MLX90614 Device Driver

1.0

Generated by Doxygen 1.8.16

Fri Sep 20 2019 15:29:42

1 Arduino Library for the MLX90614 Temperature Sensor

This library was written to enable remote sensing of the temperature of the rotors of outrunner style brushless DC motors used in remotely piloted aircraft, for the purpose of real time data logging and air to ground telemetry.

These sensors use the SMB bus protocol to communicate. This is similar, though not identical, to the I2C bus. There is enough similarity to enable the Arduino standard Wire library to communicate with the device, however not all features can be implemented, for example it is not possible to read the flags register with standard Wire functions. 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 ***/mlx90614***.
Check that the ***/mlx90614*** folder contains the following files;

MLX90614.cpp
MLX90614.h
MLX90614.chm
MLX90614.pdf
Crc8.cpp
Crc8.h
property.h
doxyfile

Place the ***/mlx90614*** library folder into your ***arduinosketchfolder/libraries/*** folder.
You may need to create the libraries subfolder if its your first library. Restart the IDE.

Documentation

MLX90614.chm and *MLX90614.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.
This library was inspired by a library written by Adafruit Industries.

License

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. EXCEPT 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 IMPLIED, 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 PROGRAM 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 COPYRIGHT 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 CONSEQUENTIAL 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 PROGRAMS), 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.

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Class Documentation

5.1 CRC8 Class Reference

```
#include <Crc8.h>
```

Public Member Functions

- CRC8 (uint8_t polynomial=CRC8_DEFAULTPOLY)

  CRC8 class constructor.
- uint8_t crc8 (void)

  Return the current value of the CRC.
- uint8_t crc8 (uint8_t data)

  Update the current value of the CRC.
- void crc8Start (uint8_t poly)

  Initialize the CRC8 object.

Private Attributes

- uint8_t _crc
- uint8_t _poly

5.1.1 Detailed Description

Definition at line 37 of file Crc8.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 CRC8() `CRC8::CRC8 ( uint8_t poly = CRC8_DEFAULTPOLY )`

CRC8 class constructor.

Parameters

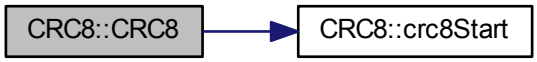
in	*poly*	8 bit CRC polynomial to use.

Definition at line 36 of file Crc8.cpp.

```
00036 {crc8Start(poly);}
```

References crc8Start().

Here is the call graph for this function:



5.1.3 Member Function Documentation

5.1.3.1 crc8() [1/2] uint8_t CRC8::crc8 (uint8_t *data*)

Update the current value of the CRC.

Parameters

in	*data*	New 8 bit data to be added to the CRC.

Returns

8 bit CRC current value.

Definition at line 49 of file Crc8.cpp.

```
00049                                {
00050     uint8_t i = 8;
00051
00052     _crc ^= data;
00053     while(i--) _crc = _crc & 0x80 ? (_crc « 1) ^ _poly : _crc « 1;
00054     return _crc;
00055 }
```

References _crc, and _poly.

5.1.3.2 crc8() [2/2] uint8_t CRC8::crc8 (void)

Return the current value of the CRC.

Returns

8 bit CRC current value.

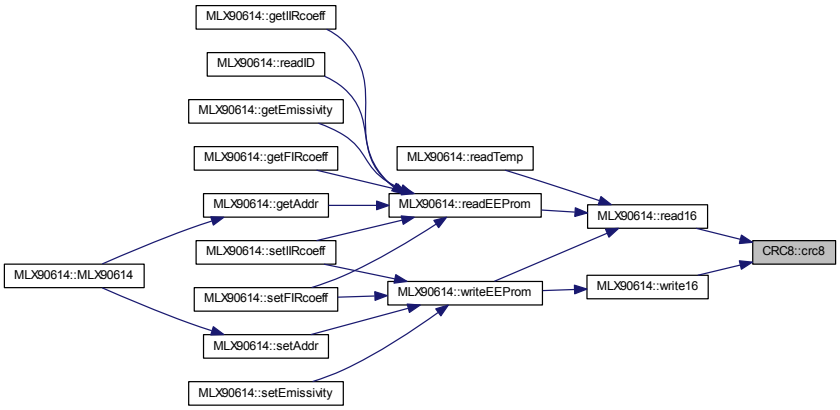
Definition at line 42 of file Crc8.cpp.

```
00042 {return _crc;}
```

References _crc.

Referenced by MLX90614::read16(), and MLX90614::write16().

Here is the caller graph for this function:



5.1.3.3 crc8Start() `void CRC8::crc8Start ( uint8_t poly )`

Initialize the CRC8 object.

Parameters

in	*poly*	8 bit CRC polynomial to use.

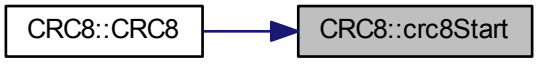
Definition at line 61 of file Crc8.cpp.

```
00061                                       {
00062     _poly = poly;
00063     _crc = 0;
00064 }
```

References _crc, and _poly.

Referenced by CRC8().

Here is the caller graph for this function:



5.1.4 Member Data Documentation

5.1.4.1 _crc uint8_t CRC8::_crc [private]

Definition at line 44 of file Crc8.h.

Referenced by crc8(), and crc8Start().

5.1.4.2 _poly uint8_t CRC8::_poly [private]

Definition at line 45 of file Crc8.h.

Referenced by crc8(), and crc8Start().

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

- Crc8.h
- Crc8.cpp

5.2 MLX90614 Class Reference

`#include <MLX90614.h>`

Public Types

- enum tempUnit_t { MLX90614_TK, MLX90614_TC, MLX90614_TF }
- enum tempSrc_t { MLX90614_SRCA, MLX90614_SRC01, MLX90614_SRC02 }

Public Member Functions

- MLX90614 (uint8_t i2caddr=MLX90614_I2CDEFAULTADDR)
  MLX90614 Device class constructor.
- boolean begin ()
  Initialize the device and the i2c interface.
- boolean isReady (void)
- uint64_t readID (void)
  Retrieve the chip ID bytes.
- uint8_t getIIRcoeff (void)
  Get the coefficients of the IIR digital filter.
- uint8_t getFIRcoeff (void)
  Get the coefficients of the FIR digital filter.
- float getEmissivity (void)
  Get the emissivity of the object.
- void setIIRcoeff (uint8_t csb=4)
  Set the coefficients of the IIR digital filter.
- void setFIRcoeff (uint8_t csb=7)
  Set the coefficients of the FIR digital filter.
- void setEmissivity (float emiss=1.0)
  Set the emissivity of the object.
- uint16_t readEEProm (uint8_t)
  Return a 16 bit value read from EEPROM.
- void writeEEProm (uint8_t, uint16_t)
  Write a 16 bit value to EEPROM after first clearing the memory.
- double readTemp (tempSrc_t=MLX90614_SRC01, tempUnit_t=MLX90614_TC)
  Return a temperature from the specified source in specified units.
- double convKtoC (double)
  Convert temperature in degrees K to degrees C.
- double convCtoF (double)
  Convert temperature in degrees C to degrees F.

Public Attributes

- Property< uint8_t, MLX90614 > busAddr
- Property< uint8_t, MLX90614 > rwError
- Property< uint8_t, MLX90614 > crc8
- Property< uint8_t, MLX90614 > pec

Private Member Functions

- uint16_t read16 (uint8_t)

  Return a 16 bit value read from RAM or EEPROM.
- void write16 (uint8_t, uint16_t)

  Write a 16 bit value to memory.
- uint8_t getRwError (void)
- uint8_t getCRC8 (void)
- uint8_t getPEC (void)
- uint8_t getAddr (void)

  Return the device SMBus address.
- void setAddr (uint8_t)

  Set device SMBus address.

Private Attributes

- boolean _ready
- uint8_t _addr
- uint8_t _rwError
- uint8_t _crc8
- uint8_t _pec

5.2.1 Detailed Description

Examples

MelexisTest.ino.

Definition at line 104 of file MLX90614.h.

5.2.2 Member Enumeration Documentation

5.2.2.1 tempSrc_t `enum MLX90614::tempSrc_t`

Enumerations for temperature measurement source.

Enumerator

MLX90614_SRCA	Chip (ambient) sensor
MLX90614_SRC01	IR source #1
MLX90614_SRC02	IR source #2

Definition at line 134 of file MLX90614.h.

```
00140         :
00141     boolean  _ready;
```

5.2.2.2 tempUnit_t enum MLX90614::tempUnit_t

Enumerations for temperature units.

Enumerator

MLX90614_TK	degrees Kelvin
MLX90614_TC	degrees Centigrade
MLX90614_TF	degrees Fahrenheit

Definition at line 129 of file MLX90614.h.

```
00131                     {MLX90614_SRCA,                    /**< Chip (ambient) sensor */
00132                      MLX90614_SRC01,                   /**< IR source #1 */
```

5.2.3 Constructor & Destructor Documentation

5.2.3.1 MLX90614() MLX90614::MLX90614 (uint8_t *i2caddr* = *MLX90614_I2CDEFAULTADDR*)

MLX90614 Device class constructor.

Parameters

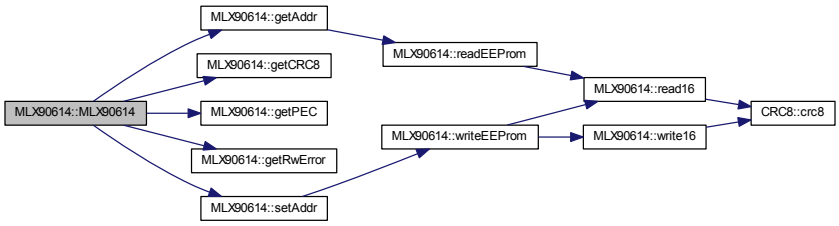
in	*i2caddr*	Device address (default: published value).

Definition at line 50 of file MLX90614.cpp.

```
00050                                    {
00051
00052     busAddr.Set_Class(this);
00053     busAddr.Set_Get(&MLX90614::getAddr);
00054     busAddr.Set_Set(&MLX90614::setAddr);
00055
00056     rwError.Set_Class(this);
00057     rwError.Set_Get(&MLX90614::getRwError);
00058
00059     pec.Set_Class(this);
00060     pec.Set_Get(&MLX90614::getPEC);
00061
00062     crc8.Set_Class(this);
00063     crc8.Set_Get(&MLX90614::getCRC8);
00064
00065     _addr = i2caddr;
00066     _ready = false;
00067 }
```

References _addr, _ready, busAddr, crc8, getAddr(), getCRC8(), getPEC(), getRwError(), pec, rwError, and setAddr().

Here is the call graph for this function:



5.2.4 Member Function Documentation

5.2.4.1 begin() `boolean MLX90614::begin ( void )`

Initialize the device and the i2c interface.

Examples

MelexisTest.ino.

Definition at line 72 of file MLX90614.cpp.

```
00072                                {
00073
00074     _rwError = _pec = _crc8 = 0;
00075     return _ready = true;
00076 }
```

References _crc8, _pec, _ready, and _rwError.

