Ethernet Module
Operations Manual

(This page intentionally left blank.)

DOCUMENT NUMBER: 1903-2021-01 REV D

JUNE 2016

Made in the United States of America

(This page intentionally left blank.)

TABLE OF CONTENTS

Page

TABLE OF CONTENTS
(Continued)

SOFTWARE LICENSE AGREEMENT

The software contained in this manual 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 library 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.

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

You can also obtain a copy of this license by contacting Teledyne Monitor Labs, Inc. at 5310 N. Pioneer Rd., Gibsonia, PA 15044, USA (supplier and licensor).

Teledyne Monitor Labs, Inc.

1-724-444-5000 Phone
1-724-444-5050 Fax

http://www.teledyne-ml.com

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to

satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser 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 Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES

PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

(This page intentionally left blank.)

1.0 INTRODUCTION

The need for simplified web browser-based remote access, configuration and control of intelligent embedded microprocessor-based devices has led Teledyne Monitor Labs (TML) to offer the Ethernet Module, which at the same time can provide HTML web pages for user interface and fast Modbus TCP access to instrument data and parameters.

The module comes in two hardware implementations:

- Internal to the Enhanced Remote Panel
- Stand-alone

Although both are similar electrically and use identical firmware, the Stand-alone version was made to interconnect in an existing instrument network, while the Enhanced Remote Panel version was made to be installed as an option in the Enhanced Remote Panel at the TML factory.

Ethernet Module firmware varies depending on product line (LightHawk, LaserHawk, etc.) since each instrument type uses unique network parameter definitions.

1.1 USING THIS MANUAL

This manual is intended for users interested in either the web browser based interface, Modbus TCP access or both. It also contains instructions and electrical requirements for initial installation on an Ethernet network. The browser based mode of operation is accessible to users via commercial readily available PC programs and is consequently much simpler than Modbus TCP communication. The latter requires development of a driver program at the PC end, hence sections dealing with this mode of communication (Section 3 and the Appendices) presume the reader has a background in software development and some familiarity with each instrument.

1.2 HARDWARE DESCRIPTION

The Ethernet Module has two processors: one responsible for LONWORKS® network communications and an ARM7 responsible for Ethernet and Internet communication. The LONWORKS® network processor binds itself (i.e., self installs) to the existing LONWORKS® network and acts as an interface to the ARM7 processor. Both processors are connected via a UART. The ARM7 processor side stores all received instrument data and provides a user interface via HTML web pages and Modbus TCP. The LONWORKS® network processor communicates with other network nodes via an FTT10A twisted shielded pair.

The version of the Ethernet Module that is factory packaged in the Enhanced Remote Panel is internally wired to power and the LONWORKS® network. The Ethernet connection is accessible via the rear panel of the Enhanced Remote Panel

through a connector labeled ETHERNET. There are two access holes below the ETHERNET connector labeled SERVICE and RESET. Usage of the SERVICE and RESET functions are outside the scope of this manual and should only be activated if instructed to do so by TML Tech Support personnel.

The Stand-alone version of the Ethernet Module is packaged in its own small enclosure. See Appendix A for installation drawings regarding mechanical details and electrical wiring. Electrical wiring for the Stand-alone Ethernet Module consists of three parts:

- Low voltage DC power supply
- LONWORKS® network
- 10/100 BASE T Ethernet network.

The connector below J1 must be wired to DC power from the Power Adapter. J1 must be wired to the LONWORKS® network of the instrument. Connection to the Ethernet network is via the 10/100 BASE T connector shown in the END VIEW of the Appendix A drawing "DIMENSIONAL DWG. STAND ALONE GATEWAY / ETHERNET MODULE ASSY." There are two access holes to the left of the ETHERNET connector labeled RST and SRV. Usage of the RST and SRV functions are outside the scope of this manual.

The Ethernet Module board contains two jumpers of interest (JU6 and JU7) to end users of the Stand-alone version because they may need to change position based on the LONWORKS® network wiring or the instrument type. ***Users of the Ethernet Module version internal to the Enhanced Remote Panel do not need to worry about these jumpers since they are factory configured and in this case will not need to change based on external wiring.*** In Direct Interface systems or in situations where the Stand-alone Ethernet Module is the last LONWORKS® network device on the line, these jumpers should be changed from factory default to the positions in the "Ethernet Module Jumper Positions for Direct Interface Systems" table. In Stand-alone Ethernet Module systems with an Enhanced Remote Panel and wiring as per the applicable Appendix A drawings, the factory defaults apply for JU6 and JU7 as per the second table below.

Note: *If JU6 and JU7 on the Ethernet Module are changed from factory default and an Enhanced Remote Panel is present (non-standard wiring), the Enhanced Remote Panel Motherboard termination jumpers (JU6 and JU7) should also be changed (see table) for optimal LONWORKS® network termination.*

JUMPER	POSITION	DESCRIPTION
JU6	IN	LONWORKS® Network Termination
JU7	OUT	LONWORKS® Network Single / Double Termination

Ethernet Module Jumper Positions for Direct Interface Systems (Systems without ERP or with ERP but Nonstandard Wiring)

JUMPER	POSITION	DESCRIPTION
JU6	OUT	LONWORKS® Network Termination
JU7	IN	LONWORKS® Network Single / Double Termination

Ethernet Module Factory Default Jumper Positions (Systems with ERP and Standard Wiring)

Jumper	Options	Default	Function
JU1	A **B**	B	NEURON Memory Select
JU2	A B **C** D	C	NEURON Memory Select
JU3	1 2 **3 4**	3-4	NEURON Program Select
JU4	1 2 **3 4**	3-4	NEURON Program Select
JU5	1 2 **3 4**	3-4	NEURON Program Select
JU8	**OPEN** …to JU9	OPEN	I/O Device Selection
JU9	OPEN to JU8 **to JU10**	to JU10	I/O Device Selection
JU10	OPEN **to JU9** to JU11	to JU9	I/O Device Selection
JU11	**OPEN** to JU10	OPEN	I/O Device Selection
JU12	1 2 3 4	OPEN	DIGI-ME Manufacturing Header
JU13	**1 2** 3	1-2 (OUT)	DIGI-ME Exec selector APP/BOOT
JU14	**IN** OUT	IN	DIGI-ME Hardware Reset
JU15	1 **2 3**	2-3 (OUT)	RS-422 +5 Volt Power
JU16	**1 3** 5 **2 4** 6	1-3, 2-4	RS-232 Serial Driver Control

Ethernet Module Board Jumpers (excluding JU6 & 7)

Part Number	Description
1903-2010-01	LightHawk 560 Ethernet Module, Internal to Enhanced Remote
1903-2010-02	Ultraflow 150 Ethernet Module, Internal to Enhanced Remote
1903-2010-03	LaserHawk 360 Ethernet Module, Internal to Enhanced Remote
1903-2010-11	LightHawk 560 Ethernet Module, Stand-alone
1903-2010-12	Ultraflow 150 Ethernet Module, Stand-alone
1903-2010-13	LaserHawk 360 Ethernet Module, Stand-alone
1905-0100-01	Stand-alone Ethernet Module Power Supply Board

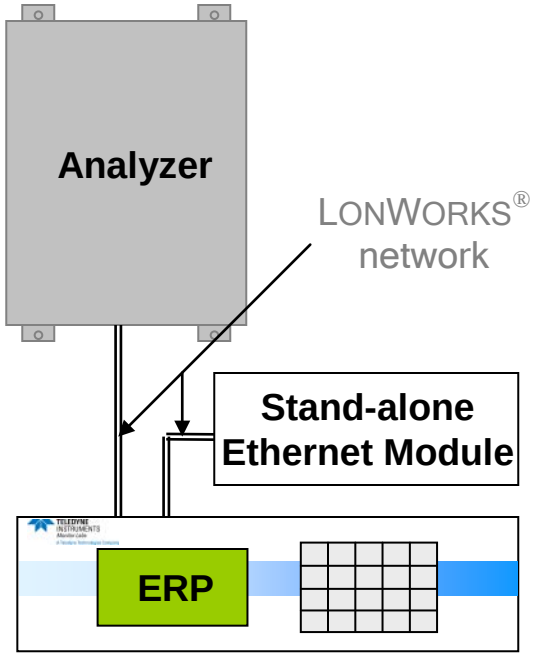
Part Numbers for Ethernet Module Circuit Boards

JUMPER	POSITION	DESCRIPTION
JU6	B - C	LONWORKS® Network Single / Double Termination
JU7	B - C	LONWORKS® Network Termination

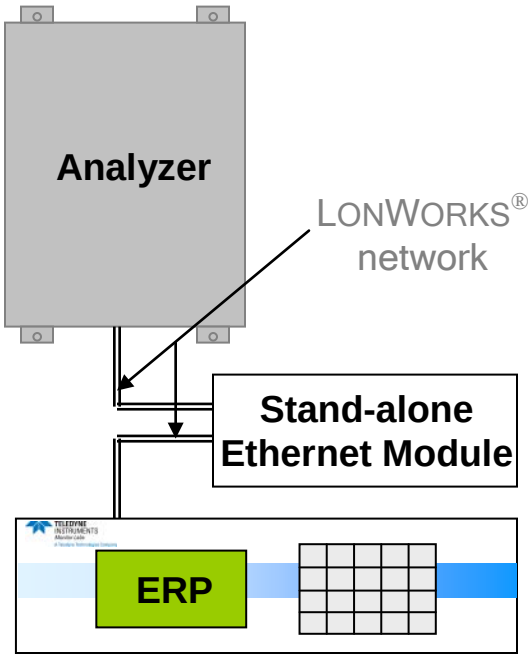
Enhanced Remote Panel Motherboard Jumper Positions (Systems with Nonstandard Wiring)

JUMPER	POSITION	DESCRIPTION
JU6	B - C	LONWORKS® Network Single / Double Termination
JU7	A - B	LONWORKS® Network Termination

Enhanced Remote Panel Motherboard Factory Default Jumper Positions (Systems with ERP and Standard Wiring)



ERP with Stand-alone Ethernet Module and Non-standard Wiring (see Tables for JU6 & JU7 Positions)



ERP with Stand-alone Ethernet Module Standard Wiring (Use Factory Defaults for JU6 & JU7)

1.3 SOFTWARE DESCRIPTION

Most of the instrument data traversing the LONWORKS® network is prearranged in data structures that are completely dependent on product line. These data structures can contain instantaneous and averaged instrument data, parameters, status and other information. In order to make the data and protocol translations fast, these structures are directly mapped to the memory on the ARM processor. This memory represents both the structured data seen on the WEB pages served by the HTML server and the input and holding registers accessed via Modbus TCP.

The Ethernet Module also provides FTP access for factory firmware updates. Description of the use of FTP access to the Ethernet Module is beyond the scope of this manual.

2.0 CONNECTING TO THE ETHERNET MODULE

Starting in May 2016 if you the customer did not specify a static IP address at the time of placing your order, the Ethernet Module comes preset by the factory with a default static IP address. A label is provided with the static IP address, and Subnet Mask programmed into the Ethernet Module. This label can be found on the rear left corner of the ERP or stand alone module or on the PCA static discharge protective packaging if a spare board was purchased. Below is a list of possible Static IP addresses your module may come with.

Possible factory set Static IP addresses for Ethernet PCA, TML P/N 1910-2010-XX	
SM8200 (-04)	172.16.83.248
UF150 (-02, -08, -12)	172.16.83.250
LH360 (-03, -13)	172.16.83.217
LH560 (-01, -05, -07, -10, -11)	172.16.83.251
ERP	172.16.83.252
SUBNET MASK FOR ALL IS 255.255.252.0	

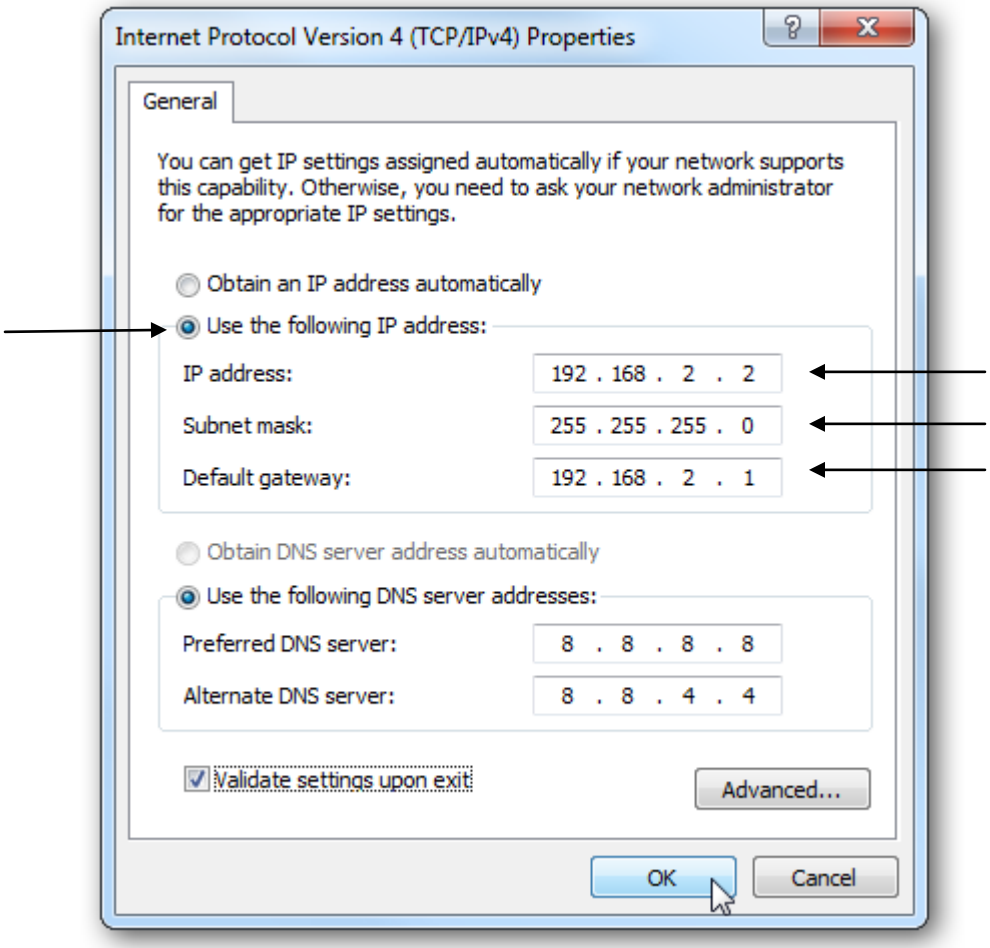
Use the above table as a reference only. Please refer to the label provided with the spare board, Stand-alone module, or ERP for the actual IP address of your Ethernet Module. In order to access the Ethernet Module web browser interface, you will need to change the IPv4 settings on your PC to match the settings. After you gain access to the web browser insterface you can reprogram the static IP address and Subnet Mask per Section 2.1 of this manual.

If an IP address was specificed at the time the order was placed, and the Ethernet Module label provided reflects that it was programmed accordingly your IPv4 settings will likely not need to be changed. You should be able to access the web browser insterface by simply entering the IP address in the address bar of your favorite internet browser. If you do not see a label with an IP address and Subnet, your Ethernet Module did not come with a static IP address. See page 2-6 for dynamic IP address procedure.

The procedure for accessing an Ethernet Module with a default static IP address is as follows for a PC using the Windows 7 operating system (It may be slightly different for other operating systems. Contact your network administrator for assistance if needed):

- Make sure your PC and the Ethernet Module are connected to the same network.
- From the Control Panel click *View network status and tasks* under the *Network and Internet* section. In the next screen, click *Change adapter settings.*
- In the next screen select the appropriate network that the PC and the Ethernet Module are connected to.
- In the pop-up window click *Properties.* Click *Internet Protocol Version 4*
- Note what the current settings are so you can reset properly at the end of this procedure.
- Check the *Use the Following IP Address option.*

- Enter an IP address that is one less than the one printed on the Ethernet Module label. For example, if the label says 172.16.83.252 set your PC to 172.16.83.251.
- Set the Subnet Mask to match the Ethernet Module Label. Click OK, no Default Gateway is needed. See below for reference.



- You should now be able to enter the Ethernet Module factory default IP address into the address bar of your favorite internet browser and gain access to the web browser interface. If successful, you will be promted for a username and password:

User Name	Password	Access Level
system	Light	Read-Write
operator	Sound	Read Only

- See section 2.1 of this manual for instructions on how to configure the Ethernet Module to your site specific network settings. Remember to reset your PC to its original IPv4 settings when you are done.

If you receive an Ethernet Module that was manufactured prior to May 2016 it will likey be setup to automatically obtain an IP address from the DHCP server on the Ethernet network or other compatible device. Therefore, since a DHCP server sets the initial IP address, to connect to the Ethernet Module one must discover the IP address the DHCP server has assigned the device. Once a connection is established with the Ethernet Module via the web browser interface a static IP address may be assigned.

In order to quickly locate the assigned IP address of the interface, use the software called WinArpSpoof that is supplied with the Ethernet Module. The procedure is as follows:

- Locate the Ethernet Module MAC address. It is usually found on a sticker attached to the side of the Stand-alone Version enclosure or the rear of the Enhanced Remote Panel. The MAC address is in a format **0050C2 - 4A5xxx** where xxx can be any combination of numbers 0-9 or letters A-F.

- Start the WinArpSpoof program. Under the "NIC" option select your Network Interface Card that is connected on the same network that the Ethernet Module is connected to.

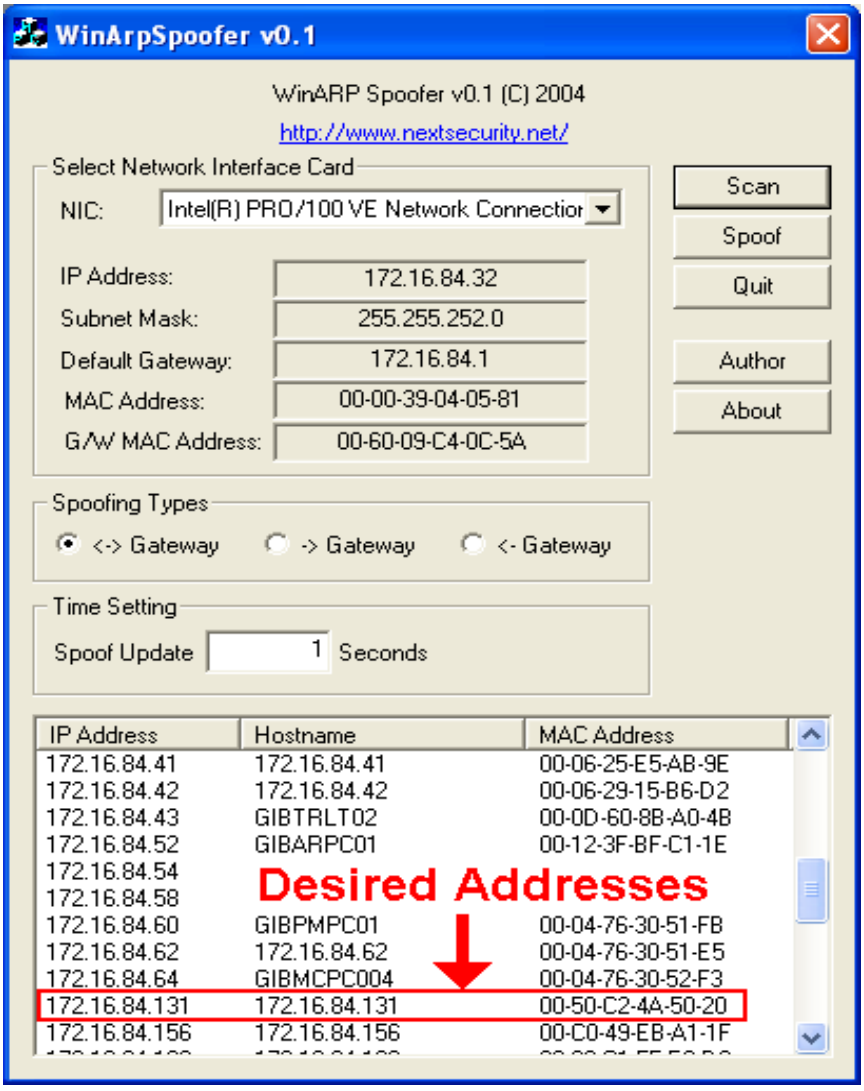
- Press the "Scan" button and wait until the list gets populated.

- Scroll until you find the MAC address of the Ethernet Module. Write down the corresponding IP address, like shown on the picture below. In our case on the picture, the factory assigned MAC address is 0050C2 – 4A5020 and the assigned IP address that we've found is 172.16.84.45.

- Open your favorite Internet browser and type the IP address that you found, for example: http://172.16.84.131

- If everything is correct, you should get a window asking you for User Name and Password. Use the values given in the table above and don't forget to change them from the default values to increase security.

In most cases, your network administrator can assist you with this process and can assign an alias (name) for the device or can give you an IP address that will always be assigned to the same Ethernet Module.



There are two types of users that can access the WEB interface and those are ***system*** and ***operator***. Please note that user name and password are case sensitive. The "system" user(s) can both view and change properties of the instrument and the Ethernet Module. The "operator" user(s) can only view the instrument properties and data. The case-sensitive default passwords are noted in the table below.

User Name	Password	Access Level
system	Light	Read-Write
operator	Sound	Read Only

NOTE: ***YOU MUST CHANGE THE ABOVE NOTED FACTORY DEFAULT PASSWORDS ONCE YOU CONNECT TO THE MODULE VIA THE USER CONFIGURATION SCREEN.***

2.1 SETTING THE IP CONFIGURATION

The Ethernet Module can be configured to have either a static IP assignment and a dynamic one. In order to configure the module to use a static IP, in the *Module Configuration* screen do the following:

- Check the *Use Static IP* box;
- Enter a value for the *IP Address*;
- Enter a value for the *Mask*;
- Enter a value for the *Gateway*;
- Press the button to change the module's IP assignments;
- After the next screen shows, reset or recycle power on the module.

In order to configure the module for a dynamic IP (one that will be assigned to the module upon request), do the following:

- Uncheck the *Use Static IP* box;
- Press the button to change the module's IP assignments;
- After the next screen shows, reset or recycle power on the module.

2.2 CHANGING THE UNIT NAME(S)

The Ethernet module enables you to (re)name the unit(s) that is connected to thus giving you an option to easily identify each module when there is more than one that you can access. This option is given on the *Module Configuration* screen.

NOTE: *The unit name has nothing to do with the IP configuration nor is it the DNS name of the device.*

2.3 SETTING USER ACCOUNTS

There are two levels of access to the WWW interface: ***system*** and ***operator.*** As mentioned previously, these accounts have different security settings and can access different parts of the instrument's data and configuration. ***It is of utmost importance that the passwords for these accounts are changed from the factory defaults as soon as possible.***

The *User Configuration* screen provides an interface that changes the *operator's* or *system's* password, but only if you are logged into the *system* account. The *operator* cannot change its own password. Remember to restart or cycle power on the module after successful change of any password

3.0 MODBUS OVER TCP ACCESS

The *Modbus TCP* interface provides means for fast data acquisition and remote instrument configuration using the Modbus protocol. The Ethernet Module allows for up to two Modbus TCP devices to be connected at a same time. Restricted access is assured by specifying the IP address, the MAC address or both of the connecting device(s). The main value of the Modbus TCP interface is as an emission, instrument status and calibration data application front end for data acquisition systems. It is of less utility as an instrument configuration and troubleshooting tool since these functions are intermittent in nature and are handled by the HTML web server interface, which does not require any extra development at the PC end as does Modbus TCP.

Using the *Modbus Configuration* screen and *system* account you can specify the security method (*IP address*, *MAC address* or *Both IP & MAC*) that each of the two entries will use. If the device requesting Modbus communication does not match the security requirement, the connection socket will instantly be terminated without notice. Since there can be up to two settings, each set of security arrangements is evaluated individually. If you don't want to allow any device to access the instrument's data and configuration via Modbus, select the **NO MODBUS** option.

3.1 MAPPING DATA INTO MODBUS REGISTERS

The Modbus registers by definition are 16 bits in size. The data that these registers contain is in general a direct translation of the LONWORKS® network data structures. Thus the data stored is either concatenated or clustered together to take as little space as possible in the registers, rounding up to the nearest 16 bits. For instance, examine the following structure:

```
struct {
      BYTE day;
      BYTE month;
      WORD year;
      FLOAT measurement;
      BYTE direction;
} nviSomeData;
```

If the above structure were mapped into the Modbus Input Registers starting at offset 20, it would look like:

Variable	Member	Type	Offset
nviSomeData	day	BYTE	20
	month	BYTE	
	year	WORD (HI BYTE)	21
		WORD (LO BYTE)	
	measurement	FLOAT (S + MSE)	22
		FLOAT (LSE + MSM)	
		FLOAT(LSM HIGHBYTE)	23
		FLOAT(LSM LOBYTE)	
	direction	BYTE	24
	(filler)	BYTE	

As one may notice, the above structure is one byte shy of a word boundary and because of it a filler byte is added instead of starting the next structure at a half-word offset.

3.2 DATA TYPES, REPRESENTATION AND SIZES

There are three basic types of data used in this implementation: **BYTE, WORD** and **FLOAT**.

BYTE is a size of 8 bits. If there is a bit-wise structure mapped into a byte, as the instance below, the bits are mapped into a byte in a Big-Endian fashion:

```
struct {
            BYTE some_bit_7 : 1;
            BYTE some_bit_6 : 1;
            BYTE some_bit_5 : 1;
            BYTE some_bit_4 : 1;
            BYTE some_bit_3 : 1;
            BYTE some_bit_2 : 1;
            BYTE some_bit_1 : 1;
            BYTE some_bit_0 : 1;
      } nviSomeBitwiseData;
```

So if the above bit 7, 6 and 5 are high (1), the structure mapped into a BYTE will read 11100000 binary, or E0 hexadecimal or 224 decimal.

WORD is a size of 16 bits and the representation is in Big Endian fashion – High-Byte first.

FLOAT is a size of 32 bits and the representation of the variables is according to the IEEE754 standard with the (Sign + Most Significant Exponent) first. The IEEE754 standard defines the floating point number as:

Byte 1	Byte 2	Byte 3	Byte 4
Sign + Exponent (S + MSE)	Exponent + Mantissa (LSE + MSM)	Mantissa (LSM)	Mantissa (LSM)
SEEEEEEE	EEEEEEEM	MMMMMMM	MMMMMMM

As an example, the number -1.23456 (0xbf9e060f hexadecimal) will be stored as:

TYPE	VALUE	MODBUS register value	MODBUS address offset
IEEE float (S+MSE)	191 (0xbf)	49054 (0xbf9e)	0
IEEE float (LSE+MSM)	159 (0x9e)		
IEEE float (HIBYTE(LSM))	6 (0x06)	1551 (0x060f)	1
IEEE float (LOBYTE(LSM))	15 (0x0f)		

3.3 BIG-ENDIAN VS. LITTLE-ENDIAN FORMAT

Most of today's INTEL based PC's store and represent data in Little-Endian fashion. This is opposite to the convention of Big-Endian used in this document. Big-Endian comes from a convention of *Big-End First*, meaning the higher byte is at the first position in a word. This is the way most humans would represent data. A Little-Endian based platform uses the concept of *Little-End First* which makes more sense when storing or retrieving data from a microprocessor stack via "pushing" and "popping" functions. If data is read from the registers by an INTEL based PC, the data needs to be converted according to the table below (assume that XX, YY, ZZ and QQ are bytes in hexadecimal representation):

Type Definition	Big Endian Representation	Little Endian Representation
WORD	XX YY	YY XX
FLOAT	XX YY ZZ QQ	QQ ZZ YY XX

3.4 STRUCTURE PACKING AND BYTE ALIGNMENT

In order to minimize the translation time between the protocols, network variables coming from the LONWORKS® network are directly mapped to the Modbus holding or input registers. This mapping is actually nothing more than making the structured network variables pointers that point to a certain holding or input register address. Thus when a network variable comes over the network in a form of message its data is copied into memory at a location assigned by the pointer to the network variable.

In the appendices to follow, the mapping of the particular variables into the input or holding registers will be detailed.

In the following example, we see a structure definition for a network variable, a variable mapping and a pointer assignment.

Variable	Member	Type	Address	Offset
nvi332Version		SNVT_time_stamp		
	year	WORD (HIBYTE)	1	0
		WORD (LOWBYTE)		
	month	BYTE	2	1
	day	BYTE		
	hour	BYTE	3	2
	minute	BYTE		
	second	BYTE	4	3
	(reserved)	BYTE		

```
#define add_nvi332Version   (BYTE)1

struct __SNVT_time_stamp__{
     WORD16   year;    /* = #### 16-bits */
     BYTE  month;      /* = ##    8-bits */
     BYTE  day;        /* = ##    8-bits */
     BYTE  hour;       /* = ##    8-bits */
     BYTE  minute;     /* = ##    8-bits */
     BYTE  second;     /* = ##    8-bits */
 } __attribute__ ((__packed__));

typedef struct __SNVT_time_stamp__ SNVT_time_stamp;
...
SNVT_time_stamp  *nvi332Version ;
...
nvi332Version = (SNVT_time_stamp *)&input_registers [add_nvi332Version];
```

As one may notice, there can be serious problems with structure alignment because some compilers or platforms allot four or more bytes for each structure element. For instance, in the example structure shown above __SNVT_time_stamp__ “year” is followed by “month” followed by “day”. One would assume that the memory location for “month” is 2 bytes after the “year” and the “day” is 1 byte offset after the “month” thus looking something like:

Address	Variable	Size
0x8000:0000	Year	2 bytes
0x8000:0002	Month	1 byte
0x8000:0003	Day	1 byte

However, this may not be the case, since as mentioned, some compilers allot up to four bytes per structure element, thus setting the address boundary at four bytes. So the actual location of the variables in the memory looks something like:

Address	Variable	Size
0x8000:0000	Year	2 bytes
0x8000:0004	Month	1 byte
0x8000:0008	Day	1 byte

If the compiler allots up to four bytes per structure element, then the actual structure members will not point to the appropriate locations in the holding or input registers. To prevent such issues, in our example we used the *__attribute__ ((__packed__))* ***structure attribute construct since it is supported in the GNU compiler used on the ARM7 processor in the Ethernet Module. This, in the given example, instructs the compiler to use absolute alignment and use the structure member's size as an address boundary. Since memory alignment and remedies such as the structure attribute construct are compiler and platform dependant, it is important to consult your compiler literature if the structure member's address boundaries become an issue.***

3.5 PROPAGATING VALUES ON THE NETWORK

Because of the nature of the Modbus protocol and the implementation of the LONWORKS® network, there is no direct method for propagating data from the registers onto the network. This is mostly because Modbus protocol is register based and master slave and the LonTalk® protocol employed by the LONWORKS® network is peer-to-peer and can employ structured network variables.

These differences impact how, for instance, the master real time clock (RTC) is set. The RTC in the LightHawk and other Teledyne Monitor Labs products is usually maintained by the instrument itself and not the Ethernet Module, so the Ethernet Module must know how and when to transfer the data from the Modbus holding registers onto the LONWORKS® network once the desired data has been written into the holding registers. Since the LONWORKS® network variables are often structures that map to multiple Modbus registers, all the Modbus holding registers that map to that LONWORKS® network variable must be updated prior to propagation on the LONWORKS® network.

To solve this, there is a trigger register for each variable that the Ethernet Module can output on the network. When a value of 1 (one) is written into this register, it transfers the contents from the registers that the particular variable occupies onto the network. For instance, look at the example mapping of the *nvoTimeSet* variable, which sets the instrument time.

Variable	Member	Type	Address	Offset
nvoTimeSet_trig		**long (HIBYTE)** **long (LOWBYTE)**	**3**	**2**
NvoTimeSet		SNVT_time_stamp		
	year	long (HIBYTE)	4	3
		long (LOWBYTE)		
	month	BYTE	5	4
	day	BYTE		
	hour	BYTE	6	5
	minute	BYTE		
	second	BYTE	7	6
	(reserved)			

In order to propagate the register's contents onto the network one must:

A. Write the desired data into the holding registers representing the nvoTimeSet output variable using the **PRESET MULTIPLE REGISTERS** Modbus command.
B. Issue a **PRESET SINGLE REGISTER** command and write 1 (one) into the location of *nvoTimeSet_trig*. This will signal the Ethernet Module that the desired data is in the registers occupied by *nvoTimeSet* and it is ready to be sent via the LONWORKS® network to the instrument. Note that if you don't use the **PRESET SINGLE REGISTER** command, no data will be transferred.

4.0 STAND-ALONE VERSION ELECTRICAL CHARACTERISTICS

This section describes the electrical and mechanical characteristics of the Stand-alone version of the Ethernet Module. It does not apply to the version that is internal to the Enhanced Remote Panel.

Physical Characteristics

Length: *9.78 inches (24.84 cm)*
Height: *2.50 inches (6.35 cm)*
Width: *3.50 inches (8.89 cm)*
Weight: *1.5 pounds (0.68 kg)*

Ambient Operating Conditions

Temperature Range: *32 to 140º F (0 to 60ºC)*
Relative Humidity: *5 to 95%*

Power Requirements

Power Adapter:

Input Voltage: *90 to 264 VAC*
Input Frequency: *47 to 63 Hz*
Output Voltage: *12VDC ± 5%*
Maximum Power: *6 VA*
(as used with Stand-alone Ethernet Module)

Wiring

Ethernet: *Double Shielded (braid and foil shields)*
Category 5 or 5e (suitable for 10 /100 BASE T)
LONWORKS® Network: *2 Conductor Shielded Twisted Pair, 16 AWG*
(Alpha 5610B1601 or equivalent)

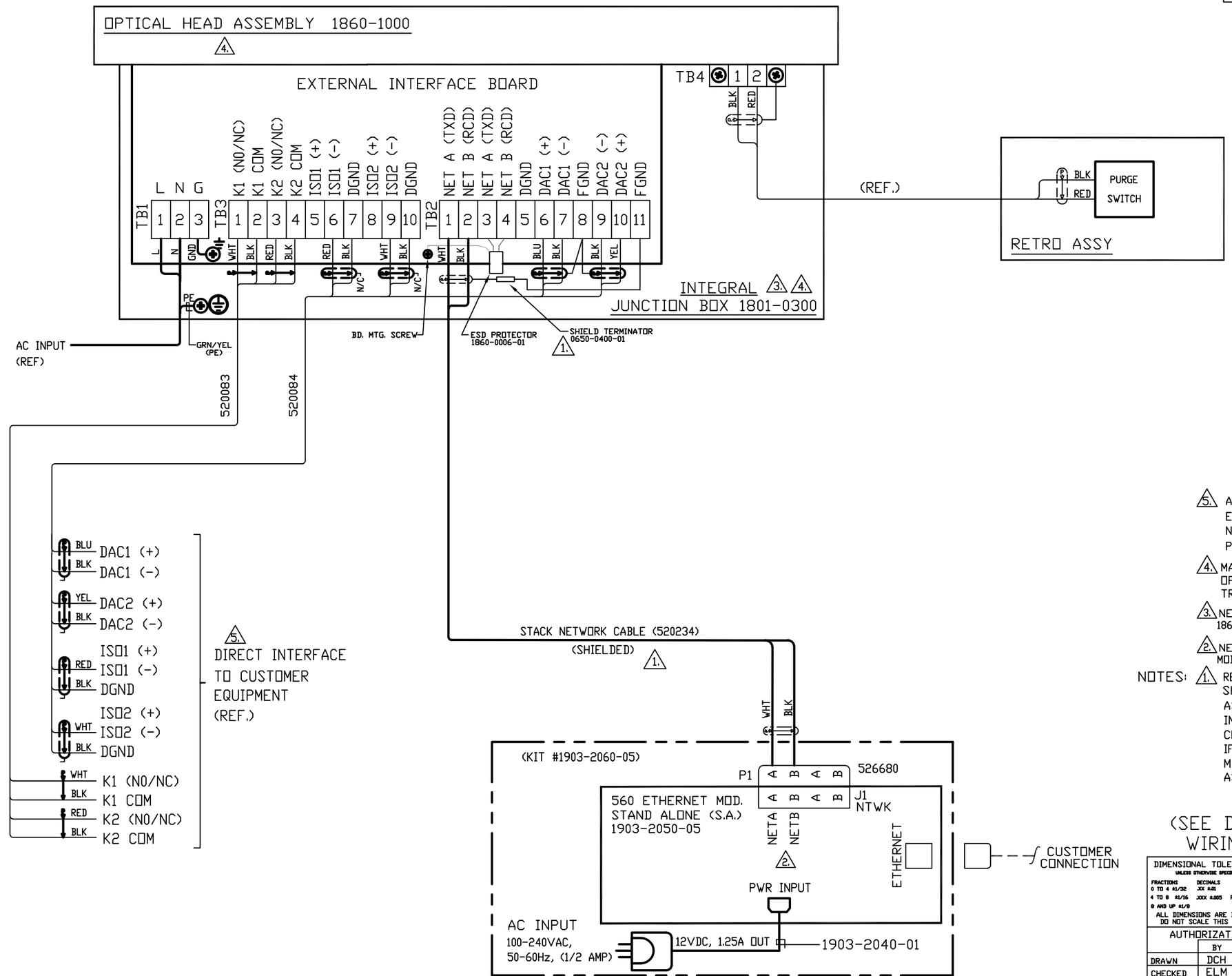
(This page intentionally left blank.)

APPENDIX A

DRAWINGS

DWG. NO 1860-0052 SH 1

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	ELM



5. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

4. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING
(SEE DWG. 1860-0001 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS .XX ±.01; .XXX ±.005
ANGLES ±0°30'
RMS FINISH ✓
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-25-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY
	TOP

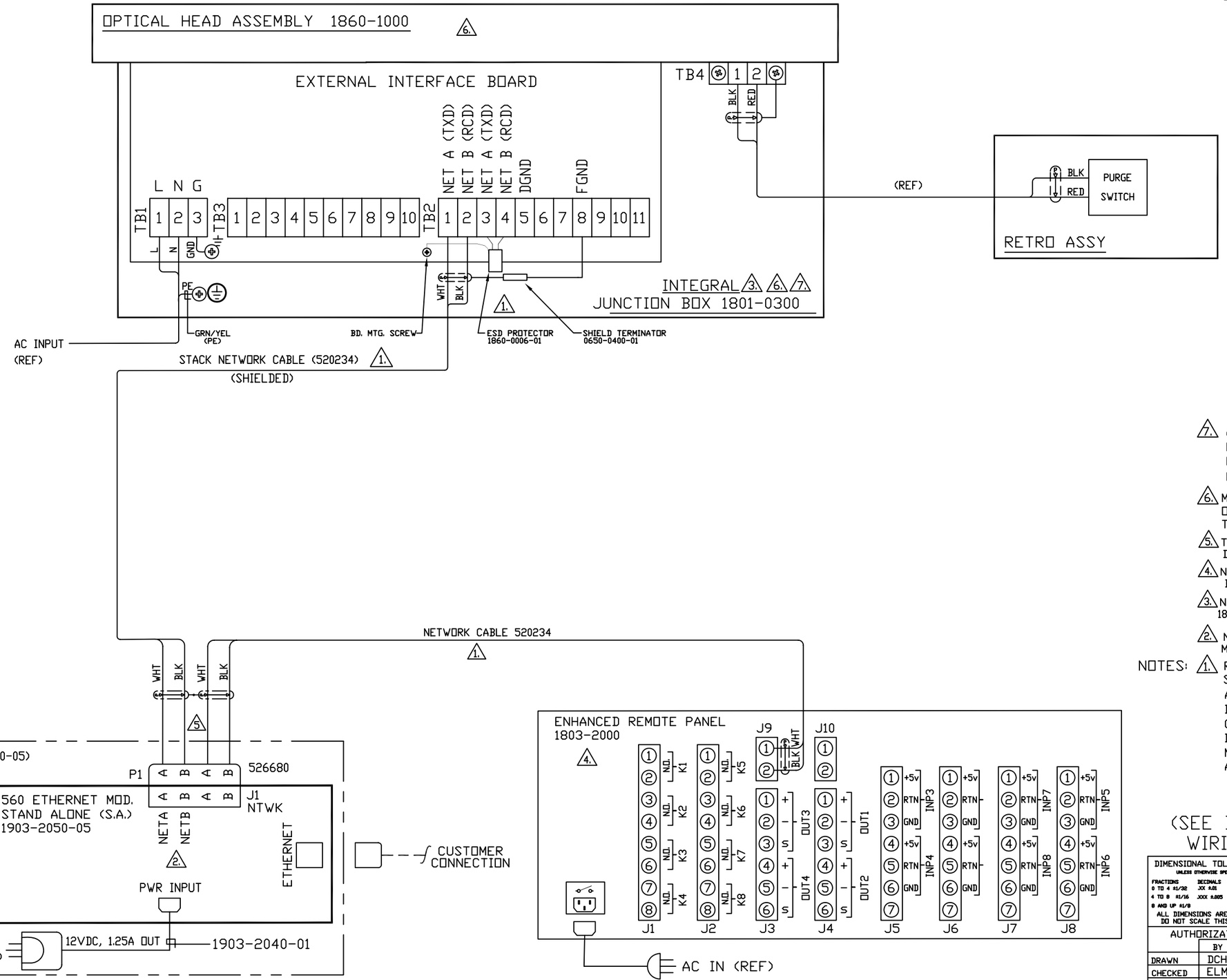
TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LIGHTHAWK 560 DIRECT INTFC. WITH S.A. ETHERNET MODULE, SUPPLEMENTAL WIRING DIAGRAM

MAT'L.
FINISH
JIG NO.
SCALE NTS
SHEET 1 OF 5
PROJECTION
D
DRAWING NO. 1860-0052
LATEST REVISION A
2-15-07
ACAD2007 FILE 18600052

DWG. NO 1860-0052 SH 2

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	ELM



7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

6. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING
(SEE DWG. 1860-0001 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES
UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32, 4 TO 8 ±1/16, 8 AND UP ±1/8
DECIMALS XX ±.01, XXX ±.005
ANGLES ±0°30'
RMS FINISH √
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION

	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-25-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LIGHTHAWK 560 W/ERP AND S.A. ETHERNET MODULE, SUPPLEMENTAL WIRING DIAGRAM

MAT'L.
FINISH
JIG NO.
SCALE NTS
SHEET 2 OF 5
DRAWING NO. 1860-0052
LATEST REVISION A
D

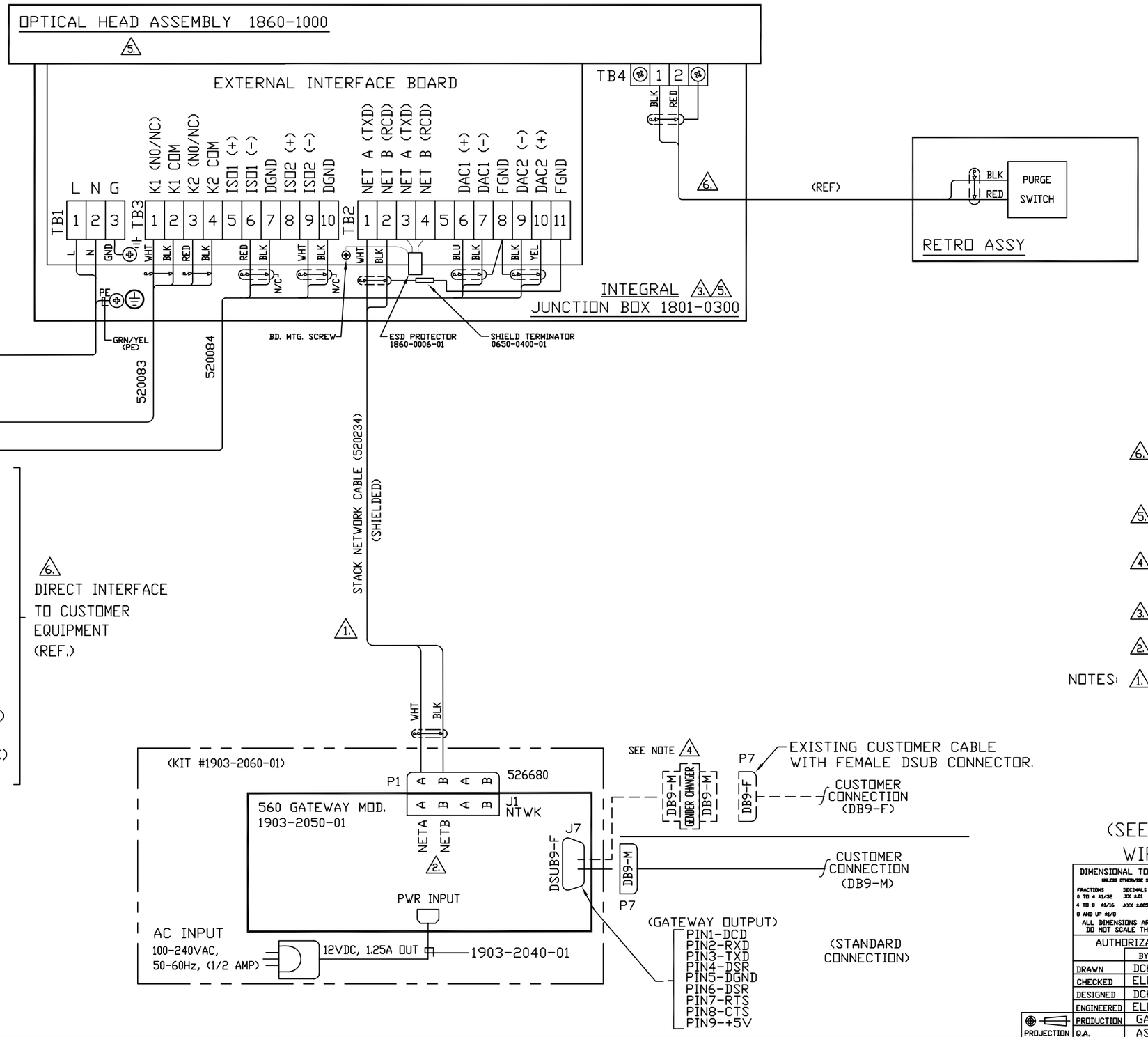
PROJECTION

2-15-07

ACAD2007 FILE 18600052-S2

DWG. NO 1860-0052 | SH 3

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	ELM



6. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

5. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

4 ONLY REQUIRED WHEN REPLACING 1903-1700-02 (OBSOLETE GATEWAY ASSEMBLY) PLUG ADAPTER INTO J7, THEN EXISTING CUSTOMER CABLE INTO ADAPTER.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON GATEWAY MODULE ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION, AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1860-0001 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32 | 4 TO 8 ±1/16 | 8 AND UP ±1/8
DECIMALS .XX ±.01 | .XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION

	BY	DATE
DRAWN	DCH	10-25-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-25-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS Monitor Labs
A Teledyne Technologies Company

TITLE
LIGHTHAWK 560 DIRECT INTFC. WITH GATEWAY MODULE, SUPPLEMENTAL WIRING DIAGRAM

MAT'L. | FINISH | JIG NO.

