_AL 0x00
```

OS polarity selection active LOW

Definition at line 70 of file LM75.h.

Referenced by LM75::setOutputPolarity().

6.3.2.13 LM75_CONF_RES

```
#define LM75_CONF_RES 0x00
```

CONFIGURATION bits masks. Manufacturer reserved bits

Definition at line 65 of file LM75.h.

6.3.2.14 LM75_I2CDEFAULTADDR

```
#define LM75_I2CDEFAULTADDR 0x48
```

Device default slave address

Definition at line 55 of file LM75.h.

6.3.2.15 LM75_TEMP

```
#define LM75_TEMP 0x00
```

RAM reg - Temperature

Definition at line 60 of file LM75.h.

Referenced by LM75::readTemp().

6.3.2.16 LM75_THYST

```
#define LM75_THYST 0x02
```

RAM reg - Hysteresis

Definition at line 62 of file LM75.h.

Referenced by LM75::readSetPoint(), and LM75::writeSetPoint().

6.3.2.17 LM75_TOS

```
#define LM75_TOS 0x03
```

RAM reg - Overtemperature shutdown threshold

Definition at line 61 of file LM75.h.

Referenced by LM75::readSetPoint(), and LM75::writeSetPoint().

6.4 LM75.h

```
00001 #ifndef _LM75_H_
00002 #define _LM75_H_
00003
00004 /*****************************************************************************************************//**
00005  *  \brief      LM75 Family Device Driver Library - CPP Header file
00006  *  \par
00007  *  \par        Details
00008  *              Based on the Melexis LM75 Family Data Sheet 3901090614 Rev 004 09jun2008.
00009  *  \li         The current implementation does not manage PWM (only digital data by I2C).
00010  *  \li         Sleep mode is not implemented yet.
00011  *
00012  *  \note       THIS IS ONLY A PARTIAL RELEASE. THIS DEVICE CLASS IS CURRENTLY UNDERGOING
00013  *              ACTIVE DEVELOPMENT AND IS STILL MISSING SOME IMPORTANT FEATURES. PLEASE KEEP
00014  *              THIS IN MIND IF YOU DECIDE TO USE THIS PARTICULAR CODE FOR ANYTHING.
00015  *
00016  *  \file       LM75.H
00017  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00018  *  \version    1.0
00019  *  \date       jan2016
00020  *  \copyright  Copyright (c) 2016-2017 John Fitter.  All right reserved.
00021  *
00022  *  \par License
00023  *              GNU Public License. Permission is hereby granted, free of charge, to any
00024  *              person obtaining a copy of this software and associated documentation files
00025  *              (the "Software"), to deal in the Software without restriction, including
00026  *              without limitation the rights to use, copy, modify, merge, publish, distribute,
00027  *              sublicense, and/or sell copies of the Software, and to permit persons to whom
00028  *              the Software is furnished to do so, subject to the following conditions:
00029  *  \par
00030  *              The above copyright notice and this permission notice shall be included in
00031  *              all copies or substantial portions of the Software.
00032  *  \par
00033  *              THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034  *              IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035  *              FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00036  *              AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00037  *              LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00038  *              OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
00039  *              THE SOFTWARE.
00040  *
00041  *****************************************************************************************************/
00042
00043 #if (ARDUINO >= 100)
00044     #include "Arduino.h"
00045 #else
```