5.2.4.2 convCtoF() `double MLX90614::convCtoF ( double degC )`

Convert temperature in degrees C to degrees F.

Parameters

in	*degC*	Temperature in degrees Centigrade.

Returns

Temperature in degrees Fahrenheit.

Examples

MelexisTest.ino.

Definition at line 393 of file MLX90614.cpp.
```
00393 {return (degC * 1.8) + 32.0;}
```

Referenced by readTemp().

Here is the caller graph for this function:

5.2.4.3 convKtoC() double MLX90614::convKtoC (double degK)

Convert temperature in degrees K to degrees C.

Parameters

in	*degK*	Temperature in degrees Kelvin.

Returns

Temperature in degrees Centigrade.

Examples

MelexisTest.ino.

Definition at line 386 of file MLX90614.cpp.
```
00386 {return degK - 273.15;}
```

Referenced by readTemp().

Here is the caller graph for this function:

5.2.4.4 getAddr() uint8_t MLX90614::getAddr (void) [private]

Return the device SMBus address.

SMB bus address getter

Remarks

- Must be only device on the bus.
- Sets the library to use the new found address.

Returns

Device address.

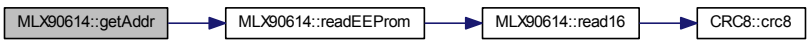
Definition at line 254 of file MLX90614.cpp.

```
00254                                {
00255     uint8_t tempAddr = _addr;
00256
00257     _rwError = 0;
00258
00259     // It is assumed we do not know the existing slave address so the broadcast address is used.
00260     // This will throw a r/w error so errors will be ignored.
00261     _addr = MLX90614_BROADCASTADDR;
00262
00263     // Reload program copy with the existing slave address.
00264     _addr = lowByte(readEEProm(MLX90614_ADDR));
00265
00266     return _addr;
00267 }
```

References _addr, _rwError, MLX90614_ADDR, MLX90614_BROADCASTADDR, and readEEProm().

Referenced by MLX90614().

Here is the call graph for this function:



Here is the caller graph for this function:

5.2.4.5 getCRC8() uint8_t MLX90614::getCRC8 (void) [inline], [private]

8 bit CRC getter

Definition at line 154 of file MLX90614.h.

Referenced by MLX90614().

Here is the caller graph for this function:

5.2.4.6 getEmissivity() float MLX90614::getEmissivity (void)

Get the emissivity of the object.

Emissivity getter

Remarks

The emissivity is stored as a 16 bit integer defined by the following:
`emissivity = dec2hex[round(65535 x emiss)]`

Returns

Physical emissivity value in range 0.1 ...1.0

Definition at line 124 of file MLX90614.cpp.

```
00124                                   {
00125
00126     _rwError = 0;
00127     uint16_t emiss = readEEProm(MLX90614_EMISS);
00128     if(_rwError) return (float)1.0;
00129     return (float)emiss / 65535.0;
00130 }
```

References _rwError, MLX90614_EMISS, and readEEProm().

Here is the call graph for this function:

5.2.4.7 getFIRcoeff() uint8_t MLX90614::getFIRcoeff (void)

Get the coefficients of the FIR digital filter.

IIR coefficient getter

Remarks

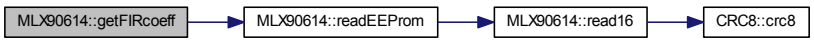
The FIR digital filter coefficient N is bits 10:8 of ConfigRegister1
The value of N is set as follows: `N = 2 ^ (csb + 3)`
The manufacturer does not recommend `N < 128`

Definition at line 212 of file MLX90614.cpp.

```
00212                                   {
00213
00214     _rwError = 0;
00215
00216     // Get the current value of ConfigRegister1 bits 10:8
00217     uint8_t fir = (readEEProm(MLX90614_CONFIG) » 8) & 7;
00218
00219     if(_rwError) return 7;
00220     return fir;
00221 }
```

References _rwError, MLX90614_CONFIG, and readEEProm().

Here is the call graph for this function:



5.2.4.8 getIIRcoeff() uint8_t MLX90614::getIIRcoeff (void)

Get the coefficients of the IIR digital filter.

IIR coefficient getter

Remarks

The IIR digital filter coefficients are set by the LS 3 bits of ConfigRegister1

Returns

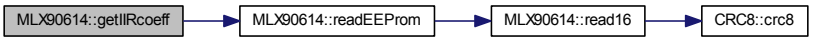
Filter coefficient table index. Range 0...7

Definition at line 170 of file MLX90614.cpp.

```
00170                                   {
00171
00172     _rwError = 0;
00173
00174     // Get the current value of ConfigRegister1 bits 2:0
00175     uint8_t iir = readEEProm(MLX90614_CONFIG) & 7;
00176
00177     if(_rwError) return 4;
00178     return iir;
00179 }
```

References _rwError, MLX90614_CONFIG, and readEEProm().

Here is the call graph for this function:



5.2.4.9 getPEC() uint8_t MLX90614::getPEC (void) [inline], [private]

PEC getter

Definition at line 155 of file MLX90614.h.

Referenced by MLX90614().

Here is the caller graph for this function:

5.2.4.10 getRwError() uint8_t MLX90614::getRwError (void) [inline], [private]

R/W error flags getter

Definition at line 153 of file MLX90614.h.

Referenced by MLX90614().

Here is the caller graph for this function:

5.2.4.11 isReady() boolean MLX90614::isReady (void) [inline]

Definition at line 109 of file MLX90614.h.

```
00126 {MLX90614_TK,                          /**< degrees Kelvin */
```

5.2.4.12 read16() uint16_t MLX90614::read16 (uint8_t *cmd*) [private]

Return a 16 bit value read from RAM or EEPROM.

Parameters

in	*cmd*	Command to send (register to read from).

Returns

Value read from memory.

Definition at line 274 of file MLX90614.cpp.

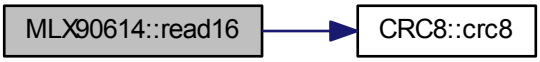
```
00274                                       {
00275     uint16_t val;
00276     CRC8 crc(MLX90614_CRC8POLY);
00277
00278     // Send the slave address then the command and set any error status bits returned by the write.
00279     Wire.beginTransmission(_addr);
00280     Wire.write(cmd);
00281     _rwError |= (1 << Wire.endTransmission(false)) >> 1;
00282
00283     // Experimentally determined delay to prevent read errors (manufacturer's data sheet has
00284     // left something out).
```

```
00285     delayMicroseconds(MLX90614_XDLY);
00286
00287     // Resend slave address then get the 3 returned bytes.
00288     Wire.requestFrom(_addr, (uint8_t)3);
00289
00290     // Data is returned as 2 bytes little endian.
00291     val = Wire.read();
00292     val |= Wire.read() << 8;
00293
00294     // Rread the PEC (CRC-8 of all bytes).
00295     _pec = Wire.read();
00296
00297     // Clear r/w errors if using broadcast address.
00298     if(_addr == MLX90614_BROADCASTADDR) _rwError &= MLX90614_NORWERROR;
00299
00300     // Build our own CRC-8 of all received bytes.
00301     crc.crc8(_addr << 1);
00302     crc.crc8(cmd);
00303     crc.crc8((_addr << 1) + 1);
00304     crc.crc8(lowByte(val));
00305     _crc8 = crc.crc8(highByte(val));
00306
00307     // Set error status bit if CRC mismatch.
00308     if(_crc8 != _pec) _rwError |= MLX90614_RXCRC;
00309
00310     return val;
00311 }
```

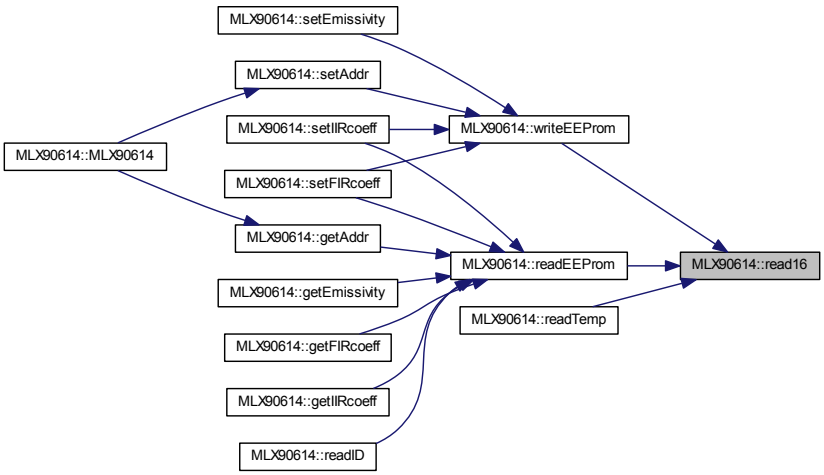
References _addr, _crc8, _pec, _rwError, CRC8::crc8(), MLX90614_BROADCASTADDR, MLX90614_CRC8POLY, MLX90614_NORWERROR, MLX90614_RXCRC, and MLX90614_XDLY.

Referenced by readEEProm(), readTemp(), and writeEEProm().

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4.13 readEEProm() `uint16_t MLX90614::readEEProm ( uint8_t addr )`

Return a 16 bit value read from EEPROM.

Parameters

in	*addr*	Register address to read from.

Returns

Value read from EEPROM.

Examples

MelexisTest.ino.

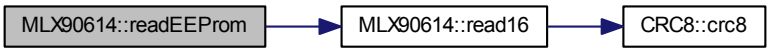
Definition at line 348 of file MLX90614.cpp.

```
00348 {return read16(addr | 0x20);}
```

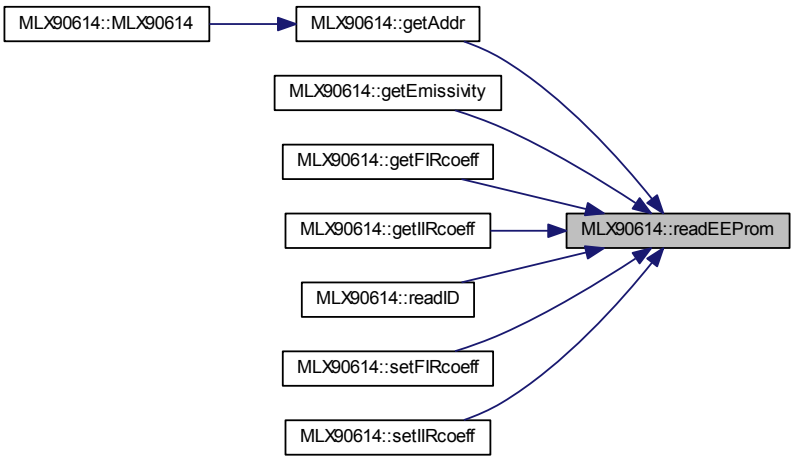
References read16().

Referenced by getAddr(), getEmissivity(), getFIRcoeff(), getIIRcoeff(), readID(), setFIRcoeff(), and setIIRcoeff().