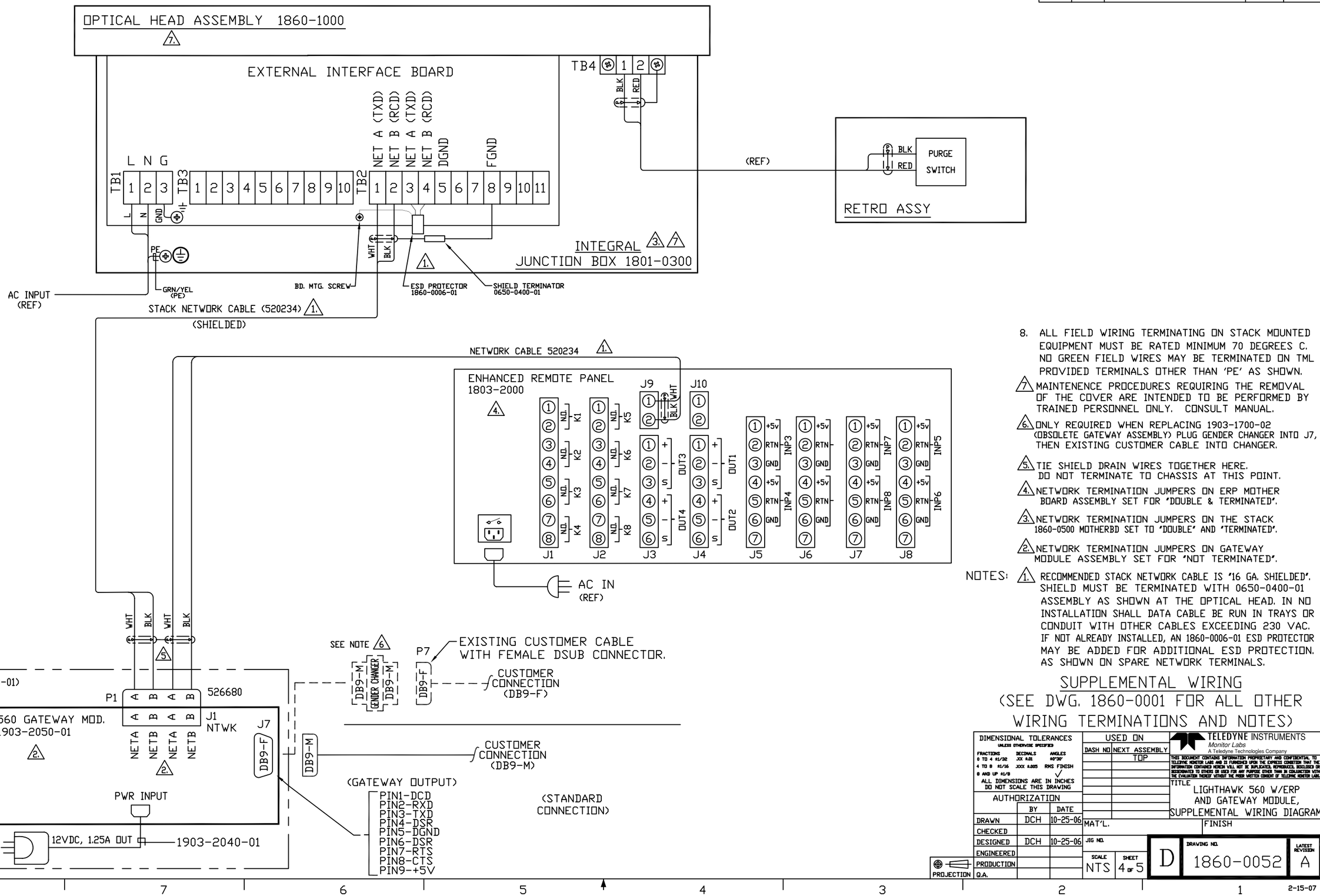
PROJECTION | SCALE NTS | SHEET 3 OF 5

D | DRAWING NO. 1860-0052 | LATEST REVISION A

2-15-07

ACAD2007 FILE 18600052-S3

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	



8. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

7. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

6. ONLY REQUIRED WHEN REPLACING 1903-1700-02 (OBSOLETE GATEWAY ASSEMBLY) PLUG GENDER CHANGER INTO J7, THEN EXISTING CUSTOMER CABLE INTO CHANGER.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMINATE TO CHASSIS AT THIS POINT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON GATEWAY MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1860-0001 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES
UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS .XX ±.01; .XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-25-06
CHECKED		
DESIGNED	DCH	10-25-06
ENGINEERED		
PRODUCTION		
Q.A.		

PROJECTION

USED ON

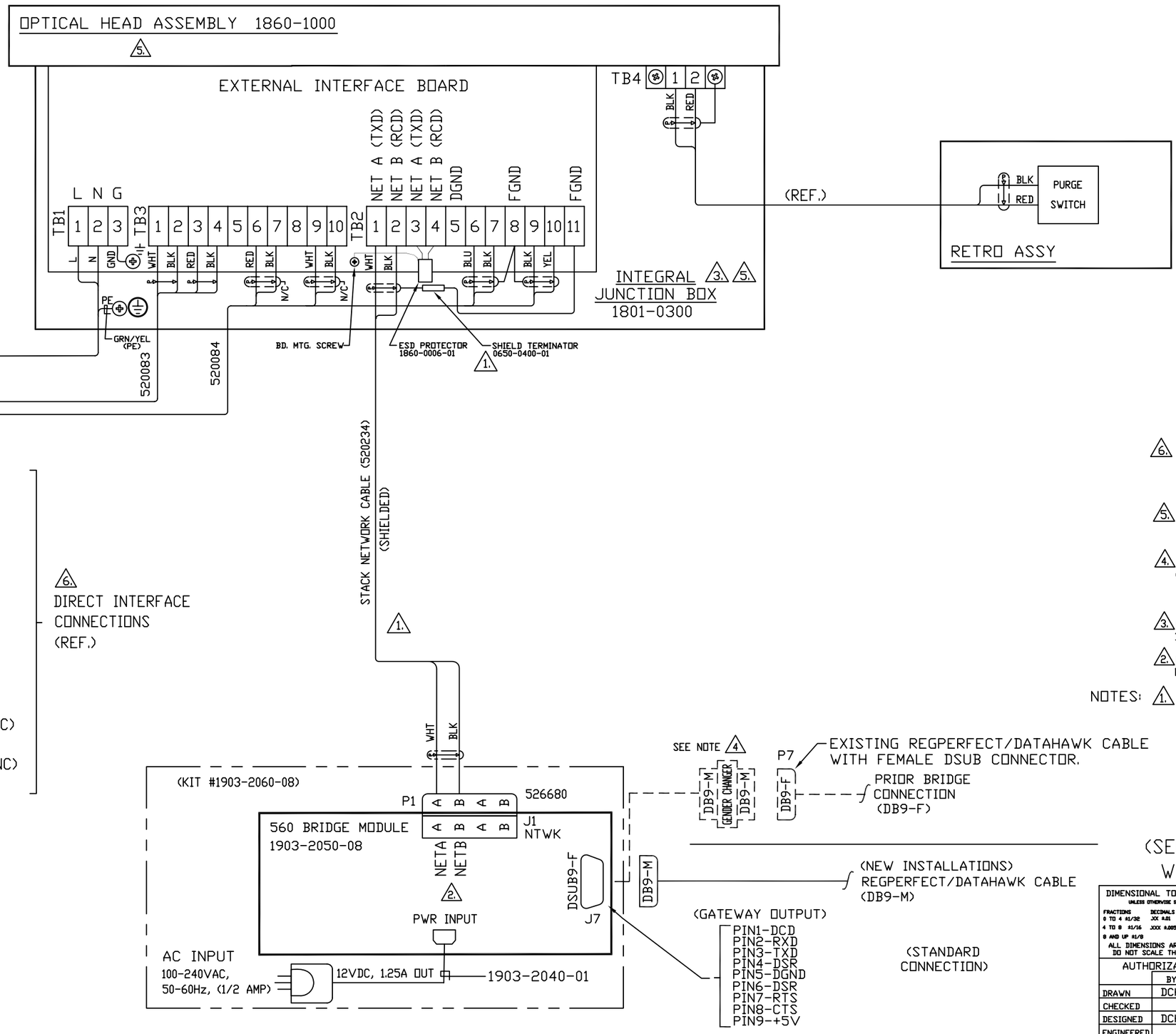
DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LIGHTHAWK 560 W/ERP AND GATEWAY MODULE, SUPPLEMENTAL WIRING DIAGRAM

MAT'L.
FINISH
JIG NO.
SCALE NTS
SHEET 4 OF 5
DRAWING NO. 1860-0052
LATEST REVISION A
2-15-07

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	



6. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

5. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

4. ONLY REQUIRED WHEN REPLACING 1860-2700-01 (OBSOLETE GATEWAY ASSEMBLY) PLUG GENDER CHANGER INTO J7, THEN EXISTING REGPERFECT/DATAHAWK CABLE INTO GENDER CHANGER.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON BRIDGE MODULE ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1860-0001 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32 — 4 TO 8 ±1/16 — 8 AND UP ±1/8
DECIMALS XX ±.01 — XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-25-06
CHECKED		
DESIGNED	DCH	10-25-06
ENGINEERED		
PRODUCTION		
Q.A.		

PROJECTION

USED ON

DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LIGHTHAWK 560 DIRECT INTFC. WITH REG PERFECT BRIDGE, SUPPLEMENTAL WIRING DIAGRAM

MAT'L. FINISH

JIG NO.

SCALE NTS | SHEET 5 OF 5

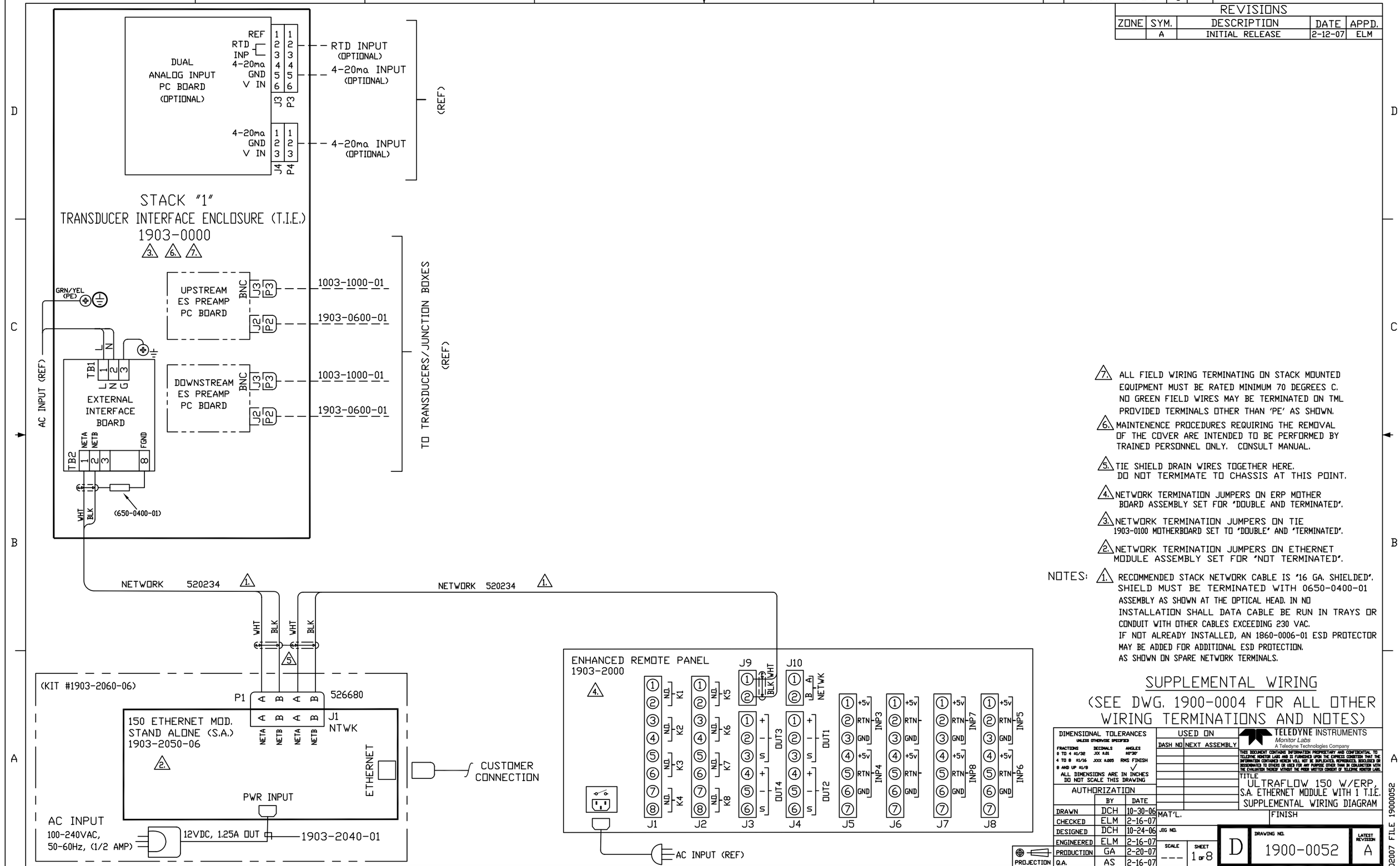
D | DRAWING NO. 1860-0052 | LATEST REVISION A

ACAD2007 FILE 18600052-S5

2-15-07

DWG. NO 1900-0052 SH 1

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-12-07	ELM



7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

6. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'DOUBLE AND TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0004 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS XX ±.01; XXX ±.005
ANGLES ±0°30'
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY

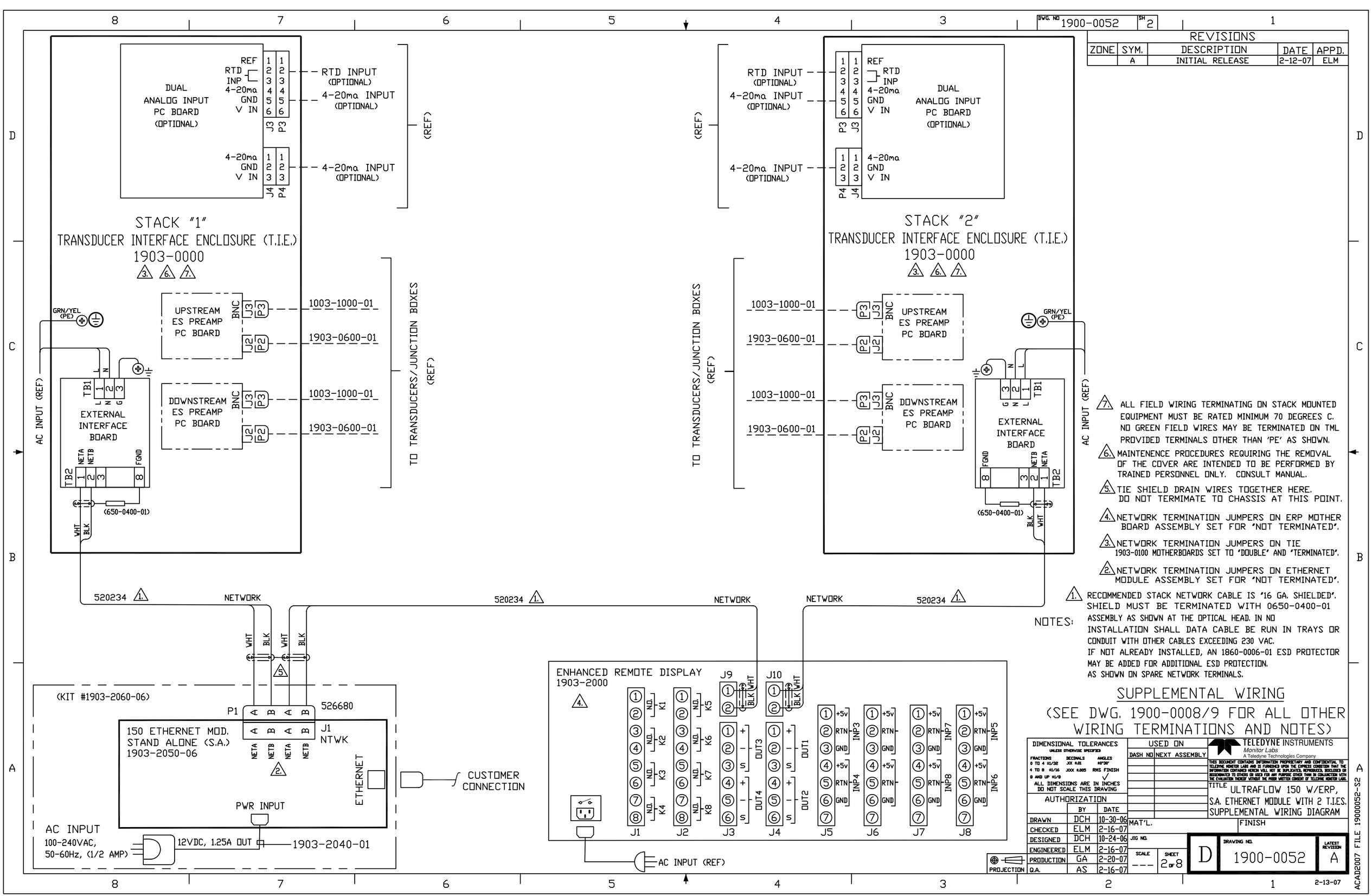
TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
ULTRAFLOW 150 W/ERP, S.A. ETHERNET MODULE WITH 1 T.I.E.
SUPPLEMENTAL WIRING DIAGRAM

MAT'L. FINISH
JIG NO.
SCALE ---
SHEET 1 OF 8
DRAWING NO. 1900-0052
LATEST REVISION A
SIZE D
PROJECTION
2-12-07
ACAD2007 FILE 19000052

DWG. NO 1900-0052 SH 2

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-12-07	ELM



7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

6. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR "NOT TERMINATED".

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARDS SET TO "DOUBLE" AND "TERMINATED".

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR "NOT TERMINATED".

1. RECOMMENDED STACK NETWORK CABLE IS "16 GA. SHIELDED". SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC.
IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

NOTES:

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0008/9 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES
UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS XX ±.01; XXX ±.005
ANGLES ±0°30'
RMS FINISH ✓
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON
DASH NO | NEXT ASSEMBLY

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE: ULTRAFLOW 150 W/ERP, S.A. ETHERNET MODULE WITH 2 T.I.E.S. SUPPLEMENTAL WIRING DIAGRAM

MAT'L. | FINISH | JIG NO. | SCALE --- | SHEET 2 OF 8

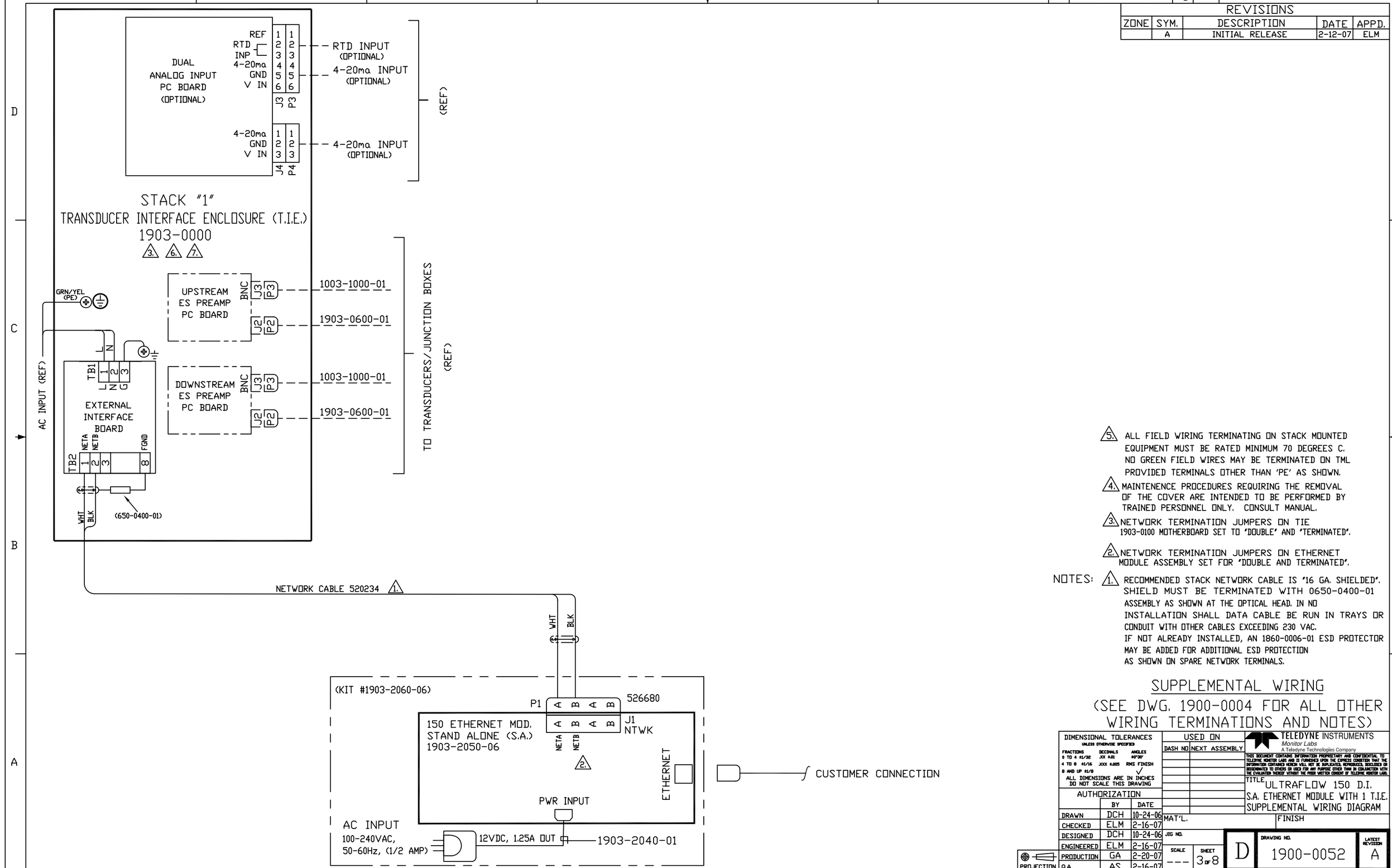
PROJECTION

DRAWING NO. 1900-0052 | D | LATEST REVISION A

2-13-07

ACAD2007 FILE 1900052-S2

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-12-07	ELM



5. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

4. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'DOUBLE AND TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0004 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS XX ±.01; XXX ±.005
ANGLES ±0°30'
RMS FINISH ✓
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-24-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS Monitor Labs
A Teledyne Technologies Company

TITLE: ULTRAFLOW 150 D.I. S.A. ETHERNET MODULE WITH 1 T.I.E. SUPPLEMENTAL WIRING DIAGRAM

MAT'L. | FINISH | JIG NO.

SCALE ---
SHEET 3 OF 8
PROJECTION

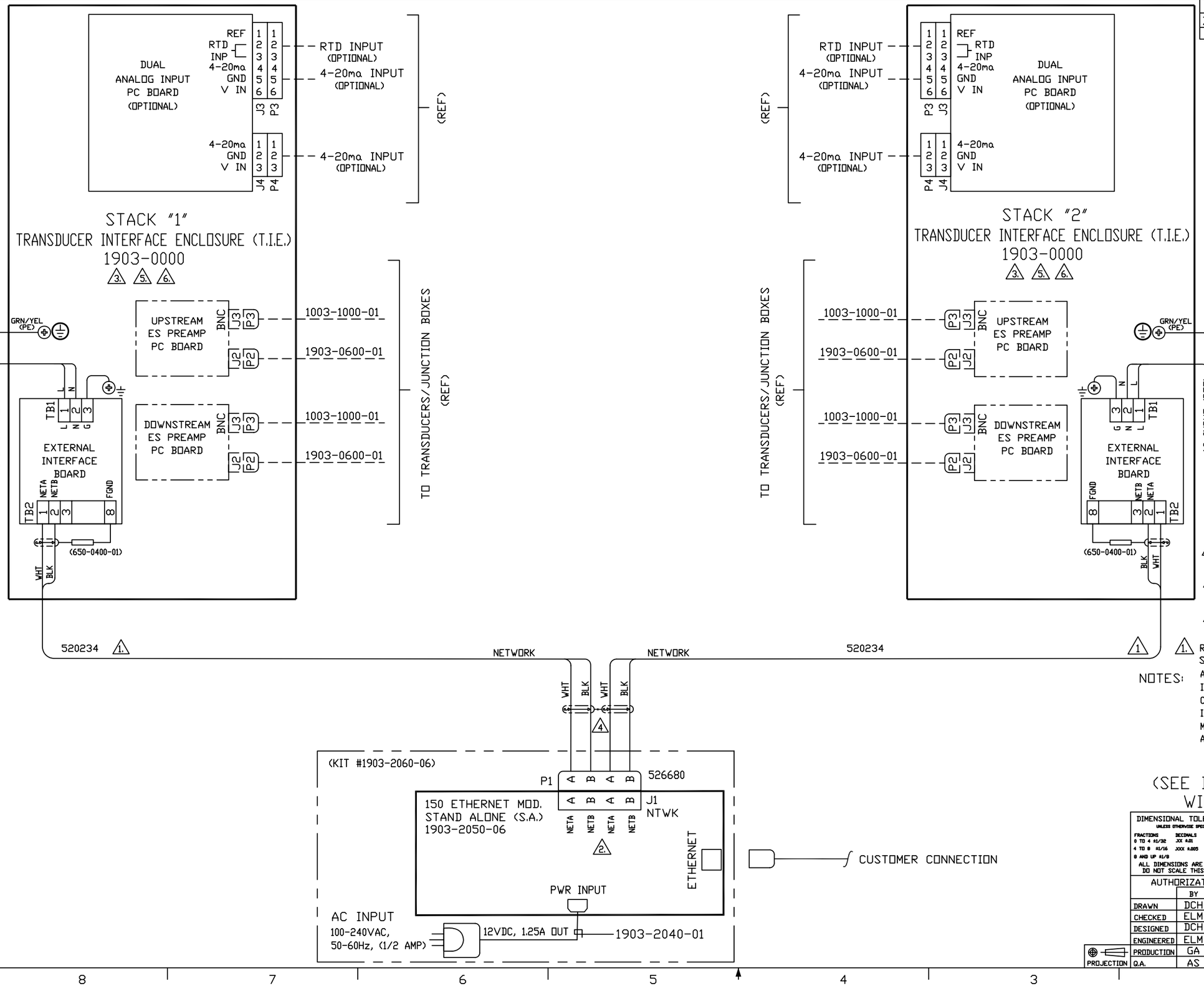
DRAWING NO. 1900-0052
LATEST REVISION A
SIZE D

2-12-07

ACAD2007 FILE 19000052-s3

DWG. NO 1900-0052 SH 4

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-12-07	ELM



6. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

5. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

4. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.

3. NETWORK TERMINATION JUMPERS ON BOTH TIE 1903-0100 MOTHERBOARDS SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPER ON ETHERNET MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

NOTES:

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0008/9 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS .XX ±.01; .XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

PROJECTION

USED ON

DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS Monitor Labs
A Teledyne Technologies Company

TITLE ULTRAFLOW 150 D.I.
S.A. ETHERNET MODULE W/2 T.I.E.
SUPPLEMENTAL WIRING DIAGRAM

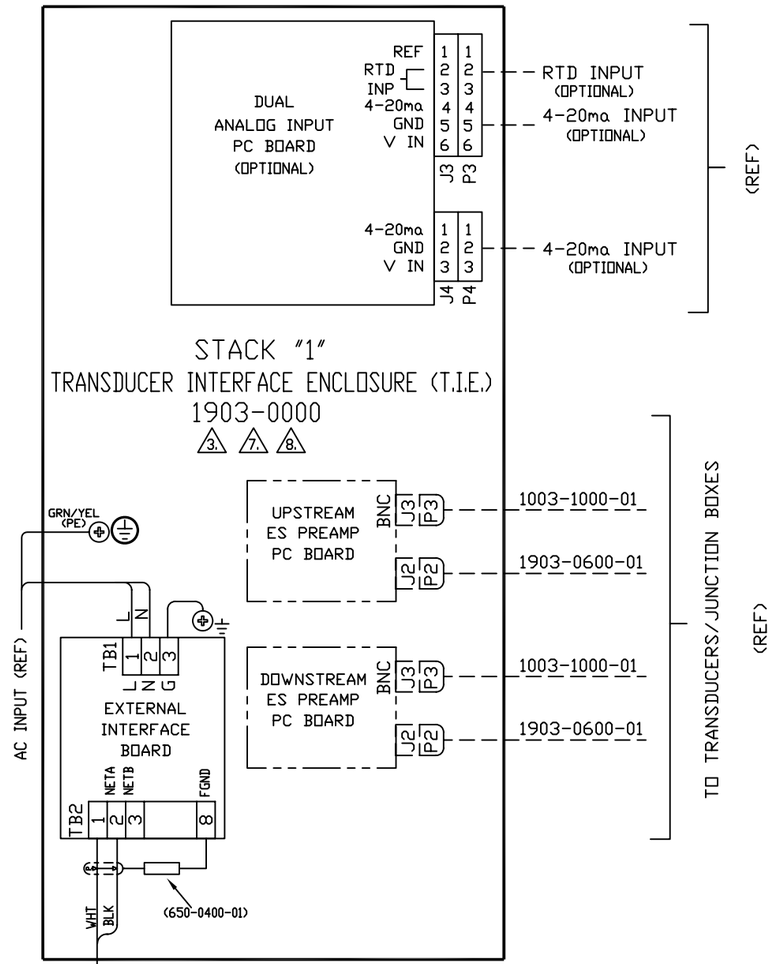
MAT'L.
FINISH
JIG NO.
SCALE ---
SHEET 4 OF 8

D | DRAWING NO. 1900-0052 | LATEST REVISION A

2-13-07

ACAD2007 FILE 1900052-s4

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-12-07	ELM



8. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

6. ONLY REQUIRED WHEN REPLACING 1903-1700-01 (OBSOLETE GATEWAY ASSEMBLY) PLUG GENDER CHANGER INTO J7, THEN EXISTING CUSTOMER CABLE INTO ADAPTER.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS PT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'DOUBLE AND TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON GATEWAY MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0004 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES
UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32, 4 TO 8 ±1/16, 8 AND UP ±1/8
DECIMALS .XX ±.01, .XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION

	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
ULTRAFLOW 150 W/ERP,
GATEWAY MODULE WITH 1 T.I.E.
SUPPLEMENTAL WIRING DIAGRAM

MAT'L.
FINISH
JIG NO.
SCALE ---
SHEET 5 OF 8

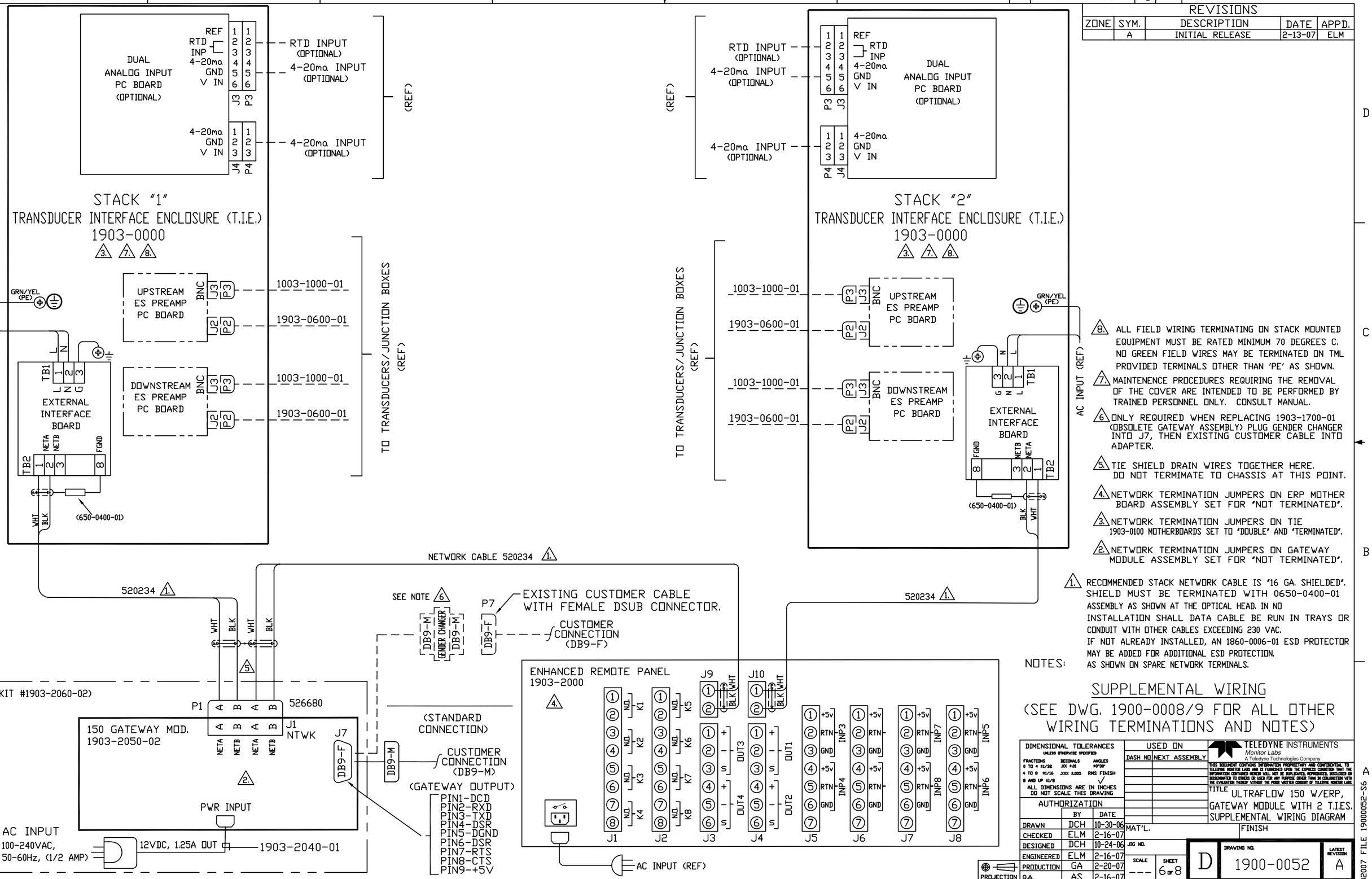
PROJECTION	DRAWING NO.	LATEST REVISION
	D 1900-0052	A

2-12-07

ACAD2007 FILE 19000052-S5

DWG. NO 1900-0052 SH 6

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-13-07	ELM



8. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

7. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

6. ONLY REQUIRED WHEN REPLACING 1903-1700-01 (OBSOLETE GATEWAY ASSEMBLY) PLUG GENDER CHANGER INTO J7, THEN EXISTING CUSTOMER CABLE INTO ADAPTER.

5. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'NOT TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARDS SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON GATEWAY MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

NOTES:

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0008/9 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS .XX ±.01; .XXX ±.005
ANGLES ±0°30'
RMS FINISH
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION

	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE ULTRAFLOW 150 W/ERP, GATEWAY MODULE WITH 2 T.I.E.S. SUPPLEMENTAL WIRING DIAGRAM

MAT'L. FINISH

JIG NO.

SCALE ---

SHEET 6 OF 8

DRAWING NO. 1900-0052

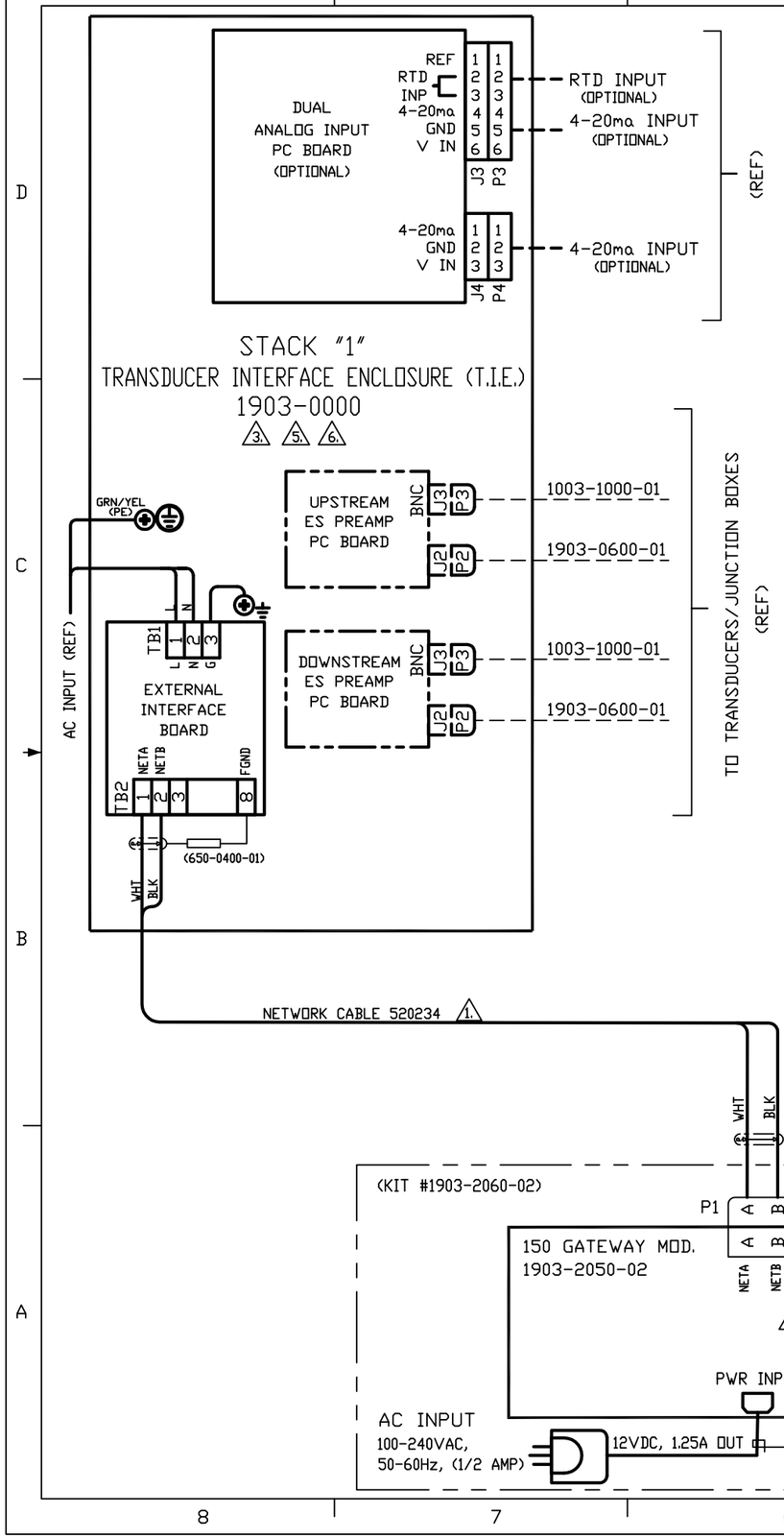
LATEST REVISION A

PROJECTION

2-12-07

ACAD2007 FILE 1900052-S6

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-13-07	ELM



6. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

5. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

4. ONLY REQUIRED WHEN REPLACING 1903-1700-01 (OBSOLETE GATEWAY ASSEMBLY) PLUG GENDER CHANGER INTO J7, THEN EXISTING CUSTOMER CABLE INTO ADAPTER.