```
00046     #include "WProgram.h"
00047 #endif
00048 #include <Wire.h>
00049 #include "Property.h"
00050
00051 /**************************************************************************************************/
00052 /* Definitions                                                                                    */
00053 /**************************************************************************************************/
00054
00055 #define LM75_I2CDEFAULTADDR     0x48    /**< Device default slave address */
00056 #define LM75_BROADCASTADDR      0       /**< Device broadcast slave address */
00057
00058 /** REGISTER addresses. */
00059 #define LM75_CONF               0x01    /**< RAM reg - Configuration */
00060 #define LM75_TEMP               0x00    /**< RAM reg - Temperature */
00061 #define LM75_TOS                0x03    /**< RAM reg - Overtemperature shutdown threshold */
00062 #define LM75_THYST              0x02    /**< RAM reg - Hysteresis */
00063
00064 /** CONFIGURATION bits masks. */
00065 #define LM75_CONF_RES           0x00    /**< Manufacturer reserved bits */
00066 #define LM75_CONF_OSFQUE_1      0x00    /**< OS fault queue programming value = 1 */
00067 #define LM75_CONF_OSFQUE_2      0x08    /**< OS fault queue programming value = 2 */
00068 #define LM75_CONF_OSFQUE_4      0x10    /**< OS fault queue programming value = 4 */
00069 #define LM75_CONF_OSFQUE_6      0x18    /**< OS fault queue programming value = 6 */
00070 #define LM75_CONF_OSPOL_AL      0x00    /**< OS polarity selection active LOW */
00071 #define LM75_CONF_OSPOL_AH      0x04    /**< OS polarity selection active HIGH */
00072 #define LM75_CONF_OSOM_COMP     0x00    /**< OS operation mode - comparator */
00073 #define LM75_CONF_OSOM_INT      0x02    /**< OS operation mode - interrupt */
00074 #define LM75_CONF_DOM_NORMAL    0x00    /**< Device operation mode - normal */
00075 #define LM75_CONF_DOM_SHUTDOWN  0x01    /**< Device operation mode - shutdown */
00076
00077 /**************************************************************************************************/
00078 /* LM75 Device class.                                                                             */
00079 /**************************************************************************************************/
00080
00081 class LM75 {
00082 public:
00083     LM75(uint8_t = LM75_I2CDEFAULTADDR);                    /**< constructor */
00084
00085     boolean  begin();
00086
00087     Property<uint8_t, LM75> busAddr;                        /**< I2C Bus address property */
00088
00089     /** Enumerations for temperature units. */
00090     enum tempUnit_t {LM75_TK,                               /**< degrees Kelvin */
00091                      LM75_TC,                               /**< degrees Centigrade */
00092                      LM75_TF                                /**< degrees Fahrenheit */
00093     };
00094     /** Enumerations for setpoint registers. */
00095     enum setPointType {SPT_OVERTEMP,                        /**< overtemp shutdown
 register */
00096                        SPT_HYSTERESIS                       /**< hysteresis register */
00097     };
00098     /** Enumerations for output polarity. */
00099     enum outputPolarity {POL_LOW,                           /**< OS output active low
*/
00100                          POL_HIGH                           /**< OS output active high */
00101     };
00102     /** Enumerations for output mode. */
00103     enum outputMode {MODE_COMP,                             /**< OS comparator mode */
00104                      MODE_INT                               /**< OS interrupt mode */
00105     };
00106     /** Enumerations for operating mode. */
00107     enum opMode {OPM_NORMAL,                                /**< normal operation mode */
00108                  OPM_SHDN                                   /**< shutdown mode */
00109     };
00110     /** Enumerations for fault queue length. */
00111     enum faultQueue {FQ_1 = 1,                              /**< fault queue value = 1 */
00112                      FQ_2 = 2,                              /**< fault queue value = 2 */
00113                      FQ_4 = 4,                              /**< fault queue value = 4 */
00114                      FQ_6 = 6                               /**< fault queue value = 6 */
00115     };
00116
00117     boolean isReady(void) { return _ready; };
00118     double readTemp(tempUnit_t = LM75_TC);
00119     double readSetPoint(tempUnit_t = LM75_TC,
      setPointType = SPT_OVERTEMP);
00120     void writeSetPoint(double, tempUnit_t = LM75_TC,
      setPointType = SPT_OVERTEMP);
00121     void setOutputPolarity(outputPolarity =
      POL_LOW);
00122     void setOutputMode(outputMode = MODE_COMP);
00123     void setOperationMode(opMode = OPM_NORMAL);
00124     void setFaultQueue(faultQueue);
00125
00126     double convCtoK(double degC);
00127     double convCtoF(double degC);
```