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4.14 readID() uint64_t MLX90614::readID (void)

Retrieve the chip ID bytes.

Chip ID getter

Returns

Chip ID as a 64 bit word.

Examples

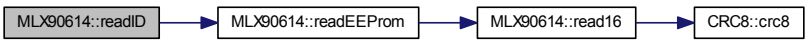
MelexisTest.ino.

Definition at line 399 of file MLX90614.cpp.

```
00399                                {
00400     uint64_t ID = 0;
00401
00402     // If we are lucky the compiler will optimise this.
00403     for(uint8_t i = 0; i < 4; i++) ID = (ID <<= 16) | readEEProm(MLX90614_ID1 + i);
00404     return ID;
00405 }
```

References MLX90614_ID1, and readEEProm().

Here is the call graph for this function:



5.2.4.15 readTemp() double MLX90614::readTemp (tempSrc_t *tsrc* = *MLX90614_SRC01*, tempUnit_t *tunit* = *MLX90614_TC*)

Return a temperature from the specified source in specified units.

Remarks

- Temperature is stored in ram as a 16 bit absolute value to a resolution of 0.02K
- Linearized sensor die temperature is available as Ta (ambient).
- One or two object temperatures are linearized to the range -38.2C...125C

Parameters

in	*tsrc*	Internal temperature source to read, default #1.
in	*tunit*	Temperature units to convert raw data to, default deg Celsius.

Returns

Temperature.

Examples

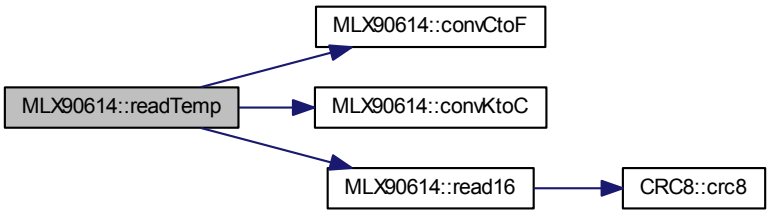
MelexisTest.ino.

Definition at line 88 of file MLX90614.cpp.

```
00088                                                                {
00089     double temp;
00090
00091     _rwError = 0;
00092     switch(tsrc) {
00093         case MLX90614_SRC01 : temp = read16(MLX90614_TOBJ1); break;
00094         case MLX90614_SRC02 : temp = read16(MLX90614_TOBJ2); break;
00095         default : temp = read16(MLX90614_TA);
00096     }
00097     temp *= 0.02;
00098     switch(tunit) {
00099         case MLX90614_TC : return convKtoC(temp);
00100         case MLX90614_TF : return convKtoC(convCtoF(temp));
00101     }
00102     return temp;
00103 }
```

References _rwError, convCtoF(), convKtoC(), MLX90614_SRC01, MLX90614_SRC02, MLX90614_TA, MLX90614_TC, MLX90614_TF, MLX90614_TOBJ1, MLX90614_TOBJ2, and read16().

Here is the call graph for this function:



5.2.4.16 setAddr() void MLX90614::setAddr (uint8_t *addr*) [private]

Set device SMBus address.

SMB bus address setter

Remarks

- Must be only device on the bus.
- Must power cycle the device after changing address.

Parameters

in	*addr*	New device address. Range 1...127

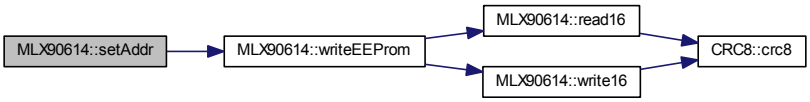
Definition at line 230 of file MLX90614.cpp.

```
00230                                    {
00231
00232     _rwError = 0;
00233
00234     // It is assumed we do not know the existing slave address so the broadcast address is used.
00235     // First ensure the new address is in the legal range (1..127)
00236     if(addr &= 0x7f) {
00237         _addr = MLX90614_BROADCASTADDR;
00238         writeEEProm(MLX90614_ADDR, addr);
00239
00240         // There will always be a r/w error using the broadcast address so we cannot respond
00241         // to r/w errors. We must just assume this worked.
00242         _addr = addr;
00243
00244     } else _rwError |= MLX90614_INVALIDATA;
00245 }
```

References _addr, _rwError, MLX90614_ADDR, MLX90614_BROADCASTADDR, MLX90614_INVALIDATA, and writeEEProm().

Referenced by MLX90614().

Here is the call graph for this function:



Here is the caller graph for this function:

5.2.4.17 setEmissivity() void MLX90614::setEmissivity (float *emiss = 1.0*)

Set the emissivity of the object.

Emissivity setter

Remarks

The emissivity is stored as a 16 bit integer defined by the following:
`emissivity = dec2hex[round(65535 x emiss)]`

Parameters

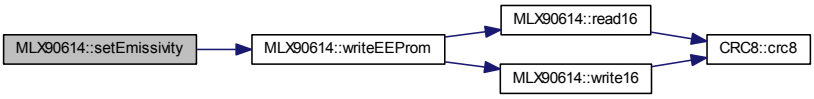
in	*emiss*	Physical emissivity value in range 0.1 ...1.0, default 1.0

Definition at line 111 of file MLX90614.cpp.

```
00111                                           {
00112
00113     _rwError = 0;
00114     uint16_t e = int(emiss * 65535. + 0.5);
00115     if((emiss > 1.0) || (e < 6553)) _rwError |= MLX90614_INVALIDATA;
00116     else writeEEProm(MLX90614_EMISS, e);
00117 }
```

References _rwError, MLX90614_EMISS, MLX90614_INVALIDATA, and writeEEProm().

Here is the call graph for this function:



5.2.4.18 setFIRcoeff() `void MLX90614::setFIRcoeff (`

`uint8_t csb = 7 )`

Set the coefficients of the FIR digital filter.

IIR coefficient setter

Remarks

The FIR digital filter coefficient N is bits 10:8 of ConfigRegister1
The value of N is set as follows: `N = 2 ^ (csb + 3)`
The manufacturer does not recommend `N < 128`

Parameters

in	*csb*	See page 12 of datasheet. Range 0...7, default = 7 (N = 1024)

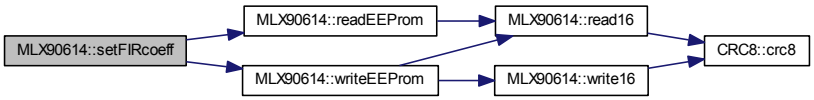
Definition at line 188 of file MLX90614.cpp.

```
00188                                           {
00189
00190     _rwError = 0;
00191
00192     // Ensure legal range by clearing all but the LS 3 bits.
00193     csb &= 7;
00194
00195     // Get the current value of ConfigRegister1
00196     uint16_t reg = readEEProm(MLX90614_CONFIG);
00197
00198     // Clear bits 10:8, mask in the new value, then write it back.
00199     if(!_rwError) {
00200         reg &= 0xf8ff;
```

```
00201         reg |= (uint16_t)csb « 8;
00202         writeEEProm(MLX90614_CONFIG, reg);
00203     }
00204 }
```

References _rwError, MLX90614_CONFIG, readEEProm(), and writeEEProm().

Here is the call graph for this function:



5.2.4.19 setIIRcoeff() void MLX90614::setIIRcoeff (uint8_t *csb* = *4*)

Set the coefficients of the IIR digital filter.

IIR coefficient setter

Remarks

The IIR digital filter coefficients are set by the LS 3 bits of ConfigRegister1
The value of the coefficients is set as follows:

```
csb = 0   a1 = 0.5    a2 = 0.5
      1        0.25       0.75
      2        0.167      0.833
      3        0.125      0.875
      4        1          0 (IIR bypassed)
      5        0.8        0.2
      6        0.67       0.33
      7        0.57       0.43
```

Parameters

in	*csb*	See page 12 of datasheet. Range 0...7, default = 4 (IIR bypassed)

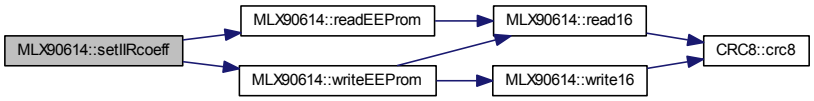
Definition at line 147 of file MLX90614.cpp.

```
00147                                       {
00148
00149     _rwError = 0;
00150
00151     // Ensure legal range by clearing all but the LS 3 bits.
00152     csb &= 7;
00153
00154     // Get the current value of ConfigRegister1
00155     uint16_t reg = readEEProm(MLX90614_CONFIG);
00156
00157     // Clear bits 2:0, mask in the new value, then write it back.
00158     if(!_rwError) {
00159         reg &= 0xfff8;
00160         reg |= (uint16_t)csb;
00161         writeEEProm(MLX90614_CONFIG, reg);
```

```
00162     }
00163 }
```

References _rwError, MLX90614_CONFIG, readEEProm(), and writeEEProm().

Here is the call graph for this function:



5.2.4.20 write16() `void MLX90614::write16 ( uint8_t cmd, uint16_t data ) [private]`

Write a 16 bit value to memory.

Parameters

in	*cmd*	Command to send (register to write to).
in	*data*	Value to write.

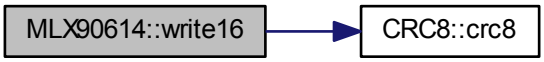
Definition at line 318 of file MLX90614.cpp.

```
00318                                                 {
00319     CRC8 crc(MLX90614_CRC8POLY);
00320
00321     // Build the CRC-8 of all bytes to be sent.
00322     crc.crc8(_addr « 1);
00323     crc.crc8(cmd);
00324     crc.crc8(lowByte(data));
00325     _crc8 = crc.crc8(highByte(data));
00326
00327     // Send the slave address then the command.
00328     Wire.beginTransmission(_addr);
00329     Wire.write(cmd);
00330
00331     // Write the data low byte first.
00332     Wire.write(lowByte(data));
00333     Wire.write(highByte(data));
00334
00335     // Then write the crc and set the r/w error status bits.
00336     Wire.write(_pec = _crc8);
00337     _rwError |= (1 « Wire.endTransmission(true)) » 1;
00338
00339     // Clear r/w errors if using broadcast address.
00340     if(_addr == MLX90614_BROADCASTADDR) _rwError &= MLX90614_NORWERROR;
00341 }
```

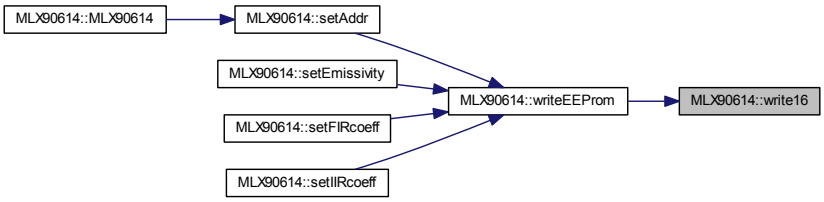
References _addr, _crc8, _pec, _rwError, CRC8::crc8(), MLX90614_BROADCASTADDR, MLX90614_CRC8POLY, and MLX90614_NORWERROR.