3. NETWORK TERMINATION JUMPERS ON TIE 1903-0100 MOTHERBOARD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'DOUBLE AND TERMINATED'.

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

NOTES:

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0004 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32, 4 TO 8 ±1/16, 8 AND UP ±1/8
DECIMALS .XX ±.01, .XXX ±.005
ANGLES ±0°30'
RMS FINISH √
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-24-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

PROJECTION

USED ON

DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE ULTRAFLOW 150 D.I. GATEWAY MODULE WITH 1 T.I.E. SUPPLEMENTAL WIRING DIAGRAM

MAT'L. FINISH

JIG NO.

SCALE ---

SHEET 7 OF 8

DRAWING NO. 1900-0052

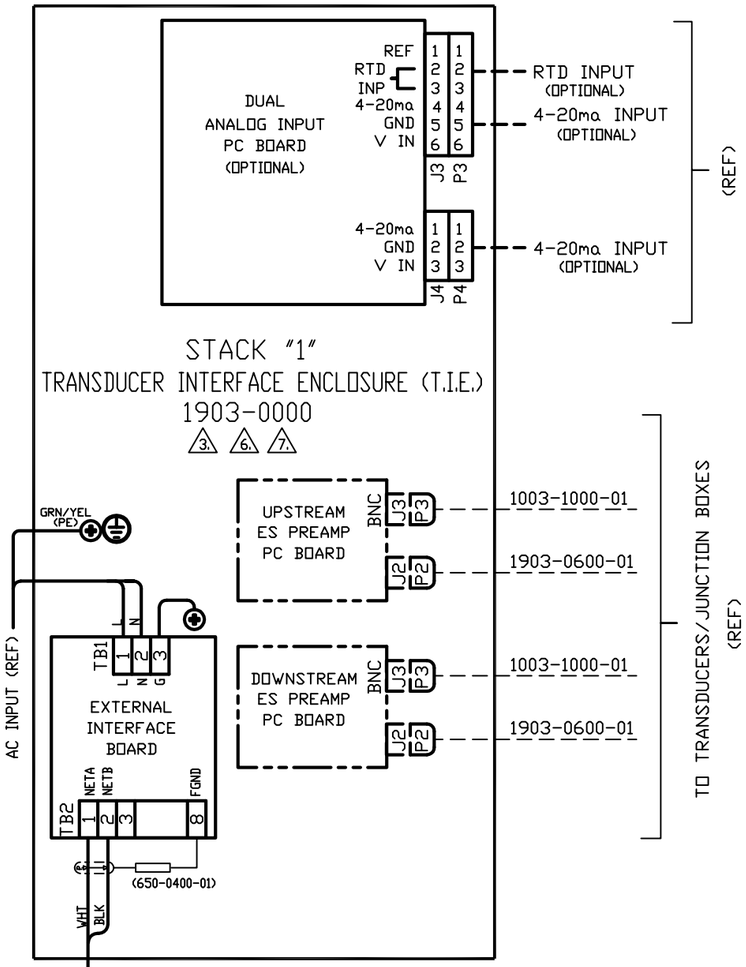
LATEST REVISION A

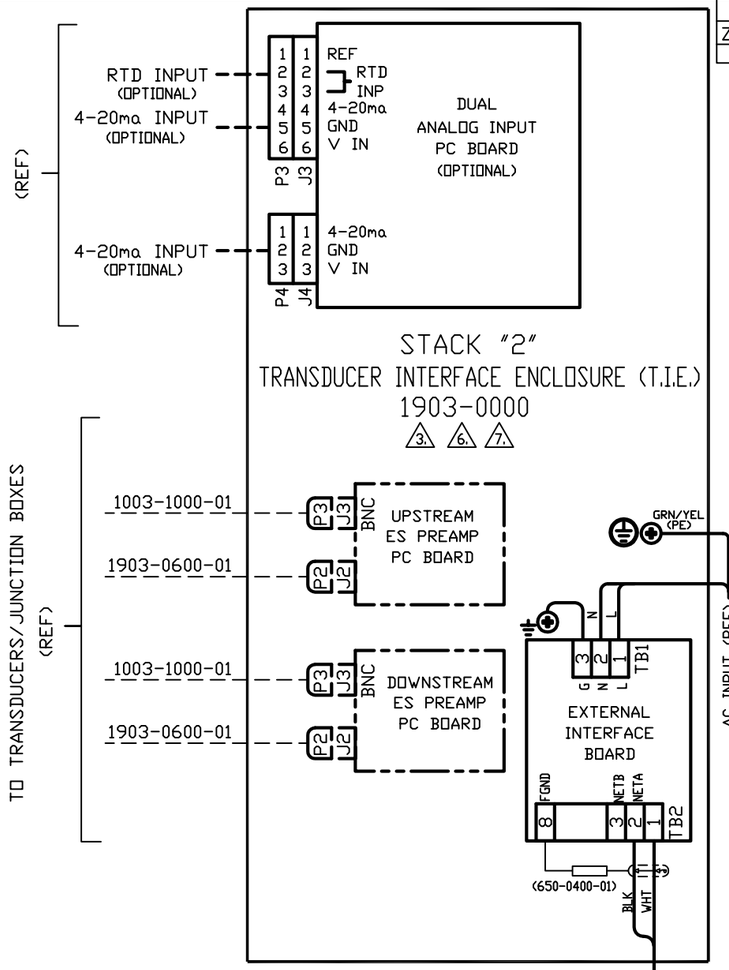
D

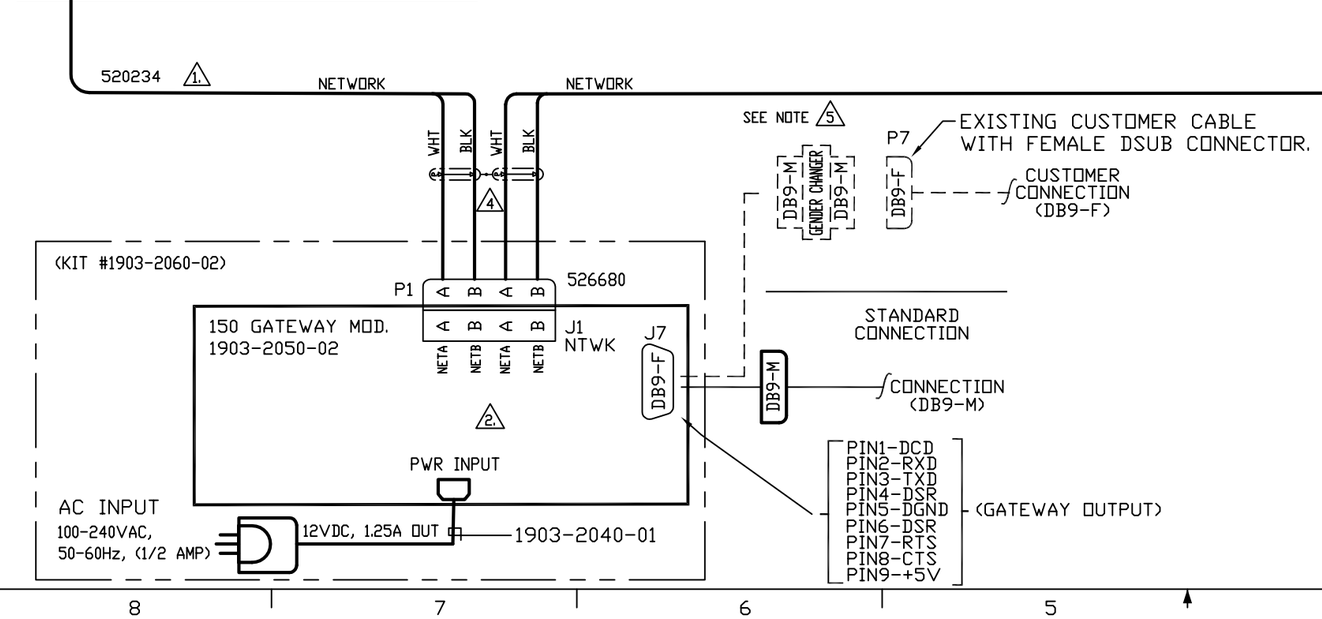
2-13-07

ACAD2007 FILE 1900052-s7

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-13-07	ELM







NOTES:

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 ASSEMBLY AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION.
2. NETWORK TERMINATION JUMPER ON ETHERNET MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.
3. NETWORK TERMINATION JUMPERS ON BOTH TIE 1903-0100 MOTHERBOARDS SET TO 'DOUBLE' AND 'TERMINATED'.
4. TIE SHIELD DRAIN WIRES TOGETHER HERE. DO NOT TERMIMATE TO CHASSIS AT THIS POINT.
5. ONLY REQUIRED WHEN REPLACING 1903-1700-01 (OBSOLETE GATEWAY ASSEMBLY) PLUG ADAPTER INTO J7, THEN EXISTING CUSTOMER CABLE INTO ADAPTER.
6. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.
7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

SUPPLEMENTAL WIRING

(SEE DWG. 1900-0008/9 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED			
FRACTIONS	DECIMALS	ANGLES	
0 TO 4 ±1/32	.XX ±.01	±0°30'	
4 TO 8 ±1/16	.XXX ±.005	RMS FINISH	
8 AND UP ±1/8			
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING			

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-30-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-24-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON	
DASH NO	NEXT ASSEMBLY

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
ULTRAFLOW 150 D.I.
GATEWAY MODULE W/2 T.I.E.
SUPPLEMENTAL WIRING DIAGRAM

MAT'L.
FINISH
JIG NO.
SCALE ---
SHEET 8 OF 8
PROJECTION

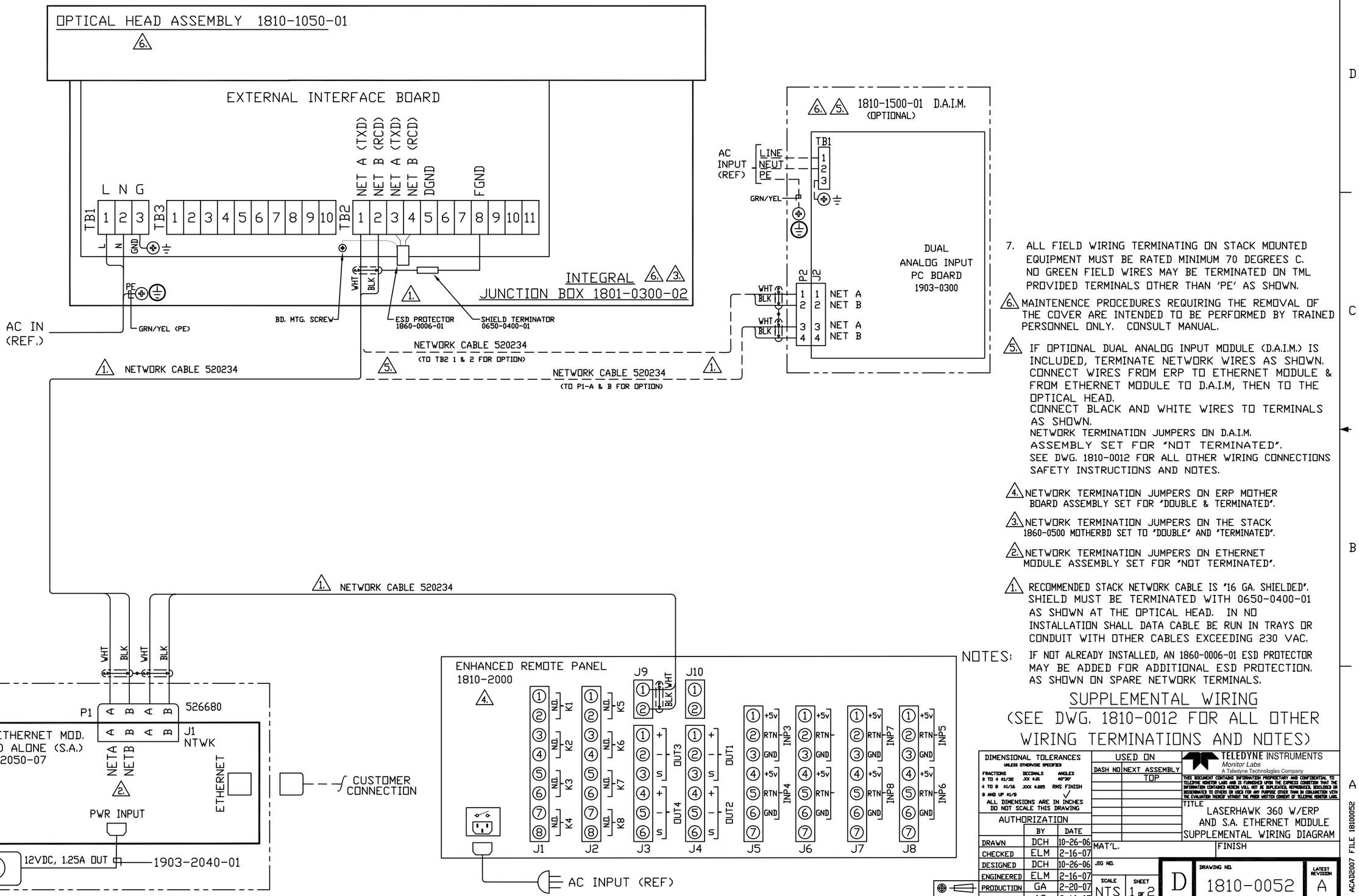
DRAWING NO.	LATEST REVISION
1900-0052	A

2-13-07

ACAD2007 FILE 19000052-s8

DWG. NO	SH
1810-0052	1

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	ELM



7. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

6. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

5. IF OPTIONAL DUAL ANALOG INPUT MODULE (D.A.I.M.) IS INCLUDED, TERMINATE NETWORK WIRES AS SHOWN. CONNECT WIRES FROM ERP TO ETHERNET MODULE & FROM ETHERNET MODULE TO D.A.I.M, THEN TO THE OPTICAL HEAD.
CONNECT BLACK AND WHITE WIRES TO TERMINALS AS SHOWN.
NETWORK TERMINATION JUMPERS ON D.A.I.M. ASSEMBLY SET FOR "NOT TERMINATED".
SEE DWG. 1810-0012 FOR ALL OTHER WIRING CONNECTIONS SAFETY INSTRUCTIONS AND NOTES.

4. NETWORK TERMINATION JUMPERS ON ERP MOTHER BOARD ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'NOT TERMINATED'.

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC.

NOTES: IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

SUPPLEMENTAL WIRING
(SEE DWG. 1810-0012 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED			
FRACTIONS	DECIMALS	ANGLES	
0 TO 4 ±1/32	.XX ±.01	±0°30'	
4 TO 8 ±1/16	.XXX ±.005	RMS FINISH	
8 AND UP ±1/8			
ALL DIMENSIONS ARE IN INCHES DO NOT SCALE THIS DRAWING			

AUTHORIZATION	BY	DATE
DRAWN	DCH	10-26-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-26-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON	
DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LASERHAWK 360 W/ERP AND S.A. ETHERNET MODULE SUPPLEMENTAL WIRING DIAGRAM

MAT'L. | FINISH | JIG NO.

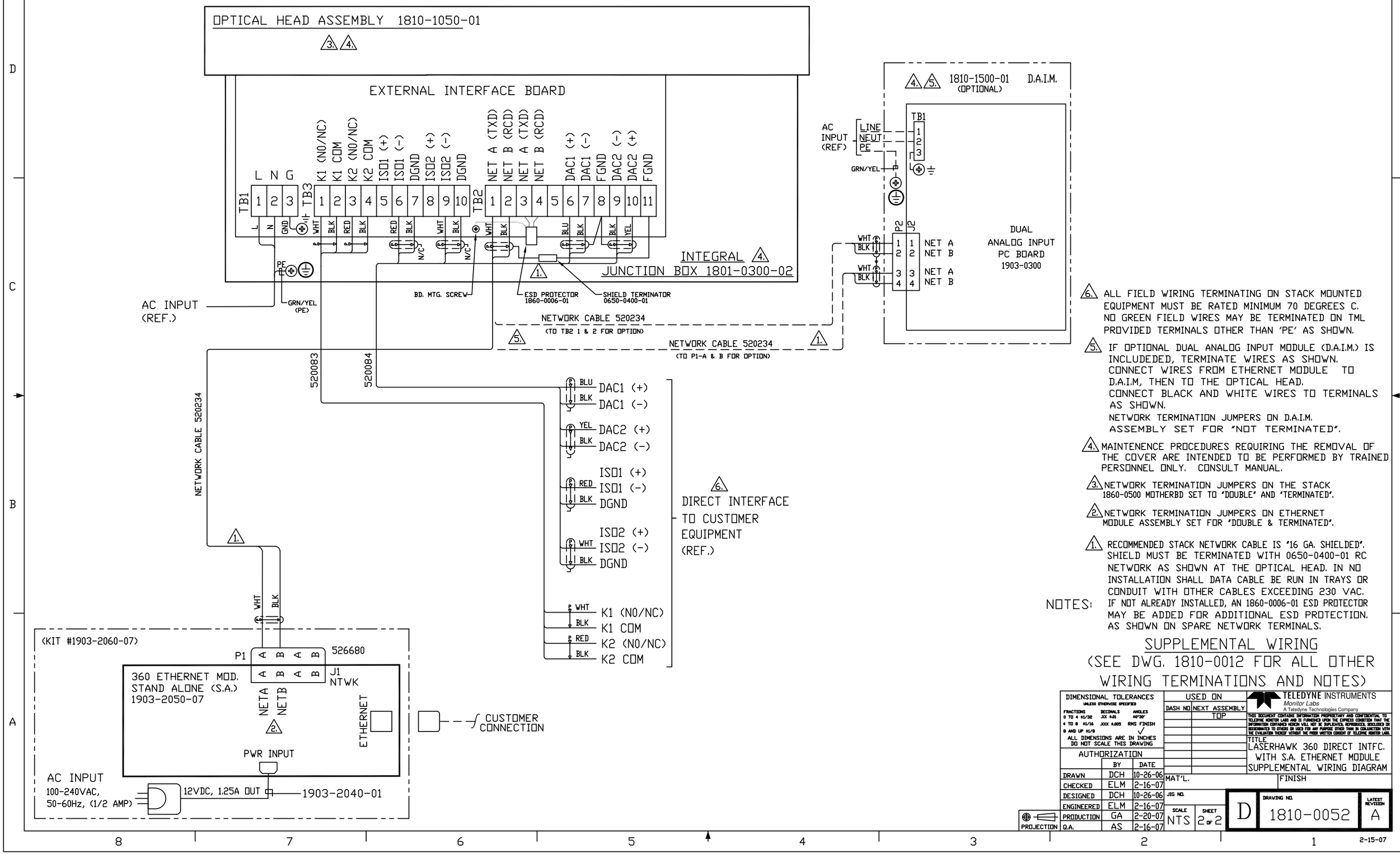
PROJECTION | SCALE NTS | SHEET 1 OF 2 | D | DRAWING NO. 1810-0052 | LATEST REVISION A

ACAD2007 FILE 18100052

2-15-07

DWG. NO 1810-0052 | SH 2

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	INITIAL RELEASE	2-15-07	ELM



6. ALL FIELD WIRING TERMINATING ON STACK MOUNTED EQUIPMENT MUST BE RATED MINIMUM 70 DEGREES C. NO GREEN FIELD WIRES MAY BE TERMINATED ON TML PROVIDED TERMINALS OTHER THAN 'PE' AS SHOWN.

5. IF OPTIONAL DUAL ANALOG INPUT MODULE (D.A.I.M.) IS INCLUDEDED, TERMINATE WIRES AS SHOWN. CONNECT WIRES FROM ETHERNET MODULE TO D.A.I.M, THEN TO THE OPTICAL HEAD. CONNECT BLACK AND WHITE WIRES TO TERMINALS AS SHOWN. NETWORK TERMINATION JUMPERS ON D.A.I.M. ASSEMBLY SET FOR "NOT TERMINATED".

4. MAINTENENCE PROCEDURES REQUIRING THE REMOVAL OF THE COVER ARE INTENDED TO BE PERFORMED BY TRAINED PERSONNEL ONLY. CONSULT MANUAL.

3. NETWORK TERMINATION JUMPERS ON THE STACK 1860-0500 MOTHERBD SET TO 'DOUBLE' AND 'TERMINATED'.

2. NETWORK TERMINATION JUMPERS ON ETHERNET MODULE ASSEMBLY SET FOR 'DOUBLE & TERMINATED'.

1. RECOMMENDED STACK NETWORK CABLE IS '16 GA. SHIELDED'. SHIELD MUST BE TERMINATED WITH 0650-0400-01 RC NETWORK AS SHOWN AT THE OPTICAL HEAD. IN NO INSTALLATION SHALL DATA CABLE BE RUN IN TRAYS OR CONDUIT WITH OTHER CABLES EXCEEDING 230 VAC. IF NOT ALREADY INSTALLED, AN 1860-0006-01 ESD PROTECTOR MAY BE ADDED FOR ADDITIONAL ESD PROTECTION. AS SHOWN ON SPARE NETWORK TERMINALS.

NOTES:

SUPPLEMENTAL WIRING

(SEE DWG. 1810-0012 FOR ALL OTHER WIRING TERMINATIONS AND NOTES)

DIMENSIONAL TOLERANCES UNLESS OTHERWISE SPECIFIED
FRACTIONS 0 TO 4 ±1/32; 4 TO 8 ±1/16; 8 AND UP ±1/8
DECIMALS .XX ±.01; .XXX ±.005
ANGLES ±0°30'
RMS FINISH ✓
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION

	BY	DATE
DRAWN	DCH	10-26-06
CHECKED	ELM	2-16-07
DESIGNED	DCH	10-26-06
ENGINEERED	ELM	2-16-07
PRODUCTION	GA	2-20-07
Q.A.	AS	2-16-07

USED ON

DASH NO	NEXT ASSEMBLY
	TOP

TELEDYNE INSTRUMENTS
Monitor Labs
A Teledyne Technologies Company

TITLE
LASERHAWK 360 DIRECT INTFC. WITH S.A. ETHERNET MODULE
SUPPLEMENTAL WIRING DIAGRAM

MAT'L. | FINISH | JIG NO.

SCALE NTS | SHEET 2 OF 2

DRAWING NO. 1810-0052 | LATEST REVISION A

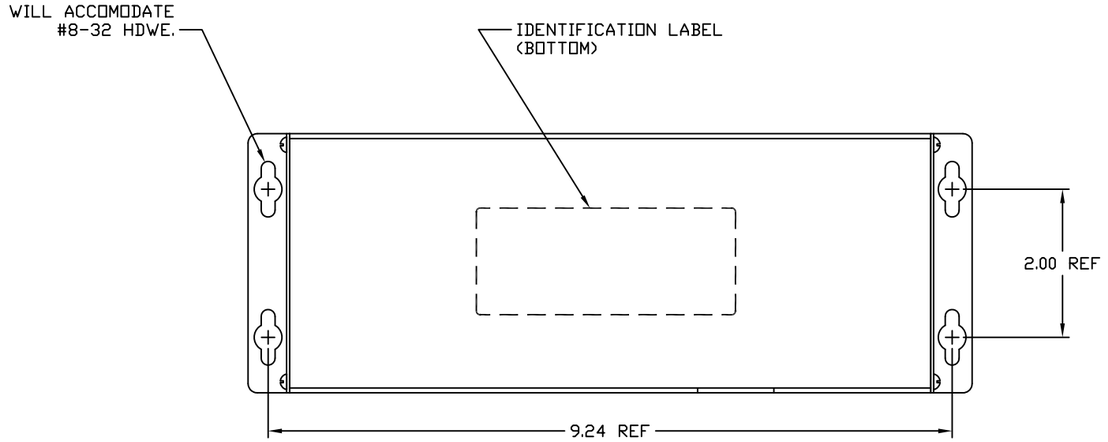
PROJECTION | SIZE D

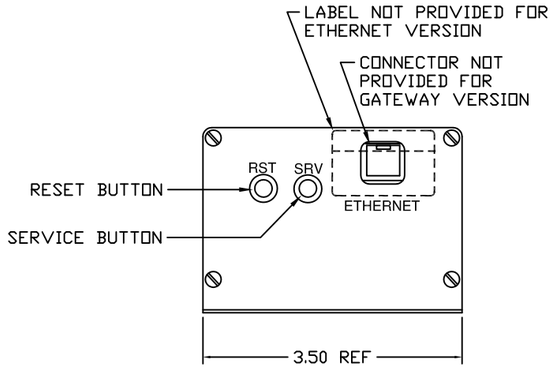
2-15-07

ACAD2007 FILE 18100052-S2

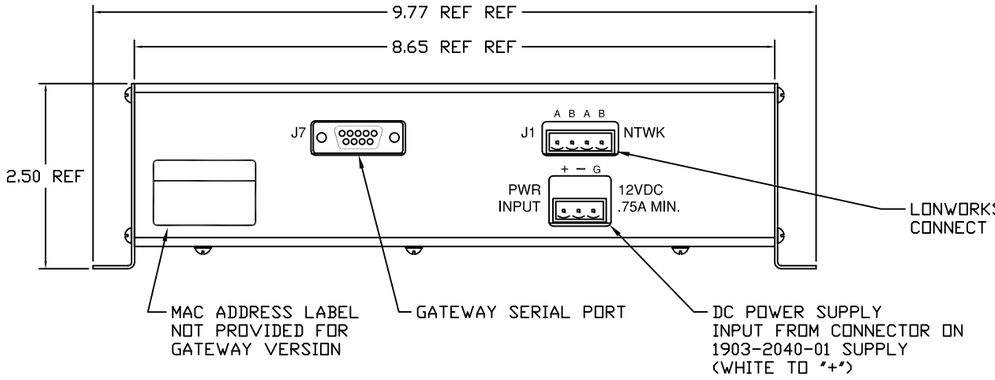
DWG. NO 1903-2060 SH 1

REVISIONS				
ZONE	SYM.	DESCRIPTION	DATE	APPD.
	A	RELEASED FOR PRODUCTION	10-11-06	ELM
	B	NO CHANGE THIS SHEET	11-21-07	ELM
	C	NO CHANGE THIS SHEET	3-18-08	DMB

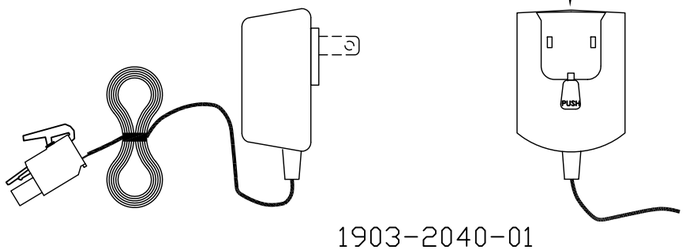




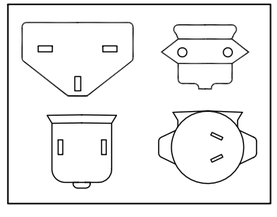
END VIEW



FRONT VIEW
1903-2050-XX
STAND ALONE ETHERNET OR GATEWAY MODULE



1903-2040-01
PLUG-IN WALL MOUNT POWER SUPPLY
100-240vac, 50-60Hz/12vdc, 1250ma
(115VAC PLUG SHOWN)



523887
UNIVERSAL AC PLUG KIT. SLIDE
PLUG FOR DESIRED SERVICE ONTO 1903-2040-01
SUPPLY AT LEFT.

P1

526680
P1 (MATING CONNECTOR FOR J1)

DIMENSIONAL TOLERANCES
UNLESS OTHERWISE SPECIFIED
FRACTIONS ±1/32 DECIMALS .XX ±.01 ANGLES ±0°30'
4 TO 8 ±1/16 .XXX ±.005 RMS FINISH
8 AND UP ±1/8
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE THIS DRAWING

AUTHORIZATION	BY	DATE
DRAWN	DCH	9-29-06
CHECKED	ELM	11-27-06
DESIGNED	DCH	9-29-06
ENGINEERED	ELM	11-27-06
PRODUCTION	G.A	11-27-06
Q.A.	A.S.	11-27-06

PROJECTION

USED ON	
DASH NO	NEXT ASSEMBLY
-01	560 GATEWAY
-02	150 GATEWAY
-04	560 BRIDGE
-05	560 ETHERNET
-06	150 ETHERNET
-07	360 ETHERNET
-08	560 8B GATWY

MAT'L. FINISH

JIG NO.

SCALE 1:1 SHEET 1 OF 1

TELEDYNE MONITOR LABS
A Teledyne Technologies Company

TITLE
CUSTOMER DRAWING,
1903-2050-xx STAND ALONE
ETHERNET/GATEWAY MODULE

DRAWING NO. 1903-2060

LATEST REVISION C

D

3-18-08

ACAD2007 FILE 19032060 A

(This page intentionally left blank.)

APPENDIX B

LIGHTHAWK 560 VARIABLES

(This page intentionally left blank.)

This section describes the variables for the Lighthawk 560 instrument. A description of each network variable is given, followed by the C/C++ definitions of the structures, constants and variable types. The Modbus register mapping is also given.

All network variables are prefixed with a three letter prefix, either “nvi” or “nvo” which translates to:

nvi – Network Variable Input;
nvo – Network Variable Output.

The direction of Input or Output is relative to the Ethernet Module.

Instrument Status Word Interpretation via Modbus TCP

On the LightHawk, the nviHeadState variable signifies instrument status. See the description for nviHeadState in the “Modbus Input Registers for LightHawk 560” table for more details.

Initiating Calibration Commands via Modbus TCP

On the LightHawk, the nvoModeRequest variable is used to command calibration on the analyzer. See the description for nvoModeRequest in the following tables, the header files for the LightHawk and the “Propagating Values On The Network” section for more details on command initiation.

nvi332Version	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make 68332 processor application code version of the Optical Head available as a network variable. Date shows date of release, hours and minutes show version. The second field is ignored and should be set to zero to avoid confusion when using browsers or other network tools. For example 2/28/1999, 2:02:00 represents Version 2.02 which was released on 2/28/1999.
nviNeuronVer	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make NEURON application code version of the Optical Head available as a network variable. See nvi332Version.
nviMIO_Version	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make NEURON application code version of the Multi I/O available as a network variable. See nvi332Version.
nviTime	Type: SNVT_time_stamp
	Units: Time and date.
	Contains the current value of the System Clock.

nviAlarms	Type: SNVT_alarm
	Units: Application Dependent
	This network variable is issued by the Optical Head once whenever an alarm changes state. The type, object, value and time stamp of the alarm are propagated. The Display Panel device accumulates an alarm history based on issuance events of this variable.
nviHeadState	Type: UNVT_HeadStatus
	Units: Dimensionless
	Represents the Optical Head status. The codes represent detailed instrument conditions which OR together.
nviHeadMode	Type: UNVT_HeadMode
	Units: Dimensionless
	This variable defines the mode of operation of various elements of the Optical Head. The AuditMode and FilterValue members are used collaboratively to describe the filter audit data.
nviMIO_Status	Type: UNVT_MIO_status
	Units: Dimensionless
	This network variable communicates the analog output mode (MIO_mode) and the error state (invalid_mode_request, invalid_AO_config and invalid_DIO_config) of the Multi I/O device. Errors exist when the bit type members are one. For normal operation they are zero. MIO_mode is an enumerated member with values such as SPAN, ZERO, NORMAL, etc.
nviOpacityAlarms, nviOptDensAlarms, nviMassAlarms	Type: UNVT_AlarmSel
	Units: Dimensionless
	These network variables represent the states of Instantaneous, One Minute and Selectable Average two level alarms for Opacity, Optical Density and Dust Mass. If a bit equals one, the alarm condition is in effect. These network variables are propagated every time the alarm condition is evaluated.
nviInstOpacity, nviInstOptDens, nviInstMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the instantaneous measurements.
nvi1MinOpacity, nvi1MinOptDens, nvi1MinMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the one minute average measurements.
nviSelAveOpacity, nviAveOptDens,	Type: Float
	Units: Application Dependent

nviAveMassLoad	These variables are the values for the selectable average measurements.
nviZeroOpacCal, nviZeroOptDens, nviZeroMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Zero Calibration Average measurements.
nviSpanOpacCal, nviSpanOptDens, nviSpanMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Span Calibration Average measurements.
nviDirtCompValue, nviDirtOptDens, nviDirtMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Dust Compensation Average measurements.
nviServiceData1	Type: UNVT_Service_One
	Units: Voltages and currents except for HeadTemperature, which is in degrees Centigrade, and CalWheelPosition.
	Represents internal instrument data of interest to service technicians more than average users.
nviServiceData2	Type: UNVT_Service_Two
	Units: Voltages.
	Represents internal power supply voltages of interest to service technicians more than average users.
nviAuditData	Type: UNVT_AuditData
	Units: AuditMode and FilterValue are dimensionless. AuditOpacity is in percent. Optical Density is dimensionless. Dust Mass is mg/m^3.
	This variable is issued via user intervention during filter audits. The current Selectable Average values for Opacity, Optical Density and Dust Mass are propagated as the audit values. The AuditMode and FilterValue members are used collaboratively to describe the filter audit data.
nviHeadConfig_1, nvoHeadConfig_1	Type: UNVT_HeadConfig1
	Units: Percent opacity for all.
	This variable defines emission alarm levels for Instantaneous Opacity, One Minute Opacity and Selectable Average Opacity. It also sets the Dust Compensation Alarm limit. The DirtCompAlarm member will not accept a value greater than 4.00 in order to comply with USEPA 40CFR60 Appendix B PS1.
nviHeadConfig_2, nvoHeadConfig_2	Type: UNVT_HeadConfig2
	Units: PLCF is dimensionless; ZeroCalSetpt, SpanCalSetpt and CalDelta are in percent opacity; and AveSendTime is in minutes.

	This variable defines the current PLCF and is mapped to PLCF(Current Value). The PLCF(Current Value) is what the Optical Head uses for exit correction. The ZeroCalSetpt defines the expected ZERO calibration value, SpanCalSetpt defines the expected UPSCALE calibration value and CalDelta defines the tolerance band around them before a calibration failure is presented. The AveSendTime defines the averaging interval for Selectable Average Opacity, Optical Density and Dust Mass.
nviHeadConfig_3, nvoHeadConfig_3	Type: UNVT_HeadConfig3
	Units: Dimensionless for all.
	This variable defines emission alarm levels for Instantaneous Optical Density, One Minute Optical Density and Selectable Average Optical Density.
nviHeadConfig_4, nvoHeadConfig_4	Type: UNVT_HeadConfig4
	Units: X values are dimensionless Optical Density; Y values are in mg/m^3.
	This variable defines three points of an interpolation curve for conversion from Optical Density to Dust Mass. The fourth point is always at the origin (0, 0).
nviHeadConfig_5, nvoHeadConfig_5	Type: UNVT_HeadConfig5
	Units: mg/m^3 for all.
	This variable defines emission alarm levels for Instantaneous Dust Mass, One Minute Dust Mass and Selectable Average Dust Mass.
nviHeadConfig_6, nvoHeadConfig_6	Type: UNVT_HeadConfig6
	Units: Temp_Deg_C is in centigrade; AbsPressure is in Pascals (SI/MKS pressure unit).
	Defines the reference temperature and pressure for correction of Dust Mass to standard conditions. The pressure is absolute.
nviHeadConfig_7, nvoHeadConfig_7	Type: UNVT_HeadConfig7
	Units: Defines the times of automatic calibration cycles. The cal_time_hour and cal_interval members are in hours, cal_time_min is in minutes. All others are in seconds.

	cal_time_hour and cal_time_min define time of day for the automatic calibration cycle in the Optical Head. cal_interval_hour defines the interval between the first calibration defined by the prior members and the next automatic calibration. If cal_interval_hour equals 0 or a value of 24 or greater, calibration occurs only once a day. If cal_time_hour = 12, cal_time_min = 30 and cal_interval_hour = 4, cals occur at 1230 hours, 1630 hours and 2030 hours. After day rollover, no more occur until 1230 hours. The span_secs, zero_secs, etc. define the number of seconds in each part of the cal cycle. If a member is zero, that part of the cycle is skipped.
nviHeadConfig_8, nvoHeadConfig_8	Type: UNVT_HeadConfig8
	Units: Dimensionless. Valid ranges are currently 0 to 255 for all. The Optical Head and Network devices should accept values only in this range.
	These members define the gain of the Optical Head optical amplifiers. CommonGain is not used. These gain controls are currently implemented with 8 bit devices.
nviMIO_Config_1, nvoMIO_Config_1	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 1 (Analog Output 1) and DAC 2 (Analog Output 2) of the Multi I/O device.
nviMIO_Config_2, nvoMIO_Config_2	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 3 (Analog Output 3) and DAC 4 (Analog Output 4) of the Multi I/O device.
nviMIO_Config_3, nvoMIO_Config_3	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping for Relay 1 (K1) through Relay 8 (K8) of the Multi I/O device.
nviMIO_Status	Type: UNVT_MIO_status
	Units: Dimensionless.

	This network variable communicates the analog output mode (MIO_mode) and the error state (invalid_mode_request, invalid_AO_config and invalid_DIO_config) of the Multi I/O device. Errors exist when the bit type members are one. For normal operation they are zero. MIO_mode is an enumerated member with values such as SPAN, ZERO, NORMAL, etc.
nviRequest	Type: SNVT_obj_request
	Units: Dimensionless.
	This network variable is not used.
nviNetworkStatus	Type: UNVT_ModeRequest
	Units: Dimensionless.
	This internal variable is used by the Neuron side of the interface. It signals one of the following values in mode structure member: Network Normal = 0, No Stack Communication = 1, Transmission Errors = 2, Network Interface Fail = 3, Status Unknown = 4.
nviPanelVersion	Type: SNVT_time_stamp
	Units: Time and Date.
	Uses time and data format to make NEURON application code version of the Ethernet Module available as a network variable. See nvi332Version.
nvoTimeSet	Type: SNVT_time_stamp
	Units: Time and Date.
	The System Clock resides in the 560 Optical Head Node. This network variable sets the value of the System Clock.
nvoModeRequest	Type: UNVT_ModeRequest
	Units: Dimensionless.

	Used to transmit commands from the Remote Display node to the Optical Head and Multi I/O nodes. The node_id member is used to designate which node the command originated from: 1 represents the Enhanced Remote Panel/Ethernet Module and 2 represents the Multi I/O. The mode member is an enumerated data type that invokes the command. For example, if (mode = ZERO), the Optical Head will move the calibration mechanism to ZERO position and perform a calibration ZERO. TEST type commands are meant for the Multi I/O only. Support for these functions for the Six Point I/O analog outputs is via its own jumpers or the Optical Head keypad. **DESIRED MODE** **nvoModeRequest VALUE** UNKNOWN 0 NORMAL 1 ZERO 2 UPSCALE 3 PLCF 4 DIRT 5 FORCE_CAL_CYCLE 6 TEST_ZERO_SCALE 7 TEST_MID_SCALE 8 TEST_FULL_SCALE 9 PUT_IN_SERVICE 10 OUT_OF_SERVICE 11
nvoDataRequest	Type: UNVT_ModeRequest
	Units: Dimensionless.
	This variable is not truly a network variable. It is used to instruct the NEURON side of the Ethernet Module to fetch (poll) a particular variable from the network, since not all variables mentioned here are continuously broadcast. Both structure members node_id and mode are set to desired beginning MODBUS address of the variable to fetch. The valid variables that can be fetched are: nviMIO_Config_1, nviMIO_Config_2, nviMIO_Config_3, PanelVersion, nvi332Version, nviNeuronVer, nviMIO_Version, nviHeadConfig_1, nviHeadConfig_2, nviHeadConfig_3, nviHeadConfig_4, nviHeadConfig_5, nviHeadConfig_6, nviHeadConfig_7 and nviHeadConfig_8.
nviML_CorrFactor	Type: Float
	Units: Application Dependent
	A fractional value proportional to the magnitude of the correction from Actual to Standard Conditions.