```
00128     double convKtoC(double degK);
00129     double convFtoC(double degF);
00130
00131 private:
00132     boolean _ready;
00133     uint8_t _addr;                                          /**< Slave address */
00134     uint8_t getAddr(void);
00135
00136     uint8_t read8(uint8_t);
00137     uint16_t read16(uint8_t);
00138     void write8(uint8_t, uint8_t);
00139     void write16(uint8_t, uint16_t);
00140 };
00141
00142 #endif /* _LM75_H_ */
```

Index

_addr
    LM75, 35

busAddr
    LM75, 35

convCtoF
    LM75, 21
convCtoK
    LM75, 22
convFtoC
    LM75, 23
convKtoC
    LM75, 24

faultQueue
    LM75, 18

getAddr
    LM75, 24

LM75, 17
    _addr, 35
    busAddr, 35
    convCtoF, 21
    convCtoK, 22
    convFtoC, 23
    convKtoC, 24
    faultQueue, 18
    getAddr, 24
    LM75, 21
    opMode, 19
    outputMode, 19
    outputPolarity, 19
    read16, 25
    read8, 26
    readSetPoint, 27
    readTemp, 28
    setFaultQueue, 29
    setOperationMode, 30
    setOutputMode, 31
    setOutputPolarity, 31
    setPointType, 20
    tempUnit_t, 20
    write16, 32
    write8, 33
    writeSetPoint, 34
LM75.cpp, 37
LM75.h, 42
    LM75_BROADCASTADDR, 44
    LM75_CONF_DOM_NORMAL, 45
    LM75_CONF_DOM_SHUTDOWN, 45
    LM75_CONF_OSFQUE_1, 45
    LM75_CONF_OSFQUE_2, 45
    LM75_CONF_OSFQUE_4, 46
    LM75_CONF_OSFQUE_6, 46
    LM75_CONF_OSOM_COMP, 46
    LM75_CONF_OSOM_INT, 46
    LM75_CONF_OSPOL_AH, 46
    LM75_CONF_OSPOL_AL, 47
    LM75_CONF_RES, 47
    LM75_CONF, 44
    LM75_I2CDEFAULTADDR, 47
    LM75_TEMP, 47
    LM75_THYST, 47
    LM75_TOS, 48
LM75_BROADCASTADDR
    LM75.h, 44
LM75_CONF_DOM_NORMAL
    LM75.h, 45
LM75_CONF_DOM_SHUTDOWN
    LM75.h, 45
LM75_CONF_OSFQUE_1
    LM75.h, 45
LM75_CONF_OSFQUE_2
    LM75.h, 45
LM75_CONF_OSFQUE_4
    LM75.h, 46
LM75_CONF_OSFQUE_6
    LM75.h, 46
LM75_CONF_OSOM_COMP
    LM75.h, 46
LM75_CONF_OSOM_INT
    LM75.h, 46
LM75_CONF_OSPOL_AH
    LM75.h, 46
LM75_CONF_OSPOL_AL
    LM75.h, 47
LM75_CONF_RES
    LM75.h, 47
LM75_CONF
    LM75.h, 44
LM75_I2CDEFAULTADDR
    LM75.h, 47
LM75_TEMP
    LM75.h, 47
LM75_THYST
    LM75.h, 47
LM75_TOS
    LM75.h, 48

opMode
LM75, 19
outputMode
LM75, 19
outputPolarity
LM75, 19

read16
LM75, 25
read8
LM75, 26
readSetPoint
LM75, 27
readTemp
LM75, 28

setFaultQueue
LM75, 29
setOperationMode
LM75, 30
setOutputMode
LM75, 31
setOutputPolarity
LM75, 31
setPointType
LM75, 20

tempUnit_t
LM75, 20

write16
LM75, 32
write8
LM75, 33
writeSetPoint
LM75, 34