Referenced by writeEEProm().

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.4.21 writeEEProm() `void MLX90614::writeEEProm (`

```
uint8_t reg,
uint16_t data )
```

Write a 16 bit value to EEPROM after first clearing the memory.

Remarks

- Erase and write time 5ms per manufacturer specification
- Manufacturer does not specify max or min erase/write times

Parameters

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

Examples

MelexisTest.ino.

Definition at line 358 of file MLX90614.cpp.

```
00358                                                   {
00359     uint16_t val;
00360     reg |= 0x20;
```

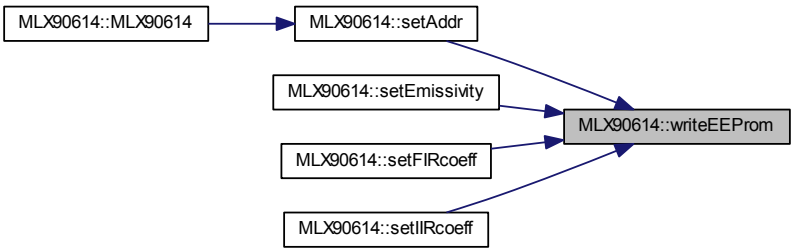
```
00361
00362     // Read current value, compare to the new value, and do nothing on a match or if there are
00363     // read errors set the error status flag only.
00364     val = read16(reg);
00365     if((val != data) && !_rwError) {
00366
00367         // On any R/W errors it is assumed the memory is corrupted.
00368         // Clear the memory and wait Terase (per manufacturer's documentation).
00369         write16(reg, 0);
00370         delay(5);
00371         if(_rwError) _rwError |= MLX90614_EECORRUPT;
00372
00373         // Write the data and wait Twrite (per manufacturer's documentation)
00374         // and set the r/w error status bits.
00375         write16(reg, data);
00376         delay(5);
00377         if(_rwError) _rwError |= MLX90614_EECORRUPT;
00378     }
00379 }
```

References _rwError, MLX90614_EECORRUPT, read16(), and write16().

Referenced by setAddr(), setEmissivity(), setFIRcoeff(), and setIIRcoeff().

Here is the call graph for this function:

Here is the caller graph for this function:



5.2.5 Member Data Documentation

5.2.5.1 _addr `uint8_t MLX90614::_addr [private]`

Slave address

Definition at line 145 of file MLX90614.h.

Referenced by getAddr(), MLX90614(), read16(), setAddr(), and write16().

5.2.5.2 _crc8 uint8_t MLX90614::_crc8 [private]

8 bit CRC

Definition at line 147 of file MLX90614.h.

Referenced by begin(), read16(), and write16().

5.2.5.3 _pec uint8_t MLX90614::_pec [private]

PEC

Definition at line 148 of file MLX90614.h.

Referenced by begin(), read16(), and write16().

5.2.5.4 _ready boolean MLX90614::_ready [private]

Definition at line 144 of file MLX90614.h.

Referenced by begin(), and MLX90614().

5.2.5.5 _rwError uint8_t MLX90614::_rwError [private]

R/W error flags

Definition at line 146 of file MLX90614.h.

Referenced by begin(), getAddr(), getEmissivity(), getFIRcoeff(), getIIRcoeff(), read16(), readTemp(), setAddr(), setEmissivity(), setFIRcoeff(), setIIRcoeff(), write16(), and writeEEProm().

5.2.5.6 busAddr Property<uint8_t, MLX90614> MLX90614::busAddr

SMBus address property

Definition at line 123 of file MLX90614.h.

Referenced by MLX90614().

5.2.5.7 crc8 `Property<uint8_t, MLX90614> MLX90614::crc8`

8 bit CRC property

Examples

MelexisTest.ino.

Definition at line 125 of file MLX90614.h.

Referenced by MLX90614().

5.2.5.8 pec `Property<uint8_t, MLX90614> MLX90614::pec`

PEC property

Examples

MelexisTest.ino.

Definition at line 126 of file MLX90614.h.

Referenced by MLX90614().

5.2.5.9 rwError `Property<uint8_t, MLX90614> MLX90614::rwError`

R/W error flags property

Examples

MelexisTest.ino.

Definition at line 124 of file MLX90614.h.

Referenced by MLX90614().

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

- MLX90614.h
- MLX90614.cpp

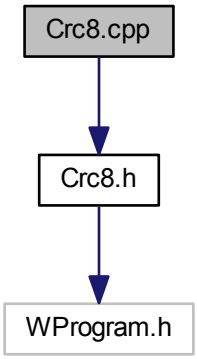
6 File Documentation

6.1 Crc8.cpp File Reference

8 bit CRC helper/utility class - CPP Source file.

```
#include "Crc8.h"
```

Include dependency graph for Crc8.cpp:



6.1.1 Detailed Description

8 bit CRC helper/utility class - CPP Source file.

Author

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

Version

1.0

Date

2014-2017

Copyright

License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 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 Lesser General Public License for more details at http://www.gnu.org/copyleft/gpl.html

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Definition in file Crc8.cpp.

6.2 Crc8.cpp

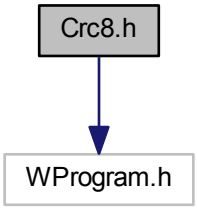
```
00001 /*************************************************************************************************//**
00002  *  \brief      8 bit CRC helper/utility class - CPP Source file.
00003  *  \file       CRC8.CPP
00004  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00005  *  \version    1.0
00006  *  \date       2014-2017
00007  *  \copyright  Copyright (c) 2017 John Fitter.  All right reserved.
00008  *
00009  *  \par        License
00010  *              This program is free software; you can redistribute it and/or modify it under
00011  *              the terms of the GNU Lesser General Public License as published by the Free
00012  *              Software Foundation; either version 2.1 of the License, or (at your option)
00013  *              any later version.
00014  *  \par
00015  *              This Program is distributed in the hope that it will be useful, but WITHOUT ANY
00016  *              WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
00017  *              PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details
00018  *              at http://www.gnu.org/copyleft/gpl.html
00019  *  \par
00020  *              You should have received a copy of the GNU Lesser General Public License along
00021  *              with this library; if not, write to the Free Software Foundation, Inc.,
00022  *              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
00023  *
00024  *//*************************************************************************************************/
00025
00026 #include "Crc8.h"
00027
00028 /*************************************************************************************************/
00029 /*  CRC8 helper class functions.                                                                 */
00030 /*************************************************************************************************/
00031
00032 /**
00033  *  \brief            CRC8 class constructor.
00034  *  \param [in] poly  8 bit CRC polynomial to use.
00035  */
00036 CRC8::CRC8(uint8_t poly) {crc8Start(poly);}
00037
00038 /**
00039  *  \brief            Return the current value of the CRC.
00040  *  \return           8 bit CRC current value.
00041  */
00042 uint8_t CRC8::crc8(void) {return _crc;}
00043
00044 /**
00045  *  \brief            Update the current value of the CRC.
00046  *  \param [in] data  New 8 bit data to be added to the CRC.
00047  *  \return           8 bit CRC current value.
00048  */
00049 uint8_t CRC8::crc8(uint8_t data) {
00050     uint8_t i = 8;
00051
00052     _crc ^= data;
00053     while(i--) _crc = _crc & 0x80 ? (_crc << 1) ^ _poly : _crc << 1;
00054     return _crc;
```

```
00055 }
00056
00057 /**
00058  *  \brief            Initialize the CRC8 object.
00059  *  \param [in] poly  8 bit CRC polynomial to use.
00060  */
00061 void CRC8::crc8Start(uint8_t poly) {
00062     _poly = poly;
00063     _crc = 0;
00064 }
00065
```

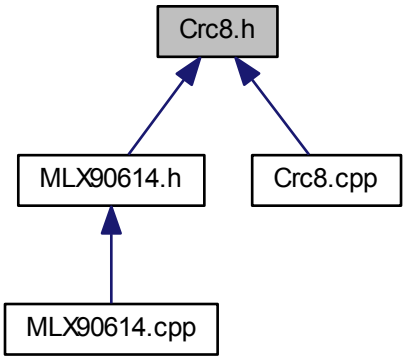
6.3 Crc8.h File Reference

8 bit CRC helper/utility class - CPP Header file.

`#include "WProgram.h"`
Include dependency graph for Crc8.h:



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



Classes

- class CRC8

Macros

- #define CRC8_DEFAULTPOLY 7

6.3.1 Detailed Description

8 bit CRC helper/utility class - CPP Header file.

Author

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

Version

1.0

Date

2014-2017

Copyright

License

Definition in file Crc8.h.

6.3.2 Macro Definition Documentation

6.3.2.1 CRC8_DEFAULTPOLY `#define CRC8_DEFAULTPOLY 7`

Default CRC polynomial = X8+X2+X1+1

Definition at line 35 of file Crc8.h.

6.4 Crc8.h

```
00001 #ifndef _CRC8_H_
00002 #define _CRC8_H_
00003
00004 /*************************************************************************************************//**
00005  *  \brief      8 bit CRC helper/utility class - CPP Header file.
00006  *  \file       CRC8.H
00007  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00008  *  \version    1.0
00009  *  \date       2014-2017
00010  *  \copyright  Copyright (c) 2017 John Fitter.  All right reserved.
00011  *
00012  *  \par        License
00013  *              This program is free software; you can redistribute it and/or modify it under
00014  *              the terms of the GNU Lesser General Public License as published by the Free
00015  *              Software Foundation; either version 2.1 of the License, or (at your option)
00016  *              any later version.
00017  *  \par
00018  *              This Program is distributed in the hope that it will be useful, but WITHOUT ANY
00019  *              WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
00020  *              PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details
00021  *              at http://www.gnu.org/copyleft/gpl.html
00022  *  \par
00023  *              You should have received a copy of the GNU Lesser General Public License along
00024  *              with this library; if not, write to the Free Software Foundation, Inc.,
00025  *              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
00026  *
00027  *//***********************************************************************************************/
00028
00029 #if (ARDUINO >= 100)
00030     #include "Arduino.h"
00031 #else
00032     #include "WProgram.h"
00033 #endif
00034
00035 #define CRC8_DEFAULTPOLY  7  /**< Default CRC polynomial = X8+X2+X1+1 */
00036
00037 class CRC8 {
00038 public:
00039     CRC8(uint8_t polynomial = CRC8_DEFAULTPOLY);
00040     uint8_t  crc8(void);
00041     uint8_t  crc8(uint8_t data);
00042     void     crc8Start(uint8_t poly);
00043 private:
00044     uint8_t  _crc;
00045     uint8_t  _poly;
00046 };
00047
00048 #endif /* _CRC8_H_ */
```

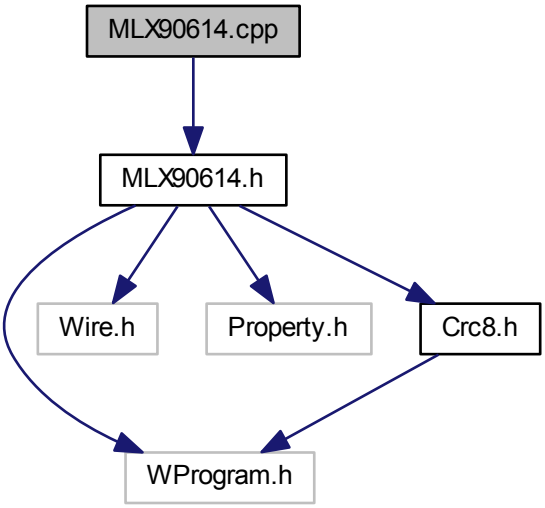
6.5 LICENSE.md File Reference

6.6 MLX90614.cpp File Reference

Melexis MLX90614 Family Device Driver Library - CPP Source file.

```
#include "MLX90614.h"
```

Include dependency graph for MLX90614.cpp:



6.6.1 Detailed Description

Melexis MLX90614 Family Device Driver Library - CPP Source file.