Header File Definitions for the LightHawk 560

```
/*
 *      Copyright (c) 2006 by Teledyne Monitor Labs Inc.
 *
 *  This software is copyrighted by and is the sole property of
 *  Teledyne Monitor Labs.  All rights, title, ownership, or other interests
 *  in the software remain the property of Teledyne Monitor Labs.  This
 *  software may only be used in accordance with the corresponding
 *  license agreement.  Any unauthorized use, duplication, transmission,
 *  distribution, or disclosure of this software is expressly forbidden.
 *
 *  This Copyright notice may not be removed or modified without prior
 *  written consent of Teledyne Monitor Labs.
 *
 *  Teledyne Monitor Labs, reserves the right to modify this software
 *  without notice.
 *
 *  Teledyne Monitor Labs, Inc.
 *  Gibsonia Facility                       Phone: 724-444-5000
 *  5310 N. Pioneer Rd.                     Fax: 724-444-5050
 *  Gibsonia, PA 15044, USA                 http://www.teledyne-ml.com
 *
 ******************************************************************************
 *
 *     Module Name: LH560DEF.H
 *         Version: 1.00
 *   Original Date: 08/04/2006
 *          Author: Ivica Eftimovski
 *        Language: Ansi C
 * Compile Options:
 * Compile defines:
 *       Libraries:
 *    Link Options:
 *
 *
 * Description.
 * ============================================================================
 *  This file defines variables needed by the LightHawk 560.
 *
 *
 * Edit Date/Ver   Edit Description
 * ==============  ====================================================
 *
 *
 */

#ifndef LIGHTHAWK560DEFINITIONS_HEADER_FILE
#define LIGHTHAWK560DEFINITIONS_HEADER_FILE

#ifdef __cplusplus
extern "C"
{
#endif
```

```
/* Constants and Enumerations */

/* Isolator Configurations */
#define ISO_FULL_SCALE    0x80 /* binary10000000 */
#define ISO_ZERO_SCALE    0x40 /* binary01000000 */
#define ISO_MID_SCALE     0xC0 /* binary11000000 */
#define ISO_FORCE_CAL     0x10 /* binary00010000 */
#define ISO_FORCE_DIRT    0x08 /* binary00001000 */
#define ISO_FORCE_PLCF    0x04 /* binary00000100 */
#define ISO_FORCE_ZERO    0x02 /* binary00000010 */
#define ISO_FORCE_UPSCALE 0x01 /* binary00000001 */

#define WITH    1 /* for with calibration configurations */
#define WITHOUT 0 /* for without calibration configurations */
#define WITH_EXPANDED_SCALE 2 /* for with expanded cal ZERO & DIRT scaling */

#define VALID     1   /* used by fault flags to indicate status */
#define INVALID   42


/* Modes */
typedef enum
{
   UNKNOWN=0,         //  0
   NORMAL,            //  1
   ZERO,              //  2
   UPSCALE,           //  3
   PLCF,              //  4
   DIRT,              //  5
   FORCE_CAL_CYCLE,   //  6
   TEST_ZERO_SCALE,   //  7
   TEST_MID_SCALE,    //  8
   TEST_FULL_SCALE,   //  9
   PUT_IN_SERVICE,    // 10
   OUT_OF_SERVICE     // 11

} node_mode ;

/* MIO Analog configuration, A_output_type, B_output_type */
typedef enum
{
   NO_SELECTION = 0,               //  0
   INSTANT_OPAC,                   //  1
   MINUTE_AVG_OPAC,                //  2
   SELECTABLE_AVG_OPAC,            //  3
   INSTANT_OPT_DENSITY,            //  4
   MINUTE_AVG_OPT_DENSITY,         //  5
   SELECTABLE_AVG_OPT_DENSITY,     //  6
   INSTANT_DUST_MASS,              //  7
   MINUTE_AVG_DUST_MASS,           //  8
   SELECTABLE_AVG_DUST_MASS,       //  9
   DIRT_OUTPUT,                    // 10
   ZERO_OUTPUT,                    // 11
   UPSCALE_OUTPUT,                 // 12
   PLCF_OUTPUT,                    // 13
   SIGNAL_VOLTAGE,                 // 14
   REFERENCE_VOLTAGE,              // 15
```

```
   LED_CURRENT_mA,                // 16
   STACK_SET_VOLTAGE,             // 17
   CAL_ZERO_SET_VOLTAGE,          // 18
   BACK_GROUND_SET_VOLTAGE,       // 19
   OPTICAL_HEAD_TEMP_C,           // 20
   CAL_MECHANISM_POSITION,        // 21
   POS_15_SUPPLY,                 // 22
   NEG_15_SUPPLY,                 // 23
   POS_5V_ANALOG_SUPPLY,          // 24
   NEG_5V_ANALOG_SUPPLY,          // 25
   POS_5V_DIGITAL_SUPPLY          // 26
} AO_type ;

/* MIO relay configuration, DIO_type */
typedef enum
{
   NO_SELECT = 0,             //  0
   INST_OPAC_LEVEL1,          //  1
   INST_OPAC_LEVEL2,          //  2
   MIN_OPAC_LEVEL1,           //  3
   MIN_OPAC_LEVEL2,           //  4
   AVG_OPAC_LEVEL1,           //  5
   AVG_OPAC_LEVEL2,           //  6
   INST_OPT_DENSITY_LEVEL1,   //  7
   INST_OPT_DENSITY_LEVEL2,   //  8
   MIN_OPT_DENSITY_LEVEL1,    //  9
   MIN_OPT_DENSITY_LEVEL2,    // 10
   AVG_OPT_DENSITY_LEVEL1,    // 11
   AVG_OPT_DENSITY_LEVEL2,    // 12
   INST_MASS_LOAD_LEVEL1,     // 13
   INST_MASS_LOAD_LEVEL2,     // 14
   MIN_MASS_LOAD_LEVEL1,      // 15
   MIN_MASS_LOAD_LEVEL2,      // 16
   AVG_MASS_LOAD_LEVEL1,      // 17
   AVG_MASS_LOAD_LEVEL2,      // 18
   INSTRUMENT_MALF,           // 19
   INSTRUMENT_ALERT,          // 20
   PURGE_FAILURE,             // 21
   EXCESS_DIRT_COMP,          // 22
   CAL_FAILURE,               // 23
   SPAN_ON_AO,                // 24
   ZERO_ON_AO,                // 25
   PLCF_ON_AO,                // 26
   DIRT_ON_AO,                // 27
   NORMAL_ON_AO,              // 28
   CAL_ON_AO                  // 29
} DIO_type ;


typedef enum
{
      NOT_AUDIT, RUN1, RUN2, RUN3, RUN4, RUN5, CLEAR_AUDIT

} AUDITMODE ;

typedef enum
{
```

```
      NO_FILT, LOW_FILT, MID_FILT, HI_FILT

} FILTERVALUE ;

/* Network Variables */

 struct __SNVT_time_stamp__{
     WORD16   year;
     BYTE  month;
     BYTE  day;
     BYTE  hour;
     BYTE  minute;
     BYTE  second;
 } __attribute__ ((__packed__));

typedef struct __SNVT_time_stamp__ SNVT_time_stamp;

extern SNVT_time_stamp  *nvi332Version ;
extern SNVT_time_stamp  *nviNeuronVer ;
extern SNVT_time_stamp  *nviMIO_Version ;
extern SNVT_time_stamp  *PanelVersion ;
extern SNVT_time_stamp  *nviTime ;
extern SNVT_time_stamp  *nvoTimeSet ;
extern SNVT_time_stamp   *TimeSet ;

struct __SNVT_alarm__{
      BYTE        location[ 6 ];
      WORD16      object_id;
      BYTE        alarm_type;
      BYTE        priority_level;
      WORD16      index_to_SNVT;
      BYTE        value[ 4 ];
      WORD16      year;
      BYTE        month;
      BYTE        day;
      BYTE        hour;
      BYTE        minute;
      BYTE        second;
      WORD16      millisecond;
      BYTE        alarm_limit[ 4 ];

 } __attribute__ ((__packed__));

typedef struct __SNVT_alarm__ SNVT_alarm;

extern SNVT_alarm       *nviAlarms ;

struct __UNVT_HeadStatus__TagName__
{
   BYTE  DirtDriftAlarm : 1; // Bit15 for Excessive Dirt Drift
   BYTE  CalSpanAlarm   : 1; // Bit14 for UPSCALE Cal bad
   BYTE  CalZeroAlarm   : 1; // Bit13 for Cal ZERO bad
   BYTE  LoSetVoltZERO  : 1; // Bit12 for ZERO SET volts out of range
   BYTE  LoSetVoltSTACK : 1; // Bit11 for CLEAR STACK SET Volts out of range
   BYTE  SET_Backgrnd   : 1; // Bit10 for BACKGROUND SET in progress
   BYTE  SET_Zero       : 1; // Bit9  for ZERO SET in progress
   BYTE  SET_STACK      : 1; // Bit8  for CLEAR STACK SET in progress
```

```
   BYTE  UpscalePosErr  : 1; // Bit7  for UPSCALE position not achived
   BYTE  ZeroPosErr     : 1; // Bit6  for ZERO position not achived
   BYTE  NormalPosErr   : 1; // Bit5  for Normal position not achived
   BYTE  ReferenceFault : 1; // Bit4  for Reference Fault
   BYTE  ADC_Fault      : 1; // Bit3  for ADC (Sync Demod) Fault
   BYTE  Out_of_Service : 1; // Bit2  for Out of Service
   BYTE  PurgeFailRetro : 1; // Bit1  for Purge Failure Retro Side
   BYTE  PurgeFailHead  : 1; // Bit0  for Purge Failure Analyzer Side

}  __attribute__ ((__packed__));

typedef struct __UNVT_HeadStatus__TagName__ UNVT_HeadStatus;

extern UNVT_HeadStatus  *nviHeadState ;


struct      __UNVT_HeadMode__TagName__
{

      BYTE   CalMechanism;   // first four enums of nodemode
      BYTE   SixPIO_AO_Mode; // analog out modes defined in mode560.h
      BYTE   AuditMode;      // (enum: NOT_AUDIT, RUN1, RUN2, RUN3,
                          //        RUN4, RUN5, CLEAR_AUDIT)
      BYTE   FilterValue;    // (enum: ZERO, LOW_FILT, MID_FILT, HI_FILT

} __attribute__ ((__packed__));

typedef struct    __UNVT_HeadMode__TagName__ UNVT_HeadMode ;

extern UNVT_HeadMode    *nviHeadMode ;


struct __UNVT_MIO_status__TagName__
{
   BYTE     invalid_mode_request  : 1 ;
   BYTE     invalid_AO_config     : 1 ;
   BYTE     invalid_DIO_config    : 1 ;
   BYTE     reserved              : 5 ;
   BYTE     MIO_mode ;  /* the current mode of the Multi_i/O Module */
}  __attribute__ ((__packed__));

typedef struct __UNVT_MIO_status__TagName__ UNVT_MIO_status ;

extern UNVT_MIO_status  *nviMIO_Status ;

struct __UNVT_AlarmSel__TagName__
{
      BYTE  Bit_7          : 1; // Bit_07 = not-used
      BYTE  Bit_6          : 1; // Bit_06 = not-used

      BYTE  AveAlarm2      : 1; // Bit_05 =
      BYTE  AveAlarm1      : 1; // Bit_04 = Set Average Alarms

      BYTE  MinAlarm2      : 1; // Bit_03 =
      BYTE  MinAlarm1      : 1; // Bit_02 = 1-Minute Alarms

      BYTE  InstAlarm2     : 1; // Bit_01 =
```

```
      BYTE  InstAlarm1     : 1; // Bit_00 = Instantaneous Alarms

}  __attribute__ ((__packed__));

typedef struct __UNVT_AlarmSel__TagName__ UNVT_AlarmSel ;

extern UNVT_AlarmSel    *nviOpacityAlarms ;
extern UNVT_AlarmSel    *nviMassAlarms ;
extern UNVT_AlarmSel    *nviOptDensAlarms ;


extern float  *nviSelAveOpacity ;
extern float  *nviInstOpacity ;
extern float  *nvi1MinOpacity ;
extern float  *nviZeroOpacCal ;
extern float  *nviSpanOpacCal ;
extern float  *nviDirtCompValue ;

extern float  *nviAveOptDens ;
extern float  *nviInstOptDens ;
extern float  *nvi1MinOptDens ;
extern float  *nviZeroOptDens ;
extern float  *nviSpanOptDens ;
extern float  *nviDirtOptDens ;

extern float  *nviInstMassLoad ;
extern float  *nviAveMassLoad ;
extern float  *nvi1MinMassLoad ;
extern float  *nviZeroMassLoad ;
extern float  *nviSpanMassLoad ;
extern float  *nviDirtMassLoad ;

extern float  *nviAbsPress ;
extern float  *nviTemp_C ;
extern float  *nviML_CorrFactor ;


struct __UNVT_Service_One__TagName__
{
      float SignalVolts ;       //(-10 to +10, VDC)
      float ReferenceVolts ;    //(-10 to +10, VDC)
      float LED_current ;       //(0 to 50, ma)
      float StackSetVols ;      //(-10 to +10, VDC)
      float ZeroSetVolts ;      //(-10 to +10, VDC)
      float BackgndSetVols ;    //(-10 to +10, VDC)
      float HeadTemperature ;   //degrees C
      WORD16      CalWheelPosition ;  //dimless

}  __attribute__ ((__packed__));

typedef struct __UNVT_Service_One__TagName__ UNVT_Service_One ;

extern UNVT_Service_One  *nviServiceData1 ;


struct __UNVT_Service_Two__TagName__
{
```

```
      float Pos_15V ;        //(-30 to +30, VDC)
      float Neg_15V ;        //(-30 to +30, VDC)
      float Pos_5Vanalog ;   //(-10 to +10, VDC)
      float Neg_5Vanalog ;   //(-10 to +10, VDC)
      float Pos_5Vdigital ;  //(-10 to +10, VDC)

}  __attribute__ ((__packed__));

typedef struct __UNVT_Service_Two__TagName__ UNVT_Service_Two ;

extern UNVT_Service_Two  *nviServiceData2 ;


struct __UNVT_AuditData__TagName__
{
      BYTE AuditMode;     //(enum: NOT_AUDIT, RUN1, RUN2, RUN3,
                                 //       RUN4, RUN5, CLEAR_AUDIT)
      BYTE FilterValue;   //(enum: ZERO, LOW_FILT, MID_FILT, HI_FILT
      float AuditOpacity;
      float AuditOptDens;
      float AuditDustMass;

} __attribute__ ((__packed__));

typedef struct __UNVT_AuditData__TagName__ UNVT_AuditData ;

extern UNVT_AuditData    *nviAuditData ;


struct __UNVT_HeadConfig1__TagName__
{
   float   inst_opac_level1 ;
   float   inst_opac_level2 ;
   float   min_opac_level1 ;
   float   min_opac_level2 ;
   float   avg_opac_level1 ;
   float   avg_opac_level2 ;
   float   DirtCompAlarm ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig1__TagName__ UNVT_HeadConfig1 ;

extern UNVT_HeadConfig1  *nviHeadConfig_1 ;
extern UNVT_HeadConfig1  *nvoHeadConfig_1 ;


struct __UNVT_HeadConfig2__TagName__
{
   float   PLCFfactor ;
   float   ZeroCalSetpt ;
   float   SpanCalSetpt ;
   float   CalDelta ;
   BYTE    AveSendTime ; // selectable average send time in minutes

}  __attribute__ ((__packed__));
```

```
typedef struct  __UNVT_HeadConfig2__TagName__ UNVT_HeadConfig2 ;

extern UNVT_HeadConfig2  *nviHeadConfig_2 ;
extern UNVT_HeadConfig2  *nvoHeadConfig_2 ;


struct __UNVT_HeadConfig3__TagName__
{
   float   inst_optD_level1 ;
   float   inst_optD_level2 ;
   float   min_optD_level1 ;
   float   min_optD_level2 ;
   float   avg_optD_level1 ;
   float   avg_optD_level2 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig3__TagName__ UNVT_HeadConfig3 ;

extern UNVT_HeadConfig3  *nviHeadConfig_3 ;
extern UNVT_HeadConfig3  *nvoHeadConfig_3 ;


struct __UNVT_HeadConfig4__TagName__
{
   float   MassLoad_X1 ;
   float   MassLoad_Y1 ;
   float   MassLoad_X2 ;
   float   MassLoad_Y2 ;
   float   MassLoad_X3 ;
   float   MassLoad_Y3 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig4__TagName__ UNVT_HeadConfig4 ;

extern UNVT_HeadConfig4  *nviHeadConfig_4 ;
extern UNVT_HeadConfig4  *nvoHeadConfig_4 ;


struct __UNVT_HeadConfig5__TagName__
{
   float   inst_Dust_level1 ;
   float   inst_Dust_level2 ;
   float   min_Dust_level1 ;
   float   min_Dust_level2 ;
   float   avg_Dust_level1 ;
   float   avg_Dust_level2 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig5__TagName__ UNVT_HeadConfig5 ;

extern UNVT_HeadConfig5  *nviHeadConfig_5 ;
extern UNVT_HeadConfig5  *nvoHeadConfig_5 ;
```

```
struct __UNVT_HeadConfig6__TagName__
{
   float   Temp_Deg_C ;
   float   AbsPressure ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig6__TagName__ UNVT_HeadConfig6 ;

extern UNVT_HeadConfig6  *nviHeadConfig_6 ;
extern UNVT_HeadConfig6  *nvoHeadConfig_6 ;


struct  __UNVT_HeadConfig7__TagName__
{
   BYTE   cal_time_hour ;
   BYTE   cal_time_min ;
   BYTE   cal_interval_hour ;
   BYTE   filler ;
   WORD16   span_secs ;
   WORD16   zero_secs ;
   WORD16   plcf_secs ;
   WORD16   dirt_secs ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig7__TagName__ UNVT_HeadConfig7 ;

extern UNVT_HeadConfig7  *nviHeadConfig_7 ;
extern UNVT_HeadConfig7  *nvoHeadConfig_7 ;


struct __UNVT_HeadConfig8__TagName__
{
   WORD16  SignalGain ;
   WORD16  ReferenceGain ;
   WORD16  CommonGain ;
}  __attribute__ ((__packed__));

typedef struct __UNVT_HeadConfig8__TagName__ UNVT_HeadConfig8 ;

extern UNVT_HeadConfig8  *nviHeadConfig_8 ;
extern UNVT_HeadConfig8  *nvoHeadConfig_8 ;


struct __UNVT_MIO_config__TagName__
{
   BYTE     A_output_type ;
   BYTE     A_calibration ;
   float    A_zero ;
   float    A_full_scale ;

   BYTE     B_output_type ;
   BYTE     B_calibration ;
   float    B_zero ;
   float    B_full_scale ;
```

```
   BYTE     digital_type[8] ;

}  __attribute__ ((__packed__));

typedef struct __UNVT_MIO_config__TagName__ UNVT_MIO_config ;

extern UNVT_MIO_config   *nviMIO_Config_1 ;
extern UNVT_MIO_config   *nvoMIO_Config_1 ;
extern UNVT_MIO_config   *nviMIO_Config_2 ;
extern UNVT_MIO_config   *nvoMIO_Config_2 ;
extern UNVT_MIO_config   *nviMIO_Config_3 ;
extern UNVT_MIO_config   *nvoMIO_Config_3 ;


struct __SNVT_obj_request__TagName__
{
      WORD16 object_id;
      BYTE object_request;

} __attribute__ ((__packed__));

typedef struct __SNVT_obj_request__TagName__ SNVT_obj_request;

extern SNVT_obj_request *nviRequest;


struct __UNVT_ModeRequest__TagName__
{
   char node_id ;
   char mode ;
}  __attribute__ ((__packed__));

typedef struct __UNVT_ModeRequest__TagName__ UNVT_ModeRequest ;

extern UNVT_ModeRequest *nvoModeRequest ;
extern UNVT_ModeRequest *nviNetworkStatus;
extern UNVT_ModeRequest *nvoDataRequest;

#ifdef __cplusplus
}
#endif

#endif
```

Modbus Input Registers for LightHawk 560

Variable	Member	Type	Large Address	Offset
nvi332Version		SNVT_time_stamp		
	year	WORD (HIBYTE)	30002	1
		WORD (LOWBYTE)		
	month	BYTE	30003	2
	day	BYTE		
	hour	BYTE	30004	3
	minute	BYTE		
	second	BYTE	30005	4
	(reserved)	BYTE		
nviNeuronVer		SNVT_time_stamp		
	year	WORD (HIBYTE)	30006	5
		WORD (LOWBYTE)		
	month	BYTE	30007	6
	day	BYTE		
	hour	BYTE	30008	7
	minute	BYTE		
	second	BYTE	30009	8
	(reserved)	BYTE		
nviMIO_Version		SNVT_time_stamp		
	year	WORD (HIBYTE)	30010	9
		WORD (LOWBYTE)		
	month	BYTE	30011	10
	day	BYTE		
	hour	BYTE	30012	11
	minute	BYTE		
	second	BYTE	30013	12
	(reserved)	BYTE		
nviTime		SNVT_time_stamp		
	year	WORD (HIBYTE)	30014	13
		WORD (LOWBYTE)		
	month	BYTE	30015	14
	day	BYTE		
	hour	BYTE	30016	15
	minute	BYTE		
	second	BYTE	30017	16
	(reserved)	BYTE		
nviAlarms		SNVT_alarm		
	location	BYTE[0]	30018	17
		BYTE[1]		
		BYTE[2]	30019	18

		BYTE[3]		
		BYTE[4]	30020	19
		BYTE[5]		
	object_id	WORD (HIBYTE)	30021	20
		WORD (LOWBYTE)		
	alarm_type	BYTE	30022	21
	priority_level	BYTE		
	index_to_SNVT	WORD (HIBYTE)	30023	22
		WORD (LOWBYTE)		
	value	BYTE[0]	30024	23
		BYTE[1]		
		BYTE[2]	30025	24
		BYTE[3]		
	year	WORD (HIBYTE)	30026	25
		WORD (LOWBYTE)		
	month	BYTE	30027	26
	day	BYTE		
	hour	BYTE	30028	27
	minute	BYTE		
	second	BYTE	30029	28
	millisecond	WORD (HIBYTE)		
		WORD (LOWBYTE)	30030	29
	alarm_limit	BYTE[0]		
		BYTE[1]	30031	30
		BYTE[2]		
		BYTE[3]	30032	31
nviHeadState		UNVT_HeadStatus		
	DirtDriftAlarm	BYTE	30033	32
	CalSpanAlarm			
	CalZeroAlarm			
	LoSetVoltZERO			
	LoSetVoltSTACK			
	SET_Backgrnd			
	SET_Zero			
	SET_STACK			
	UpscalePosErr	BYTE		
	ZeroPosErr			
	NormalPosErr			
	ReferenceFault			
	ADC_Fault			
	Out_of_Service			
	PurgeFailRetro			

	PurgeFailHead			
nviHeadMode		UNVT_HeadMode		
	CalMechanism	node_mode (enum)	30034	33
	SixPIO_AO_Mode	node_mode (enum)		
	AuditMode	AUDITMODE (enum)	30035	34
	FilterValue	FILTERVALUE (enum)		
nviMIO_Status		UNVT_MIO_status		
	invalid_mode_request	BYTE	30036	35
	invalid_AO_config			
	invalid_DIO_config			
	reserved			
	MIO_mode	node_mode (enum)		
nviOpacityAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30037	36
	Bit_6			
	AveAlarm2			
	AveAlarm1			
	MinAlarm2			
	MinAlarm1			
	InstAlarm2			
	InstAlarm1			
	(reserved)	BYTE		
nviSelAveOpacity		float		
		IEEE float (S+MSE)	30038	37
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30039	38
		IEEE float (LOBYTE(LSM))		
nviInstOpacity		float		
		IEEE float (S+MSE)	30040	39
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30041	40
		IEEE float (LOBYTE(LSM))		
nvi1MinOpacity		float		
		IEEE float (S+MSE)	30042	41
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30043	42
		IEEE float (LOBYTE(LSM))		
nviZeroOpacCal		float		
		IEEE float (S+MSE)	30044	43
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30045	44
		IEEE float (LOBYTE(LSM))		
nviSpanOpacCal		float		

		IEEE float (S+MSE)	30046	45
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30047	46
		IEEE float (LOBYTE(LSM))		
nviDirtCompValue		float		
		IEEE float (S+MSE)	30048	47
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30049	48
		IEEE float (LOBYTE(LSM))		
nviServiceData1		UNVT_Service_One		
	SignalVolts	IEEE float (S+MSE)	30050	49
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30051	50
		IEEE float (LOBYTE(LSM))		
	ReferenceVolts	IEEE float (S+MSE)	30052	51
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30053	52
		IEEE float (LOBYTE(LSM))		
	LED_current	IEEE float (S+MSE)	30054	53
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30055	54
		IEEE float (LOBYTE(LSM))		
	StackSetVols	IEEE float (S+MSE)	30056	55
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30057	56
		IEEE float (LOBYTE(LSM))		
	ZeroSetVolts	IEEE float (S+MSE)	30058	57
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30059	58
		IEEE float (LOBYTE(LSM))		
	BackgndSetVols	IEEE float (S+MSE)	30060	59
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30061	60
		IEEE float (LOBYTE(LSM))		
	HeadTemperature	IEEE float (S+MSE)	30062	61
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30063	62
		IEEE float (LOBYTE(LSM))		
	CalWheelPosition	WORD (HIBYTE)	30064	63
		WORD (LOWBYTE)		
nviServiceData2		UNVT_Service_Two		
	Pos_15V	IEEE float (S+MSE)	30065	64
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30066	65
		IEEE float (LOBYTE(LSM))		
	Neg_15V	IEEE float (S+MSE)	30067	66
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30068	67
		IEEE float (LOBYTE(LSM))		
	Pos_5Vanalog	IEEE float (S+MSE)	30069	68
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30070	69
		IEEE float (LOBYTE(LSM))		
	Neg_5Vanalog	IEEE float (S+MSE)	30071	70
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30072	71
		IEEE float (LOBYTE(LSM))		
	Pos_5Vdigital	IEEE float (S+MSE)	30073	72
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30074	73
		IEEE float (LOBYTE(LSM))		
nviAuditData		UNVT_AuditData		
	AuditMode	AUDITMODE (enum)	30075	74
	FilterValue	FILTERVALUE (enum)		
	AuditOpacity	IEEE float (S+MSE)	30076	75
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30077	76
		IEEE float (LOBYTE(LSM))		
	AuditOptDens	IEEE float (S+MSE)	30078	77
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30079	78
		IEEE float (LOBYTE(LSM))		
	AuditDustMass	IEEE float (S+MSE)	30080	79
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30081	80
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_1		UNVT_HeadConfig1		
	inst_opac_level1	IEEE float (S+MSE)	30082	81
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30083	82
		IEEE float (LOBYTE(LSM))		
	inst_opac_level2	IEEE float (S+MSE)	30084	83
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30085	84
		IEEE float (LOBYTE(LSM))		
	min_opac_level1	IEEE float (S+MSE)	30086	85

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30087	86
		IEEE float (LOBYTE(LSM))		
	min_opac_level2	IEEE float (S+MSE)	30088	87
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30089	88
		IEEE float (LOBYTE(LSM))		
	avg_opac_level1	IEEE float (S+MSE)	30090	89
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30091	90
		IEEE float (LOBYTE(LSM))		
	avg_opac_level2	IEEE float (S+MSE)	30092	91
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30093	92
		IEEE float (LOBYTE(LSM))		
	DirtCompAlarm	IEEE float (S+MSE)	30094	93
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30095	94
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_2		UNVT_HeadConfig2		
	PLCFfactor	IEEE float (S+MSE)	30096	95
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30097	96
		IEEE float (LOBYTE(LSM))		
	ZeroCalSetpt	IEEE float (S+MSE)	30098	97
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30099	98
		IEEE float (LOBYTE(LSM))		
	SpanCalSetpt	IEEE float (S+MSE)	30100	99
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30101	100
		IEEE float (LOBYTE(LSM))		
	CalDelta	IEEE float (S+MSE)	30102	101
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30103	102
		IEEE float (LOBYTE(LSM))		
	AveSendTime	BYTE	30104	103
	(reserved)			
nviHeadConfig_3		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	30105	104
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30106	105
		IEEE float (LOBYTE(LSM))		

	inst_optD_level2	IEEE float (S+MSE)	30107	106
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30108	107
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	30109	108
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30110	109
		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	30111	110
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30112	111
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	30113	112
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30114	113
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	30115	114
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30116	115
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_4		UNVT_HeadConfig4		
	MassLoad_X1	IEEE float (S+MSE)	30117	116
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30118	117
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y1	IEEE float (S+MSE)	30119	118
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30120	119
		IEEE float (LOBYTE(LSM))		
	MassLoad_X2	IEEE float (S+MSE)	30121	120
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30122	121
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y2	IEEE float (S+MSE)	30123	122
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30124	123
		IEEE float (LOBYTE(LSM))		
	MassLoad_X3	IEEE float (S+MSE)	30125	124
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30126	125
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y3	IEEE float (S+MSE)	30127	126
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30128	127
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_5		UNVT_HeadConfig5		
	inst_Dust_level1	IEEE float (S+MSE)	30129	128
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30130	129
		IEEE float (LOBYTE(LSM))		
	inst_Dust_level2	IEEE float (S+MSE)	30131	130
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30132	131
		IEEE float (LOBYTE(LSM))		
	min_Dust_level1	IEEE float (S+MSE)	30133	132
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30134	133
		IEEE float (LOBYTE(LSM))		
	min_Dust_level2	IEEE float (S+MSE)	30135	134
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30136	135
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level1	IEEE float (S+MSE)	30137	136
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30138	137
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level2	IEEE float (S+MSE)	30139	138
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30140	139
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_6		UNVT_HeadConfig6		
	Temp_Deg_C	IEEE float (S+MSE)	30141	140
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30142	141
		IEEE float (LOBYTE(LSM))		
	AbsPressure	IEEE float (S+MSE)	30143	142
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30144	143
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_7		UNVT_HeadConfig7		
	cal_time_hour	BYTE	30145	144
	cal_time_min	BYTE		
	cal_interval_hour	BYTE	30146	145
	filler	BYTE		
	span_secs	WORD (HIBYTE)	30147	146
		WORD (LOWBYTE)		

	zero_secs	WORD (HIBYTE)	30148	147
		WORD (LOWBYTE)		
	plcf_secs	WORD (HIBYTE)	30149	148
		WORD (LOWBYTE)		
	dirt_secs	WORD (HIBYTE)	30150	149
		WORD (LOWBYTE)		
nviHeadConfig_8		UNVT_HeadConfig8		
	SignalGain	WORD (HIBYTE)	30151	150
		WORD (LOWBYTE)		
	ReferenceGain	WORD (HIBYTE)	30152	151
		WORD (LOWBYTE)		
	CommonGain	WORD (HIBYTE)	30153	152
		WORD (LOWBYTE)		
nviMIO_Config_1		UNVT_MIO_config		
	A_output_type	BYTE	30154	153
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30155	154
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30156	155
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30157	156
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30158	157
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	30159	158
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30160	159
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30161	160
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30162	161
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30163	162
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30164	163
		BYTE[1]		
		BYTE[2]	30165	164
		BYTE[3]		
		BYTE[4]	30166	165
		BYTE[5]		
		BYTE[6]	30167	166
		BYTE[7]		
nviMIO_Config_2		UNVT_MIO_config		

	A_output_type	BYTE	30168	167
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30169	168
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30170	169
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30171	170
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30172	171
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	30173	172
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30174	173
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30175	174
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30176	175
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30177	176
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30178	177
		BYTE[1]		
		BYTE[2]	30179	178
		BYTE[3]		
		BYTE[4]	30180	179
		BYTE[5]		
		BYTE[6]	30181	180
		BYTE[7]		
nviMIO_Config_3		UNVT_MIO_config		
	A_output_type	BYTE	30182	181
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30183	182
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30184	183
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30185	184
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30186	185
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	30187	186
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30188	187
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30189	188
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30190	189
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30191	190
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30192	191
		BYTE[1]		
		BYTE[2]	30193	192
		BYTE[3]		
		BYTE[4]	30194	193
		BYTE[5]		
		BYTE[6]	30195	194
		BYTE[7]		
nviInstMassLoad		float		
		IEEE float (S+MSE)	30196	195
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30197	196
		IEEE float (LOBYTE(LSM))		
nviAveMassLoad		float		
		IEEE float (S+MSE)	30198	197
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30199	198
		IEEE float (LOBYTE(LSM))		
nvi1MinMassLoad		float		
		IEEE float (S+MSE)	30200	199
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30201	200
		IEEE float (LOBYTE(LSM))		
nviZeroMassLoad		float		
		IEEE float (S+MSE)	30202	201
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30203	202
		IEEE float (LOBYTE(LSM))		
nviSpanMassLoad		float		
		IEEE float (S+MSE)	30204	203
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30205	204
		IEEE float (LOBYTE(LSM))		
nviDirtMassLoad		float		
		IEEE float (S+MSE)	30206	205
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30207	206

		IEEE float (LOBYTE(LSM))		
nviAbsPress		SNVT_press_f		
		IEEE float (S+MSE)	30208	207
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30209	208
		IEEE float (LOBYTE(LSM))		
nviTemp_C		SNVT_temp_f		
		IEEE float (S+MSE)	30210	209
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30211	210
		IEEE float (LOBYTE(LSM))		
nviML_CorrFactor		float		
		IEEE float (S+MSE)	30212	211
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30213	212
		IEEE float (LOBYTE(LSM))		
nviMassAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30214	213
	Bit_6	BYTE		
	AveAlarm2	BYTE	30215	214
	AveAlarm1	BYTE		
	MinAlarm2	BYTE	30216	215
	MinAlarm1	BYTE		
	InstAlarm2	BYTE	30217	216
	InstAlarm1	BYTE		
nviRequest		SNVT_obj_request		
	object_id	WORD (HIBYTE)	30218	217
		WORD (LOWBYTE)		
	object_request	BYTE	30219	218
	(reserved)			
nviAveOptDens		float		
		IEEE float (S+MSE)	30220	219
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30221	220
		IEEE float (LOBYTE(LSM))		
nviInstOptDens		float		
		IEEE float (S+MSE)	30222	221
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30223	222
		IEEE float (LOBYTE(LSM))		
nvi1MinOptDens		float		
		IEEE float (S+MSE)	30224	223
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30225	224
		IEEE float (LOBYTE(LSM))		
nviZeroOptDens		float		
		IEEE float (S+MSE)	30226	225
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30227	226
		IEEE float (LOBYTE(LSM))		
nviSpanOptDens		float		
		IEEE float (S+MSE)	30228	227
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30229	228
		IEEE float (LOBYTE(LSM))		
nviDirtOptDens		float		
		IEEE float (S+MSE)	30230	229
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30231	230
		IEEE float (LOBYTE(LSM))		
nviOptDensAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30232	231
	Bit_6	BYTE		
	AveAlarm2	BYTE	30233	232
	AveAlarm1	BYTE		
	MinAlarm2	BYTE	30234	233
	MinAlarm1	BYTE		
	InstAlarm2	BYTE	30235	234
	InstAlarm1	BYTE		
nviNetworkStatus		UNVT_ModeRequest		
	node_id	BYTE	30236	235
	mode	BYTE		
PanelVersion		SNVT_time_stamp		
	year	WORD (HIBYTE)	30237	236
		WORD (LOWBYTE)		
	month	BYTE	30238	237
	day	BYTE		
	hour	BYTE	30239	238
	minute	BYTE		
	second	BYTE	30240	239
	(reserved)	BYTE		

Modbus Holding Registers for LighHawk 560

Variable	Member	Type	Large Address	Offset
nvoModeRequest_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40002**	**1**
nvoModeRequest		UNVT_ModeRequest		
	node_id	BYTE	40003	2
	mode	BYTE		
nvoTimeSet_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40004**	**3**
nvoTimeSet		SNVT_time_stamp		
	year	WORD (HIBYTE)	40005	4
		WORD (LOWBYTE)		
	month	BYTE	40006	5
	day	BYTE		
	hour	BYTE	40007	6
	minute	BYTE		
	second	BYTE	40008	7
	(reserved)			
nvoHeadConfig_1_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40009**	**8**
nvoHeadConfig_1		UNVT_HeadConfig1		
	inst_opac_level1	IEEE float (S+MSE)	40010	9
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40011	10
		IEEE float (LOBYTE(LSM))		
	inst_opac_level2	IEEE float (S+MSE)	40012	11
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40013	12
		IEEE float (LOBYTE(LSM))		
	min_opac_level1	IEEE float (S+MSE)	40014	13
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40015	14
		IEEE float (LOBYTE(LSM))		
	min_opac_level2	IEEE float (S+MSE)	40016	15
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40017	16
		IEEE float (LOBYTE(LSM))		
	avg_opac_level1	IEEE float (S+MSE)	40018	17
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40019	18
		IEEE float (LOBYTE(LSM))		
	avg_opac_level2	IEEE float (S+MSE)	40020	19

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40021	20
		IEEE float (LOBYTE(LSM))		
	DirtCompAlarm	IEEE float (S+MSE)	40022	21
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40023	22
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_2_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40024**	**23**
nvoHeadConfig_2		UNVT_HeadConfig2		
	PLCFfactor	IEEE float (S+MSE)	40025	24
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40026	25
		IEEE float (LOBYTE(LSM))		
	ZeroCalSetpt	IEEE float (S+MSE)	40027	26
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40028	27
		IEEE float (LOBYTE(LSM))		
	SpanCalSetpt	IEEE float (S+MSE)	40029	28
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40030	29
		IEEE float (LOBYTE(LSM))		
	CalDelta	IEEE float (S+MSE)	40031	30
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40032	31
		IEEE float (LOBYTE(LSM))		
	AveSendTime	BYTE	40033	32
	(reserved)			
nvoHeadConfig_3_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40034**	**33**
nvoHeadConfig_3		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	40035	34
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40036	35
		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	40037	36
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40038	37
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	40039	38
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40040	39
		IEEE float (LOBYTE(LSM))		

	min_optD_level2	IEEE float (S+MSE)	40041	40
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40042	41
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	40043	42
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40044	43
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	40045	44
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40046	45
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_4_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40047**	**46**
nvoHeadConfig_4		UNVT_HeadConfig4		
	MassLoad_X1	IEEE float (S+MSE)	40048	47
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40049	48
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y1	IEEE float (S+MSE)	40050	49
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40051	50
		IEEE float (LOBYTE(LSM))		
	MassLoad_X2	IEEE float (S+MSE)	40052	51
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40053	52
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y2	IEEE float (S+MSE)	40054	53
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40055	54
		IEEE float (LOBYTE(LSM))		
	MassLoad_X3	IEEE float (S+MSE)	40056	55
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40057	56
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y3	IEEE float (S+MSE)	40058	57
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40059	58
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_5_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40060**	**59**
nvoHeadConfig_5		UNVT_HeadConfig5		
	inst_Dust_level1	IEEE float (S+MSE)	40061	60

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40062	61
		IEEE float (LOBYTE(LSM))		
	inst_Dust_level2	IEEE float (S+MSE)	40063	62
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40064	63
		IEEE float (LOBYTE(LSM))		
	min_Dust_level1	IEEE float (S+MSE)	40065	64
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40066	65
		IEEE float (LOBYTE(LSM))		
	min_Dust_level2	IEEE float (S+MSE)	40067	66
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40068	67
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level1	IEEE float (S+MSE)	40069	68
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40070	69
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level2	IEEE float (S+MSE)	40071	70
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40072	71
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_6_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40073**	**72**
nvoHeadConfig_6		UNVT_HeadConfig6		
	Temp_Deg_C	IEEE float (S+MSE)	40074	73
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40075	74
		IEEE float (LOBYTE(LSM))		
	AbsPressure	IEEE float (S+MSE)	40076	75
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40077	76
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_7_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40078**	**77**
nvoHeadConfig_7		UNVT_HeadConfig7		
	cal_time_hour	BYTE	40079	78
	cal_time_min	BYTE		
	cal_interval_hour	BYTE	40080	79
	filler	BYTE		
	span_secs	WORD (HIBYTE)	40081	80
		WORD (LOWBYTE)		

	zero_secs	WORD (HIBYTE)	40082	81
		WORD (LOWBYTE)		
	plcf_secs	WORD (HIBYTE)	40083	82
		WORD (LOWBYTE)		
	dirt_secs	WORD (HIBYTE)	40084	83
		WORD (LOWBYTE)		
nvoHeadConfig_8_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40085**	**84**
nvoHeadConfig_8		UNVT_HeadConfig8		
	SignalGain	WORD (HIBYTE)	40086	85
		WORD (LOWBYTE)		
	ReferenceGain	WORD (HIBYTE)	40087	86
		WORD (LOWBYTE)		
	CommonGain	WORD (HIBYTE)	40088	87
		WORD (LOWBYTE)		
nvoMIO_Config_1_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40089**	**88**
nvoMIO_Config_1		UNVT_MIO_config		
	A_output_type	BYTE	40090	89
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40091	90
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40092	91
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40093	92
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40094	93
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40095	94
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40096	95
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40097	96
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40098	97
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40099	98
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40100	99
		BYTE[1]		
		BYTE[2]	40101	100
		BYTE[3]		
		BYTE[4]	40102	101

		BYTE[5]		
		BYTE[6]	40103	102
		BYTE[7]		
nvoMIO_Config_2_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40104**	**103**
nvoMIO_Config_2		UNVT_MIO_config		
	A_output_type	BYTE	40105	104
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40106	105
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40107	106
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40108	107
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40109	108
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40110	109
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40111	110
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40112	111
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40113	112
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40114	113
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40115	114
		BYTE[1]		
		BYTE[2]	40116	115
		BYTE[3]		
		BYTE[4]	40117	116
		BYTE[5]		
		BYTE[6]	40118	117
		BYTE[7]		
nvoMIO_Config_3_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40119**	**118**
nvoMIO_Config_3		UNVT_MIO_config		
	A_output_type	BYTE	40120	119
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40121	120
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40122	121
		IEEE float (LOBYTE(LSM))		

	A_full_scale	IEEE float (S+MSE)	40123	122
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40124	123
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40125	124
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40126	125
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40127	126
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40128	127
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40129	128
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40130	129
		BYTE[1]		
		BYTE[2]	40131	130
		BYTE[3]		
		BYTE[4]	40132	131
		BYTE[5]		
		BYTE[6]	40133	132
		BYTE[7]		
nvoDataRequest_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40134**	**133**
nvoDataRequest		UNVT_ModeRequest		
	node_id	BYTE	40135	134
	mode	BYTE		

(This page intentionally left blank.)

APPENDIX C

ULTRAFLOW 150 VARIABLES

(This page intentionally left blank.)

This section describes the variables for the Ultraflow 150 instrument. A description of each network variable is given, followed by the C/C++ definitions of the structures, constants and variable types. The Modbus register mapping is also given along with the table of indexed values for the "indexed network variables". These indexed network variables are used by the Ultraflow 150 for configuration or displaying additional data. They are named: nviFloatConfig1, nviFloatConfig2, nvoFloatConfig1, nvoFloatConfig2, nviFloatData1, nviFloatData2, nvoFloatData1, nvoFloatData2, nviUnLngConfig1, nvoUnLngConfig1, nviUnLngConfig2 and nvoUnLngConfig2.

All network variables are prefixed with a three letter prefix, either "nvi" or "nvo" which translates to:

nvi – Network Variable Input;
nvo – Network Variable Output.

The direction of Input or Output is relative to the Ethernet Module.

NOTE: TIE is an acronym for Transducer Interface Enclosure.