Details

Based on the Melexis MLX90614 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

2014-2017

Copyright

Copyright (c) 2017 John Fitter. All right reserved.

License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 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 Lesser General Public License for more details at http://www.gnu.org/copyleft/gpl.html

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Definition in file MLX90614.cpp.

6.7 MLX90614.cpp

```
00001 /**************************************************************************************************//**
00002  *  \brief      Melexis MLX90614 Family Device Driver Library - CPP Source file
00003  *  \par
00004  *  \par        Details
00005  *              Based on the Melexis MLX90614 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       MLX90614.CPP
00014  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00015  *  \version    1.0
00016  *  \date       2014-2017
00017  *  \copyright  Copyright (c) 2017 John Fitter.  All right reserved.
00018  *
00019  *  \par        License
00020  *              This program is free software; you can redistribute it and/or modify it under
00021  *              the terms of the GNU Lesser General Public License as published by the Free
00022  *              Software Foundation; either version 2.1 of the License, or (at your option)
00023  *              any later version.
00024  *  \par
00025  *              This Program is distributed in the hope that it will be useful, but WITHOUT ANY
00026  *              WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
00027  *              PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details
00028  *              at http://www.gnu.org/copyleft/gpl.html
00029  *  \par
00030  *              You should have received a copy of the GNU Lesser General Public License along
00031  *              with this library; if not, write to the Free Software Foundation, Inc.,
00032  *              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
00033  *
```

```
00034  *//*****************************************************************************************************/
00035
00036 #include "MLX90614.h"
00037
00038 /*****************************************************************************************************/
00039 /*  MLX90614 Device class functions.                                                                   */
00040 /*****************************************************************************************************/
00041 /**
00042  *  \example{lineno}     MelexisTest.ino
00043  *                       An example of how to use the MLX90614 class.
00044  */
00045
00046 /**
00047  *  \brief               MLX90614 Device class constructor.
00048  *  \param [in] i2caddr  Device address (default: published value).
00049  */
00050 MLX90614::MLX90614(uint8_t i2caddr) {
00051
00052     busAddr.Set_Class(this);
00053     busAddr.Set_Get(&MLX90614::getAddr);
00054     busAddr.Set_Set(&MLX90614::setAddr);
00055
00056     rwError.Set_Class(this);
00057     rwError.Set_Get(&MLX90614::getRwError);
00058
00059     pec.Set_Class(this);
00060     pec.Set_Get(&MLX90614::getPEC);
00061
00062     crc8.Set_Class(this);
00063     crc8.Set_Get(&MLX90614::getCRC8);
00064
00065     _addr = i2caddr;
00066     _ready = false;
00067 }
00068
00069 /**
00070  *  \brief  Initialize the device and the i2c interface.
00071  */
00072 boolean MLX90614::begin(void) {
00073
00074     _rwError = _pec = _crc8 = 0;
00075     return _ready = true;
00076 }
00077
00078 /**
00079  *  \brief             Return a temperature from the specified source in specified units.
00080  *  \remarks
00081  *  \li                Temperature is stored in ram as a 16 bit absolute value to a resolution of
      0.02K
00082  *  \li                Linearized sensor die temperature is available as Ta (ambient).
00083  *  \li                One or two object temperatures are linearized to the range -38.2C...125C
00084  *  \param [in] tsrc   Internal temperature source to read, default #1.
00085  *  \param [in] tunit  Temperature units to convert raw data to, default deg Celsius.
00086  *  \return            Temperature.
00087  */
00088 double MLX90614::readTemp(tempSrc_t tsrc, tempUnit_t tunit) {
00089     double temp;
00090
00091     _rwError = 0;
00092     switch(tsrc) {
00093         case MLX90614_SRC01 : temp = read16(MLX90614_TOBJ1); break;
00094         case MLX90614_SRC02 : temp = read16(MLX90614_TOBJ2); break;
00095         default : temp = read16(MLX90614_TA);
00096     }
00097     temp *= 0.02;
00098     switch(tunit) {
00099         case MLX90614_TC : return convKtoC(temp);
00100         case MLX90614_TF : return convKtoC(convCtoF(temp));
00101     }
00102     return temp;
00103 }
00104
00105 /**
00106  *  \brief             Set the emissivity of the object.
00107  *  \remarks           The emissivity is stored as a 16 bit integer defined by the following:
00108  *  \n<tt>             emissivity = dec2hex[round(65535 x emiss)]</tt>
00109  *  \param [in] emiss  Physical emissivity value in range 0.1 ...1.0, default 1.0
00110  */
00111 void MLX90614::setEmissivity(float emiss) {
00112
00113     _rwError = 0;
00114     uint16_t e = int(emiss * 65535. + 0.5);
00115     if((emiss > 1.0) || (e < 6553)) _rwError |= MLX90614_INVALIDATA;
00116     else writeEEProm(MLX90614_EMISS, e);
00117 }
00118 /**
00119  *  \brief             Get the emissivity of the object.
```

```
00120  *  \remarks           The emissivity is stored as a 16 bit integer defined by the following:
00121  *  \n<tt>             emissivity = dec2hex[round(65535 x emiss)]</tt>
00122  *  \return            Physical emissivity value in range 0.1 ...1.0
00123  */
00124 float MLX90614::getEmissivity(void) {
00125
00126     _rwError = 0;
00127     uint16_t emiss = readEEProm(MLX90614_EMISS);
00128     if(_rwError) return (float)1.0;
00129     return (float)emiss / 65535.0;
00130 }
00131
00132 /**
00133  *  \brief            Set the coefficients of the IIR digital filter.
00134  *  \remarks          The IIR digital filter coefficients are set by the LS 3 bits of ConfigRegister1
00135  *  \n                The value of the coefficients is set as follows:
00136  *  \n <tt> \verbatim
00137  csb = 0   a1 = 0.5    a2 = 0.5
00138        1        0.25        0.75
00139        2        0.167       0.833
00140        3        0.125       0.875
00141        4        1           0 (IIR bypassed)
00142        5        0.8         0.2
00143        6        0.67        0.33
00144        7        0.57        0.43 \endverbatim </tt>
00145  *  \param [in] csb   See page 12 of datasheet. Range 0...7, default = 4 (IIR bypassed)
00146  */
00147 void MLX90614::setIIRcoeff(uint8_t csb) {
00148
00149     _rwError = 0;
00150
00151     // Ensure legal range by clearing all but the LS 3 bits.
00152     csb &= 7;
00153
00154     // Get the current value of ConfigRegister1
00155     uint16_t reg = readEEProm(MLX90614_CONFIG);
00156
00157     // Clear bits 2:0, mask in the new value, then write it back.
00158     if(!_rwError) {
00159         reg &= 0xfff8;
00160         reg |= (uint16_t)csb;
00161         writeEEProm(MLX90614_CONFIG, reg);
00162     }
00163 }
00164
00165 /**
00166  *  \brief            Get the coefficients of the IIR digital filter.
00167  *  \remarks          The IIR digital filter coefficients are set by the LS 3 bits of ConfigRegister1
00168  *  \return           Filter coefficient table index. Range 0...7
00169  */
00170 uint8_t MLX90614::getIIRcoeff(void) {
00171
00172     _rwError = 0;
00173
00174     // Get the current value of ConfigRegister1 bits 2:0
00175     uint8_t iir = readEEProm(MLX90614_CONFIG) & 7;
00176
00177     if(_rwError) return 4;
00178     return iir;
00179 }
00180
00181 /**
00182  *  \brief            Set the coefficients of the FIR digital filter.
00183  *  \remarks          The FIR digital filter coefficient N is bits 10:8 of ConfigRegister1
00184  *  \n                The value of N is set as follows:  <tt>N = 2 ^ (csb + 3)</tt>
00185     \n                The manufacturer does not recommend <tt>N < 128</tt>
00186  *  \param [in] csb   See page 12 of datasheet. Range 0...7, default = 7 (N = 1024)
00187  */
00188 void MLX90614::setFIRcoeff(uint8_t csb) {
00189
00190     _rwError = 0;
00191
00192     // Ensure legal range by clearing all but the LS 3 bits.
00193     csb &= 7;
00194
00195     // Get the current value of ConfigRegister1
00196     uint16_t reg = readEEProm(MLX90614_CONFIG);
00197
00198     // Clear bits 10:8, mask in the new value, then write it back.
00199     if(!_rwError) {
00200         reg &= 0xf8ff;
00201         reg |= (uint16_t)csb « 8;
00202         writeEEProm(MLX90614_CONFIG, reg);
00203     }
00204 }
00205
00206 /**
```

```
00207  *  \brief            Get the coefficients of the FIR digital filter.
00208  *  \remarks          The FIR digital filter coefficient N is bits 10:8 of ConfigRegister1
00209  *  \n                The value of N is set as follows:  <tt>N = 2 ^ (csb + 3)</tt>
00210     \n                The manufacturer does not recommend <tt>N < 128</tt>
00211  */
00212 uint8_t MLX90614::getFIRcoeff(void) {
00213
00214     _rwError = 0;
00215
00216     // Get the current value of ConfigRegister1 bits 10:8
00217     uint8_t fir = (readEEProm(MLX90614_CONFIG) >> 8) & 7;
00218
00219     if(_rwError) return 7;
00220     return fir;
00221 }
00222
00223 /**
00224  *  \brief            Set device SMBus address.
00225  *  \remarks
00226  *  \li               Must be only device on the bus.
00227  *  \li               Must power cycle the device after changing address.
00228  *  \param [in] addr  New device address. Range 1...127
00229  */
00230 void MLX90614::setAddr(uint8_t addr) {
00231
00232     _rwError = 0;
00233
00234     // It is assumed we do not know the existing slave address so the broadcast address is used.
00235     // First ensure the new address is in the legal range (1..127)
00236     if(addr &= 0x7f) {
00237         _addr = MLX90614_BROADCASTADDR;
00238         writeEEProm(MLX90614_ADDR, addr);
00239
00240         // There will always be a r/w error using the broadcast address so we cannot respond
00241         // to r/w errors. We must just assume this worked.
00242         _addr = addr;
00243
00244     } else _rwError |= MLX90614_INVALIDATA;
00245 }
00246
00247 /**
00248  *  \brief            Return the device SMBus address.
00249  *  \remarks
00250  *  \li               Must be only device on the bus.
00251  *  \li               Sets the library to use the new found address.
00252  *  \return           Device address.
00253  */
00254 uint8_t MLX90614::getAddr(void) {
00255     uint8_t tempAddr = _addr;
00256
00257     _rwError = 0;
00258
00259     // It is assumed we do not know the existing slave address so the broadcast address is used.
00260     // This will throw a r/w error so errors will be ignored.
00261     _addr = MLX90614_BROADCASTADDR;
00262
00263     // Reload program copy with the existing slave address.
00264     _addr = lowByte(readEEProm(MLX90614_ADDR));
00265
00266     return _addr;
00267 }
00268
00269 /**
00270  *  \brief            Return a 16 bit value read from RAM or EEPROM.
00271  *  \param [in] cmd   Command to send (register to read from).
00272  *  \return           Value read from memory.
00273  */
00274 uint16_t MLX90614::read16(uint8_t cmd) {
00275     uint16_t val;
00276     CRC8 crc(MLX90614_CRC8POLY);
00277
00278     // Send the slave address then the command and set any error status bits returned by the write.
00279     Wire.beginTransmission(_addr);
00280     Wire.write(cmd);
00281     _rwError |= (1 << Wire.endTransmission(false)) >> 1;
00282
00283     // Experimentally determined delay to prevent read errors (manufacturer's data sheet has
00284     // left something out).
00285     delayMicroseconds(MLX90614_XDLY);
00286
00287     // Resend slave address then get the 3 returned bytes.
00288     Wire.requestFrom(_addr, (uint8_t)3);
00289
00290     // Data is returned as 2 bytes little endian.
00291     val = Wire.read();
00292     val |= Wire.read() << 8;
00293
```

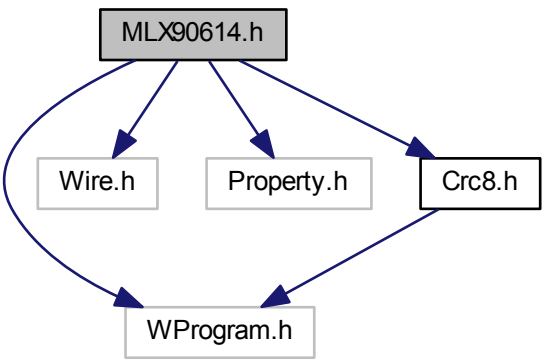
```
00294     // Rread the PEC (CRC-8 of all bytes).
00295     _pec = Wire.read();
00296 
00297     // Clear r/w errors if using broadcast address.
00298     if(_addr == MLX90614_BROADCASTADDR) _rwError &= MLX90614_NORWERROR;
00299 
00300     // Build our own CRC-8 of all received bytes.
00301     crc.crc8(_addr « 1);
00302     crc.crc8(cmd);
00303     crc.crc8((_addr « 1) + 1);
00304     crc.crc8(lowByte(val));
00305     _crc8 = crc.crc8(highByte(val));
00306 
00307     // Set error status bit if CRC mismatch.
00308     if(_crc8 != _pec) _rwError |= MLX90614_RXCRC;
00309 
00310     return val;
00311 }
00312 
00313 /**
00314  *  \brief            Write a 16 bit value to memory.
00315  *  \param [in] cmd   Command to send (register to write to).
00316  *  \param [in] data  Value to write.
00317  */
00318 void MLX90614::write16(uint8_t cmd, uint16_t data) {
00319     CRC8 crc(MLX90614_CRC8POLY);
00320 
00321     // Build the CRC-8 of all bytes to be sent.
00322     crc.crc8(_addr « 1);
00323     crc.crc8(cmd);
00324     crc.crc8(lowByte(data));
00325     _crc8 = crc.crc8(highByte(data));
00326 
00327     // Send the slave address then the command.
00328     Wire.beginTransmission(_addr);
00329     Wire.write(cmd);
00330 
00331     // Write the data low byte first.
00332     Wire.write(lowByte(data));
00333     Wire.write(highByte(data));
00334 
00335     // Then write the crc and set the r/w error status bits.
00336     Wire.write(_pec = _crc8);
00337     _rwError |= (1 « Wire.endTransmission(true)) » 1;
00338 
00339     // Clear r/w errors if using broadcast address.
00340     if(_addr == MLX90614_BROADCASTADDR) _rwError &= MLX90614_NORWERROR;
00341 }
00342 
00343 /**
00344  *  \brief            Return a 16 bit value read from EEPROM.
00345  *  \param [in] addr  Register address to read from.
00346  *  \return           Value read from EEPROM.
00347  */
00348 uint16_t MLX90614::readEEProm(uint8_t addr) {return read16(addr | 0x20);}
00349 
00350 /**
00351  *  \brief            Write a 16 bit value to EEPROM after first clearing the memory.
00352  *  \remarks
00353  *  \li               Erase and write time 5ms per manufacturer specification
00354  *  \li               Manufacturer does not specify max or min erase/write times
00355  *  \param [in] reg   Address to write to.
00356  *  \param [in] data  Value to write.
00357  */
00358 void MLX90614::writeEEProm(uint8_t reg, uint16_t data) {
00359     uint16_t val;
00360     reg |= 0x20;
00361 
00362     // Read current value, compare to the new value, and do nothing on a match or if there are
00363     // read errors set the error status flag only.
00364     val = read16(reg);
00365     if((val != data) && !_rwError) {
00366 
00367         // On any R/W errors it is assumed the memory is corrupted.
00368         // Clear the memory and wait Terase (per manufacturer's documentation).
00369         write16(reg, 0);
00370         delay(5);
00371         if(_rwError) _rwError |= MLX90614_EECORRUPT;
00372 
00373         // Write the data and wait Twrite (per manufacturer's documentation)
00374         // and set the r/w error status bits.
00375         write16(reg, data);
00376         delay(5);
00377         if(_rwError) _rwError |= MLX90614_EECORRUPT;
00378     }
00379 }
00380 
```

```
00381 /**
00382  *  \brief            Convert temperature in degrees K to degrees C.
00383  *  \param [in] degK  Temperature in degrees Kelvin.
00384  *  \return           Temperature in degrees Centigrade.
00385  */
00386 double MLX90614::convKtoC(double degK) {return degK - 273.15;}
00387
00388 /**
00389  *  \brief            Convert temperature in degrees C to degrees F.
00390  *  \param [in] degC  Temperature in degrees Centigrade.
00391  *  \return           Temperature in degrees Fahrenheit.
00392  */
00393 double MLX90614::convCtoF(double degC) {return (degC * 1.8) + 32.0;}
00394
00395 /**
00396  *  \brief            Retrieve the chip ID bytes.
00397  *  \return           Chip ID as a 64 bit word.
00398  */
00399 uint64_t MLX90614::readID(void) {
00400     uint64_t ID = 0;
00401
00402     // If we are lucky the compiler will optimise this.
00403     for(uint8_t i = 0; i < 4; i++) ID = (ID «= 16) | readEEProm(MLX90614_ID1 + i);
00404     return ID;
00405 }
00406
```

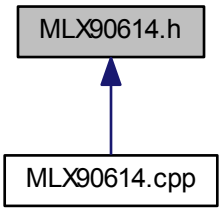
6.8 MLX90614.h File Reference

```
#include "WProgram.h"
#include <Wire.h>
#include "Property.h"
#include "Crc8.h"
```

Include dependency graph for MLX90614.h:



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



Classes

- class MLX90614

Macros

- #define MLX90614_I2CDEFAULTADDR 0x5A
- #define MLX90614_BROADCASTADDR 0
- #define MLX90614_CRC8POLY 7
- #define MLX90614_XDLY 25
- #define MLX90614_RAWIR1 0x04
- #define MLX90614_RAWIR2 0x05
- #define MLX90614_TA 0x06
- #define MLX90614_TOBJ1 0x07
- #define MLX90614_TOBJ2 0x08
- #define MLX90614_TOMAX 0x00
- #define MLX90614_TOMIN 0x01
- #define MLX90614_PWMCTRL 0x02
- #define MLX90614_TARANGE 0x03
- #define MLX90614_EMISS 0x04
- #define MLX90614_CONFIG 0x05
- #define MLX90614_ADDR 0x0E
- #define MLX90614_ID1 0x1C
- #define MLX90614_ID2 0x1D
- #define MLX90614_ID3 0x1E
- #define MLX90614_ID4 0x1F
- #define MLX90614_RFLAGCMD 0xF0
- #define MLX90614_EEBUSY 0x80
- #define MLX90614_EE_DEAD 0x20
- #define MLX90614_INIT 0x10
- #define MLX90614_NORWERROR 0
- #define MLX90614_DATATOOLONG 1
- #define MLX90614_TXADDRNACK 2
- #define MLX90614_TXDATANACK 4
- #define MLX90614_TXOTHER 8
- #define MLX90614_RXCRC 0x10
- #define MLX90614_INVALIDATA 0x20
- #define MLX90614_EECORRUPT 0x40
- #define MLX90614_RFLGERR 0x80

6.8.1 Macro Definition Documentation

6.8.1.1 MLX90614_ADDR `#define MLX90614_ADDR 0x0E`

EEPROM reg - SMBus address

Examples

MelexisTest.ino.

Definition at line 76 of file MLX90614.h.

6.8.1.2 MLX90614_BROADCASTADDR `#define MLX90614_BROADCASTADDR 0`

Device broadcast slave address

Examples

MelexisTest.ino.

Definition at line 53 of file MLX90614.h.

6.8.1.3 MLX90614_CONFIG `#define MLX90614_CONFIG 0x05`

EEPROM reg - Configuration register

Definition at line 75 of file MLX90614.h.

6.8.1.4 MLX90614_CRC8POLY `#define MLX90614_CRC8POLY 7`

CRC polynomial = X8+X2+X1+1

Definition at line 54 of file MLX90614.h.

6.8.1.5 MLX90614_DATATOOLONG `#define MLX90614_DATATOOLONG 1`

R/W error bitmask - Data is too long

Examples

MelexisTest.ino.

Definition at line 91 of file MLX90614.h.

6.8.1.6 MLX90614_EE_DEAD `#define MLX90614_EE_DEAD 0x20`

R/W flag bitmask - EEProm double error has occurred

Definition at line 86 of file MLX90614.h.

6.8.1.7 MLX90614_EEBUSY `#define MLX90614_EEBUSY 0x80`

Read flags - bitmask. R/W flag bitmask - EEProm is busy (writing/erasing)

Definition at line 85 of file MLX90614.h.

6.8.1.8 MLX90614_EECORRUPT `#define MLX90614_EECORRUPT 0x40`

R/W error bitmask - The EEProm is likely to be corrupted

Examples

MelexisTest.ino.

Definition at line 97 of file MLX90614.h.

6.8.1.9 MLX90614_EMISS `#define MLX90614_EMISS 0x04`

EEPROM reg - Object emissivity register

Definition at line 74 of file MLX90614.h.