Instrument Status Word Interpretation via Modbus TCP
On the Ultraflow 150, the "primary_status" and "extended_status" members of the nviVelocity1A, nviVelocity1AB, nviVelocity1B, nviVelocity2A, nviVelocity2AB and nviVelocity2B variables signify instrument status. All status bits in each structure member are "OR-ed" together to create an individual status word.

The velocity variables ending in "A" are for Path A, "B" are for Path B and "AB" are a logical "OR" of Paths A and B. The number "1" represents TIE1 and "2" represents TIE 2 (the Ultraflow 150 Ethernet Module, like the Enhanced Remote Panel, can interface to two TIE's simultaneously).

Primary Status Code Definition
Bit_15 = *(Not Used)*
Bit_14 = Signal Energy Ratio >6X
Bit_13 = S/N Ratio Warning Upstream
Bit_12 = S/N Ratio Warning, Downstream
Bit_11 = SPAN LOW Calibration bad
Bit_10 = SPAN HIGH Calibration bad
Bit_09 = ZERO Calibration bad
Bit_08 = SPAN LOW Offset TD for CAL bad
Bit_07 = SPAN HIGH Offset TD for CAL bad
Bit_06 = Minimum Signal Energy Not Achieved, Upstream
Bit_05 = Minimum Signal Energy Not Achieved, Downstream
Bit_04 = Temperature Out of Range
Bit_03 = FPGA Config. Fault
Bit_02 = Out of Service
Bit_01 = Purge Failure Upstream
Bit_00 = Purge Failure Downstream

Extended Status Code Definition
Bit_15 = *(Not Used)*
Bit_14 = *(Not Used)*
Bit_13 = *(Not Used)*
Bit_12 = *(Not Used)*
Bit_11 = *(Not Used)*
Bit_10 =*(Not Used)*
Bit_09 = *(Not Used)*
Bit_08 =*(Not Used)*
Bit_07 =*(Not Used)*
Bit_06 = Noise Threshold Disabled
Bit_05 = Gain Disabled
Bit_04 = BoxCar Overflow
Bit_03 = Filter Mismatch
Bit_02 = 3 X FPGA Timeout
Bit_01 = FPGA Timeout
Bit_00 = Divide By Zero

Initiating Calibration Commands via Modbus TCP

On the Ultraflow 150, the nvoModeRequest variable is used to command calibration on the analyzer. See the description for nvoModeRequest in the following tables, the header files for the Ultraflow 150 and the “Propagating Values On The Network” section for more details on command initiation.

NviAlarms	Type: SNVT_alarm
	Units: Time and date.
	This network variable is issued by the Ultraflow-150 TIE once whenever an alarm changes state. The type, object, value and time stamp of the alarm are propagated. The Display Panel device accumulates an alarm history based on issuance events of this variable.
nviAlarmState1, nviAlarmState2	Type: UNVT_AlarmsU150
	Units: Dimensionless.
	These network variables represent the states of TIE1's Instantaneous and Average single level alarms for Linearized Actual Volume, Standard Volume, Medium Internal Temperature and Medium External Temperature. The variable and path actuating the alarm are selectable from another network variable. If a bit equals one, the alarm condition is in effect. These network variables are propagated every time the alarm condition is evaluated.
nviFloatConfig1, nviFloatConfig2, nvoFloatConfig1, nvoFloatConfig2	Type: UNVT_FloatArray
	Units: Dependent on array index number. The unit for each array element is defined later in this document.
	An array of floating point data for use when time or mode stamping is not required. The meaning of each array element is defined later in this document. This variable communicates floating point instrument configuration data between TIE1 and Enhanced Remote Panel. Notes on the (.command) portion of the structure: (.command = set) > The initiating 68332 uses this command in (nvoFloatConfig1) to tell the receiving 68332 to set the new value contained in (.float_data) into its memory location for (.variable_index). On the receiving end, when nviFloatConfig1 is received, the 68332 will set the value in (.float_data) into its memory location for (.variable_index). (.command = read) > The initiating 68332 uses this command in (nvoFloatConfig1) to request the receiving device to send the current value of (.variable_index) back to the requesting device. On the receiving end, when nviFloatConfig1 sees this command the 68332 will send the value for (.variable_index) back to the requesting device as nvoFloatConfig1 with the (.command) set to (set). (.command = loop) > Since this variable has the possibility of being over-written in the Neuron receiving device, the (.command = loop) is used by the Neuron as a loop back to indicate that the nvoFloatConfig1 was received and a new nvoFloatConfig1 can be sent.

nviFloatData1, **nviFloatData2,** **nvoFloatData1,** **nvoFloatData2**	Type: UNVT_FloatArray
	Units: Dependent on array index number. The unit for each array element is defined later in this document.
	An array of floating point data for use when time or mode stamping is not required. The meaning of each array element is defined later in this document. In general these variables communicate auxiliary output data from TIE1 to the Enhanced Remote and Multi I/O. A description of the each indexed variable is included in the "description" variable. Notes on the (.command) portion of the structure: (.command = set) > The initiating 68332 uses this command in (nvoFloatData1) to tell the receiving 68332 to set the new value contained in (.float_data) into its memory location for (.variable_index). On the receiving end, when nviFloatData1 is received, the 68332 will set the value in (.float_data) into its memory location for (.variable_index). (.command = read) > The initiating 68332 uses this command in (nvoFloatData1) to request the receiving device to send the current value of (.variable_index) back to the requesting device. On the receiving end, when nviFloatData1 sees this command the 68332 will send the value for (.variable_index) back to the requesting device as nvoFloatData1 with the (.command) set to (set). (.command = loop) > Since this variable has the possibility of being over-written in the Neuron receiving device, the (.command = loop) is used by the Neuron as a loop back to indicate that the nvoFloatData1 was received and a new nvoFloatData1 can be sent.
nviMIO_Config_1, **nvoMIO_Config_1**	Type: UNVT_AIO_Config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 1 (Analog Output 1) and DAC 2 (Analog Output 2) of the Multi I/O device.
nviMIO_Config_2, **nvoMIO_Config_2**	Type: UNVT_AIO_Config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 3 (Analog Output 3) and DAC 4 (Analog Output 4) of the Multi I/O device.
nviMIO_Config_3,	Type: UNVT_MIO_Config

nvoMIO_Config_3	Units: Dimensionless.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping for Relay 1 (K1) through Relay 8 (K8) of the Multi I/O device.
nviMIO_Status	Type: UNVT_MIO_Status
	Units: Dimensionless.
	This network variable communicates the analog output mode (MIO_mode) and the error state (invalid_mode_request, invalid_AO_config and invalid_DIO_config) of the Multi I/O device. Errors exist when the bit type members are one. For normal operation they are zero. MIO_mode is an enumerated member with values such as SPAN, ZERO, NORMAL, etc.
nviPressTemp1, nviPressTemp2	Type: UNVT_FloatStamp
	Units: Dependent on geometry unit variable (If meters, temperature unit is degrees Centigrade and pressure unit is Kilo Pascals; if feet, temperature unit is degrees Fahrenheit and pressure unit is inches of mercury).
	A structure of floating point data from TIE1 that is time and mode stamped. The "data1" variables are RTD derived instantaneous and average medium temperature. The "data2" variables are medium absolute pressure.
nviServceData1A, nviServceData2A, nviServceData1B, nviServceData2B	Type: UNVT_ServiceData
	Units: All variables in this structure are dimensionless.
	A structure of integer and floating point data from TIE1/2 measurement Path-A/B that is time and mode stamped. The "peak_RAM_counts" variables are the integer time increments into the digitizing window where the peak occurs, where "up" signifies upstream and "down" signifies downstream. The "calc_RAM_counts" are floating point variables representing RAM counts after processing by a peak detection algorithm. The "SN_ratio" variables represent signal-to-noise ratio of the upstream ("up") and downstream ("down") transducers with linear scaling. The "gain" variables represent the current upstream and downstream digital and preamp gains in the system. All designations are with respect to the receive mode of the transducer.
nviSoundTemp1A, nviSoundTemp1AB, nviSoundTemp1B, nviSoundTemp2A, nviSoundTemp2AB, nviSoundTemp2B	Type: UNVT_FloatStamp
	Units: Dependent on geometry unit variable (If meters, temperature unit is degrees Centigrade and speed of sound unit is meters/second; if feet, temperature unit is degrees Fahrenheit and speed of sound unit is feet/second).

	A structure of floating point data from TIE1/2 measurement Path-A/AB/B that is time and mode stamped. The "data1" variables are ultrasonically derived instantaneous and average medium temperature. The "data2" variables are medium speed of sound.
nviSpanHigh1, nviSpanHigh2	Type: UNVT_StampedCal
	Units: Dependent on geometry and flow volume unit variables.
	A structure of floating point data from TIE1/2 measurement paths A and B that is time stamped. The floating point data is the last average span high calibration data from TIE1/2 measurement paths A and B. The time stamp is the end time of the averaging interval for when the data first propagated.
nviSpanLow1, nviSpanLow2	Type: UNVT_StampedCal
	Units: Dependent on geometry and flow volume unit variables.
	A structure of floating point data from TIE1/2 measurement paths A and B that is time stamped. The floating point data is the last average span low calibration data from TIE1/2 measurement paths A and B. The time stamp is the end time of the averaging interval for when the data first propagated.
nviTime1, nviTime2	Type: SNVT_time_stamp
	Units: Time and date.
	Contains the current value of the TIE1/2 Real Time. The Enhanced Remote Panel uses the input of this network variable as a flag to determine that TIE1/2 is on the network so its variables may be polled.
nviUnLngConfig1, nvoUnLngConfig1, nviUnLngConfig2,	Type: UNVT_UnLongArray
	Units: Dependent on array index number. The unit for each array element is defined later in this document.

nvoUnLngConfig2	An array of unsigned long integer data for use when time or mode stamping is not required. The meaning of each array element is defined later in this document. A description of each indexed variable is included in the "description" variable. This variable communicates unsigned long integer instrument configuration data between TIE1/2 and Enhanced Remote Display. Notes on the (.command) portion of the structure: (.command = set) > The initiating 68332 uses this command in (nvoUnLngConfig1) to tell the receiving 68332 to set the new value contained in (.float_data) into its memory location for (.variable_index). On the receiving end, when nviUnLngConfig1 is received, the 68332 will set the value in (.float_data) into its memory location for (.variable_index). (.command = read) > The initiating 68332 uses this command in (nvoUnLngConfig1) to request the receiving device to send the current value of (.variable_index) back to the requesting device. On the receiving end, when nviUnLngConfig1 sees this command the 68332 will send the value for (.variable_index) back to the requesting device as nvoUnLngConfig1 with the (.command) set to (set). (.command = loop) > Since this variable has the possibility of being over-written in the Neuron receiving device, the (.command = loop) is used by the Neuron as a loop back to indicate that the nvoUnLngConfig1 was received and a new nvoUnLngConfig1 can be sent.
nviVelocity1A, **nviVelocity1AB,** **nviVelocity1B,** **nviVelocity2A,** **nviVelocity2AB,** **nviVelocity2B**	Type: UNVT_MixedStamp
	Units: Dependent on geometry unit variable (meters or feet, therefore meters/sec or feet/sec).
	A structure of floating point and integer data from TIE1/2 measurement Path-A/AB/B that is time and mode stamped. The "_lin_velocity" variables are instantaneous and average linearized actual flow velocity. The "_raw_velocity" variables are instantaneous and average raw actual flow velocity (prior to linearization). The "primary_status" variable is the primary status word. The "extended_status" variable is the extended status word.
nviVersionMIO	Type: SNVT_time_stamp
	Units: Dimensionless

	A structure of integer data from MIO to indicate what version of software is in place. definition of Neuron SNVT_time_stamp structure used for version outputs unsigned long year ; #### year of version unsigned short month ; ## month of version unsigned short day ; ## day of version unsigned short hour ; ##._ version integer unsigned short minute ; _.## version fraction unsigned short second ; ## intermediate engineering versions (typ 0)
nviVersionTIE1, **nviVersionTIE2**	Type: UNVT_VersionData
	Units: Dimensionless
	A structure of integer data from TIE1/2 to indicate what version of software is in place. definition of Neuron SNVT_time_stamp structure used for version outputs unsigned long year ; #### year of version unsigned short month ; ## month of version unsigned short day ; ## day of version unsigned short hour ; ##._ version integer unsigned short minute ; _.## version fraction unsigned short second ; ## intermediate engineering versions (typ 0) FPGA is an 8-bit byte in the form of VVVVVFFF where (VVVVV) is the version number of the FPGA firmware, and FFF is the filter type. FFF = (001=14kHz, 010=20kHz, 100=50kHz)
nviVolume1A, **nviVolume1AB,** **nviVolume1B,** **nviVolume2A,** **nviVolume2AB,** **nviVolume2B**	Type: UNVT_FloatStamp
	Units: Dependent on Geometry Unit (meters or feet) and Flow Measurement Unit variables [Kx/Hour, x/Hour, Kx/Minute, x/Minute, Kx/Sec, x/Sec (where x = m^3 or ft^3 based on "Geometry Unit" variable)].
	A structure of floating point and integer data from TIE1/2 measurement Path-A/AB/B that is time and mode stamped. The "data1" variables are instantaneous and average linearized actual flow volume. The "data2" variables are linearized instantaneous and average standard flow volume when temperature and pressure correction are both enabled (each correction may be individually enabled by another variable). If both are disabled, they are equal to actual volume.
nviZero1, **nviZero2**	Type: UNVT_StampedCal
	Units: Dependent on geometry and flow volume unit variables.

	A structure of floating point data from TIE1/2 measurement paths A and B that is time stamped. The floating point data is the last average zero calibration data from TIE1/2 measurement paths A and B. The time stamp is the end time of the averaging interval for when the data first propagated.
nvoModeRequest	Type: UNVT_ModeRequest
	Units: Not applicable.
	Used to transmit commands from the Remote Display and Multi I/O nodes to the TIE1 and TIE2 and Multi I/O nodes. This command may need to be sent multiple times if more than one node needs to change mode. The node_id member is used to designate which node should receive the command: 0 undefined destination 1 represents the TIE1 2 represents the TIE2 3 represents the Multi I/O. The mode member is an enumerated data type that invokes the command. The values for the enumerated type mode are as follows: 0 = UNKNOWN, 1 = NORMAL, 2 = SPANHIGH, 3 = SPANLOW, 4 = ZERO, 5 = NORMAL_ACQUIRE, 6 = SPANHIGH_ACQUIRE, 7 = SPANLOW_ACQUIRE, 8 = ZERO_ACQUIRE, 9 = DIAGNOSTIC, 10 = TEST_FULL_SCALE, 11 = TEST_MID_SCALE, 12 = TEST_ZERO_SCALE, 13 = CAL_CYCLE, 14 = PUT_OUT_OF_SERVICE, 15 = PUT_IN_SERVICE, 16 = RESET, 17 = RECONFIG "Acquire" type modes are invoked during transitions. For example, if a TIE was in NORMAL mode but then went into SPANHIGH, the SPANHIGH mode will not be invoked until enough averages have been accumulated to conclude that the average values are now composed of all SPANHIGH data. During this transition phase, the SPANHIGH_ACQUIRE mode is invoked.

Header File Definitions for the Ultraflow 150

```
/*
 *      Copyright (c) 2006 by Teledyne Monitor Labs Inc.
 *
 *  This software is copyrighted by and is the sole property of
 *  Teledyne Monitor Labs.  All rights, title, ownership, or other interests
 *  in the software remain the property of Teledyne Monitor Labs.  This
 *  software may only be used in accordance with the corresponding
 *  license agreement.  Any unauthorized use, duplication, transmission,
 *  distribution, or disclosure of this software is expressly forbidden.
 *
 *  This Copyright notice may not be removed or modified without prior
 *  written consent of Teledyne Monitor Labs.
 *
 *  Teledyne Monitor Labs, reserves the right to modify this software
 *  without notice.
 *
 *  Teledyne Monitor Labs, Inc.
 *  Gibsonia Facility                       Phone: 724-444-5000
 *  5310 N. Pioneer Rd.                     Fax: 724-444-5050
 *  Gibsonia, PA 15044, USA                 http://www.teledyne-ml.com
 *
 ******************************************************************************
 *
 *     Module Name: U150DEF.H
 *         Version: 1.00
 *   Original Date: 08/16/2006
 *          Author: Ivica Eftimovski
 *        Language: Ansi C
 * Compile Options:
 * Compile defines:
 *       Libraries:
 *    Link Options:
 *
 *
 * Description.
 * ============================================================================
 *  This file defines variables needed by the Ultraflow 150.
 *
 *
 * Edit Date/Ver   Edit Description
 * ==============  ====================================================
 *
 *
 */

#ifndef ULTRAFLOW150DEFINITIONS_HEADER_FILE
#define ULTRAFLOW150DEFINITIONS_HEADER_FILE

#ifdef __cplusplus
extern "C"
{
#endif
```

```
/* Constants and Enumerations */

/* TIE Modes */
typedef  enum  node_mode
{
    UNKNOWN=0,          //  0
    NORMAL,             //  1
    SPANHIGH,           //  2
    SPANLOW,            //  3
    ZERO,               //  4
    NORMAL_ACQUIRE,     //  5
    SPANHIGH_ACQUIRE,   //  6
    SPANLOW_ACQUIRE,    //  7
    ZERO_ACQUIRE,       //  8
    DIAGNOSTIC,         //  9
    TEST_FULL_SCALE,    // 10
    TEST_MID_SCALE,     // 11
    TEST_ZERO_SCALE,    // 12
    CAL_CYCLE,          // 13
    PUT_OUT_OF_SERVICE, // 14
    PUT_IN_SERVICE,     // 15
    RESET,              // 16
    RECONFIG            // 17
} node_mode ;

/*****************************************************************/

#define  LAST_OBJECT  LAST_ONE-1
#define  MAX_NV_NUM   LAST_ONE-1
#define  MAX_N_V_NUM  LAST_ONE-1
#define  BASE_ID      1
/*****************************************************************/

/* Index variables commands */
typedef  enum  loop_commands
{
   SET_CONFIG = 0,    //  0
   READ_CONFIG,       //  1
   LOOP_BACK          //  2
} loop_commands ;
/*****************************************************************/

/* MIO Isolator Settings */
#define ISO_FULL_SCALE      0x80 /* binary10000000 */
#define ISO_ZERO_SCALE      0x40 /* binary01000000 */
#define ISO_MID_SCALE       0xC0 /* binary11000000 */

#define ISO_CAL_CYCLE_TIE2  0x20 /* binary00100000 */
#define ISO_ZERO_TIE2       0x10 /* binary00010000 */
#define ISO_SPAN_LOW_TIE2   0x18 /* binary00011000 */
#define ISO_SPAN_HIGH_TIE2  0x08 /* binary00001000 */

#define ISO_CAL_CYCLE_TIE1  0x04 /* binary00000100 */
#define ISO_ZERO_TIE1       0x02 /* binary00000010 */
#define ISO_SPAN_LOW_TIE1   0x03 /* binary00000011 */
#define ISO_SPAN_HIGH_TIE1  0x01 /* binary00000001 */
```

```
#define EXP       2     /* for expanded calibration configurations */
#define WITH      1     /* for with calibration configurations     */
#define WITHOUT   0     /* for without calibration configurations  */

#define VALID     1     /* used by fault flags to indicate status  */
#define INVALID   42
/*****************************************************************/

// for analog configuration, A_output_type, B_output_type
typedef enum
{
      NO = 0,                       // 00
      INSTANT_VELOCITY,             // 01
      AVG_VELOCITY,                 // 02
      INSTANT_RAW_VELOCITY,         // 03
      AVG_RAW_VELOCTY,              // 04
      INSTANT_ACT_VOLUME,           // 05
      AVG_ACT_VOLUME,               // 06
      INSTANT_STD_VOLUME,           // 07
      AVG_STD_VOLUME,               // 08
      INST_MEDIUM_INT_TEMP,         // 09
      AVG_MEDIUM_INT_TEMP,          // 10
      INSTANT_SOUND_SPEED,          // 11
      AVG_SOUND_SPEED,              // 12
      AVG_SPAN_HIGH,                // 13
      AVG_SPAN_LOW,                 // 14
      AVG_ZERO,                     // 15

      SN_RATIO_UPSTREAM,            // 16
      SN_RATIO_DOWNSTREAM,          // 17
      DIGITAL_GAIN_UPSTREAM,        // 18
      DIGITAL_GAIN_DOWNSTREAM,      // 19
      PREAMP_GAIN_UPSTREAM,         // 20
      PREAMP_GAIN_DOWNSTREAM,       // 21
      PEAK_RAM_COUNT_UPSTREAM,      // 22
      PEAK_RAM_COUNT_DOWNSTREAM,    // 23
      CALC_RAM_COUNT_UPSTREAM,      // 24
      CALC_RAM_COUNT_DOWNSTREAM,    // 25

      INSTANT_MEDIUM_EXT_TEMP,      // 26
      AVG_MEDIUM_EXT_TEMP,          // 27
      INSTANT_MEDIUM_PRESSURE,      // 28
      AVG_MEDIUM_PRESSURE,          // 29
} AO_type;
/*****************************************************************/

typedef enum  //for digital configuration, DIO_type
{
      NO_SELECTION = 0,                       // 00

      INST_VOLUME_ALARM_TIE1,                 // 01
      AVG_VOLUME_ALARM_TIE1,                  // 02

      INST_MEDIUM_TEMP_ALARM_TIE1,            // 03
      AVG_MEDIUM_TEMP_ALARM_TIE1,             // 04
      SPAN_HIGH_ON_AO_TIE1,                   // 05
```

```
SPAN_LOW_ON_AO_TIE1,                    // 06
ZERO_ON_AO_TIE1,                        // 07
NORMAL_ON_AO_TIE1,                      // 08
CAL_ON_AO_TIE1,                         // 09
FATAL_FAULT_TIE1A,                      // 10
NONFATAL_FAULT_TIE1A,                   // 11
DATA_VALID_TIE1A,                       // 12

INTERFERENCE_TEST_TIE1A,                // 13
PURGE_FAILURE_TIE1A,                    // 14
CAL_FAILURE_TIE1A,                      // 15
SN_RATIO_UPSTREAM_ALARM_TIE1A,          // 16
SN_RATIO_DOWNSTREAM_ALARM_TIE1A,        // 17
SN_RATIO_BOTH_ALARM_TIE1A,              // 18

FATAL_FAULT_TIE1B,                      // 19
NONFATAL_FAULT_TIE1B,                   // 20
DATA_VALID_TIE1B,                       // 21
INTERFERENCE_TEST_TIE1B,                // 22
PURGE_FAILURE_TIE1B,                    // 23

CAL_FAILURE_TIE1B,                      // 24
SN_RATIO_UPSTREAM_ALARM_TIE1B,          // 25
SN_RATIO_DOWNSTREAM_ALARM_TIE1B,        // 26
SN_RATIO_BOTH_ALARM_TIE1B,              // 27

FATAL_FAULT_TIE1AorB,                   // 28
NONFATAL_FAULT_TIE1AorB,                // 29
DATA_VALID_TIE1AorB,                    // 30
INTERFERENCE_TEST_TIE1AorB,             // 31
PURGE_FAILURE_TIE1AorB,                 // 32
CAL_FAILURE_TIE1AorB,                   // 33
SN_RATIO_UPSTREAM_ALARM_TIE1AorB,       // 34
SN_RATIO_DOWNSTREAM_ALARM_TIE1AorB,     // 35
SN_RATIO_BOTH_ALARM_TIE1AORB,           // 36

INST_VOLUME_ALARM_TIE2,                 // 37
AVG_VOLUME_ALARM_TIE2,                  // 38
INST_MEDIUM_TEMP_ALARM_TIE2,            // 39
AVG_MEDIUM_TEMP_ALARM_TIE2,             // 40
SPAN_HIGH_ON_AO_TIE2,                   // 41
SPAN_LOW_ON_AO_TIE2,                    // 42
ZERO_ON_AO_TIE2,                        // 43
NORMAL_ON_AO_TIE2,                      // 44
CAL_ON_AO_TIE2,                         // 45

FATAL_FAULT_TIE2A,                      // 46
NONFATAL_FAULT_TIE2A,                   // 47
DATA_VALID_TIE2A,                       // 48
INTERFERENCE_TEST_TIE2A,                // 49
PURGE_FAILURE_TIE2A,                    // 50
CAL_FAILURE_TIE2A,                      // 51
SN_RATIO_UPSTREAM_ALARM_TIE2A,          // 52
SN_RATIO_DOWNSTREAM_ALARM_TIE2A,        // 53
SN_RATIO_BOTH_ALARM_TIE2A,              // 54
FATAL_FAULT_TIE2B,                      // 55
NONFATAL_FAULT_TIE2B,                   // 56
```

```
      DATA_VALID_TIE2B,                         // 57
      INTERFERENCE_TEST_TIE2B,                  // 58
      PURGE_FAILURE_TIE2B,                      // 59
      CAL_FAILURE_TIE2B,                        // 60
      SN_RATIO_UPSTREAM_ALARM_TIE2B,            // 61
      SN_RATIO_DOWNSTREAM_ALARM_TIE2B,          // 62
      SN_RATIO_BOTH_ALARM_TIE2B,                // 63

      FATAL_FAULT_TIE2AorB,                     // 64
      NONFATAL_FAULT_TIE2AorB,                  // 65
      DATA_VALID_TIE2AorB,                      // 66
      INTERFERENCE_TEST_TIE2AorB,               // 67

      PURGE_FAILURE_TIE2AorB,                   // 68
      CAL_FAILURE_TIE2AorB,                     // 69
      SN_RATIO_UPSTREAM_ALARM_TIE2AorB,         // 70
      SN_RATIO_DOWNSTREAM_ALARM_TIE2AorB,       // 71
      SN_RATIO_BOTH_ALARM_TIE2AORB,             // 72

      SPAN_HIGH_OR_ACQ_ON_AO_TIE1,              // 73
      SPAN_LOW_OR_ACQ_ON_AO_TIE1,               // 74
      ZERO_OR_ACQ_ON_AO_TIE1,                   // 75
      NORMAL_OR_ACQ_ON_AO_TIE1,                 // 76
      CAL_OR_ACQ_ON_AO_TIE1,                    // 77
      SPAN_HIGH_OR_ACQ_ON_AO_TIE2,              // 78

      SPAN_LOW_OR_ACQ_ON_AO_TIE2,               // 79
      ZERO_OR_ACQ_ON_AO_TIE2,                   // 80
      NORMAL_OR_ACQ_ON_AO_TIE2,                 // 81
      CAL_OR_ACQ_ON_AO_TIE2,                    // 82

} DIO_type;
/*****************************************************************/

//
// Note: Created TIE_type and UNVT_AIO_Config.
// Only path_and_TIE[0] & path_and_TIE[1] are used.
// Analog output configuration variables for Ultraflow 150:
//
//  For analog output A, path_and_TIE[0] defines
//  TIE1_PATH_A,TIE1_PATH_B,TIE1_PATHSAB_AVG,
//  TIE2_PATH_A,TIE2_PATH_B,TIE2_PATHSAB_AVG,
//
//  For analog output B, path_and_TIE[1] defines
//  TIE1_PATH_A,TIE1_PATH_B,TIE1_PATHSAB_AVG,
//  TIE2_PATH_A,TIE2_PATH_B,TIE2_PATHSAB_AVG,
// (uses from TIE_type enum)
/* ------------------------------------------------------------- */

typedef enum  //for analog output configuration, TIE_type
{
      TIE1_PATH_A,        // 00 For AO_type configuration only.
      TIE1_PATH_B,        // 01 For AO_type configuration only.
      TIE1_PATHSAB_AVG,   // 02 (A+B)/2, For AO_type configuration only.
      TIE2_PATH_A,        // 03 For AO_type configuration only.
      TIE2_PATH_B,        // 04 For AO_type configuration only.
```

```
      TIE2_PATHSAB_AVG,   // 05 For AO_type configuration only.
} TIE_type;
/*****************************************************************/

/* NETWORK VARIABLES */

 struct __SNVT_time_stamp__{
     WORD16   year;   /* = #### 16-bits */
     BYTE  month;  /* = ##    8-bits */
     BYTE  day;    /* = ##    8-bits */
     BYTE  hour;   /* = ##    8-bits */
     BYTE  minute; /* = ##    8-bits */
     BYTE  second; /* = ##    8-bits */
 } __attribute__ ((__packed__));

typedef struct __SNVT_time_stamp__ SNVT_time_stamp;

extern    SNVT_time_stamp    *nviTime1 ;
extern    SNVT_time_stamp    *nviTime2 ;
extern    SNVT_time_stamp    *nvoTimeSet ;
extern    SNVT_time_stamp    *nviVersionMIO ;


struct __UNVT_VersionData__{
     WORD16  year332;   /* #### year of version          */
     BYTE   month332;  /* ##   month of version          */
     BYTE   day332;    /* ##   day of version            */
     BYTE   hour332;   /* ##._ version integer           */
     BYTE   minute332; /* _.## version fraction          */
     BYTE   second332; /* ##   intermed. engr ver(typ 0) */

     BYTE   FPGA;

     WORD16  yearN;   /* #### year of version            */
     BYTE   monthN;  /* ##   month of version            */
     BYTE   dayN;    /* ##   day of version              */
     BYTE   hourN;   /* ##._ version integer             */
     BYTE   minuteN; /* _.## version fraction            */
     BYTE   secondN; /* ##   intermediate engr ver(typ 0)*/
} __attribute__ ((__packed__));

typedef struct __UNVT_VersionData__ UNVT_VersionData;

extern    UNVT_VersionData   *nviVersionTIE1 ;
extern    UNVT_VersionData   *nviVersionTIE2 ;


struct __UNVT_ModeRequest__{
   BYTE  node_id;      /* Source=>  1=ERP  2=Multi I/O */
   BYTE  mode;         /* (Enum: UNKNOWN,NORMAL,etc.)  */
}__attribute__ ((__packed__));

typedef struct __UNVT_ModeRequest__ UNVT_ModeRequest;

extern    UNVT_ModeRequest   *nvoModeRequest ;
```

```
struct __UNVT_UnLongArray__{
   WORD16  variable_index;      /* 0 to 65,535           */
   WORD16  un_long_data;        /* 0 to 65,535           */
   WORD16  command;             /* 0=set, 1=read, 2=loop */
}__attribute__ ((__packed__));

typedef struct __UNVT_UnLongArray__ UNVT_UnLongArray;

extern    UNVT_UnLongArray   *nviUnLngConfig1 ;
extern    UNVT_UnLongArray   *nviUnLngConfig2 ;
extern    UNVT_UnLongArray   *nvoUnLngConfig1 ;
extern    UNVT_UnLongArray   *nvoUnLngConfig2 ;


struct __UNVT_FloatArray__{
      WORD16      variable_index;      /* 0 to 65,535           */
      float float_data;          /*                       */
      WORD16      command;             /* 0=set, 1=read, 2=loop */
}__attribute__ ((__packed__));

typedef struct __UNVT_FloatArray__ UNVT_FloatArray;

extern    UNVT_FloatArray    *nviFloatData1 ;
extern    UNVT_FloatArray    *nviFloatData2 ;
extern    UNVT_FloatArray    *nvoFloatData1 ;
extern    UNVT_FloatArray    *nvoFloatData2 ;
extern    UNVT_FloatArray    *nviFloatConfig1 ;
extern    UNVT_FloatArray    *nviFloatConfig2 ;
extern    UNVT_FloatArray    *nvoFloatConfig1 ;
extern    UNVT_FloatArray    *nvoFloatConfig2 ;


struct __UNVT_ServiceData__{
   float    up_SN_ratio;
   float    down_SN_ratio;
   BYTE     up_digital_gain;              /* 0 to 255    */
   BYTE     down_digital_gain;            /* 0 to 255    */
   WORD16   up_preamp_gain;               /* 0 to 4095   */
   WORD16   down_preamp_gain;             /* 0 to 4095   */
   WORD16   up_peak_RAM_counts;           /* 0 to 65,535 */
   WORD16   down_peak_RAM_counts;         /* 0 to 65,535 */
   float    up_calc_RAM_counts;
   float    down_calc_RAM_counts;
} __attribute__ ((__packed__));

typedef struct __UNVT_ServiceData__ UNVT_ServiceData;

extern    UNVT_ServiceData   *nviServceData1A ;
extern    UNVT_ServiceData   *nviServceData1B ;
extern    UNVT_ServiceData   *nviServceData2A ;
extern    UNVT_ServiceData   *nviServceData2B ;


struct __UNVT_FloatStamp__{
   WORD16   year;
   BYTE     month;
   BYTE     day;
```

```
   BYTE     hour;
   BYTE     minute;
   BYTE     second;

   BYTE     mode;

   float    inst_float_data1;
   float    avg_float_data1;
   float    inst_float_data2;
   float    avg_float_data2;
} __attribute__ ((__packed__));

typedef struct __UNVT_FloatStamp__ UNVT_FloatStamp;

extern    UNVT_FloatStamp    *nviPressTemp1 ;
extern    UNVT_FloatStamp    *nviPressTemp2 ;
extern    UNVT_FloatStamp    *nviSoundTemp1A ;
extern    UNVT_FloatStamp    *nviSoundTemp1AB ;
extern    UNVT_FloatStamp    *nviSoundTemp1B ;
extern    UNVT_FloatStamp    *nviSoundTemp2A ;
extern    UNVT_FloatStamp    *nviSoundTemp2AB ;
extern    UNVT_FloatStamp    *nviSoundTemp2B ;
extern    UNVT_FloatStamp    *nviVolume1A ;
extern    UNVT_FloatStamp    *nviVolume1AB ;
extern    UNVT_FloatStamp    *nviVolume1B ;
extern    UNVT_FloatStamp    *nviVolume2A ;
extern    UNVT_FloatStamp    *nviVolume2AB ;
extern    UNVT_FloatStamp    *nviVolume2B ;

struct __UNVT_MixedStamp__{
   WORD16   year;
   BYTE     month;
   BYTE     day;
   BYTE     hour;
   BYTE     minute;
   BYTE     second;

   BYTE     mode;

   float    inst_lin_velocity;     /* linearized velocity */
   float    avg_lin_velocity;      /* linearized velocity */
   float    inst_raw_velocity;     /* raw velocity        */
   float    avg_raw_velocity;      /* raw velocity        */
   WORD16   primary_status;        /* primary status      */
   WORD16   extended_status;       /* extended status     */
} __attribute__ ((__packed__));

typedef struct __UNVT_MixedStamp__ UNVT_MixedStamp;

extern    UNVT_MixedStamp    *nviVelocity1A ;
extern    UNVT_MixedStamp    *nviVelocity1AB ;
extern    UNVT_MixedStamp    *nviVelocity1B ;
extern    UNVT_MixedStamp    *nviVelocity2A ;
extern    UNVT_MixedStamp    *nviVelocity2AB ;
extern    UNVT_MixedStamp    *nviVelocity2B ;
```

```
struct __UNVT_StampedCal__{
   WORD16   year;
   BYTE     month;
   BYTE     day;
   BYTE     hour;
   BYTE     minute;
   BYTE     second;

   BYTE     filler;
   float    avg_pathA_data;
   float    avg_pathB_data;
   float    avg_pathAB_data;
} __attribute__ ((__packed__));

typedef struct __UNVT_StampedCal__ UNVT_StampedCal;

extern    UNVT_StampedCal    *nviSpanHigh1 ;
extern    UNVT_StampedCal    *nviSpanHigh2 ;
extern    UNVT_StampedCal    *nviSpanLow1 ;
extern    UNVT_StampedCal    *nviSpanLow2 ;
extern    UNVT_StampedCal    *nviZero1 ;
extern    UNVT_StampedCal    *nviZero2 ;


struct __UNVT_AlarmsU150__{
      BYTE  Bit_7      : 1;     /* Bit_00 = 07 not used          */
      BYTE  Bit_6      : 1;     /* Bit_01 = 06 not used          */
      BYTE  Bit_5      : 1;     /* Bit_02 = 05 not used          */
      BYTE  Bit_4      : 1;     /* Bit_03 = 04 not used          */
      BYTE  AveTemp    : 1;     /* Bit_04 = 03 Ave Medium Temp.  */
      BYTE  InstTemp   : 1;     /* Bit_05 = 02 Inst Medium Temp. */
      BYTE  AvgVolume  : 1;     /* Bit_06 = 01 Ave Volume        */
      BYTE  InstVolume : 1;     /* Bit_07 = 00 Inst Volume       */

} __attribute__ ((__packed__));

typedef struct __UNVT_AlarmsU150__ UNVT_AlarmsU150;

extern    UNVT_AlarmsU150    *nviAlarmState1 ;
extern    UNVT_AlarmsU150    *nviAlarmState2 ;


struct __SNVT_alarm__{
   BYTE      location[ 6 ];
   WORD16    object_id;
   BYTE      alarm_type;
   BYTE      priority_level;
   WORD16    index_to_SNVT;
   BYTE      value[ 4 ];
   WORD16    year;
   BYTE      month;
   BYTE      day;
   BYTE      hour;
   BYTE      minute;
   BYTE      second;
   WORD16    millisecond;
   BYTE      alarm_limit[ 4 ];
```

```
 } __attribute__ ((__packed__));

typedef struct __SNVT_alarm__ SNVT_alarm;

extern    SNVT_alarm         *nviAlarms ;


struct __UNVT_AIO_Config__
{
     BYTE         A_output_type ;
     BYTE         A_calibration ;
     float        A_zero ;
     float        A_full_scale ;

     BYTE         B_output_type ;
     BYTE         B_calibration ;
     float        B_zero ;
     float        B_full_scale ;

     BYTE         path_and_TIE[8] ;

} __attribute__ ((__packed__));

typedef struct __UNVT_AIO_Config__ UNVT_AIO_Config;

extern    UNVT_AIO_Config    *nviMIO_Config_1 ;
extern    UNVT_AIO_Config    *nvoMIO_Config_1 ;
extern    UNVT_AIO_Config    *nviMIO_Config_2 ;
extern    UNVT_AIO_Config    *nvoMIO_Config_2 ;


struct __UNVT_MIO_Config__
{
     BYTE         A_output_type ;
     BYTE         A_calibration ;
     float        A_zero ;
     float        A_full_scale ;

     BYTE         B_output_type ;
     BYTE         B_calibration ;
     float        B_zero ;
     float        B_full_scale ;

     BYTE         digital_type[8] ;

} __attribute__ ((__packed__));

typedef struct __UNVT_MIO_Config__ UNVT_MIO_Config;

extern    UNVT_MIO_Config    *nviMIO_Config_3 ;
extern    UNVT_MIO_Config    *nvoMIO_Config_3 ;


struct __UNVT_MIO_Status__
{
     BYTE         invalid_mode_request    : 1 ;
     BYTE         invalid_AO_config       : 1 ;
```

```
    BYTE        invalid_DIO_config      : 1 ;
    BYTE        reserved                : 5 ;  /* Mode of TIE2 */
    BYTE        MIO_mode ;                     /* Mode of TIE1 */

}  __attribute__ ((__packed__));

typedef struct __UNVT_MIO_Status__ UNVT_MIO_Status;

extern    UNVT_MIO_Status    *nviMIO_Status ;


#ifdef __cplusplus
}
#endif

#endif
```

Modbus Holding Registers for Ultraflow 150

Variable	Member	Type	Large Address	Offset
nviAlarms		**SNVT_alarm**		
	location[0]	BYTE	40002	1
	location[1]	BYTE		
	location[2]	BYTE	40003	2
	location[3]	BYTE		
	location[4]	BYTE	40004	3
	location[5]	BYTE		
	object_id	WORD (HIBYTE)	40005	4
		WORD (LOWBYTE)		
	alarm_type	BYTE	40006	5
	priority_level	BYTE		
	index_to_SNVT	WORD (HIBYTE)	40007	6
		WORD (LOWBYTE)		
	value[0]	BYTE	40008	7
	value[1]	BYTE		
	value[2]	BYTE	40009	8
	value[3]	BYTE		
	year	WORD (HIBYTE)	40010	9
		WORD (LOWBYTE)		
	month	BYTE	40011	10
	day	BYTE		
	hour	BYTE	40012	11
	minute	BYTE		
	second	BYTE	40013	12
	millisecond	WORD (HIBYTE)		
		WORD (LOWBYTE)	40014	13
	alarm_limit[0]	BYTE		
	alarm_limit[1]	BYTE	40015	14
	alarm_limit[2]	BYTE		
	alarm_limit[3]	BYTE	40016	15
	FILLER	BYTE		
nviAlarmState1		**UNVT_AlarmsU150**		
	enum	BYTE	40017	16
	FILLER	BYTE		
nviAlarmState2		**UNVT_AlarmsU150**		
	enum	BYTE	40018	17
	FILLER	BYTE		
nviFloatConfig1		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40019	18