6.8.1.10 MLX90614_I2CDEFAULTADDR `#define MLX90614_I2CDEFAULTADDR 0x5A`

Device default slave address

Definition at line 52 of file MLX90614.h.

6.8.1.11 MLX90614_ID1 `#define MLX90614_ID1 0x1C`

EEPROM reg - ID numer (w1)

Definition at line 77 of file MLX90614.h.

6.8.1.12 MLX90614_ID2 `#define MLX90614_ID2 0x1D`

EEPROM reg - ID numer (w2)

Definition at line 78 of file MLX90614.h.

6.8.1.13 MLX90614_ID3 `#define MLX90614_ID3 0x1E`

EEPROM reg - ID numer (w3)

Definition at line 79 of file MLX90614.h.

6.8.1.14 MLX90614_ID4 `#define MLX90614_ID4 0x1F`

EEPROM reg - ID numer (w4)

Definition at line 80 of file MLX90614.h.

6.8.1.15 MLX90614_INIT `#define MLX90614_INIT 0x10`

R/W flag bitmask - POR initialization is still ongoing

Definition at line 87 of file MLX90614.h.

6.8.1.16 MLX90614_INVALIDATA `#define MLX90614_INVALIDATA 0x20`

R/W error bitmask - RX/TX Data fails selection criteria

Examples

MelexisTest.ino.

Definition at line 96 of file MLX90614.h.

6.8.1.17 MLX90614_NORWERROR `#define MLX90614_NORWERROR 0`

R/W Error flags - bitmask. R/W error bitmask - No Errors

Examples

MelexisTest.ino.

Definition at line 90 of file MLX90614.h.

6.8.1.18 MLX90614_PWMCTRL `#define MLX90614_PWMCTRL 0x02`

EEPROM reg - Pulse width modulation output control register

Definition at line 72 of file MLX90614.h.

6.8.1.19 MLX90614_RAWIR1 `#define MLX90614_RAWIR1 0x04`

RAM addresses. RAM reg - Raw temperature, source #1

Definition at line 63 of file MLX90614.h.

6.8.1.20 MLX90614_RAWIR2 `#define MLX90614_RAWIR2 0x05`

RAM reg - Raw temperature, source #2

Definition at line 64 of file MLX90614.h.

6.8.1.21 MLX90614_RFLAGCMD `#define MLX90614_RFLAGCMD 0xF0`

Read R/W Flags register command

Definition at line 82 of file MLX90614.h.

6.8.1.22 MLX90614_RFLGERR `#define MLX90614_RFLGERR 0x80`

R/W error bitmask - R/W flags register access error

Examples

MelexisTest.ino.

Definition at line 98 of file MLX90614.h.

6.8.1.23 MLX90614_RXCRC `#define MLX90614_RXCRC 0x10`

R/W error bitmask - Receiver CRC mismatch

Examples

MelexisTest.ino.

Definition at line 95 of file MLX90614.h.

6.8.1.24 MLX90614_TA `#define MLX90614_TA 0x06`

RAM reg - Linearized temperature, ambient

Definition at line 65 of file MLX90614.h.

6.8.1.25 MLX90614_TARANGE `#define MLX90614_TARANGE 0x03`

EEPROM reg - Customer dependent ambient temperature range

Definition at line 73 of file MLX90614.h.

6.8.1.26 MLX90614_TOBJ1 `#define MLX90614_TOBJ1 0x07`

RAM reg - Linearized temperature, source #1

Definition at line 66 of file MLX90614.h.

6.8.1.27 MLX90614_TOBJ2 `#define MLX90614_TOBJ2 0x08`

RAM reg - Linearized temperature, source #2

Definition at line 67 of file MLX90614.h.

6.8.1.28 MLX90614_TOMAX `#define MLX90614_TOMAX 0x00`

EEPROM addresses. EEPROM reg - Customer dependent object temperature range maximum

Definition at line 70 of file MLX90614.h.

6.8.1.29 MLX90614_TOMIN `#define MLX90614_TOMIN 0x01`

EEPROM reg - Customer dependent object temperature range minimum

Definition at line 71 of file MLX90614.h.

6.8.1.30 MLX90614_TXADDRNACK `#define MLX90614_TXADDRNACK 2`

R/W error bitmask - TX address not acknowledged

Examples

MelexisTest.ino.

Definition at line 92 of file MLX90614.h.

6.8.1.31 MLX90614_TXDATANACK `#define MLX90614_TXDATANACK 4`

R/W error bitmask - TX data not acknowledged

Examples

MelexisTest.ino.

Definition at line 93 of file MLX90614.h.

6.8.1.32 MLX90614_TXOTHER `#define MLX90614_TXOTHER 8`

R/W error bitmask - Unknown error

Examples

MelexisTest.ino.

Definition at line 94 of file MLX90614.h.

6.8.1.33 MLX90614_XDLY `#define MLX90614_XDLY 25`

Experimentally determined delay to prevent read errors after calling Wire.endTransmission() *(possibly due to incompatibility between Wire library and SMBus protocol)*.

Definition at line 55 of file MLX90614.h.

6.9 MLX90614.h

```
00001 #ifndef _MLX90614_H_
00002 #define _MLX90614_H_
00003
00004 /**************************************************************************************************//**
00005  *  \brief      Melexis MLX90614 Family Device Driver Library - CPP Source file
00006  *  \par
00007  *  \par        Details
00008  *              Based on the Melexis MLX90614 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       MLX90614.CPP
00017  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00018  *  \version    1.0
00019  *  \date       2014-2017
00020  *  \copyright  Copyright (c) 2017 John Fitter.  All right reserved.
00021  *
00022  *  \par        License
00023  *              This program is free software; you can redistribute it and/or modify it under
00024  *              the terms of the GNU Lesser General Public License as published by the Free
00025  *              Software Foundation; either version 2.1 of the License, or (at your option)
00026  *              any later version.
00027  *  \par
00028  *              This Program is distributed in the hope that it will be useful, but WITHOUT ANY
00029  *              WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
00030  *              PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details
00031  *              at http://www.gnu.org/copyleft/gpl.html
00032  *  \par
00033  *              You should have received a copy of the GNU Lesser General Public License along
00034  *              with this library; if not, write to the Free Software Foundation, Inc.,
00035  *              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
00036  *
00037  *//**************************************************************************************************/
00038
00039 #if (ARDUINO >= 100)
00040     #include "Arduino.h"
00041 #else
00042     #include "WProgram.h"
00043 #endif
00044 #include <Wire.h>
00045 #include "Property.h"
00046 #include "Crc8.h"
00047
00048 /**************************************************************************************************/
00049 /* Definitions                                                                                    */
00050 /**************************************************************************************************/
00051
00052 #define MLX90614_I2CDEFAULTADDR 0x5A    /**< Device default slave address */
00053 #define MLX90614_BROADCASTADDR  0       /**< Device broadcast slave address */
00054 #define MLX90614_CRC8POLY       7       /**< CRC polynomial = X8+X2+X1+1 */
00055 #define MLX90614_XDLY           25      /**< Experimentally determined delay to prevent read
00056                                              errors after calling Wire.endTransmission()
00057                                              <em>(possibly due to incompatibility between Wire
00058                                              library and SMBus protocol)</em>. */
00059 /** RAM addresses. */
00060 #define MLX90614_RAWIR1         0x04    /**< RAM reg - Raw temperature, source #1 */
00061 #define MLX90614_RAWIR2         0x05    /**< RAM reg - Raw temperature, source #2 */
00062 #define MLX90614_TA             0x06    /**< RAM reg - Linearized temperature, ambient */
00063 #define MLX90614_TOBJ1          0x07    /**< RAM reg - Linearized temperature, source #1 */
00064 #define MLX90614_TOBJ2          0x08    /**< RAM reg - Linearized temperature, source #2 */
00065
00066 /** EEPROM addresses. */
00067 #define MLX90614_TOMAX          0x00    /**< EEPROM reg - Customer dependent object temperature range
       maximum */
00068 #define MLX90614_TOMIN          0x01    /**< EEPROM reg - Customer dependent object temperature range
       minimum */
00069 #define MLX90614_PWMCTRL        0x02    /**< EEPROM reg - Pulse width modulation output control
       register */
00070 #define MLX90614_TARANGE        0x03    /**< EEPROM reg - Customer dependent ambient temperature range
       */
00071 #define MLX90614_EMISS          0x04    /**< EEPROM reg - Object emissivity register */
00072 #define MLX90614_CONFIG         0x05    /**< EEPROM reg - Configuration register */
00073 #define MLX90614_ADDR           0x0E    /**< EEPROM reg - SMBus address */
00074 #define MLX90614_ID1            0x1C    /**< EEPROM reg - ID numer (w1) */
00075 #define MLX90614_ID2            0x1D    /**< EEPROM reg - ID numer (w2) */
00076 #define MLX90614_ID3            0x1E    /**< EEPROM reg - ID numer (w3) */
00077 #define MLX90614_ID4            0x1F    /**< EEPROM reg - ID numer (w4) */
00078
00079 #define MLX90614_RFLAGCMD       0xF0    /**< Read R/W Flags register command */
00080
00081 /** Read flags - bitmask. */
```

```
00082 #define MLX90614_EEBUSY         0x80    /**< R/W flag bitmask - EEProm is busy (writing/erasing) */
00083 #define MLX90614_EE_DEAD        0x20    /**< R/W flag bitmask - EEProm double error has occurred */
00084 #define MLX90614_INIT           0x10    /**< R/W flag bitmask - POR initialization is still ongoing */
00085
00086 /** R/W Error flags - bitmask. */
00087 #define MLX90614_NORWERROR      0       /**< R/W error bitmask - No Errors */
00088 #define MLX90614_DATATOOLONG    1       /**< R/W error bitmask - Data is too long */
00089 #define MLX90614_TXADDRNACK     2       /**< R/W error bitmask - TX address not acknowledged */
00090 #define MLX90614_TXDATANACK     4       /**< R/W error bitmask - TX data not acknowledged */
00091 #define MLX90614_TXOTHER        8       /**< R/W error bitmask - Unknown error */
00092 #define MLX90614_RXCRC          0x10    /**< R/W error bitmask - Receiver CRC mismatch */
00093 #define MLX90614_INVALIDATA     0x20    /**< R/W error bitmask - RX/TX Data fails selection criteria
       */
00094 #define MLX90614_EECORRUPT      0x40    /**< R/W error bitmask - The EEProm is likely to be corrupted
       */
00095 #define MLX90614_RFLGERR        0x80    /**< R/W error bitmask - R/W flags register access error */
00096
00097 /**************************************************************************************************/
00098 /* MLX90614 Device class.                                                                         */
00099 /**************************************************************************************************/
00100
00101 class MLX90614 {
00102 public:
00103     MLX90614(uint8_t i2caddr = MLX90614_I2CDEFAULTADDR);
00104
00105     boolean  begin();
00106     boolean  isReady(void) { return _ready; };
00107     uint64_t readID(void);                                  /**< Chip ID getter */
00108
00109     uint8_t  getIIRcoeff(void);                             /**< IIR coefficient getter */
00110     uint8_t  getFIRcoeff(void);                             /**< IIR coefficient getter */
00111     float    getEmissivity(void);                           /**< Emissivity getter */
00112
00113     void     setIIRcoeff(uint8_t csb = 4);                  /**< IIR coefficient setter */
00114     void     setFIRcoeff(uint8_t csb = 7);                  /**< IIR coefficient setter */
00115     void     setEmissivity(float emiss = 1.0);              /**< Emissivity setter */
00116
00117     uint16_t readEEProm(uint8_t);
00118     void     writeEEProm(uint8_t, uint16_t);
00119
00120     Property<uint8_t, MLX90614> busAddr;                    /**< SMBus address property */
00121     Property<uint8_t, MLX90614> rwError;                    /**< R/W error flags property */
00122     Property<uint8_t, MLX90614> crc8;                       /**< 8 bit CRC property */
00123     Property<uint8_t, MLX90614> pec;                        /**< PEC property */
00124
00125     /** Enumerations for temperature units. */
00126     enum tempUnit_t {MLX90614_TK,                           /**< degrees Kelvin */
00127                      MLX90614_TC,                           /**< degrees Centigrade */
00128                      MLX90614_TF                            /**< degrees Fahrenheit */
00129     };
00130     /** Enumerations for temperature measurement source. */
00131     enum tempSrc_t  {MLX90614_SRCA,                         /**< Chip (ambient) sensor */
00132                      MLX90614_SRC01,                        /**< IR source #1 */
00133                      MLX90614_SRC02                         /**< IR source #2 */
00134                     };
00135
00136     double   readTemp(tempSrc_t = MLX90614_SRC01, tempUnit_t = MLX90614_TC);
00137     double   convKtoC(double);
00138     double   convCtoF(double);
00139
00140 private:
00141     boolean  _ready;
00142     uint8_t  _addr;                                         /**< Slave address */
00143     uint8_t  _rwError;                                      /**< R/W error flags */
00144     uint8_t  _crc8;                                         /**< 8 bit CRC */
00145     uint8_t  _pec;                                          /**< PEC */
00146
00147     uint16_t read16(uint8_t);
00148     void     write16(uint8_t, uint16_t);
00149
00150     uint8_t  getRwError(void)   {return _rwError;}          /**< R/W error flags getter */
00151     uint8_t  getCRC8(void)      {return _crc8;}             /**< 8 bit CRC getter */
00152     uint8_t  getPEC(void)       {return _pec;}              /**< PEC getter */
00153
00154     uint8_t  getAddr(void);                                 /**< SMB bus address getter */
00155     void     setAddr(uint8_t);                              /**< SMB bus address setter */
00156
00157 };
00158
00159 #endif /* _MLX90614_H_ */
```