		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40020	19
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40021	20
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40022	21
		WORD (LOWBYTE)		
nviFloatConfig2		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40023	22
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40024	23
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40025	24
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40026	25
		WORD (LOWBYTE)		
nviFloatData1		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40027	26
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40028	27
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40029	28
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40030	29
		WORD (LOWBYTE)		
nviFloatData1		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40031	30
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40032	31
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40033	32
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40034	33
		WORD (LOWBYTE)		
nviMIO_Config_1		**UNVT_MIO_Config**		
	A_output_type	BYTE	40035	34
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40036	35
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40037	36
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40038	37
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40039	38
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40040	39
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40041	40
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40042	41
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40043	42
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40044	43
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40045	44
	digital_type[1]	BYTE		
	digital_type[2]	BYTE	40046	45
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40047	46
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40048	47
	digital_type[7]	BYTE		
nviMIO_Config_2		**UNVT_MIO_Config**		
	A_output_type	BYTE	40049	48
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40050	49
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40051	50
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40052	51
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40053	52
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40054	53
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40055	54
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40056	55
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40057	56
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40058	57
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40059	58
	digital_type[1]	BYTE		

	digital_type[2]	BYTE	40060	59
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40061	60
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40062	61
	digital_type[7]	BYTE		
nviMIO_Config_3		**UNVT_MIO_Config**		
	A_output_type	BYTE	40063	62
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40064	63
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40065	64
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40066	65
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40067	66
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40068	67
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40069	68
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40070	69
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40071	70
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40072	71
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40073	72
	digital_type[1]	BYTE		
	digital_type[2]	BYTE	40074	73
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40075	74
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40076	75
	digital_type[7]	BYTE		
nviMIO_Status		**UNVT_MIO_Status**		
	status & reserved	BYTE	40077	76
	MIO_mode	BYTE		
nviPressTemp1		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40078	77
		WORD (LOWBYTE)		
	month	BYTE	40079	78
	day	BYTE		

	hour	BYTE	40080	79
	minute	BYTE		
	second	BYTE	40081	80
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40082	81
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40083	82
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40084	83
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40085	84
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40086	85
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40087	86
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40088	87
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40089	88
		IEEE float (LOBYTE(LSM))		
nviPressTemp2		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40090	89
		WORD (LOWBYTE)		
	month	BYTE	40091	90
	day	BYTE		
	hour	BYTE	40092	91
	minute	BYTE		
	second	BYTE	40093	92
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40094	93
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40095	94
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40096	95
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40097	96
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40098	97
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40099	98
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40100	99
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40101	100
		IEEE float (LOBYTE(LSM))		
nviServceData1A		**UNVT_ServiceData**		
	up_SN_ratio	IEEE float (S+MSE)	40102	101
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40103	102
		IEEE float (LOBYTE(LSM))		
	down_SN_ratio	IEEE float (S+MSE)	40104	103
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40105	104
		IEEE float (LOBYTE(LSM))		
	up_digital_gain	BYTE	40106	105
	down_digital_gain	BYTE		
	up_analog_gain	WORD (HIBYTE)	40107	106
		WORD (LOWBYTE)		
	down_analog_gain	WORD (HIBYTE)	40108	107
		WORD (LOWBYTE)		
	up_peak_RAM_counts	WORD (HIBYTE)	40109	108
		WORD (LOWBYTE)		
	down_peak_RAM_counts	WORD (HIBYTE)	40110	109
		WORD (LOWBYTE)		
	up_calc_RAM_counts	IEEE float (S+MSE)	40111	110
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40112	111
		IEEE float (LOBYTE(LSM))		
	down_calc_RAM_counts	IEEE float (S+MSE)	40113	112
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40114	113
		IEEE float (LOBYTE(LSM))		
nviServceData1B		**UNVT_ServiceData**		
	up_SN_ratio	IEEE float (S+MSE)	40115	114
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40116	115
		IEEE float (LOBYTE(LSM))		
	down_SN_ratio	IEEE float (S+MSE)	40117	116
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40118	117
		IEEE float (LOBYTE(LSM))		
	up_digital_gain	BYTE	40119	118
	down_digital_gain	BYTE		
	up_analog_gain	WORD (HIBYTE)	40120	119
		WORD (LOWBYTE)		
	down_analog_gain	WORD (HIBYTE)	40121	120

		WORD (LOWBYTE)		
	up_peak_RAM_counts	WORD (HIBYTE)	40122	121
		WORD (LOWBYTE)		
	down_peak_RAM_counts	WORD (HIBYTE)	40123	122
		WORD (LOWBYTE)		
	up_calc_RAM_counts	IEEE float (S+MSE)	40124	123
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40125	124
		IEEE float (LOBYTE(LSM))		
	down_calc_RAM_counts	IEEE float (S+MSE)	40126	125
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40127	126
		IEEE float (LOBYTE(LSM))		
nviServceData2A		**UNVT_ServiceData**		
	up_SN_ratio	IEEE float (S+MSE)	40128	127
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40129	128
		IEEE float (LOBYTE(LSM))		
	down_SN_ratio	IEEE float (S+MSE)	40130	129
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40131	130
		IEEE float (LOBYTE(LSM))		
	up_digital_gain	BYTE	40132	131
	down_digital_gain	BYTE		
	up_analog_gain	WORD (HIBYTE)	40133	132
		WORD (LOWBYTE)		
	down_analog_gain	WORD (HIBYTE)	40134	133
		WORD (LOWBYTE)		
	up_peak_RAM_counts	WORD (HIBYTE)	40135	134
		WORD (LOWBYTE)		
	down_peak_RAM_counts	WORD (HIBYTE)	40136	135
		WORD (LOWBYTE)		
	up_calc_RAM_counts	IEEE float (S+MSE)	40137	136
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40138	137
		IEEE float (LOBYTE(LSM))		
	down_calc_RAM_counts	IEEE float (S+MSE)	40139	138
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40140	139
		IEEE float (LOBYTE(LSM))		
nviServceData2B		**UNVT_ServiceData**		
	up_SN_ratio	IEEE float (S+MSE)	40141	140
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40142	141
		IEEE float (LOBYTE(LSM))		
	down_SN_ratio	IEEE float (S+MSE)	40143	142
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40144	143
		IEEE float (LOBYTE(LSM))		
	up_digital_gain	BYTE	40145	144
	down_digital_gain	BYTE		
	up_analog_gain	WORD (HIBYTE)	40146	145
		WORD (LOWBYTE)		
	down_analog_gain	WORD (HIBYTE)	40147	146
		WORD (LOWBYTE)		
	up_peak_RAM_counts	WORD (HIBYTE)	40148	147
		WORD (LOWBYTE)		
	down_peak_RAM_counts	WORD (HIBYTE)	40149	148
		WORD (LOWBYTE)		
	up_calc_RAM_counts	IEEE float (S+MSE)	40150	149
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40151	150
		IEEE float (LOBYTE(LSM))		
	down_calc_RAM_counts	IEEE float (S+MSE)	40152	151
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40153	152
		IEEE float (LOBYTE(LSM))		
nviSoundTemp1A		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40154	153
		WORD (LOWBYTE)		
	month	BYTE	40155	154
	day	BYTE		
	hour	BYTE	40156	155
	minute	BYTE		
	second	BYTE	40157	156
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40158	157
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40159	158
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40160	159
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40161	160
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40162	161
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40163	162
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40164	163
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40165	164
		IEEE float (LOBYTE(LSM))		
nviSoundTemp1AB		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40166	165
		WORD (LOWBYTE)		
	month	BYTE	40167	166
	day	BYTE		
	hour	BYTE	40168	167
	minute	BYTE		
	second	BYTE	40169	168
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40170	169
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40171	170
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40172	171
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40173	172
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40174	173
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40175	174
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40176	175
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40177	176
		IEEE float (LOBYTE(LSM))		
nviSoundTemp1B		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40178	177
		WORD (LOWBYTE)		
	month	BYTE	40179	178
	day	BYTE		
	hour	BYTE	40180	179
	minute	BYTE		
	second	BYTE	40181	180
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40182	181
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40183	182

		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40184	183
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40185	184
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40186	185
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40187	186
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40188	187
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40189	188
		IEEE float (LOBYTE(LSM))		
nviSoundTemp2A		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40190	189
		WORD (LOWBYTE)		
	month	BYTE	40191	190
	day	BYTE		
	hour	BYTE	40192	191
	minute	BYTE		
	second	BYTE	40193	192
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40194	193
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40195	194
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40196	195
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40197	196
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40198	197
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40199	198
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40200	199
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40201	200
		IEEE float (LOBYTE(LSM))		
nviSoundTemp2AB		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40202	201
		WORD (LOWBYTE)		
	month	BYTE	40203	202
	day	BYTE		

	hour	BYTE	40204	203
	minute	BYTE		
	second	BYTE	40205	204
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40206	205
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40207	206
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40208	207
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40209	208
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40210	209
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40211	210
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40212	211
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40213	212
		IEEE float (LOBYTE(LSM))		
nviSoundTemp2B		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40214	213
		WORD (LOWBYTE)		
	month	BYTE	40215	214
	day	BYTE		
	hour	BYTE	40216	215
	minute	BYTE		
	second	BYTE	40217	216
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40218	217
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40219	218
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40220	219
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40221	220
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40222	221
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40223	222
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40224	223
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40225	224
		IEEE float (LOBYTE(LSM))		
nviSpanHigh1		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40226	225
		WORD (LOWBYTE)		
	month	BYTE	40227	226
	day	BYTE		
	hour	BYTE	40228	227
	minute	BYTE		
	second	BYTE	40229	228
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40230	229
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40231	230
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40232	231
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40233	232
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	40234	233
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40235	234
		IEEE float (LOBYTE(LSM))		
nviSpanHigh2		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40236	235
		WORD (LOWBYTE)		
	month	BYTE	40237	236
	day	BYTE		
	hour	BYTE	40238	237
	minute	BYTE		
	second	BYTE	40239	238
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40240	239
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40241	240
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40242	241
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40243	242
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	40244	243
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40245	244

		IEEE float (LOBYTE(LSM))		
nviSpanLow1		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40246	245
		WORD (LOWBYTE)		
	month	BYTE	40247	246
	day	BYTE		
	hour	BYTE	40248	247
	minute	BYTE		
	second	BYTE	40249	248
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40250	249
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40251	250
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40252	251
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40253	252
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	40254	253
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40255	254
		IEEE float (LOBYTE(LSM))		
nviSpanLow2		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40256	255
		WORD (LOWBYTE)		
	month	BYTE	40257	256
	day	BYTE		
	hour	BYTE	40258	257
	minute	BYTE		
	second	BYTE	40259	258
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40260	259
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40261	260
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40262	261
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40263	262
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	40264	263
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40265	264
		IEEE float (LOBYTE(LSM))		

nviTime1		**SNVT_time_stamp**		
	year	WORD (HIBYTE)	40266	265
		WORD (LOWBYTE)		
	month	BYTE	40267	266
	day	BYTE		
	hour	BYTE	40268	267
	minute	BYTE		
	second	BYTE	40269	268
	FILLER	BYTE		
nviTime2		**SNVT_time_stamp**		
	year	WORD (HIBYTE)	40270	269
		WORD (LOWBYTE)		
	month	BYTE	40271	270
	day	BYTE		
	hour	BYTE	40272	271
	minute	BYTE		
	second	BYTE	40273	272
	FILLER	BYTE		
nviUnLngConfig1		**UNVT_UnLongArray**		
	variable_index	WORD (HIBYTE)	40274	273
		WORD (LOWBYTE)		
	un_long_data	WORD (HIBYTE)	40275	274
		WORD (LOWBYTE)		
	command	WORD (HIBYTE)	40276	275
		WORD (LOWBYTE)		
nviUnLngConfig2		**UNVT_UnLongArray**		
	variable_index	WORD (HIBYTE)	40277	276
		WORD (LOWBYTE)		
	un_long_data	WORD (HIBYTE)	40278	277
		WORD (LOWBYTE)		
	command	WORD (HIBYTE)	40279	278
		WORD (LOWBYTE)		
nviVelocity1A		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40280	279
		WORD (LOWBYTE)		
	month	BYTE	40281	280
	day	BYTE		
	hour	BYTE	40282	281
	minute	BYTE		
	second	BYTE	40283	282
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40284	283
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	40285	284
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40286	285
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40287	286
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40288	287
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40289	288
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40290	289
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40291	290
		IEEE float (LOBYTE(LSM))		
	primary_status	WORD (HIBYTE)	40292	291
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40293	292
		WORD (LOWBYTE)		
nviVelocity1AB		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40294	293
		WORD (LOWBYTE)		
	month	BYTE	40295	294
	day	BYTE		
	hour	BYTE	40296	295
	minute	BYTE		
	second	BYTE	40297	296
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40298	297
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40299	298
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40300	299
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40301	300
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40302	301
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40303	302
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40304	303
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40305	304
		IEEE float (LOBYTE(LSM))		

	primary_status	WORD (HIBYTE)	40306	305
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40307	306
		WORD (LOWBYTE)		
nviVelocity1B		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40308	307
		WORD (LOWBYTE)		
	month	BYTE	40309	308
	day	BYTE		
	hour	BYTE	40310	309
	minute	BYTE		
	second	BYTE	40311	310
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40312	311
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40313	312
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40314	313
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40315	314
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40316	315
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40317	316
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40318	317
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40319	318
		IEEE float (LOBYTE(LSM))		
	primary_status	WORD (HIBYTE)	40320	319
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40321	320
		WORD (LOWBYTE)		
nviVelocity2A		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40322	321
		WORD (LOWBYTE)		
	month	BYTE	40323	322
	day	BYTE		
	hour	BYTE	40324	323
	minute	BYTE		
	second	BYTE	40325	324
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40326	325

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40327	326
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40328	327
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40329	328
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40330	329
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40331	330
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40332	331
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40333	332
		IEEE float (LOBYTE(LSM))		
	primary_status	WORD (HIBYTE)	40334	333
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40335	334
		WORD (LOWBYTE)		
nviVelocity2AB		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40336	335
		WORD (LOWBYTE)		
	month	BYTE	40337	336
	day	BYTE		
	hour	BYTE	40338	337
	minute	BYTE		
	second	BYTE	40339	338
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40340	339
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40341	340
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40342	341
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40343	342
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40344	343
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40345	344
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40346	345
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40347	346

		IEEE float (LOBYTE(LSM))		
	primary_status	WORD (HIBYTE)	40348	347
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40349	348
		WORD (LOWBYTE)		
nviVelocity2B		**UNVT_MixedStamp**		
	year	WORD (HIBYTE)	40350	349
		WORD (LOWBYTE)		
	month	BYTE	40351	350
	day	BYTE		
	hour	BYTE	40352	351
	minute	BYTE		
	second	BYTE	40353	352
	mode	BYTE		
	inst_lin_velocity	IEEE float (S+MSE)	40354	353
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40355	354
		IEEE float (LOBYTE(LSM))		
	avg_lin_velocity	IEEE float (S+MSE)	40356	355
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40357	356
		IEEE float (LOBYTE(LSM))		
	inst_raw_velocity	IEEE float (S+MSE)	40358	357
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40359	358
		IEEE float (LOBYTE(LSM))		
	avg_raw_velocity	IEEE float (S+MSE)	40360	359
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40361	360
		IEEE float (LOBYTE(LSM))		
	primary_status	WORD (HIBYTE)	40362	361
		WORD (LOWBYTE)		
	extended_status	WORD (HIBYTE)	40363	362
		WORD (LOWBYTE)		
nviVersionMIO		**SNVT_time_stamp**		
	year	WORD (HIBYTE)	40364	363
		WORD (LOWBYTE)		
	month	BYTE	40365	364
	day	BYTE		
	hour	BYTE	40366	365
	minute	BYTE		
	second	BYTE	40367	366
	FILLER	BYTE		

nviVersionTIE1		**UNVT_VersionData**		
	year	WORD (HIBYTE)	40368	367
		WORD (LOWBYTE)		
	month	BYTE	40369	368
	day	BYTE		
	hour	BYTE	40370	369
	minute	BYTE		
	second	BYTE	40371	370
	FPGA	BYTE		
	year	WORD (HIBYTE)	40372	371
		WORD (LOWBYTE)		
	month	BYTE	40373	372
	day	BYTE		
	hour	BYTE	40374	373
	minute	BYTE		
	second	BYTE	40375	374
	FILLER	BYTE		
nviVersionTIE2		**UNVT_VersionData**		
	year	WORD (HIBYTE)	40376	375
		WORD (LOWBYTE)		
	month	BYTE	40377	376
	day	BYTE		
	hour	BYTE	40378	377
	minute	BYTE		
	second	BYTE	40379	378
	FPGA	BYTE		
	year	WORD (HIBYTE)	40380	379
		WORD (LOWBYTE)		
	month	BYTE	40381	380
	day	BYTE		
	hour	BYTE	40382	381
	minute	BYTE		
	second	BYTE	40383	382
	FILLER	BYTE		
nviVolume1A		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40384	383
		WORD (LOWBYTE)		
	month	BYTE	40385	384
	day	BYTE		
	hour	BYTE	40386	385
	minute	BYTE		
	second	BYTE	40387	386
	mode	BYTE		

	inst_float_data1	IEEE float (S+MSE)	40388	387
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40389	388
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40390	389
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40391	390
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40392	391
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40393	392
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40394	393
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40395	394
		IEEE float (LOBYTE(LSM))		
nviVolume1AB		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40396	395
		WORD (LOWBYTE)		
	month	BYTE	40397	396
	day	BYTE		
	hour	BYTE	40398	397
	minute	BYTE		
	second	BYTE	40399	398
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40400	399
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40401	400
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40402	401
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40403	402
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40404	403
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40405	404
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40406	405
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40407	406
		IEEE float (LOBYTE(LSM))		
nviVolume1B		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40408	407

		WORD (LOWBYTE)		
	month	BYTE	40409	408
	day	BYTE		
	hour	BYTE	40410	409
	minute	BYTE		
	second	BYTE	40411	410
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40412	411
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40413	412
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40414	413
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40415	414
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40416	415
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40417	416
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40418	417
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40419	418
		IEEE float (LOBYTE(LSM))		
nviVolume2A		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40420	419
		WORD (LOWBYTE)		
	month	BYTE	40421	420
	day	BYTE		
	hour	BYTE	40422	421
	minute	BYTE		
	second	BYTE	40423	422
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40424	423
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40425	424
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40426	425
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40427	426
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40428	427
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40429	428

		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40430	429
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40431	430
		IEEE float (LOBYTE(LSM))		
nviVolume2AB		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40432	431
		WORD (LOWBYTE)		
	month	BYTE	40433	432
	day	BYTE		
	hour	BYTE	40434	433
	minute	BYTE		
	second	BYTE	40435	434
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40436	435
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40437	436
		IEEE float (LOBYTE(LSM))		
	avg_float_data1	IEEE float (S+MSE)	40438	437
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40439	438
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40440	439
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40441	440
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40442	441
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40443	442
		IEEE float (LOBYTE(LSM))		
nviVolume2B		**UNVT_FloatStamp**		
	year	WORD (HIBYTE)	40444	443
		WORD (LOWBYTE)		
	month	BYTE	40445	444
	day	BYTE		
	hour	BYTE	40446	445
	minute	BYTE		
	second	BYTE	40447	446
	mode	BYTE		
	inst_float_data1	IEEE float (S+MSE)	40448	447
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40449	448
		IEEE float (LOBYTE(LSM))		

	avg_float_data1	IEEE float (S+MSE)	40450	449
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40451	450
		IEEE float (LOBYTE(LSM))		
	inst_float_data2	IEEE float (S+MSE)	40452	451
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40453	452
		IEEE float (LOBYTE(LSM))		
	avg_float_data2	IEEE float (S+MSE)	40454	453
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40455	454
		IEEE float (LOBYTE(LSM))		
nviZero1		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40456	455
		WORD (LOWBYTE)		
	month	BYTE	40457	456
	day	BYTE		
	hour	BYTE	40458	457
	minute	BYTE		
	second	BYTE	40459	458
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40460	459
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40461	460
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40462	461
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40463	462
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	40464	463
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40465	464
		IEEE float (LOBYTE(LSM))		
nviZero2		**UNVT_StampedCal**		
	year	WORD (HIBYTE)	40466	465
		WORD (LOWBYTE)		
	month	BYTE	40467	466
	day	BYTE		
	hour	BYTE	40468	467
	minute	BYTE		
	second	BYTE	40469	468
	filler	BYTE		
	avg_pathA_data	IEEE float (S+MSE)	40470	469

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40471	470
		IEEE float (LOBYTE(LSM))		
	avg_pathB_data	IEEE float (S+MSE)	40472	471
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40473	472
		IEEE float (LOBYTE(LSM))		
	avg_pathAB_data	IEEE float (S+MSE)	**40474**	473
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40475	474
		IEEE float (LOBYTE(LSM))		
nvoFloatConfig1_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40476**	**475**
nvoFloatConfig1		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40477	476
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40478	477
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40479	478
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40480	479
		WORD (LOWBYTE)		
nvoFloatConfig2_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40481**	**480**
nvoFloatConfig2		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40482	481
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40483	482
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40484	483
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40485	484
		WORD (LOWBYTE)		
nvoFloatData1_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40486**	**485**
nvoFloatData1		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40487	486
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40488	487
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40489	488
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40490	489

		WORD (LOWBYTE)		
nvoFloatData2_SET		**WORD (HIBYTE) WORD (LOWBYTE)**	**40491**	**490**
nvoFloatData2		**UNVT_FloatArray**		
	variable_index	WORD (HIBYTE)	40492	491
		WORD (LOWBYTE)		
	float_data	IEEE float (S+MSE)	40493	492
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40494	493
		IEEE float (LOBYTE(LSM))		
	command	WORD (HIBYTE)	40495	494
		WORD (LOWBYTE)		
nvoMIO_Config_1_SET		**WORD (HIBYTE) WORD (LOWBYTE)**	**40496**	**495**
nvoMIO_Config_1		**UNVT_MIO_Config**		
	A_output_type	BYTE	40497	496
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40498	497
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40499	498
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40500	499
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40501	500
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40502	501
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40503	502
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40504	503
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40505	504
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40506	505
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40507	506
	digital_type[1]	BYTE		
	digital_type[2]	BYTE	40508	507
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40509	508
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40510	509
	digital_type[7]	BYTE		

nvoMIO_Config_2_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40511**	**510**
nvoMIO_Config_2		**UNVT_MIO_Config**		
	A_output_type	BYTE	40512	511
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40513	512
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40514	513
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40515	514
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40516	515
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40517	516
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40518	517
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40519	518
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40520	519
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40521	520
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40522	521
	digital_type[1]	BYTE		
	digital_type[2]	BYTE	40523	522
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40524	523
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40525	524
	digital_type[7]	BYTE		
nvoMIO_Config_3_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40526**	**525**
nvoMIO_Config_3		**UNVT_MIO_Config**		
	A_output_type	BYTE	40527	526
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40528	527
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40529	528
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40530	529
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40531	530

		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40532	531
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40533	532
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40534	533
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40535	534
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40536	535
		IEEE float (LOBYTE(LSM))		
	digital_type[0]	BYTE	40537	536
	digital_type[1]	BYTE		
	digital_type[2]	BYTE	40538	537
	digital_type[3]	BYTE		
	digital_type[4]	BYTE	40539	538
	digital_type[5]	BYTE		
	digital_type[6]	BYTE	40540	539
	digital_type[7]	BYTE		
nvoModeRequest_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40541**	**540**
nvoModeRequest		**UNVT_ModeRequest**		
	node_id	BYTE	40542	541
	mode	BYTE		
nvoTimeSet_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40543**	**542**
nvoTimeSet		**SNVT_time_stamp**		
	year	WORD (HIBYTE)	40544	543
		WORD (LOWBYTE)		
	month	BYTE	40545	544
	day	BYTE		
	hour	BYTE	40546	545
	minute	BYTE		
	second	BYTE	40547	546
	FILLER	BYTE		
nvoUnLngConfig1_SET		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40548**	**547**
nvoUnLngConfig1		**UNVT_UnLongArray**		
	variable_index	WORD (HIBYTE)	40549	548
		WORD (LOWBYTE)		
	un_long_data	WORD (HIBYTE)	40550	549
		WORD (LOWBYTE)		
	command	WORD (HIBYTE)	40551	550

		WORD (LOWBYTE)		
nvoUnLngConfig2_SET		**WORD (HIBYTE) WORD (LOWBYTE)**	**40552**	**551**
nvoUnLngConfig2		**UNVT_UnLongArray**		
	variable_index	WORD (HIBYTE)	40553	552
		WORD (LOWBYTE)		
	un_long_data	WORD (HIBYTE)	40554	553
		WORD (LOWBYTE)		
	command	WORD (HIBYTE)	40555	554
		WORD (LOWBYTE)		

Indexed Variables for Ultraflow 150

Float Data Descriptions
nvoFloatData1
nvoFloatData2

Floating point TIE OUTPUT variables, unstamped.

Note: Defaults and RESET state values are undefined for these variables because they are output data and therefore such defaults and RESET states have little meaning.

Index #	Element Name	Range, Unit Code
0 to 99	Actual Flow Velocity, Path-A, vA(k), buffer of 100	G
100 to 199	Actual Flow Volume, Path-A, fA(k), buffer of 100	GF
200 to 299	Standard Flow Volume, Path-A, sfA(k), buffer of 100	GF
300 to 399	Speed of Sound, Path-A, CsA(k), buffer of 100	G
400 to 499	Medium Temperature, Internal, Path-A, MTIA(k), buffer of 100	G
500	TimeOfFlightDownstreamA	S
501	TimeOfFlightUpstreamA	S
502	SpanHighOffsetTD_A	D
503	SpanLowOffsetTD_A	D
504	R_Factor_A	W
505	Upper SpanHigh Path-A estimate	GF
506	Lower SpanHigh Path-A estimate	GF
507	Upper SpanLow Path-A estimate	GF
508	Lower SpanLow Path-A estimate	GF
509	Acquire Mode Echo (for MIO outputs) {0=step, 1=ramp}	D
510 to 519	Span High, Path-A, buffer of 10	GF
520 to 529	Span Low, Path-A, buffer of 10	GF
530 to 539	Zero, Path-A, buffer of 10	GF
600	Temperature / Pressure Correction Factor STD Volume = Actual Volume * nvoFloatData1[600]	D
601	A/D Counts for ext temperature input channel	D
602	A/D Counts for pressure input channel	D
603	Volume to velocity conversion factor = 1/((stack area)*(units factors)) Velocity = Volume * nvoFloatData1[603]	GF
604	Upstream Noise Threshold Level - Path-A	0 to 4095, D
605	Downstream Noise Threshold Level - Path-A	0 to 4095, D
606	Upstream Noise Threshold Level - Path-B	0 to 4095, D
607	Downstream Noise Threshold Level - Path-B	0 to 4095, D

(continued on next page)

nvoFloatData1 & nvoFloatData2 (continued)

Index #	Element Name	Range, Unit Code
608	Upstream Peak Magnitude - Path-A	D
609	Downstream Peak Magnitude - Path-A	D
610	Upstream Peak Magnitude - Path-B	D
611	Downstream Peak Magnitude - Path-B	D
612	Amplitude Setpoint	D

1000 to 1099	Actual Flow Velocity, Path-B, vB(k), buffer of 100	G
1100 to 1199	Actual Flow Volume, Path-B, fB(k), buffer of 100	GF
1200 to 1299	Standard Flow Volume, Path-B, sfB(k), buffer of 100	GF
1300 to 1399	Speed of Sound, Path-B, CsB(k), buffer of 100	G
1400 to 1499	Medium Temperature, Internal, Path-B, MTIB(k), buffer of 100	G

1500	TimeOfFlightDownstreamB	S
1501	TimeOfFlightUpstreamB	S
1502	SpanHighOffsetTD_B	D
1503	SpanLowOffsetTD_B	D
1504	R_Factor_B	W
1505	Upper SpanHigh Path-B estimate	GF
1506	Lower SpanHigh Path-B estimate	GF
1507	Upper SpanLow Path-B estimate	GF
1508	Lower SpanLow Path-B estimate	GF

1510 to 1519	Span High, Path-B, buffer of 10	GF
1520 to 1529	Span Low, Path-B, buffer of 10	GF
1530 to 1539	Zero, Path-B, buffer of 10	GF

2000 to 2099	Actual Flow Velocity, (A+B)/2, vAB2(k), buffer of 100	G
2100 to 2199	Actual Flow Volume, (A+B)/2, sAB2(k), buffer of 100	GF
2200 to 2299	Standard Flow Volume, (A+B)/2, sfAB2(k), buffer of 100	GF
2300 to 2399	Speed of Sound, (A+B)/2, CsAB2(k), buffer of 100	G
2400 to 2499	Medium Temperature, Internal, (A+B)/2, MTIAB2(k), buffer of 100	G
2510 to 2519	Span High, Path-(A+B)/2, buffer of 10	GF
2520 to 2529	Span Low, Path-(A+B)/2, buffer of 10	GF
2530 to 2539	Zero, Path-(A+B)/2, buffer of 10	GF

3000 to 3099	Medium Temperature, External, MTE(k), buffer of 100	G
3100 to 3199	Medium Pressure, MP(k), buffer of 100	G

Unsigned Long Data Descriptions
nvoUnLngConfig1
nvoUnLngConfig2

Unsigned long (16 bit) integer TIE INPUT/OUTPUT configuration variables, unstamped.

Index #	Element Name	RESET State	Default	Range, Unit Code
0 **	(this variable not-used)			
1	Flow Volume Units [0=Kx/MINUTE, 1=x/MINUTE, 2=Kx/HOUR, 3=x/HOUR, 4=Kx/SECOND, 5=x/SECOND] (where x = METERS^3 or FEET^3 based on "Geometry Unit" variable)	NV	Kx/Minute	Enum, D
2	Measurement Paths [0=A, 1=A and B]	NV	1	Enum, D
3	Transducer Type [0=ES, 1=ESFS, 2=(not-used), 3=LR003, 4=LR004, 5=LR005, 6=LR006, 7=LR007] (determines transducer drive and intrinsic delay. Intrinsic delay can be modified to a non-standard value via a separate variable.)]	NV	ES	Enum, D
4	Tone Bursts Per Transit [1, 2 or 3]	NV	1	Enum, D
5	Time Between Transmits, seconds/count	NV	TBD	TBD, S
6	Integration Periods in Average	NV	3	1 to 100, D
7	Standard Pressure Correction [0=ENABLE, 1=DISABLE]	NV	DISABLE	Enum, D
8	Temperature Source [0=INTERNAL, 1=EXTERNAL]	NV	INTERNAL	Enum, D
9	Standard Temperature Correction Enable [0=ENABLE, 1=DISABLE]	NV	DISABLE	Enum, D
10	Flow Alarm Selection [0=LIN. ACT. VOLUME A, 1=STANDARD VOLUME A, 2=LIN. ACT. VOLUME B, 3=STANDARD VOLUME B, 4=LIN. ACT. VOLUME AB AVG., 5=STANDARD VOLUME AB AVG.]	NV	LINEAR, ACTUAL, VOLUME A	Enum, D
11	Flow Alarm Actuation Mode [0=ON GREATER THAN, 1=ON LESS THAN]	NV	ON LESS THAN	Enum, D
12	Temperature Alarm Selection [0=MEDIUM INTERNAL TEMP. A, 1=MEDIUM INTERNAL TEMP. B, 2=MEDIUM INTERNAL TEMP. AB AVG., 3=MEDIUM EXTERNAL TEMP.]	NV	MEDIUM INTERNAL TEMP. A	Enum, D
13	Temperature Alarm Actuation Mode [0=ON GREATER THAN, 1=ON LESS THAN]	NV	ON GREATER THAN	Enum, D
14	Integration Periods of Span High	NV	4	
15	Integration Periods of Span Low	NV	4	
16	Integration Periods of Zero	NV	4	
17	Hour of Calibration	NV	0	0 to 23
18	Minute of Calibration	NV	0	0 to 59
19	Interval Hours Between Calibrations	NV	25	0 to 25
20	Minimum Digital Gain	NV	1	0 to 255, D
21	Maximum Digital Gain	NV	255	0 to 255, D
22	Minimum Preamp Gain	NV	1	0 to 4095, D
23	Maximum Preamp Gain	NV	4095	0 to 4095, D
24	Automatic Gain Control [0=ENABLE, 1=DISABLE]	ENABLE	ENABLE	Enum, D
25	Automatic Noise Threshold [0=ENABLE, 1=DISABLE]	NV	ENABLE	Enum, D
26	Seconds per RAM Count, [0=0.5E-6 SECONDS, 1=1.0 E-6 SECONDS]	NV	0	0 OR 1, D
27 **	Geometry Units [0=FEET, 1=METERS]	NV	FEET	Enum, D

** Variable 0 was moved to Variable 27

(continued on next page)

nvoUnLngConfig1 & nvoUnLngConfig2 (continued)

Index #	Element Name	RESET State	Default	Range, Unit Code
30	Seconds per TD [0=0.5E-6 seconds, 1=1.0 E-6 seconds]	NV	0	0 OR 1, D
31	TD_DownstreamA	NV	9750	0 to 65,535, D
32	TD_UpstreamA	NV	9750	0 to 65,535, D
33	Acquire Mode Averaging {0=step, 1=ramp}	NV	0	0 OR 1, D
40	Flow Correlation Curve Source, Path-A [0=Polynomial, 1=LUT]	NV	POLY	Enum, D
50	Upstream Digital Gain Level - Path-A	NV	TBD	0 to 255, D
51	Downstream Digital Gain Level - Path-A	NV	TBD	0 to 255, D
52	Upstream Preamp Gain Level - Path-A	NV	TBD	0 to 4095, D
53	Downstream Preamp Gain Level - Path-A	NV	TBD	0 to 4095, D
60 **	Upstream Noise Threshold Level - Path-A	NV	TBD	0 to 4095, D
61 **	Downstream Noise Threshold Level - Path-A	NV	TBD	0 to 4095, D
1031	TD_DownstreamB	NV	9750	0 to 65,535, D
1032	TD_UpstreamB	NV	9750	0 to 65,535, D
1040	Flow Correlation Curve Source, Path-B [0=Polynomial, 1=LUT]	NV	POLY	Enum, D
1050	Upstream Digital Gain Level - Path-B	NV	TBD	0 to 255, D
1051	Downstream Digital Gain Level - Path-B	NV	TBD	0 to 255, D
1052	Upstream Preamp Gain Level - Path-B	NV	TBD	0 to 4095, D
1053	Downstream Preamp Gain Level - Path-B	NV	TBD	0 to 4095, D
1060 **	Upstream Noise Threshold Level - Path-B	NV	TBD	0 to 4095, D
1061 **	Downstream Noise Threshold Level - Path-B	NV	TBD	0 to 4095, D

** See nvoUnLngConfig1(25) for enable of Noise Threshold Level

Float Configuration Data Descriptions
nvoFloatConfig1
nvoFloatConfig2

Floating point TIE INPUT/OUTPUT configuration variables, unstamped.

Index #	Element Name	RESET State	Default	Range, Unit Code
0	(this variable not-used)			
1	Seconds per RAM Count, (read only)	NV	0.5E-6	0.5E-6 OR 1.0E-6, S
2	Seconds per TD, (read only)	NV	0.5E-6	0.5E-6 OR 1.0E-6, S
3	Integration Period [seconds for boxcar build-up]	NV	30	1 to 720, S
4	Minimum Medium Temperature	NV	TBD	TDB, G
5	Maximum Medium Temperature	NV	TBD	TDB, G
6	Minimum Flow Velocity	NV	TBD	TDB, G
7	Maximum Flow Velocity	NV	TBD	TDB, G
8	Time Between Transmits, seconds, (read only)	NV	0.031232	0.000512 to 0.131072, S
9	Amplitude Factor	NV	1	.1 to 1, D
10	SpanHigh Volume Setpoint	NV	236 KCFM	TBD, GF
11	SpanLow Volume Setpoint	NV	118 KCFM	TBD, GF
12	Zero Volume Setpoint	NV	0	TBD, GF
13	Calibration Tolerance	NV	14 KCFM	TBD, GF
20	Pressure Scaling, Pressure at Point 1	NV	TBD	TBD, G
21	Pressure Scaling, A/D Counts at Point 1	NV	4	0 to 4095, D
22	Pressure Scaling, Pressure at Point 2	NV	TBD	TBD, G
23	Pressure Scaling, A/D Counts at Point 2	NV	20	0 to 4095, D
30	Temperature Scaling, Temperature at Point 1	NV	TBD	TBD, G
31	Temperature Scaling, A/D Counts at Point 1	NV	4	0 to 4095, D
32	Temperature Scaling, Temperature at Point 2	NV	TBD	TBD, G
33	Temperature Scaling, A/D Counts at Point 2	NV	20	0 to 4095, D

(continued on next page)

nvoFloatConfig1 & nvoFloatConfig2 (continued)

Index #	Element Name	RESET State	Default	Range, Unit Code
40	Instantaneous Flow Alarm Threshold	NV	0	TBD, GF
41	Average Flow Alarm Threshold	NV	0	TBD, GF
42	Instantaneous Temperature Alarm Threshold	NV	0	TDB, G
43	Average Temperature Alarm Threshold	NV	0	TBD, G
50	Reference Temperature	NV	If geometry unit variable = feet, default is 68; if meters, default is 20	TBD, G If geometry unit variable = feet, unit is deg. F; if meters, unit is deg C.
51	Reference Pressure	NV	If geometry unit variable = feet, default is 29.92; if meters, default is 101.5	TBD, G If geometry unit variable = feet, unit is in Hg; if meters, unit is Kpa.
53	Cross Sectional Area [meters^2, feet^2]	NV	78.54 ft^2	0 to 10^4 ft^2, G
60 ***note1**	Transducer-to-Transducer Distance, Path-A	NV	14.14 ft	0 to 100ft, G
61 ***note1**	Offset, Path-A	NV	10 ft	0 to 100ft,G
62 ***note1**	Downstream nozzle length, Path-A	NV	0.7071 ft	TDB, G
63 ***note1**	Upstream nozzle length, Path-A	NV	0.7071 ft	TDB, G
64	NozzleDelayDownstreamA	NV	626.5E-6	TBD, S
65	NozzleDelayUpstreamA	NV	626.5E-6	TBD, S
66	IntrinsicDelayA	NV	170E-6	TBD, S
70	SN Ratio Alarm Threshold, Downstream, Path-A	NV	TBD	TBD, D
71	SN Ratio Alarm Threshold, Upstream, Path-A	NV	TBD	TBD, D

***note1** is at the end of this section.

(continued on next page)

nvoFloatConfig1 & nvoFloatConfig2 (continued)

Index #	Element Name	RESET State	Default	Range, Unit Code
80 to 85	Flow X1,Y1,X2,Y2,X3,Y3, Path-A [LUT] X1=80, Y1=81, X2=82, Y2=83, X3=84, Y3=85	NV	TBD	TBD, D
90 to 95	Flow A0,A1,A2,A3,A4,A5, Path-A [Polynomial] A0=90, A1=91, A2=92, A3=93, A4=94, A5=95	NV	TBD	TBD, D
100 to 105	Temperature R1,CS1, R2,CS2, R3,CS3, Path-A [LUT], R is "R factor", CS is speed of sound R1=100, CS1=101, R2=102, CS2=103, R3=104, CS3=105	NV	TBD	TBD, W

(continued on next page)

nvoFloatConfig1 & nvoFloatConfig2 (continued)

Index #	Element Name	RESET State	Default	Range, Unit Code
1060 ***note1**	Transducer-to-Transducer Distance, Path-B	NV	14.14 ft	0 to 100ft, G
1061 ***note1**	Offset, Path-B	NV	10 ft	0 to 100ft, G
1062 ***note1**	Downstream nozzle length, Path-B	NV	0.7071 ft	TDB, G
1063 ***note1**	Upstream nozzle length, Path-B	NV	0.7071 ft	TDB, G
1064	NozzleDelayDownstreamB	NV	626.5E-6	TBD, S
1065	NozzleDelayUpstreamB	NV	626.5E-6	TBD, S
1066	IntrinsicDelayB	NV	170E-6	TBD, S
1070	SN Ratio Alarm Threshold, Downstream, Path-B	NV	TBD	TBD, D
1071	SN Ratio Alarm Threshold, Upstream, Path-B	NV	TBD	TBD, D
1080 to 1085	Flow X1,Y1,X2,Y2,X3,Y3, Path-B [LUT] X1=1080, Y1=1081, X2=1082, Y2=1083, X3=1084, Y3=1085	NV	TBD	TBD, D
1090 to 1095	Flow A0,A1,A2,A3,A4,A5, Path-B [Polynomial] A0=1090, A1=1091, A2=1092, A3=1093, A4=1094, A5=1095	NV	TBD	TBD, D
1100 to 1105	Temperature R1,CS1, R2,CS2, R3,CS3, Path-B [LUT] R1=1100, CS1=1101, R2=1102, CS2=1103, R3=1104, CS3=1105	NV	TBD	TBD, W

***note1** - Variables 60-63 and 1060-1063 are the actual dimensions measured on the stack. The true path and offset must be calculated from these values.

APPENDIX D

LASERHAWK 360 VARIABLES

(This page intentionally left blank.)

The following section describes the variables for the LaserHawk 360 instrument. A description of each network variable is given, followed by the C/C++ definitions of the structures, constants and variable types. The Modbus register mapping is also given.

Note that most of the variables and the network bindings are reused from the LightHawk 560 instrument.

All network variables are prefixed with a three letter prefix, either “nvi” or “nvo” which translates to:

nvi – Network Variable Input;
nvo – Network Variable Output.

The direction of Input or Output is relative to the Ethernet Module.

Instrument Status Word Interpretation via Modbus TCP
On the LaserHawk, the nviHeadState variable signifies instrument status. See the description for nviHeadState in the “Modbus Input Registers for LaserHawk 560” table for more details.

Initiating Calibration Commands via Modbus TCP
On the LaserHawk, the nvoModeRequest variable is used to command calibration on the analyzer. See the description for nvoModeRequest in the following tables, the header files for the LaserHawk and the “Propagating Values On The Network” section for more details on command initiation.

Multiple Use of Network Variable nvoheadconfig_3
To eliminate the need for a new network binding for the NEURON processors, a single variable was selected to enable the transfer of additional LaserHawk data not used on the LightHawk (nviHeadConfig3). This complicates writing data to nviHeadConfig3 variables. However, reading is accomplished via normal read access to the Modbus Input registers for these variables.

nviHeadConfig_3 is used to transfer four (4) kinds of data (Optical Density Alarm levels, Temperature Scaling, Pressure Scaling or Linearization Curve). The meaning of the data contained in the structure is controlled by the value in (nviHeadConfig_3.inst_OptD_level1). The data in the structure is identified as follows:

SELECTOR VARIABLE	VALUE	STRUCTURE MEANING
nviHeadConfig_3.inst_OptD_level1	>=0	Optical Density Alarm levels
nviHeadConfig_3.inst_OptD_level1	- 5	Temperature Scaling
nviHeadConfig_3.inst_OptD_level1	- 10	Pressure Scaling
nviHeadConfig_3.inst_OptD_level1	- 20	Linearization Curve

Note that the variable type itself is not changed, only the usage for the elements within the structure.

Due to the overloading of nviHeadConfig_3, normal network polling is no longer satisfactory since only the last network variable message for the structure can be polled by the network. Overloading nviHeadConfig_3 also creates the need for a method for controlling the direction of transfer so that one can write to the variables. This functionality is provided by nviHeadConfig_3. inst_OptD_level2, which is a set to –1 to read a variable and –2 to write.

To ease a programmer's interaction with these variables and avoid the possibility of multiple incoming nviHeadConfig_3's overwriting each other, four separate nviHeadConfig_3 locations are implemented in the Modbus Input Registers. Thus it appears that there are 4 independent nviHeadConfig_3 variables: nviHeadConfig_3_1, nviHeadConfig_3_2, nviHeadConfig_3_3 and nviHeadConfig_3_4.

Only one output nvoHeadConfig_3 is specified in the Modbus Holding Registers because only one at time can be processed in the LaserHawk. It is this variable that must be changed to write to the variables mapped to nviHeadConfig_3.

Linearization Curve Parameter Input
The variables, (a0, a1, a2) will be entered from the Enhanced Remote Panel using network variable (nviHeadConfig_3). As mentioned previously, this network variable will be used to transfer four (4) types of data.

The structure element definition of this variable is as follows.

Original definition	**New definition for POLYNOMIAL**	**New definition for LOGARITHMIC**	**New definition for EXPONENTIAL**
Equation=>	**Y = a0 + a1*X + a2*X*X**	**Y = a * (ln (a1X+a2))**	**Y = a * exp(a1X) + a2**
**********	**********	**********	**********
nvoHeadConfig_3. .inst_OptD_level1	**- 20** (-19.8 to -20.2)	**- 20** (-19.8 to -20.2)	**- 20** (-19.8 to -20.2)
nvoHeadConfig_3. .inst_OptD_level2	**- 2 = WRITE** (-.2 to +.2) **- 1 = READ** (-.8 to -1.2)	**- 2 = WRITE** (-.2 to +.2) **- 1 = READ** (-.8 to -1.2)	**- 2 = WRITE** (-.2 to +.2) **- 1 = READ** (-.8 to -1.2)
nvoHeadConfig_3. .min_OptD_level1	**Select = 0**	**Select = 1**	**Select = 2**
nvoHeadConfig_3. .min_OptD_level2	a0	a0	a0
nvoHeadConfig_3. .avg_OptD_level1	a1	a1	a1
nvoHeadConfig_3. .avg_OptD_level2	a2	a2	a2

Note that the structure itself is not being changed; only what the elements of the structure will mean.

nvi332Version	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make 68332 application code version of the Optical Head available as a network variable. Date shows date of release, hours and minutes show version. The second field is ignored and should be set to zero to avoid confusion when using browsers or other network tools. For example 2/28/1999, 2:02:00 represents Version 2.02 which was released on 2/28/1999.
nviNeuronVer	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make NEURON application code version of the Optical Head available as a network variable. See nvi332Version.
nviMIO_Version	Type: SNVT_time_stamp
	Units: Time and date.
	Uses time and data format to make the NEURON application code version of the Multi I/O available as a network variable in the same format as nvi332Version.
nviTime	Type: SNVT_time_stamp
	Units: Time and date.
	Contains the current value of the System Clock.
NviAlarms	Type: SNVT_alarm
	Units: Application Dependent
	This network variable is issued by the Optical Head once whenever an alarm changes state. The type, object, value and time stamp of the alarm are propagated. The Display Panel device accumulates an alarm history based on issuance events of this variable.
nviHeadState	Type: UNVT_HeadStatus
	Units: Dimensionless
	Represents the Optical Head status. The codes represent detailed instrument conditions which OR together.
nviHeadMode	Type: UNVT_HeadMode
	Units: Dimensionless
	This variable defines the mode of operation of various elements of the Optical Head. The AuditMode and FilterValue members are used collaboratively to describe the filter audit data.
nviMIO_Status	Type: UNVT_MIO_status
	Units: Dimensionless

	This network variable communicates the analog output mode (MIO_mode) and the error state (invalid_mode_request, invalid_AO_config and invalid_DIO_config) of the Multi I/O device. Errors exist when the bit type members are one. For normal operation they are zero. MIO_mode is an enumerated member with values such as SPAN, ZERO, NORMAL, etc.
nviOpacityAlarms, nviOptDensAlarms, nviMassAlarms	Type: UNVT_AlarmSel
	Units: Dimensionless
	These network variables represent the states of Instantaneous, One Minute and Selectable Average two level alarms for Backscatter, Optical Density and Dust Mass. If a bit equals one, the alarm condition is in effect. These network variables are propagated every time the alarm condition is evaluated.
nviInstOpacity, nviInstOptDens, nviInstMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the instantaneous measurements. nviInstOpacity actually is Instant Backscatter %. nviInstOptDens is now Zero Drift Particulate Mass.
nvi1MinOpacity, nvi1MinOptDens, nvi1MinMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the one minute average measurements. nvi1MinOpacity actually is % Backscatter. nvi1MinOptDens is actually Upscale Cal Drift Particulate Mass.
nviSelAveOpacity, nviAveOptDens, nviAveMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the selectable average measurements. Opacity actually is Backscatter %. nviAveOptDens is actually the Upscale Set Value in % Backscatter that was observed by the analyzer after its most recent Upscale Set Function. This is different that the Upscale Cal Setpoint, which is a manually.entered.
nviZeroOpacCal, nviZeroOptDens, nviZeroMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Zero Calibration Average measurements. Opacity actually is Backscatter. nviZeroOptDens is actually Zero Particulate Mass Setpoint.
nviSpanOpacCal, nviSpanOptDens, nviSpanMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Span Calibration Average measurements. Opacity actually is Backscatter. nviSpanOptDens is actually Upscale Cal Particulate Mass Setpoint.
nviDirtCompValue, nviDirtOptDens, nviDirtMassLoad	Type: Float
	Units: Application Dependent
	These variables are the values for the Dust Compensation Average measurements.
nviServiceData1	Type: UNVT_Service_One

	Units: Voltages and currents except for HeadTemperature, which is in degrees Centigrade, and CalWheelPosition, which is dimensionless.
	Represents internal instrument data of interest to service technicians more than average users.
nviServiceData2	Type: UNVT_Service_Two
	Units: Voltages.
	Represents internal power supply voltages of interest to service technicians more than average users.
nviAuditData	Type: UNVT_AuditData
	Units: AuditMode and FilterValue are dimensionless. AuditOpacity is in percent. Optical Density is dimensionless. Dust Mass is mg/m^3.
	Not used on the LaserHawk.
nviHeadConfig_1, nvoHeadConfig_1	Type: UNVT_HeadConfig1
	Units: Percent opacity for all.
	This variable defines emission alarm levels for Instantaneous Backscatter, One Minute Backscatter and Selectable Average Backscatter. It also sets the Dust Compensation Alarm limit. The DirtCompAlarm member will not accept a value greater than 8.00.
nviHeadConfig_2, nvoHeadConfig_2	Type: UNVT_HeadConfig2
	Units: PLCF is dimensionless; ZeroCalSetpt, SpanCalSetpt and CalDelta are in percent opacity; and AveSendTime is in minutes.
	This variable defines the current PLCF and is mapped to now as PART MASS UNITS. The ZeroCalSetpt defines the expected ZERO calibration value for Backscatter, SpanCalSetpt defines the expected UPSCALE calibration value and CalDelta defines the tolerance band around them before a calibration failure is presented. The AveSendTime defines the averaging interval for Selectable Average Backscatter and Particulate Mass.
nviHeadConfig_3_1, nviHeadConfig_3_2, nviHeadConfig_3_3, nviHeadConfig_3_4, nvoHeadConfig_3	Type: UNVT_HeadConfig3
	Units: Dimensionless for all.
	See the explanation at the beginning of this chapter.
nviHeadConfig_4, nvoHeadConfig_4	Type: UNVT_HeadConfig4
	Units: Application Dependent.

<table>
<tr><td></td><td>This variable defines ENABLE/DISABLE functions for medium temperature and pressure correction to standard conditions and the medium temperature and pressure limits for a valid measurement. Its member correspond to:
<table>
<tr><td>TEMPERATURE CORRECTION ENABLE</td><td>UNVT_HeadConfig4.MassLoad_X1</td></tr>
<tr><td>PRESSURE CORRECTION ENABLE</td><td>UNVT_HeadConfig4.MassLoad_Y1</td></tr>
<tr><td>MIN TEMPERATURE</td><td>UNVT_HeadConfig4.MassLoad_X2</td></tr>
<tr><td>MAX TEMPERATURE</td><td>UNVT_HeadConfig4.MassLoad_Y2</td></tr>
<tr><td>MIN PRESSURE</td><td>UNVT_HeadConfig4.MassLoad_X3</td></tr>
<tr><td>MAX PRESSURE</td><td>UNVT_HeadConfig4.MassLoad_Y3</td></tr>
</table></td></tr>
<tr><td rowspan="3">nviHeadConfig_5,
nvoHeadConfig_5</td><td>Type: UNVT_HeadConfig5</td></tr>
<tr><td>Units: Application Dependent.</td></tr>
<tr><td>This variable defines emission alarm levels for Particulate Mass:
<table>
<tr><td>INST PART MASS LEVEL1</td><td>UNVT_HeadConfig5.inst_Dust_level1</td></tr>
<tr><td>1MIN PART MASS LEVEL1</td><td>UNVT_HeadConfig5.min_Dust_level1</td></tr>
<tr><td>AVG PART MASS LEVEL1</td><td>UNVT_HeadConfig5.avg_Dust_level1</td></tr>
<tr><td>INST PART MASS LEVEL2</td><td>UNVT_HeadConfig5.inst_Dust_level2</td></tr>
<tr><td>1MIN PART MASS LEVEL2</td><td>UNVT_HeadConfig5.min_Dust_level2</td></tr>
<tr><td>AVG PART MASS LEVEL2</td><td>UNVT_HeadConfig5.avg_Dust_level2</td></tr>
</table></td></tr>
<tr><td rowspan="3">nviHeadConfig_6,
nvoHeadConfig_6</td><td>Type: UNVT_HeadConfig6</td></tr>
<tr><td>Units: Temp_Deg_C is in centigrade; AbsPressure is in Pascals (SI/MKS pressure unit).</td></tr>
<tr><td>Defines the reference temperature and pressure for correction of Dust Mass to standard conditions. The pressure is absolute.</td></tr>
<tr><td rowspan="3">nviHeadConfig_7,
nvoHeadConfig_7</td><td>Type: UNVT_HeadConfig7</td></tr>
<tr><td>Units: The cal_time_hour and cal_interval members are in hours, cal_time_min is in minutes. All others are in seconds.</td></tr>
<tr><td>cal_time_hour and cal_time_min define time of day for the automatic calibration cycle in the Optical Head. cal_interval_hour defines the interval between the first calibration defined by the prior members and the next automatic calibration. If cal_interval_hour equals 0 or a value of 24 or greater, calibration occurs only once a day. If cal_time_hour = 12, cal_time_min = 30 and cal_interval_hour = 4, cals occur at 1230 hours, 1630 hours and 2030 hours. After day rollover, no more occur until 1230 hours. The span_secs, zero_secs, etc. define the number of seconds in each part of the cal cycle. If a member is zero, that part of the cycle is skipped.</td></tr>
<tr><td rowspan="2">nviHeadConfig_8,
nvoHeadConfig_8</td><td>Type: UNVT_HeadConfig8</td></tr>
<tr><td>Units: Dimensionless. Valid ranges are currently 0 to 255 for all. The Optical Head and Network devices should accept values only in this range.</td></tr>
</table>

	These members define the gain of the Optical Head optical amplifiers. CommonGain affects both the signal and reference paths. These gain controls are currently implemented with 8 bit devices.
nviMIO_Config_1, nvoMIO_Config_1	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 1 (Analog Output 1) and DAC 2 (Analog Output 2) of the Multi I/O device.
nviMIO_Config_2, nvoMIO_Config_2	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping and scaling for DAC 3 (Analog Output 3) and DAC 4 (Analog Output 4) of the Multi I/O device.
nviMIO_Config_3, nvoMIO_Config_3	Type: UNVT_MIO_config
	Units: Configuration dependent.
	This network variable structure can define up to 2 analog outputs and 8 discrete inputs. However, this specific network variable defines the mapping for Relay 1 (K1) through Relay 8 (K8) of the Multi I/O device.
nviMIO_Status	Type: UNVT_MIO_status
	Units: Dimensionless.
	This network variable communicates the analog output mode (MIO_mode) and the error state (invalid_mode_request, invalid_AO_config and invalid_DIO_config) of the Multi I/O device. Errors exist when the bit type members are one. For normal operation they are zero. MIO_mode is an enumerated member with values such as SPAN, ZERO, NORMAL, etc.
nviRequest	Type: SNVT_obj_request
	Units: Dimensionless.
	This network variable is not used.
nviNetworkStatus	Type: UNVT_ModeRequest
	Units: Dimensionless.
	This internal variable used by the Neuron side of the interface. It signals one of the following values in mode structure member: Network Normal = 0, No Stack Communication = 1, Transmission Errors = 2, Network Interface Fail = 3, Status Unknown = 4.
nviPanelVersion	Type: SNVT_time_stamp
	Units: Time and Date.
	Uses time and data format to make NEURON application code version of the Ethernet Module available as a network variable. See nvi332Version.
nvoTimeSet	Type: SNVT_time_stamp
	Units: Time and Date.

	The System Clock resides in the 560 Optical Head Node. This network variable sets the value of the System Clock.
nvoModeRequest	Type: UNVT_ModeRequest
	Units: Dimensionless.
	Used to transmit commands from the Remote Display node to the Optical Head and Multi I/O nodes. The node_id member is used to designate which node the command originated from: 1 represents the Enhanced Remote Panel/Ethernet Module and 2 represents the Multi I/O. The mode member is an enumerated data type that invokes the command. For example, if (mode = ZERO), the Optical Head will move the calibration mechanism to ZERO position and perform a calibration ZERO. TEST type commands are meant for the Multi I/O only. Support for these functions for the Six Point I/O analog outputs is via its own jumpers or the Optical Head keypad. **DESIRED MODE** **nvoModeRequest VALUE** UNKNOWN 0 NORMAL 1 ZERO 2 UPSCALE 3 *(NOT USED)* 4 DIRT 5 FORCE_CAL_CYCLE 6 TEST_ZERO_SCALE 7 TEST_MID_SCALE 8 TEST_FULL_SCALE 9 PUT_IN_SERVICE 10 OUT_OF_SERVICE 11
nvoDataRequest	Type: UNVT_ModeRequest
	Units: Dimensionless.
	This variable is not truly a network variable. It is used to instruct the NEURON side of the Ethernet Module to fetch (poll) a particular variable from the network, since not all variables mentioned here are continuously broadcast. Both structure members node_id and mode are set to desired beginning MODBUS address of the variable to fetch. The valid variables that can be fetched are: nviMIO_Config_1, nviMIO_Config_2, nviMIO_Config_3, PanelVersion, nvi332Version, nviNeuronVer, nviMIO_Version, nviHeadConfig_1, nviHeadConfig_2, nviHeadConfig_3, nviHeadConfig_4, nviHeadConfig_5, nviHeadConfig_6, nviHeadConfig_7 and nviHeadConfig_8.
nviML_CorrFactor	Type: Float
	Units: Application Dependent
	A fractional value proportional to the magnitude of the correction from Actual to Standard Conditions.

Header File Definitions for the LaserHawk 360

```
/*
 *      Copyright (c) 2006 by Teledyne Monitor Labs Inc.
 *
 *  This software is copyrighted by and is the sole property of
 *  Teledyne Monitor Labs.  All rights, title, ownership, or other interests
 *  in the software remain the property of Teledyne Monitor Labs.  This
 *  software may only be used in accordance with the corresponding
 *  license agreement.  Any unauthorized use, duplication, transmission,
 *  distribution, or disclosure of this software is expressly forbidden.
 *
 *  This Copyright notice may not be removed or modified without prior
 *  written consent of Teledyne Monitor Labs.
 *
 *  Teledyne Monitor Labs, reserves the right to modify this software
 *  without notice.
 *
 *  Teledyne Monitor Labs, Inc.
 *  Gibsonia Facility                       Phone: 724-444-5000
 *  5310 N. Pioneer Rd.                     Fax: 724-444-5050
 *  Gibsonia, PA 15044, USA                 http://www.teledyne-ml.com
 *
 ******************************************************************************
 *
 *     Module Name: LH3600DEF.H
 *         Version: 1.00
 *   Original Date: 08/17/2006
 *          Author: Ivica Eftimovski
 *        Language: Ansi C
 * Compile Options:
 * Compile defines:
 *       Libraries:
 *    Link Options:
 *
 *
 * Description.
 * ============================================================================
 *  This file defines variables needed by the Ultraflow 150.
 *
 *
 * Edit Date/Ver   Edit Description
 * ==============  ====================================================
 *
 *
 */

#ifndef LASERHAWK360DEFINITIONS_HEADER_FILE
#define LASERHAWK360DEFINITIONS_HEADER_FILE

#ifdef __cplusplus
extern "C"
{
#endif
```

```
/* Constants and Enumerations */

/* LaserHawk 360 Modes */
typedef enum
{
   UNKNOWN=0,         //  0
   NORMAL,            //  1
   ZERO,              //  2
   UPSCALE,           //  3
   PLCF,              //  4
   DIRT,              //  5
   FORCE_CAL_CYCLE,   //  6
   TEST_ZERO_SCALE,   //  7
   TEST_MID_SCALE,    //  8
   TEST_FULL_SCALE,   //  9
   PUT_IN_SERVICE,    // 10
   OUT_OF_SERVICE     // 11
} node_mode ;

/* MIO Isolator Values */
#define ISO_FULL_SCALE    0x80 /* binary10000000 */
#define ISO_ZERO_SCALE    0x40 /* binary01000000 */
#define ISO_MID_SCALE     0xC0 /* binary11000000 */
#define ISO_FORCE_CAL     0x10 /* binary00010000 */
#define ISO_FORCE_DIRT    0x08 /* binary00001000 */
#define ISO_FORCE_PLCF    0x04 /* binary00000100 */
#define ISO_FORCE_ZERO    0x02 /* binary00000010 */
#define ISO_FORCE_UPSCALE 0x01 /* binary00000001 */

#define WITH    1 /* for with calibration configurations */
#define WITHOUT 0 /* for without calibration configurations */
#define WITH_EXPANDED_SCALE 2 /* for with expanded cal ZERO & DIRT scaling */

#define VALID     1   /* used by fault flags to indicate status */
#define INVALID   42


typedef enum  // for analog configuration, A_output_type, B_output_type
{
   NO_SELECTION = 0,              //  0 added 8-5-98 by ELM
   INSTANT_OPAC=1,                //  1
   MINUTE_AVG_OPAC,               //  2
   SELECTABLE_AVG_OPAC,           //  3
   INSTANT_OPT_DENSITY,           //  4
   MINUTE_AVG_OPT_DENSITY,        //  5
   SELECTABLE_AVG_OPT_DENSITY,    //  6
   INSTANT_DUST_MASS,             //  7
   MINUTE_AVG_DUST_MASS,          //  8
   SELECTABLE_AVG_DUST_MASS,      //  9
   DIRT_OUTPUT,                   // 10
   ZERO_OUTPUT,                   // 11
   UPSCALE_OUTPUT,                // 12
   PLCF_OUTPUT,                   // 13
   SIGNAL_VOLTAGE,                // 14
   REFERENCE_VOLTAGE,             // 15
   LED_CURRENT_mA,                // 16
   STACK_SET_VOLTAGE,             // 17
```

```
   CAL_ZERO_SET_VOLTAGE,          // 18
   BACK_GROUND_SET_VOLTAGE,       // 19
   OPTICAL_HEAD_TEMP_C,           // 20
   CAL_MECHANISM_POSITION,        // 21
   POS_15_SUPPLY,                 // 22
   NEG_15_SUPPLY,                 // 23
   POS_5V_ANALOG_SUPPLY,          // 24
   NEG_5V_ANALOG_SUPPLY,          // 25
   POS_5V_DIGITAL_SUPPLY          // 26
} AO_type ;                     // 27

typedef enum  //for digital configuration, DIO_type
{
   NO_SELECT = 0,             //  0 added 8-5-98 by ELM
   INST_OPAC_LEVEL1=1,        //  1
   INST_OPAC_LEVEL2,          //  2
   MIN_OPAC_LEVEL1,           //  3
   MIN_OPAC_LEVEL2,           //  4
   AVG_OPAC_LEVEL1,           //  5
   AVG_OPAC_LEVEL2,           //  6
   INST_OPT_DENSITY_LEVEL1,   //  7
   INST_OPT_DENSITY_LEVEL2,   //  8
   MIN_OPT_DENSITY_LEVEL1,    //  9
   MIN_OPT_DENSITY_LEVEL2,    // 10
   AVG_OPT_DENSITY_LEVEL1,    // 11
   AVG_OPT_DENSITY_LEVEL2,    // 12
   INST_MASS_LOAD_LEVEL1,     // 13
   INST_MASS_LOAD_LEVEL2,     // 14
   MIN_MASS_LOAD_LEVEL1,      // 15
   MIN_MASS_LOAD_LEVEL2,      // 16
   AVG_MASS_LOAD_LEVEL1,      // 17
   AVG_MASS_LOAD_LEVEL2,      // 18
   INSTRUMENT_MALF,           // 19
   INSTRUMENT_ALERT,          // 20
   PURGE_FAILURE,             // 21
   EXCESS_DIRT_COMP,          // 22
   CAL_FAILURE,               // 23
   SPAN_ON_AO,                // 24
   ZERO_ON_AO,                // 25
   PLCF_ON_AO,                // 26
   DIRT_ON_AO,                // 27
   NORMAL_ON_AO,              // 28 added 7-20-98 by ELM
   CAL_ON_AO                  // 29 added 7-20-98 by ELM
} DIO_type ;

typedef enum
{

      NOT_AUDIT, RUN1, RUN2, RUN3, RUN4, RUN5, CLEAR_AUDIT

} AUDITMODE ;

typedef enum
{

      NO_FILT, LOW_FILT, MID_FILT, HI_FILT
```

```
} FILTERVALUE ;


/* Network Variables */

struct __UNVT_ModeRequest__TagName__
{
      BYTE node_id ;
      BYTE mode ;

}  __attribute__ ((__packed__));

typedef struct __UNVT_ModeRequest__TagName__ UNVT_ModeRequest ;

extern UNVT_ModeRequest *nvoModeRequest ;
extern UNVT_ModeRequest *nviNetworkStatus;
extern UNVT_ModeRequest *nvoDataRequest;


struct __UNVT_AlarmSel__TagName__
{

      BYTE  Bit_7          : 1; // Bit_07 = not-used
      BYTE  Bit_6          : 1; // Bit_06 = not-used

      BYTE  AveAlarm2      : 1; // Bit_05 =
      BYTE  AveAlarm1      : 1; // Bit_04 = Set Average Alarms

      BYTE  MinAlarm2      : 1; // Bit_03 =
      BYTE  MinAlarm1      : 1; // Bit_02 = 1-Minute Alarms

      BYTE  InstAlarm2     : 1; // Bit_01 =
      BYTE  InstAlarm1     : 1; // Bit_00 = Instantaneous Alarms

}  __attribute__ ((__packed__));

typedef struct __UNVT_AlarmSel__TagName__ UNVT_AlarmSel ;

extern UNVT_AlarmSel     *nviMassAlarms ;
extern UNVT_AlarmSel     *MassAlarms ;
extern UNVT_AlarmSel     *nviOptDensAlarms ;
extern UNVT_AlarmSel     *OptDensAlarms ;


struct __UNVT_Service_One__TagName__
{

      float SignalVolts ;       //(-10 to +10, VDC)
      float ReferenceVolts ;    //(-10 to +10, VDC)
      float LED_current ;       //(0 to 50, ma)
      float StackSetVols ;      //(-10 to +10, VDC)
      float ZeroSetVolts ;      //(-10 to +10, VDC)
      float BackgndSetVols ;    //(-10 to +10, VDC)
      float HeadTemperature ;   //degrees C
      WORD16 CalWheelPosition ;  //dimless
```

```
}  __attribute__ ((__packed__));

typedef struct __UNVT_Service_One__TagName__ UNVT_Service_One ;

extern UNVT_Service_One  *nviServiceData1 ;


struct __UNVT_Service_Two__TagName__
{

      float Pos_15V ;        //(-30 to +30, VDC)
      float Neg_15V ;        //(-30 to +30, VDC)
      float Pos_5Vanalog ;   //(-10 to +10, VDC)
      float Neg_5Vanalog ;   //(-10 to +10, VDC)
      float Pos_5Vdigital ;  //(-10 to +10, VDC)

}  __attribute__ ((__packed__));

typedef struct __UNVT_Service_Two__TagName__ UNVT_Service_Two ;

extern UNVT_Service_Two  *nviServiceData2 ;


struct __UNVT_HeadStatus__TagName__
{
      BYTE  DirtDriftAlarm : 1; // Bit15 for Excessive Dirt Drift
      BYTE  CalSpanAlarm   : 1; // Bit14 for UPSCALE Cal bad
      BYTE  CalZeroAlarm   : 1; // Bit13 for Cal ZERO bad
      BYTE  LoSetVoltZERO  : 1; // Bit12 for ZERO SET volts out of range
      BYTE  LoSetVoltSTACK : 1; // Bit11 for CLEAR STACK SET Volts out of
range
      BYTE  SET_Backgrnd   : 1; // Bit10 for BACKGROUND SET in progress
      BYTE  SET_Zero       : 1; // Bit9  for ZERO SET in progress
      BYTE  SET_STACK      : 1; // Bit8  for CLEAR STACK SET in progress
      BYTE  UpscalePosErr  : 1; // Bit7  for UPSCALE position not achived
      BYTE  ZeroPosErr     : 1; // Bit6  for ZERO position not achived
      BYTE  NormalPosErr   : 1; // Bit5  for Normal position not achived
      BYTE  ReferenceFault : 1; // Bit4  for Reference Fault
      BYTE  ADC_Fault      : 1; // Bit3  for ADC (Sync Demod) Fault
      BYTE  Out_of_Service : 1; // Bit2  for Out of Service
      BYTE  PurgeFailRetro : 1; // Bit1  for Purge Failure Retro Side
      BYTE  PurgeFailHead  : 1; // Bit0  for Purge Failure Analyzer Side

}    __attribute__ ((__packed__));

typedef struct __UNVT_HeadStatus__TagName__ UNVT_HeadStatus;

extern UNVT_HeadStatus  *nviHeadState ;


 struct __SNVT_time_stamp__{
     WORD16   year;   /* = #### 16-bits */
     BYTE  month;  /* = ##    8-bits */
     BYTE  day;    /* = ##    8-bits */
     BYTE  hour;   /* = ##    8-bits */
     BYTE  minute; /* = ##    8-bits */
     BYTE  second; /* = ##    8-bits */
```

```
 } __attribute__ ((__packed__));

typedef struct __SNVT_time_stamp__ SNVT_time_stamp;

extern SNVT_time_stamp  *nvi332Version ;
extern SNVT_time_stamp  *nviNeuronVer ;
extern SNVT_time_stamp  *nviMIO_Version ;
extern SNVT_time_stamp  *PanelVersion ; // Neuron software version
extern SNVT_time_stamp  *nviTime ;
extern SNVT_time_stamp  *nvoTimeSet ;


struct __SNVT_alarm__
{
      BYTE      location[ 6 ];
      WORD16    object_id;
      BYTE      alarm_type;
      BYTE      priority_level;
      WORD16    index_to_SNVT;
      BYTE      value[ 4 ];
      WORD16    year;
      BYTE   month;
      BYTE   day;
      BYTE   hour;
      BYTE   minute;
      BYTE   second;
      WORD16    millisecond;
      BYTE      alarm_limit[ 4 ];

} __attribute__ ((__packed__));

typedef struct __SNVT_alarm__ SNVT_alarm;

extern SNVT_alarm       *nviAlarms ;


struct __SNVT_obj_request__TagName__
{
      WORD16 object_id;
      BYTE object_request;

} __attribute__ ((__packed__));

typedef struct __SNVT_obj_request__TagName__ SNVT_obj_request;

extern SNVT_obj_request  *nviRequest ;


struct __UNVT_HeadConfig1__TagName__
{
      float   inst_opac_level1 ;
      float   inst_opac_level2 ;
      float   min_opac_level1 ;
      float   min_opac_level2 ;
      float   avg_opac_level1 ;
      float   avg_opac_level2 ;
      float   DirtCompAlarm ;
```

```
}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig1__TagName__ UNVT_HeadConfig1 ;

extern UNVT_HeadConfig1  *nviHeadConfig_1 ;
extern UNVT_HeadConfig1  *nvoHeadConfig_1 ;


struct __UNVT_HeadConfig2__TagName__
{
      float   PLCFfactor ;
      float   ZeroCalSetpt ;
      float   SpanCalSetpt ;
      float   CalDelta ;
      BYTE    AveSendTime ; // selectable average send time in minutes

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig2__TagName__ UNVT_HeadConfig2 ;

extern UNVT_HeadConfig2  *nviHeadConfig_2 ;
extern UNVT_HeadConfig2  *nvoHeadConfig_2 ;


struct __UNVT_HeadConfig3__TagName__
{
      float   inst_optD_level1 ;
      float   inst_optD_level2 ;
      float   min_optD_level1 ;
      float   min_optD_level2 ;
      float   avg_optD_level1 ;
      float   avg_optD_level2 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig3__TagName__ UNVT_HeadConfig3 ;

extern UNVT_HeadConfig3  *nviHeadConfig_3_1 ;
extern UNVT_HeadConfig3  *nviHeadConfig_3_2 ;
extern UNVT_HeadConfig3  *nviHeadConfig_3_3 ;
extern UNVT_HeadConfig3  *nviHeadConfig_3_4 ;
extern UNVT_HeadConfig3  *nvoHeadConfig_3 ;


struct __UNVT_HeadConfig4__TagName__
{
      float   MassLoad_X1 ;
      float   MassLoad_Y1 ;
      float   MassLoad_X2 ;
      float   MassLoad_Y2 ;
      float   MassLoad_X3 ;
      float   MassLoad_Y3 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig4__TagName__ UNVT_HeadConfig4 ;
```

```
extern UNVT_HeadConfig4  *nviHeadConfig_4 ;
extern UNVT_HeadConfig4  *nvoHeadConfig_4 ;


struct __UNVT_HeadConfig5__TagName__
{
      float   inst_Dust_level1 ;
      float   inst_Dust_level2 ;
      float   min_Dust_level1 ;
      float   min_Dust_level2 ;
      float   avg_Dust_level1 ;
      float   avg_Dust_level2 ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig5__TagName__ UNVT_HeadConfig5 ;

extern UNVT_HeadConfig5  *nviHeadConfig_5 ;
extern UNVT_HeadConfig5  *nvoHeadConfig_5 ;


struct __UNVT_HeadConfig6__TagName__
{
      float   Temp_Deg_C ;
      float   AbsPressure ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig6__TagName__ UNVT_HeadConfig6 ;

extern UNVT_HeadConfig6  *nviHeadConfig_6 ;
extern UNVT_HeadConfig6  *nvoHeadConfig_6 ;


struct  __UNVT_HeadConfig7__TagName__
{
      BYTE   cal_time_hour ;
      BYTE   cal_time_min ;
      BYTE   cal_interval_hour ;
      BYTE   filler ;
      WORD16   span_secs ;
      WORD16   zero_secs ;
      WORD16   plcf_secs ;
      WORD16   dirt_secs ;

}  __attribute__ ((__packed__));

typedef struct  __UNVT_HeadConfig7__TagName__ UNVT_HeadConfig7 ;

extern UNVT_HeadConfig7  *nviHeadConfig_7 ;
extern UNVT_HeadConfig7  *nvoHeadConfig_7 ;


struct __UNVT_HeadConfig8__TagName__
{
      WORD16  SignalGain ;
```

```
      WORD16  ReferenceGain ;
      WORD16  CommonGain ;

}  __attribute__ ((__packed__));

typedef struct __UNVT_HeadConfig8__TagName__ UNVT_HeadConfig8 ;

extern UNVT_HeadConfig8  *nviHeadConfig_8 ;
extern UNVT_HeadConfig8  *nvoHeadConfig_8 ;


struct __UNVT_MIO_config__TagName__
{
      BYTE    A_output_type ;
      BYTE    A_calibration ;
      float   A_zero ;
      float   A_full_scale ;
      BYTE    B_output_type ;
      BYTE    B_calibration ;
      float   B_zero ;
      float   B_full_scale ;
      BYTE    digital_type[8] ;

}  __attribute__ ((__packed__));

typedef struct __UNVT_MIO_config__TagName__ UNVT_MIO_config ;
/* contains 2 analog output & 1 digital configuration */
extern UNVT_MIO_config   *nviMIO_Config_1 ;
extern UNVT_MIO_config   *nviMIO_Config_2 ;
extern UNVT_MIO_config   *nviMIO_Config_3 ;
extern UNVT_MIO_config   *nvoMIO_Config_1 ;
extern UNVT_MIO_config   *nvoMIO_Config_2 ;
extern UNVT_MIO_config   *nvoMIO_Config_3 ;



struct __UNVT_MIO_status__TagName__
{
      BYTE     invalid_mode_request  : 1 ;
      BYTE     invalid_AO_config     : 1 ;
      BYTE     invalid_DIO_config    : 1 ;
      BYTE     reserved              : 5 ;
      BYTE     MIO_mode ;  /* the current mode of the Multi_i/O Module */

}  __attribute__ ((__packed__));

typedef struct __UNVT_MIO_status__TagName__ UNVT_MIO_status ;

extern UNVT_MIO_status  *nviMIO_Status ;


struct __UNVT_AuditData__TagName__
{

      BYTE AuditMode;     //(enum: NOT_AUDIT, RUN1, RUN2, RUN3,
                                  //       RUN4, RUN5, CLEAR_AUDIT)
      BYTE FilterValue;   //(enum: ZERO, LOW_FILT, MID_FILT, HI_FILT
```

```
      float AuditOpacity;
      float AuditOptDens;
      float AuditDustMass;

} __attribute__ ((__packed__));

typedef struct __UNVT_AuditData__TagName__ UNVT_AuditData ;

extern UNVT_AuditData    *nviAuditData ;


struct      __UNVT_HeadMode__TagName__
{

      BYTE   CalMechanism;   // first four enums of nodemode
      BYTE   SixPIO_AO_Mode; // analog out modes defined in mode560.h
      BYTE   AuditMode;      // (enum: NOT_AUDIT, RUN1, RUN2, RUN3,
                                    //        RUN4, RUN5, CLEAR_AUDIT)
      BYTE   FilterValue;    // (enum: ZERO, LOW_FILT, MID_FILT, HI_FILT

} __attribute__ ((__packed__));

typedef struct    __UNVT_HeadMode__TagName__ UNVT_HeadMode ;

extern UNVT_HeadMode    *nviHeadMode ;

extern float  *nviSelAveOpacity ;
extern float  *nviInstOpacity ;
extern float  *nvi1MinOpacity ;
extern float  *nviZeroOpacCal ;
extern float  *nviSpanOpacCal ;
extern float  *nviDirtCompValue ;
extern float  *nviInstMassLoad ;
extern float  *nviAveMassLoad ;
extern float  *nvi1MinMassLoad ;
extern float  *nviZeroMassLoad ;
extern float  *nviSpanMassLoad ;
extern float  *nviDirtMassLoad ;
extern float  *nviAbsPress ;
extern float  *nviTemp_C ;
extern float  *nviML_CorrFactor ;
extern float  *nviAveOptDens ;
extern float  *nviInstOptDens ;
extern float  *nvi1MinOptDens ;
extern float  *nviZeroOptDens ;
extern float  *nviSpanOptDens ;
extern float  *nviDirtOptDens ;


#ifdef __cplusplus
}
#endif

#endif
```

Modbus Input Registers for LaserHawk 360

Variable	Member	Type	Large Address	Offset
nvi332Version		SNVT_time_stamp		
	year	WORD (HIBYTE)	30002	1
		WORD (LOWBYTE)		
	month	BYTE	30003	2
	day	BYTE		
	hour	BYTE	30004	3
	minute	BYTE		
	second	BYTE	30005	4
	(reserved)	BYTE		
nviNeuronVer		SNVT_time_stamp		
	year	WORD (HIBYTE)	30006	5
		WORD (LOWBYTE)		
	month	BYTE	30007	6
	day	BYTE		
	hour	BYTE	30008	7
	minute	BYTE		
	second	BYTE	30009	8
	(reserved)	BYTE		
nviMIO_Version		SNVT_time_stamp		
	year	WORD (HIBYTE)	30010	9
		WORD (LOWBYTE)		
	month	BYTE	30011	10
	day	BYTE		
	hour	BYTE	30012	11
	minute	BYTE		
	second	BYTE	30013	12
	(reserved)	BYTE		
nviTime		SNVT_time_stamp		
	year	WORD (HIBYTE)	30014	13
		WORD (LOWBYTE)		
	month	BYTE	30015	14
	day	BYTE		
	hour	BYTE	30016	15
	minute	BYTE		
	second	BYTE	30017	16
	(reserved)	BYTE		
nviAlarms		SNVT_alarm		
	location	BYTE[0]	30018	17
		BYTE[1]		

		BYTE[2]	30019	18
		BYTE[3]		
		BYTE[4]	30020	19
		BYTE[5]		
	object_id	WORD (HIBYTE)	30021	20
		WORD (LOWBYTE)		
	alarm_type	BYTE	30022	21
	priority_level	BYTE		
	index_to_SNVT	WORD (HIBYTE)	30023	22
		WORD (LOWBYTE)		
	value	BYTE[0]	30024	23
		BYTE[1]		
		BYTE[2]	30025	24
		BYTE[3]		
	year	WORD (HIBYTE)	30026	25
		WORD (LOWBYTE)		
	month	BYTE	30027	26
	day	BYTE		
	hour	BYTE	30028	27
	minute	BYTE		
	second	BYTE	30029	28
	millisecond	WORD (HIBYTE)		
		WORD (LOWBYTE)	30030	29
	alarm_limit	BYTE[0]		
		BYTE[1]	30031	30
		BYTE[2]		
		BYTE[3]	30032	31
		FILLER		
nviHeadState		UNVT_HeadStatus		
	DirtDriftAlarm	BYTE	30033	32
	CalSpanAlarm			
	CalZeroAlarm			
	LoSetVoltZERO			
	LoSetVoltSTACK			
	SET_Backgrnd			
	SET_Zero			
	SET_STACK			
	UpscalePosErr	BYTE		
	ZeroPosErr			
	NormalPosErr			
	ReferenceFault			
	ADC_Fault			
	Out_of_Service			

	PurgeFailRetro			
	PurgeFailHead			
nviHeadMode		UNVT_HeadMode		
	CalMechanism	node_mode (BYTE)	30034	33
	SixPIO_AO_Mode	node_mode (BYTE)		
	AuditMode	AUDITMODE (BYTE)	30035	34
	FilterValue	FILTERVALUE (BYTE)		
nviMIO_Status		UNVT_MIO_status		
	invalid_mode_request	BYTE	30029	35
	invalid_AO_config			
	invalid_DIO_config			
	reserved			
	MIO_mode	node_mode (BYTE)		
nviOpacityAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30029	36
	Bit_6			
	AveAlarm2			
	AveAlarm1			
	MinAlarm2			
	MinAlarm1			
	InstAlarm2			
	InstAlarm1			
	(reserved)	BYTE		
nviSelAveOpacity		SNVT_count_inc_f		
		IEEE float (S+MSE)	30038	37
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30039	38
		IEEE float (LOBYTE(LSM))		
nviInstOpacity		SNVT_count_inc_f		
		IEEE float (S+MSE)	30040	39
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30041	40
		IEEE float (LOBYTE(LSM))		
nvi1MinOpacity		SNVT_count_inc_f		
		IEEE float (S+MSE)	30042	41
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30043	42
		IEEE float (LOBYTE(LSM))		
nviZeroOpacCal		SNVT_count_inc_f		
		IEEE float (S+MSE)	30044	43
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30045	44
		IEEE float (LOBYTE(LSM))		

nviSpanOpacCal		SNVT_count_inc_f		
		IEEE float (S+MSE)	30046	45
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30047	46
		IEEE float (LOBYTE(LSM))		
nviDirtCompValue		SNVT_count_inc_f		
		IEEE float (S+MSE)	30048	47
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30049	48
		IEEE float (LOBYTE(LSM))		
nviServiceData1		UNVT_Service_One		
	SignalVolts	IEEE float (S+MSE)	30050	49
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30051	50
		IEEE float (LOBYTE(LSM))		
	ReferenceVolts	IEEE float (S+MSE)	30052	51
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30053	52
		IEEE float (LOBYTE(LSM))		
	LED_current	IEEE float (S+MSE)	30054	53
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30055	54
		IEEE float (LOBYTE(LSM))		
	StackSetVols	IEEE float (S+MSE)	30056	55
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30057	56
		IEEE float (LOBYTE(LSM))		
	ZeroSetVolts	IEEE float (S+MSE)	30058	57
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30059	58
		IEEE float (LOBYTE(LSM))		
	BackgndSetVols	IEEE float (S+MSE)	30060	59
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30061	60
		IEEE float (LOBYTE(LSM))		
	HeadTemperature	IEEE float (S+MSE)	30062	61
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30063	62
		IEEE float (LOBYTE(LSM))		
	CalWheelPosition	WORD (HIBYTE)	30064	63
		WORD (LOWBYTE)		
nviServiceData2		UNVT_Service_Two		
	Pos_15V	IEEE float (S+MSE)	30065	64

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30066	65
		IEEE float (LOBYTE(LSM))		
	Neg_15V	IEEE float (S+MSE)	30067	66
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30068	67
		IEEE float (LOBYTE(LSM))		
	Pos_5Vanalog	IEEE float (S+MSE)	30069	68
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30070	69
		IEEE float (LOBYTE(LSM))		
	Neg_5Vanalog	IEEE float (S+MSE)	30071	70
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30072	71
		IEEE float (LOBYTE(LSM))		
	Pos_5Vdigital	IEEE float (S+MSE)	30073	72
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30074	73
		IEEE float (LOBYTE(LSM))		
nviAuditData		UNVT_AuditData		
	AuditMode	AUDITMODE (BYTE)	30075	74
	FilterValue	FILTERVALUE (BYTE)		
	AuditOpacity	IEEE float (S+MSE)	30076	75
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30077	76
		IEEE float (LOBYTE(LSM))		
	AuditOptDens	IEEE float (S+MSE)	30078	77
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30079	78
		IEEE float (LOBYTE(LSM))		
	AuditDustMass	IEEE float (S+MSE)	30080	79
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30081	80
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_1		UNVT_HeadConfig1		
	inst_opac_level1	IEEE float (S+MSE)	30082	81
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30083	82
		IEEE float (LOBYTE(LSM))		
	inst_opac_level2	IEEE float (S+MSE)	30084	83
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30085	84
		IEEE float (LOBYTE(LSM))		

	min_opac_level1	IEEE float (S+MSE)	30086	85
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30087	86
		IEEE float (LOBYTE(LSM))		
	min_opac_level2	IEEE float (S+MSE)	30088	87
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30089	88
		IEEE float (LOBYTE(LSM))		
	avg_opac_level1	IEEE float (S+MSE)	30090	89
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30091	90
		IEEE float (LOBYTE(LSM))		
	avg_opac_level2	IEEE float (S+MSE)	30092	91
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30093	92
		IEEE float (LOBYTE(LSM))		
	DirtCompAlarm	IEEE float (S+MSE)	30094	93
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30095	94
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_2		UNVT_HeadConfig2		
	PLCFfactor	IEEE float (S+MSE)	30096	95
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30097	96
		IEEE float (LOBYTE(LSM))		
	ZeroCalSetpt	IEEE float (S+MSE)	30098	97
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30099	98
		IEEE float (LOBYTE(LSM))		
	SpanCalSetpt	IEEE float (S+MSE)	30100	99
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30101	100
		IEEE float (LOBYTE(LSM))		
	CalDelta	IEEE float (S+MSE)	30102	101
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30103	102
		IEEE float (LOBYTE(LSM))		
	AveSendTime	BYTE	30104	103
	(reserved)			
nviHeadConfig_3_1		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	30105	104
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30106	105