6.10 README.md File Reference

7 Example Documentation

7.1 MelexisTest.ino

An example of how to use the MLX90614 class.

```
00001 /**************************************************************************************************//**
00002  *  \brief      Melexis MCX90614BAA Test Program - Sensor test implementation.
00003  *  \details    Arduino test implementation of Melexis MCX90614 PIR temperature sensor driver.
00004  *
00005  *  \note       THIS IS ONLY A PARTIAL RELEASE. THIS DEVICE CLASS IS CURRENTLY UNDERGOING
00006  *              ACTIVE DEVELOPMENT AND IS STILL MISSING SOME IMPORTANT FEATURES. PLEASE KEEP
00007  *              THIS IN MIND IF YOU DECIDE TO USE THIS PARTICULAR CODE FOR ANYTHING.
00008  *
00009  *  \file       MelexisTest.ino
00010  *  \author     J. F. Fitter <jfitter@eagleairaust.com.au>
00011  *  \version    1.0
00012  *  \date       2014-2017
00013  *  \copyright  Copyright (c) 2017 John Fitter.  All right reserved.
00014  *
00015  *  \par        License
00016  *              This program is free software; you can redistribute it and/or modify it under
00017  *              the terms of the GNU Lesser General Public License as published by the Free
00018  *              Software Foundation; either version 2.1 of the License, or (at your option)
00019  *              any later version.
00020  *  \par
00021  *              This Program is distributed in the hope that it will be useful, but WITHOUT ANY
00022  *              WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
00023  *              PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details
00024  *              at http://www.gnu.org/copyleft/gpl.html
00025  *  \par
00026  *              You should have received a copy of the GNU Lesser General Public License along
00027  *              with this library; if not, write to the Free Software Foundation, Inc.,
00028  *              51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
00029  *
00030  *//**********************************************************************************************/
00031
00032 #define MELEXISTEST_C
00033 #define __STDC_LIMIT_MACROS
00034 #define __STDC_CONSTANT_MACROS
00035
00036 #include <Arduino.h>
00037 #include <Wire.h>
00038 #include <MLX90614.h>
00039 #include "printf.h"
00040
00041 MLX90614 mlx = MLX90614(MLX90614_BROADCASTADDR);      // *** must be only one device on bus ***
00042
00043 /**
00044  *  \brief  Program setup.
00045  */
00046 void setup(void) {
00047
00048     Wire.begin(); // library does not do this by default
00049     Serial.begin(115200);
00050     printf_begin();
00051     mlx.begin();
00052
00053     Serial.println(F("\nMelexis MLX90614 Temperature Sensor Test Program"));
00054     Serial.print(F("SMBus address ="));
00055     printf(" %02Xh", (uint8_t)mlx.readEEProm(MLX90614_ADDR));
00056     Serial.print(F("  Chip ID ="));
00057
00058     uint64_t id = mlx.readID();
00059     printf(" %04X-%04X-%04X-%04X\n\n", (uint16_t)(id >> 48), (uint16_t)(id >> 32),
00060                                        (uint16_t)(id >> 16), (uint16_t)id);
00061     dumpEEProm();
00062     Serial.println("");
00063 }
00064
00065 /**
00066  *  \brief  Main processing loop.
00067  */
00068 void loop(void) {
00069     static uint16_t smpcount = 0, errcount = 0;
00070
00071     // read ambient temperature from chip and print out
00072     printlnTemp(mlx.readTemp(MLX90614::MLX90614_SRCA, MLX90614::MLX90614_TK), 'A');
00073     if(mlx.rwError) ++errcount;
```

```
00074
00075     // read object temperature from source #1 and print out
00076     printlnTemp(mlx.readTemp(MLX90614::MLX90614_SRC01, MLX90614::MLX90614_TK), 'O');
00077     if(mlx.rwError) ++errcount;
00078
00079     // print running total of samples and errors
00080     Serial.print(F("        Samples:Errors "));
00081     printf("%u:%u\r\n", smpcount += 2, errcount);
00082
00083     // slow down to human speed
00084     delay(250);
00085 }
00086
00087 /**
00088  *  \brief           Print a line of temperature, crc, pec, and error string.
00089  *  \param [in] temp Temperature
00090  *  \param [in] src  Temperature source
00091  */
00092 void printlnTemp(double temp, char src) {
00093     char str[20];
00094
00095     if(mlx.rwError) Serial.print(F("No valid temperatures                                "));
00096     else {
00097         if(src == 'A') Serial.print(F("Ambient temperature"));
00098         else Serial.print(F("Object  temperature"));
00099         printf(" = %sK ", floatToStr(str, temp));
00100         printf("%sC ",    floatToStr(str, mlx.convKtoC(temp)));
00101         printf("%sF    ", floatToStr(str, mlx.convCtoF(mlx.convKtoC(temp))));
00102     }
00103     printCRC(mlx.crc8, mlx.pec);
00104     printErrStr(mlx.rwError);
00105     Serial.println("");
00106 }
00107
00108 /**
00109  *  \brief Print a complete memory dump of the EEPROM.
00110  */
00111 void dumpEEProm() {
00112
00113     Serial.println(F("EEProm Dump"));
00114     for(uint8_t j=0; j<8; j++) {
00115         for(uint8_t i=0; i<4; i++) printf("%02Xh-%04Xh    ", j*4+i, mlx.readEEProm(j*4+i));
00116         printCRC(mlx.crc8, mlx.pec);
00117         printErrStr(mlx.rwError);
00118         Serial.println("");
00119     }
00120 }
00121
00122 /**
00123  *  \brief          Utility to stringify a float.
00124  *  \param [in] str String to receive converted result
00125  *  \param [in] val Float value
00126  *  \return         Float as string
00127  */
00128 char* floatToStr(char *str, double val) {
00129
00130     sprintf(str, "%4d.%02u", int(val), int(val * 100) % 100);
00131     return str;
00132 }
00133
00134 /**
00135  *  \brief          Just print the crc and pec.
00136  *  \param [in] crc CRC
00137  *  \param [in] pec PEC
00138  */
00139 void printCRC(uint8_t crc, uint8_t pec) {printf("crc=%02Xh pec=%02Xh", crc, pec);}
00140
00141 /**
00142  *  \brief          Convert error flags to diagnostic strings and print.
00143  *  \param [in] err Error flags
00144  */
00145 void printErrStr(uint8_t err) {
00146
00147     Serial.print(F("  *** "));
00148     if(err == MLX90614_NORWERROR) Serial.print(F("RW Success"));
00149     else {
00150         Serial.print(F("Errors: "));
00151         if(err &  MLX90614_DATATOOLONG) Serial.print(F("Data too long / "));
00152         if(err &  MLX90614_TXADDRNACK)  Serial.print(F("TX addr NACK / "));
00153         if(err &  MLX90614_TXDATANACK)  Serial.print(F("TX data NACK / "));
00154         if(err &  MLX90614_TXOTHER)     Serial.print(F("Unknown / "));
00155         if(err &  MLX90614_RXCRC)       Serial.print(F("RX CRC / "));
00156         if(err &  MLX90614_INVALIDATA)  Serial.print(F("Invalid data / "));
00157         if(err &  MLX90614_EECORRUPT)   Serial.print(F("EEPROM / "));
00158         if(err &  MLX90614_RFLGERR)     Serial.print(F("RFlags / "));
00159     }
00160 }
```

```
00161
00162 /**
00163  *  \brief  EEPROM memory contents factory default values.
00164  */
00165 const struct defaultEEPromData {
00166     uint8_t  address;
00167     uint16_t data;
00168 } eDat[] =  {{0x20, 0x9993}, {0x21, 0x62E3}, {0x22, 0x0201},
00169              {0x23, 0xF71C}, {0x24, 0xFFFF}, {0x25, 0x9FB4},
00170              {0x2E, 0xBE5A}, {0x2F, 0x0000}, {0x39, 0x0000}};
00171
00172 /**
00173  *  \brief    Set EEPROM memory contents to factory default values.
00174  *  \remarks  A device with default adress must not be on the bus.
00175  *            \n<tt>Only user allowed memory locations are written.</tt>
00176  */
00177 void setEEPromDefaults(void) {
00178
00179     for(uint8_t i = 0; i < sizeof(eDat)/sizeof(defaultEEPromData),
00180         !mlx.rwError; i++) {
00181         mlx.writeEEProm(eDat[i].address, eDat[i].data);
00182     }
00183 }
00184
```

Index