		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	30107	106
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30108	107
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	30109	108
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30110	109
		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	30111	110
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30112	111
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	30113	112
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30114	113
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	30115	114
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30116	115
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_3_2		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	30117	116
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30118	117
		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	30119	118
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30120	119
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	30121	120
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30122	121
		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	30123	122
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30124	123
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	30125	124
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30126	125
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	30127	126

		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30128	127
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_3_3		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	30129	128
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30130	129
		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	30131	130
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30132	131
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	30133	132
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30134	133
		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	30135	134
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30136	135
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	30137	136
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30138	137
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	30139	138
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30140	139
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_3_4		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	30141	140
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30142	141
		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	30143	142
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30144	143
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	30145	144
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30146	145
		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	30147	146
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30148	147
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	30149	148
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30150	149
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	30151	150
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30152	151
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_4		UNVT_HeadConfig4		
	MassLoad_X1	IEEE float (S+MSE)	30153	152
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30154	153
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y1	IEEE float (S+MSE)	30155	154
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30156	155
		IEEE float (LOBYTE(LSM))		
	MassLoad_X2	IEEE float (S+MSE)	30157	156
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30158	157
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y2	IEEE float (S+MSE)	30159	158
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30160	159
		IEEE float (LOBYTE(LSM))		
	MassLoad_X3	IEEE float (S+MSE)	30161	160
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30162	161
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y3	IEEE float (S+MSE)	30163	162
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30164	163
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_5		UNVT_HeadConfig5		
	inst_Dust_level1	IEEE float (S+MSE)	30165	164
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30166	165
		IEEE float (LOBYTE(LSM))		
	inst_Dust_level2	IEEE float (S+MSE)	30167	166
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30168	167

		IEEE float (LOBYTE(LSM))		
	min_Dust_level1	IEEE float (S+MSE)	30169	168
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30170	169
		IEEE float (LOBYTE(LSM))		
	min_Dust_level2	IEEE float (S+MSE)	30171	170
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30172	171
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level1	IEEE float (S+MSE)	30173	172
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30174	173
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level2	IEEE float (S+MSE)	30175	174
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30176	175
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_6		UNVT_HeadConfig6		
	Temp_Deg_C	IEEE float (S+MSE)	30177	176
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30178	177
		IEEE float (LOBYTE(LSM))		
	AbsPressure	IEEE float (S+MSE)	30179	178
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30180	179
		IEEE float (LOBYTE(LSM))		
nviHeadConfig_7		UNVT_HeadConfig7		
	cal_time_hour	BYTE	30181	180
	cal_time_min	BYTE		
	cal_interval_hour	BYTE	30182	181
	filler	BYTE		
	span_secs	WORD (HIBYTE)	30183	182
		WORD (LOWBYTE)		
	zero_secs	WORD (HIBYTE)	30184	183
		WORD (LOWBYTE)		
	plcf_secs	WORD (HIBYTE)	30185	184
		WORD (LOWBYTE)		
	dirt_secs	WORD (HIBYTE)	30186	185
		WORD (LOWBYTE)		
nviHeadConfig_8		UNVT_HeadConfig8		
	SignalGain	WORD (HIBYTE)	30187	186
		WORD (LOWBYTE)		
	ReferenceGain	WORD (HIBYTE)	30188	187

		WORD (LOWBYTE)		
	CommonGain	WORD (HIBYTE)	30189	188
		WORD (LOWBYTE)		
nviMIO_Config_1		UNVT_MIO_config		
	A_output_type	BYTE	30190	189
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30191	190
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30192	191
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30193	192
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30194	193
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	30195	194
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30196	195
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30197	196
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30198	197
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30199	198
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30200	199
		BYTE[1]		
		BYTE[2]	30201	200
		BYTE[3]		
		BYTE[4]	30202	201
		BYTE[5]		
		BYTE[6]	30203	202
		BYTE[7]		
nviMIO_Config_2		UNVT_MIO_config		
	A_output_type	BYTE	30204	203
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30205	204
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30206	205
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30207	206
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30208	207
		IEEE float (LOBYTE(LSM))		

	B_output_type	BYTE	30209	208
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30210	209
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30211	210
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30212	211
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30213	212
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30214	213
		BYTE[1]		
		BYTE[2]	30215	214
		BYTE[3]		
		BYTE[4]	30216	215
		BYTE[5]		
		BYTE[6]	30217	216
		BYTE[7]		
nviMIO_Config_3		UNVT_MIO_config		
	A_output_type	BYTE	30218	217
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	30219	218
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30220	219
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	30221	220
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30222	221
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	30223	222
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	30224	223
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30225	224
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	30226	225
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30227	226
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	30228	227
		BYTE[1]		
		BYTE[2]	30229	228
		BYTE[3]		

		BYTE[4]	30230	229
		BYTE[5]		
		BYTE[6]	30231	230
		BYTE[7]		
nviInstMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30232	231
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30233	232
		IEEE float (LOBYTE(LSM))		
nviAveMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30234	233
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30235	234
		IEEE float (LOBYTE(LSM))		
nvi1MinMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30236	235
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30237	236
		IEEE float (LOBYTE(LSM))		
nviZeroMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30238	237
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30239	238
		IEEE float (LOBYTE(LSM))		
nviSpanMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30240	239
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30241	240
		IEEE float (LOBYTE(LSM))		
nviDirtMassLoad		SNVT_count_inc_f		
		IEEE float (S+MSE)	30242	241
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30243	242
		IEEE float (LOBYTE(LSM))		
nviAbsPress		SNVT_press_f		
		IEEE float (S+MSE)	30244	243
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30245	244
		IEEE float (LOBYTE(LSM))		
nviTemp_C		SNVT_temp_f		
		IEEE float (S+MSE)	30246	245
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30247	246

		IEEE float (LOBYTE(LSM))		
nviML_CorrFactor		SNVT_count_inc_f		
		IEEE float (S+MSE)	30248	247
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30249	248
		IEEE float (LOBYTE(LSM))		
nviMassAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30250	249
	Bit_6	BYTE		
	AveAlarm2	BYTE	30251	250
	AveAlarm1	BYTE		
	MinAlarm2	BYTE	30252	251
	MinAlarm1	BYTE		
	InstAlarm2	BYTE	30253	252
	InstAlarm1	BYTE		
nviRequest		SNVT_obj_request		
	object_id	WORD (HIBYTE)	30254	253
		WORD (LOWBYTE)		
	object_request	BYTE	30255	254
	(reserved)			
nviAveOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30256	255
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30257	256
		IEEE float (LOBYTE(LSM))		
nviInstOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30258	257
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30259	258
		IEEE float (LOBYTE(LSM))		
nvi1MinOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30260	259
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30261	260
		IEEE float (LOBYTE(LSM))		
nviZeroOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30262	261
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30263	262
		IEEE float (LOBYTE(LSM))		
nviSpanOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30264	263
		IEEE float (LSE+MSM)		

		IEEE float (HIBYTE(LSM))	30265	264
		IEEE float (LOBYTE(LSM))		
nviDirtOptDens		SNVT_count_inc_f		
		IEEE float (S+MSE)	30266	265
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	30267	266
		IEEE float (LOBYTE(LSM))		
nviOptDensAlarms		UNVT_AlarmSel		
	Bit_7	BYTE	30268	267
	Bit_6	BYTE		
	AveAlarm2	BYTE	30269	268
	AveAlarm1	BYTE		
	MinAlarm2	BYTE	30270	269
	MinAlarm1	BYTE		
	InstAlarm2	BYTE	30271	270
	InstAlarm1	BYTE		
nviNetworkStatus		UNVT_ModeRequest		
	node_id	BYTE	30272	271
	mode	BYTE		
PanelVersion		SNVT_time_stamp		
	year	WORD (HIBYTE)	30273	272
		WORD (LOWBYTE)		
	month	BYTE	30274	273
	day	BYTE		
	hour	BYTE	30275	274
	minute	BYTE		
	second	BYTE	30276	275
	(reserved)	BYTE		

Modbus Holding Registers for LaserHawk 360

Variable	Member	Type	Large Address	Offset
nvoModeRequest_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40002**	**1**
nvoModeRequest		UNVT_ModeRequest		
	node_id	BYTE	40003	2
	mode	BYTE		
nvoTimeSet_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40004**	**3**
nvoTimeSet		SNVT_time_stamp		
	year	WORD (HIBYTE)	40005	4
		WORD (LOWBYTE)		
	month	BYTE	40006	5
	day	BYTE		
	hour	BYTE	40007	6
	minute	BYTE		
	second	BYTE	40008	7
	(reserved)	BYTE		
nvoHeadConfig_1_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40009**	**8**
nvoHeadConfig_1		UNVT_HeadConfig1		
	inst_opac_level1	IEEE float (S+MSE)	40010	9
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40011	10
		IEEE float (LOBYTE(LSM))		
	inst_opac_level2	IEEE float (S+MSE)	40012	11
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40013	12
		IEEE float (LOBYTE(LSM))		
	min_opac_level1	IEEE float (S+MSE)	40014	13
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40015	14
		IEEE float (LOBYTE(LSM))		
	min_opac_level2	IEEE float (S+MSE)	40016	15
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40017	16
		IEEE float (LOBYTE(LSM))		
	avg_opac_level1	IEEE float (S+MSE)	40018	17
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40019	18
		IEEE float (LOBYTE(LSM))		

	avg_opac_level2	IEEE float (S+MSE)	40020	19
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40021	20
		IEEE float (LOBYTE(LSM))		
	DirtCompAlarm	IEEE float (S+MSE)	40022	21
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40023	22
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_2_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40024**	**23**
nvoHeadConfig_2		UNVT_HeadConfig2		
	PLCFfactor	IEEE float (S+MSE)	40025	24
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40026	25
		IEEE float (LOBYTE(LSM))		
	ZeroCalSetpt	IEEE float (S+MSE)	40027	26
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40028	27
		IEEE float (LOBYTE(LSM))		
	SpanCalSetpt	IEEE float (S+MSE)	40029	28
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40030	29
		IEEE float (LOBYTE(LSM))		
	CalDelta	IEEE float (S+MSE)	40031	30
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40032	31
		IEEE float (LOBYTE(LSM))		
	AveSendTime	BYTE	40033	32
	(reserved)			
nvoHeadConfig_3_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40034**	**33**
nvoHeadConfig_3		UNVT_HeadConfig3		
	inst_optD_level1	IEEE float (S+MSE)	40035	34
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40036	35
		IEEE float (LOBYTE(LSM))		
	inst_optD_level2	IEEE float (S+MSE)	40037	36
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40038	37
		IEEE float (LOBYTE(LSM))		
	min_optD_level1	IEEE float (S+MSE)	40039	38
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40040	39

		IEEE float (LOBYTE(LSM))		
	min_optD_level2	IEEE float (S+MSE)	40041	40
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40042	41
		IEEE float (LOBYTE(LSM))		
	avg_optD_level1	IEEE float (S+MSE)	40043	42
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40044	43
		IEEE float (LOBYTE(LSM))		
	avg_optD_level2	IEEE float (S+MSE)	40045	44
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40046	45
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_4_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40047**	**46**
nvoHeadConfig_4		UNVT_HeadConfig4		
	MassLoad_X1	IEEE float (S+MSE)	40048	47
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40049	48
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y1	IEEE float (S+MSE)	40050	49
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40051	50
		IEEE float (LOBYTE(LSM))		
	MassLoad_X2	IEEE float (S+MSE)	40052	51
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40053	52
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y2	IEEE float (S+MSE)	40054	53
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40055	54
		IEEE float (LOBYTE(LSM))		
	MassLoad_X3	IEEE float (S+MSE)	40056	55
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40057	56
		IEEE float (LOBYTE(LSM))		
	MassLoad_Y3	IEEE float (S+MSE)	40058	57
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40059	58
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_5_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40060**	**59**
nvoHeadConfig_5		UNVT_HeadConfig5		

	inst_Dust_level1	IEEE float (S+MSE)	40061	60
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40062	61
		IEEE float (LOBYTE(LSM))		
	inst_Dust_level2	IEEE float (S+MSE)	40063	62
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40064	63
		IEEE float (LOBYTE(LSM))		
	min_Dust_level1	IEEE float (S+MSE)	40065	64
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40066	65
		IEEE float (LOBYTE(LSM))		
	min_Dust_level2	IEEE float (S+MSE)	40067	66
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40068	67
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level1	IEEE float (S+MSE)	40069	68
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40070	69
		IEEE float (LOBYTE(LSM))		
	avg_Dust_level2	IEEE float (S+MSE)	40071	70
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40072	71
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_6_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40073**	**72**
nvoHeadConfig_6		UNVT_HeadConfig6		
	Temp_Deg_C	IEEE float (S+MSE)	40074	73
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40075	74
		IEEE float (LOBYTE(LSM))		
	AbsPressure	IEEE float (S+MSE)	40076	75
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40077	76
		IEEE float (LOBYTE(LSM))		
nvoHeadConfig_7_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40078**	**77**
nvoHeadConfig_7		UNVT_HeadConfig7		
	cal_time_hour	BYTE	40079	78
	cal_time_min	BYTE		
	cal_interval_hour	BYTE	40080	79
	filler	BYTE		
	span_secs	WORD (HIBYTE)	40081	80

		WORD (LOWBYTE)		
	zero_secs	WORD (HIBYTE)	40082	81
		WORD (LOWBYTE)		
	plcf_secs	WORD (HIBYTE)	40083	82
		WORD (LOWBYTE)		
	dirt_secs	WORD (HIBYTE)	40084	83
		WORD (LOWBYTE)		
nvoHeadConfig_8_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40085**	**84**
nvoHeadConfig_8		UNVT_HeadConfig8		
	SignalGain	WORD (HIBYTE)	40086	85
		WORD (LOWBYTE)		
	ReferenceGain	WORD (HIBYTE)	40087	86
		WORD (LOWBYTE)		
	CommonGain	WORD (HIBYTE)	40088	87
		WORD (LOWBYTE)		
nvoMIO_Config_1_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40089**	**88**
nvoMIO_Config_1		UNVT_MIO_config		
	A_output_type	BYTE	40090	89
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40091	90
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40092	91
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40093	92
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40094	93
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40095	94
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40096	95
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40097	96
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40098	97
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40099	98
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40100	99
		BYTE[1]		
		BYTE[2]	40101	100
		BYTE[3]		

		BYTE[4]	40102	101
		BYTE[5]		
		BYTE[6]	40103	102
		BYTE[7]		
nvoMIO_Config_2_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40104**	**103**
nvoMIO_Config_2		UNVT_MIO_config		
	A_output_type	BYTE	40105	104
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40106	105
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40107	106
		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40108	107
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40109	108
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40110	109
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40111	110
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40112	111
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40113	112
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40114	113
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40115	114
		BYTE[1]		
		BYTE[2]	40116	115
		BYTE[3]		
		BYTE[4]	40117	116
		BYTE[5]		
		BYTE[6]	40118	117
		BYTE[7]		
nvoMIO_Config_3_trig		**WORD (HIBYTE) WORD (LOWBYTE)**	**40119**	**118**
nvoMIO_Config_3		UNVT_MIO_config		
	A_output_type	BYTE	40120	119
	A_calibration	BYTE		
	A_zero	IEEE float (S+MSE)	40121	120
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40122	121

		IEEE float (LOBYTE(LSM))		
	A_full_scale	IEEE float (S+MSE)	40123	122
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40124	123
		IEEE float (LOBYTE(LSM))		
	B_output_type	BYTE	40125	124
	B_calibration	BYTE		
	B_zero	IEEE float (S+MSE)	40126	125
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40127	126
		IEEE float (LOBYTE(LSM))		
	B_full_scale	IEEE float (S+MSE)	40128	127
		IEEE float (LSE+MSM)		
		IEEE float (HIBYTE(LSM))	40129	128
		IEEE float (LOBYTE(LSM))		
	digital_type	BYTE[0]	40130	129
		BYTE[1]		
		BYTE[2]	40131	130
		BYTE[3]		
		BYTE[4]	40132	131
		BYTE[5]		
		BYTE[6]	40133	132
		BYTE[7]		
nvoDataRequest_trig		**WORD (HIBYTE)** **WORD (LOWBYTE)**	**40134**	**133**
nvoDataRequest		UNVT_ModeRequest		
	node_id	BYTE	40135	134
	mode	BYTE		

APPENDIX E

SM8200 VARIABLES

(This page intentionally left blank.)

This section describes the variables for the SM8200 instrument. A description of each network variable is given, followed by the C/C++ definitions of the structures, constants and variable types. The Modbus register mapping is also given along with the table of indexed values for the "indexed network variables". These indexed network variables are used by the SM8200 for configuration or displaying additional data. They are named: nviDataSM8200, nviData_RESERVED, nviCfgSM8200, nviCfg_RESERVED, nviCfgX, nvoDataSM8200, nvoData_RESERVED, nvoCfgSM8200, nvoCfg_RESERVED and nvoCfgX..

All network variables are prefixed with a three letter prefix, either "nvi" or "nvo" which translates to:

nvi – Network Variable Input;
nvo – Network Variable Output.

The direction of Input or Output is relative to the Ethernet Module.

nviStatusSM8200
nviStatus_RESERVED
nviStatusMIO1
nviStatusMIO2

Type: UNVT_Status

Members and Units:

year	Current time
month	Current time
day	Current time
hour	Current time
minute	Current time
sec	Current time
com_flag	Communication status (See header file “compdef.h” #12)
primary_status	Current status (See header file “compdef.h” #13 for SM8200, and #47 for MIO1 and MIO2)
extended_status	Current status (See header file “compdef.h” #14 for SM8200, and #48 for MIO1 and MIO2)
mode	Current mode (See header file “compdef.h” #18 for SM8200, and #45 for MIO1 and MIO2)
command	Command acknowledgement (not used)
major_ver_NC	Software version NEURON
minor_ver_NC	Software version NEURON
major_ver_332	Software version instrument
minor_ver_332	Software version instrument
major_ver_FPGA	FPGA version
minor_ver_FPGA	FPGA version
major_ver_TS	Instrument configuration version
minor_ver_TS	Instrument configuration version

Description:
This network variable is issued by the Analyzer or the MIO every second to reflect the analyzer time, mode, status, and firmware versions.

nviInstValSM8200
nviConcAvgSM8200
nviZeroAvgSM8200
nviSpanAvgSM8200
nviInstValMBTU
nviConcAvgMBTU
nviZeroAvgMBTU
nviSpanAvgMBTU
nviInstVal_RESERVED

Type: UNVT_ConcData

Members and Units:

year	Time at sample
month	Time at sample
day	Time at sample
hour	Time at sample
minute	Time at sample

nviConcAvg_RESERVED
nviZeroAvg_RESERVED
nviSpanAvg_RESERVED

sec	Time at sample
mode	Mode in which the measurement is taken (See header file “compdef.h” #18 for SM8200)
status_summary	Status for DAS (See header file “compdef.h” #15 for SM8200)
extended_summary	Status for DAS (See header file “compdef.h” #16 for SM8200)
conc1	Concentration number 1 (NO for SM8200)
conc2	Concentration number 2 (SO2 for SM8200)
conc3	Concentration number 3 (NO2 for SM8200)
conc4	Concentration number 4 (NH3 for SM8200)

Description:
This network variable is reports the concentration measurement. Instantaneous (InstVal) variable reports the instantaneous data regardless of the mode, while Concentration Average (ConcAvg) reports average values only in NORMAL/NORMAL ACQUIRE mode; Zero Average reports average values only in ZERO/ZERO ACQUIRE modes; Span Average reports average values only in SPAN/SPAN ACQUIRE/EO CAL/EO CAL ACQUIRE/CGA/CGA ACQUIRE modes. The MBTU variables have the SM8200 data in MBTU units.

nviConcO
nviZeroO
nviSpanO

Type: UNVT_OData

Members and Units:

year	Time at sample
month	Time at sample
day	Time at sample
hour	Time at sample
minute	Time at sample
sec	Time at sample
source	1 = External Analyzer 2 = Calculated
mode	Mode in which the measurement is taken (See header file “compdef.h” #18 for SM8200)

	primary_status — Current status (See header file "compdef.h" #13 for SM8200) *extended_status* — Current status (See header file "compdef.h" #14 for SM8200) *status_summary* — Status for DAS (See header file "compdef.h" #15 for SM8200) *extended_summary* — Status for DAS (See header file "compdef.h" #16 for SM8200) *inst* — Instantaneous oxygen concentration *avg* — Average oxygen concentration
	Description: This network variable is reports the oxygen concentration measurement. ConcO reports average values only in NORMAL/NORMAL ACQUIRE mode; ZeroO reports average values only in ZERO/ZERO ACQUIRE modes; SpanO reports average values only in SPAN/SPAN ACQUIRE/EO CAL/EO CAL ACQUIRE/CGA/CGA ACQUIRE modes.
nviDataX	Type: UNVT_AUXData
	Members and Units: This variable is a union of FloatOutput type and ScaledOutput. *fltdata* — FloatOutput type union member structure. *scldata* — ScaledOutput (integer) type union member structure. *FloatOutput* *type* — Data type of the current structure. This element designates which union member is used. (See header file "compdef.h" #9 for SM8200) *num_io* — I/O board number *year* — Time at sample *month* — Time at sample *day* — Time at sample *hour* — Time at sample *minute* — Time at sample *sec* — Time at sample

	com_flag — Communication status (See header file “compdef.h” #12)
	status — Internal status – factory use only.
	digital — State of the digital (isolator) inputs.
	analog1 — Analog value – floating point.
	analog2 — Analog value – floating point.
	analog3 — Analog value – floating point.
	analog4 — Analog value – floating point.

ScaledOutput	
type	Data type of the current structure. This element designates which union member is used. (See header file “compdef.h” #9 for SM8200)
num_io	I/O board number
year	Time at sample
month	Time at sample
day	Time at sample
hour	Time at sample
minute	Time at sample
sec	Time at sample
com_flag	Communication status (See header file “compdef.h” #12)
status	Internal status – factory use only.
digital	State of the digital (isolator) inputs.
analog1	Analog value – integer value.
analog2	Analog value – integer value.
analog3	Analog value – integer value.
analog4	Analog value – integer value.
filler[4]	Not used.

Description:
This network variable is reports the data values from the Plant I/O module.

nvoRTimeSet	Type: UNVT_Time
	Members and Units:

<table>
<tr><td></td><td>
<table>
<tr><td>year</td><td>Desired time</td></tr>
<tr><td>month</td><td>Desired time</td></tr>
<tr><td>day</td><td>Desired time</td></tr>
<tr><td>hour</td><td>Desired time</td></tr>
<tr><td>minute</td><td>Desired time</td></tr>
<tr><td>sec</td><td>Desired time</td></tr>
<tr><td>filler</td><td>Not used</td></tr>
</table>

Description:
The ERP or the Ethernet module sets and/or synchronizes the time of the instruments on the network using this network variable.</td></tr>
<tr><td>nvoMIO1Analog1
nvoMIO1Analog2
nvoMIO2Analog1
nvoMIO2Analog2
nviMIO1Analog1
nviMIO1Analog2
nviMIO2Analog1
nviMIO2Analog2</td><td>Type: UNVT_MIOAnalog

Members and Units:
<table>
<tr><td>instrument_A</td><td>Source selection (See header file “compdef.h” #30)</td></tr>
<tr><td>CalEnable_A</td><td>Configure the output to display calibration data (See header file “compdef.h” #33)</td></tr>
<tr><td>output_type_A</td><td>Output type selection based on the source selection in instrument_A (See header file “compdef.h” #31 for SM8200)</td></tr>
<tr><td>zero_scale_A</td><td>Zero scale value</td></tr>
<tr><td>full_scale_A</td><td>Full scale value</td></tr>
<tr><td>instrument_B</td><td>Source selection (See header file “compdef.h” #30)</td></tr>
<tr><td>CalEnable_B</td><td>Configure the output to display calibration data (See header file “compdef.h” #33)</td></tr>
<tr><td>output_type_B</td><td>Output type selection based on the source selection in instrument_B (See header file “compdef.h” #31 for SM8200)</td></tr>
<tr><td>zero_scale_B</td><td>Zero scale value</td></tr>
<tr><td>full_scale_B</td><td>Full scale value</td></tr>
</table>

Description:
The ERP or Ethernet module issues this network variable to configure or read the MIO’s analog outputs (analog 1 and 2). Error detection of illegal codes will be detected by the MIO and reported in the MIO status.</td></tr>
<tr><td>nviMIO1Relays
nvoMIO1Relays</td><td>Type: UNVT_MIORelays</td></tr>
</table>

nviMIO2Relays
nvoMIO2Relays

Members and Units:

instr_sel[8]	Source selection for relays 1 through 8 (See header file "compdef.h" #34)
relay_sel[8]	Relay selection for relays 1 through 8 based on the above *instr_sel* source selection (See header file "compdef.h" #35, #36, #37 for SM8200 and #46 for external Oxygen analyzer)

Description:
The ERP or Ethernet module issues this network variable to configure or read the MIO's relays. Error detection of illegal codes will be detected by the MIO and reported in the MIO status.

nvoCmdSM8200
nvoCmd_RESERVED
nvoCmdMIO1
nvoCmdMIO2

Type: UNVT_Command

Members and Units:

year	Time at sample
month	Time at sample
day	Time at sample
hour	Time at sample
minute	Time at sample
sec	Time at sample
command	Command sent to the instrument (See header file "compdef.h" #17 for SM8200, and #28 for MIO1 and MIO2)

Description:
The ERP, Ethernet module, MIO1 or MIO2 command an action to the SM8200 analyzer or the MIOs via this network variable.

nviDataSM8200
nviData_RESERVED
nviCfgSM8200
nviCfg_RESERVED
nviCfgX
nvoDataSM8200
nvoData_RESERVED
nvoCfgSM8200
nvoCfg_RESERVED
nvoCfgX

Type: UNVT_Array

Members and Units:

These variables are a union of two different structures. The first structure is for transmitting float point type of variables and the second structure is for transmitting two-byte (WORD) type of variables.

flt_elem	FloatElement type union

	member structure.
wrd_elem	WordElement (integer) type union member structure.

FloatElement	
type	Data type of the current structure. This element designates which union member is used. (See header file “compdef.h” #9 for SM8200)
control	This element is used for transmission control, thus implementing the read/write/acknowledgment model. (See header file “compdef.h” #7)
index	Index of the particular array element. See the index tables in this appendix for the corresponding values.
float_data	The contents of the indexed data or configuration variable in floating-point format.

WordElement	
type	Data type of the current structure. This element designates which union member is used. (See header file “compdef.h” #9 for SM8200)
control	This element is used for transmission control, thus implementing the read/write/acknowledgment model. (See header file “compdef.h” #7)
index	Index of the particular array element. See the index tables in this appendix for the corresponding values.
word_data	The contents of the indexed data or configuration variable in unsigned integer format.
filler	Not used

<table>
<tr><td></td><td></td></tr>
<tr><td></td><td>Description:
The ERP or the Ethernet module use these indexed variables to read/write configuration data to and from the SM8200 instruments, and auxiliary board (X).
To read a particular index, set the type of the output (nvo) variable to the requested type, set the index element to required value, and set its control to READ value. If the index exists the input variable (nvi) will return the value and its control element will be set to WRITE.
To write a particular index, set the type of the output (nvo) variable to the requested type, set the index element to required value, set its control to WRITE value. If the index exists the input variable (nvi) will return the index, value and its control element will be set to WRITE, effectively writing back the value sent.
If there is any failure to READ or WRITE, the returned control will be NACK.</td></tr>
<tr><td rowspan="2">nvoBufferSM8200
nvoBuffer_RESERVED
nvoBufferO
nvoBufferX
nviBufferSM8200
nviBuffer_RESERVED
nviBufferO
nviBufferX</td><td>Type: UNVT_DataBuffer</td></tr>
<tr><td>Members and Units:

<table>
<tr><td>control</td><td>This element is used for transmission control, thus implementing the read/write/acknowledgment model. (See header file “compdef.h” #7)</td></tr>
<tr><td>mode</td><td>Mode in which the measurement is taken (See header file “compdef.h” #18 for SM8200)</td></tr>
<tr><td>index</td><td>Index number of the particular sample in the circular buffer.</td></tr>
<tr><td>status_summary</td><td>Status for DAS (See header file “compdef.h” #15 for SM8200)</td></tr>
<tr><td>extended_summary</td><td>Status for DAS (See header file “compdef.h” #16 for SM8200)</td></tr>
<tr><td>conc1</td><td>Concentration number 1 (NO for SM8200, instantaneous O_2 for oxygen)</td></tr>
<tr><td>conc2</td><td>Concentration number 2 (SO2 for SM8200, average O_2 for oxygen)</td></tr>
</table></td></tr>
</table>

	conc3	Concentration number 3 (NO2 for SM8200, stack temperature for oxygen)
	conc4	Concentration number 4 (NH3 for SM8200, stack pressure for oxygen)
	year	Time at sample
	month	Time at sample
	day	Time at sample
	hour	Time at sample
	minute	Time at sample
	sec	Time at sample

Description:
These variables are used to read buffered data SM8200 (BufferSM8200), external oxygen instrument (BufferO), and auxiliary Plant I/O board (BufferX).
The buffer is circular and can store up to seven days of data depending on the integration period of the SM8200. Oldest data is overwritten by the newest and the index number rolls back to zero once it passes the maximum size.
There are several ways of reading the buffer (as per header file “compdef.h” #7):

- To read a particular index, set the output variable (nvo) *index* field to desired, and set the *control* field to READ. The input variable (nvi) will return with WRITE in *control*, the requested in *index* and the values if the requested index exists. If not, a NACK is returned in *control* and the requested index in *index*.
- To read the oldest value from the buffer, set the output variable (nvo) set the *control* field to READ_FRONT. The input variable (nvi) will return with WRITE_FRONT in *control*, the oldest index in *index* and the values if any data exists. If not, a NACK is returned in *control*.
- To read the newest value from the buffer, set the output variable (nvo) *control* field to READ_REAR. The input variable (nvi) will return with WRITE_REAR in *control*, the newest index in *index* and the values if any data exists. If not, a NACK is returned in *control*.
- To read the current size of the buffer, set the output

<table>
<tr><td></td><td>variable (nvo) control field to READ_SIZE. The input variable (nvi) will return with WRITE_SIZE in control and the current size in index if any data exists. If not, a NACK is returned in control.
• To read the maximum available size of the buffer, set the output variable (nvo) control field to READ_MAX_SIZE. The input variable (nvi) will return with WRITE_MAX_SIZE in control and the maximum size in index.
• To clear the buffer and start anew, set the output variable (nvo) control field to EMPTY_QUEUE.

Note that when the buffer starts from empty, the first recorded value goes into index 1 not index 0, although index 0 may be available later when the buffer fills. This is due to the circular implementation of the buffering which requires the pointer to the front of the buffer to be an unused field.</td></tr>
<tr><td rowspan="3">nviTempPressSM8200
nviTempPress_RESERVED</td><td>Type: UNVT_EtherCfg</td></tr>
<tr><td>Members and Units:

<table>
<tr><td>detect_temp</td><td>Detector temperature in SM8200</td></tr>
<tr><td>bench_temp</td><td>Optical bench temperature in SM8200</td></tr>
<tr><td>calgas_temp</td><td>Calibration gas temperature in SM8200</td></tr>
<tr><td>probe_temp</td><td>Probe temperature in SM8200</td></tr>
<tr><td>stack_temp</td><td>Stack temperature in SM8200</td></tr>
<tr><td>stack_press</td><td>Stack pressure in SM8200</td></tr>
<tr><td>year</td><td>Time at sample</td></tr>
<tr><td>month</td><td>Time at sample</td></tr>
<tr><td>day</td><td>Time at sample</td></tr>
<tr><td>hour</td><td>Time at sample</td></tr>
<tr><td>minute</td><td>Time at sample</td></tr>
<tr><td>sec</td><td>Time at sample</td></tr>
</table></td></tr>
<tr><td>Description:
These variables contain temperature and pressure data.</td></tr>
<tr><td rowspan="2">nviEtherCfg
nvoEtherCfg</td><td>Type: UNVT_EtherCfg</td></tr>
<tr><td>Members and Units:

<table>
<tr><td>control</td><td>This element is used for</td></tr>
</table></td></tr>
</table>

		transmission control, thus implementing the read/write/acknowledgment model. (See header file “compdef.h” #7)
	DHCP_enable	Enable DHCP
	IP[4]	Internet Protocol address of the Ethernet module.
	gateway[4]	Network gateway address.
	mask[4]	Network address mask.
	MAC[6]	Unique MAC address of the Ethernet module.
	major_ver_eth	Version of Ethernet ARM Processor
	minor_ver_eth	Version of Ethernet ARM Processor
	major_ver_REM	ERP version
	minor_ver_REM	ERP version
	major_ver_NEU	Version of network software
	minor_ver_NEU	Version of network software
	Description: These variables contain the Ethernet setup and they cannot be changed via the MODBUS TCP/IP interface. They can only be changed via the ERP or web browser screens.	
nviAlarmStateSM8200 **nviAlarmState_RESERVED**	Type: UNVT_AlarmState	
	Members and Units:	
	conc_alarms_low	Concentration alarms triggered when value under low tolerance (See header file “compdef.h” #26 for SM8200)
	conc_alarms_high	Concentration alarms triggered when value above high tolerance (See header file “compdef.h” #26 for SM8200)
	MBTU_alarms_low	MBTU concentration alarms triggered when value under low tolerance (See header file “compdef.h” #26 for SM8200)
	MBTU_alarms_high	MBTU concentration alarms triggered when value above high tolerance (See header file “compdef.h” #26 for SM8200)

	O2_alarms_low	Oxygen concentration alarms triggered when value under low tolerance (See header file “compdef.h” #26 for SM8200)
	O2_alarms_high	Oxygen concentration alarms triggered when value above high tolerance (See header file “compdef.h” #26 for SM8200)
	AUX_alarms_low	Auxiliary Plant I/O board data alarms triggered when value under low tolerance (See header file “compdef.h” #44 for SM8200)
	AUX_alarms_high	Auxiliary Plant I/O board data alarms triggered when value above high tolerance (See header file “compdef.h” #44 for SM8200)
	reserved1	Reserved for future use
	reserved2	Reserved for future use
	Description: These variables contain the alarm status of the particular instrument.	

Header File Definitions for the SM8200

```
/**** ************************** ************************** ****/
/****  Filename: COMPDEF.H                                  ****/
/****  Author: IVICA EFTIMOVSKI                             ****/
/**** ************************** ************************** ****/
/**** Date of this file: MAR-24-2011                        ****/
/**** ************************** ************************** ****/
/**** NET Definitions for the SM8200 Analyzer               ****/
/**** Teledyne Monitor Labs Inc.                            ****/
/**** Copyright 2004  /  All rights reserved                ****/
/**** ************************** ************************** ****/
/**** THIS FILE IS CONTROLLED BY THE AUTHOR (DMB/IE)        ****/
/**** DO NOT EDIT WITHOUT APPROVAL FROM THE ENGINEERING MANAGER.****/
/**** ************************** ************************** ****/

#ifndef   COMPOSITE_DEFINITIONS_HEADER
#define   COMPOSITE_DEFINITIONS_HEADER

/**** ************************** ************************** ****/
/* The following condition is used to detect if C++ compiler is used.*/
/* If so, avoid mangling the function names */
#ifdef __cplusplus
extern "C" {
#endif
/**** ************************** ************************** ****/
/* Variable types by platform */
#ifdef INTEL386
     typedef unsigned char         BYTE;
     typedef unsigned short        WORD;
     typedef float                 FLOAT;
     typedef char                  IBYTE;
     typedef short                 IWORD;
     typedef unsigned long         DWORD;
     typedef long                  TIME_T;
#endif

/*  Some useful macros                                      */

#ifndef _MSC_VER
#define   LOBYTE(w)              ((BYTE)(w))
#define   HIBYTE(w)              ((BYTE)(((WORD)(w) >> 8) & 0xFF))
#endif

#define   LO4BITS(b)             ((b) & 0x0F)
#define   HI4BITS(b)             ((b) >> 4)

#define   BITS2BYTE(hi,lo)       (((BYTE)(hi<<4)) | (lo & 0x0F))
#define   BYTE2WORD(hi,lo)       (((WORD)(hi<<8)) | (lo & 0xFF))
/**** ************************** ************************** ****/
/*  #1 = Diluent Types                                       */
#define   O2                                    (BYTE)    1
#define   CO2                                   (BYTE)    2
#define   NO_DILUENT                            (BYTE)    3
```

```
/**** ***************************** ***************************** ****/
/*  #2 = Gas Types                                                  */
#define   NO                                    (BYTE)    1
#define   SO2                                   (BYTE)    2
#define   NO2                                   (BYTE)    3
#define   NH3                                   (BYTE)    4
/**** ***************************** ***************************** ****/
/*  #3 = Index Poly/Lut Defs                                        */
#define   POLY                                  (BYTE)    0x01
#define   LUT                                   (BYTE)    0x02
/**** ***************************** ***************************** ****/
/*  #4 = Index Enable/Disable Defs                                  */
#define   DISABLE                               (BYTE)    0x01
#define   ENABLE                                (BYTE)    0x02
#define   DEFAULT                               (BYTE)    0x03
/**** ***************************** ***************************** ****/
/*  #5 = Units of Measurement                                       */
#define   ENGLISH                               (BYTE)    0x01
#define   METRIC_WGT                            (BYTE)    0x02
#define   METRIC_VOL                            (BYTE)    0x03
/**** ***************************** ***************************** ****/
/*  #6 = Index Shutter/Solenoid Defs                                */
#define   DEENERGIZE                            (BYTE)    0x01
#define   ENERGIZE                              (BYTE)    0x02

/**** ***************************** ***************************** ****/
/*  #6a = Index Shutter/Solenoid Defs                                */
#define   INDIVID                               (BYTE)    0x1
#define   DIFF                                  (BYTE)    0x2

/**** ***************************** ***************************** ****/
/*  #6b = Index Shutter/Solenoid Defs                                */
#define   CURVE_A                               (BYTE)    0x1
#define   CURVE_B                               (BYTE)    0x2

/**** ***************************** ***************************** ****/
/*  #6c = Index Shutter/Solenoid Defs                                */
#define   BOTTLE_GAS                            (BYTE)    0x1
#define   EO_CALSPAN                            (BYTE)    0x2

/**** ***************************** ***************************** ****/
/*  #7 = Index Command Defs                                         */
#define   READ                                  (BYTE)    0x01
#define   WRITE                                 (BYTE)    0x02
#define   ACK                                   (BYTE)    0x03
#define   NACK                                  (BYTE)    0x04
#define   LOOP_BACK                             (BYTE)    0x05
#define   READ_FRONT                            (BYTE)    0x06
#define   READ_REAR                             (BYTE)    0x07
#define   READ_SIZE                             (BYTE)    0x08
#define   WRITE_FRONT                           (BYTE)    0x09
#define   WRITE_REAR                            (BYTE)    0x0a
#define   WRITE_SIZE                            (BYTE)    0x0b
#define   READ_MAX_SIZE                         (BYTE)    0x0c
#define   WRITE_MAX_SIZE                        (BYTE)    0x0d
#define   EMPTY_QUEUE                           (BYTE)    0x0e
/**** ***************************** ***************************** ****/
```

```
/*  #8 = FACTORY RESERVED                                              */

/**** ************************** ************************** ****/
/*  #9 = Array Element Type, use with UNVT_Array and UNVT_AUXData  */
#define   WORD_TYPE                              (BYTE)    0x01
#define   FLOAT_TYPE                             (BYTE)    0x02
// VERSION_TYPE is used for the Plant-IO Board to output software ver.
#define   VERSION_TYPE                           (BYTE)    0x03
/**** ************************** ************************** ****/
/* #10 = DHCP settings                                               */
#define   WITH_DHCP                              (BYTE)    0x01
#define   STATIC_IP                              (BYTE)    0x02
/**** ************************** ************************** ****/
/* #11 = CGA Selection                                               */
#define   DAILY_CGA_MID                          (BYTE)    0x01
#define   DAILY_CGA_HIGH                         (BYTE)    0x02
/**** ************************** ************************** ****/
/* #12 = Communication status with all (.com_flag) elements          */
#define   COMM_NOT_PRESENT                       (BYTE)    0x01
#define   COMM_NORMAL                            (BYTE)    0x02
#define   COMM_LOST                              (BYTE)    0x03
/* ************************** ************************** ********** */
/* #13 = SM8200 Primary Status Definitions                           */

#define   TRAINING_SET_CRC_FAIL                  (WORD)    0x8000
#define   WAVELENGTH_CHECK_FAILURE               (WORD)    0x4000
#define   GAS_4_SPAN_FAIL                        (WORD)    0x2000
#define   GAS_4_ZERO_FAIL                        (WORD)    0x1000
#define   REFERENCE_FAULT                        (WORD)    0x0800
#define   GAS_3_SPAN_FAIL                        (WORD)    0x0400
#define   GAS_3_ZERO_FAIL                        (WORD)    0x0200
#define   HIGH_DARK_CURRENT_FAULT                (WORD)    0x0100
#define   GAS_2_SPAN_FAIL                        (WORD)    0x0080
#define   GAS_2_ZERO_FAIL                        (WORD)    0x0040
#define   LOW_DARK_CURRENT_FAULT                 (WORD)    0x0020
#define   GAS_1_SPAN_FAIL                        (WORD)    0x0010
#define   GAS_1_ZERO_FAIL                        (WORD)    0x0008
#define   BENCH_TEMPERATURE_FAULT                (WORD)    0x0004
#define   PDA_TEMPERATURE_FAULT                  (WORD)    0x0002
#define   PDA_THERMISTOR_FAULT                   (WORD)    0x0001
/**** ************************** ************************** ****/
/* #14 = SM8200 Extended Status Definitions                          */

#define   PROBE_TEMPERATURE_FAULT                (WORD)    0x8000
#define   CAL_GAS_TEMPERATURE_FAULT              (WORD)    0x4000
#define   GAS_4_TOTAL_ADJUST_FAULT               (WORD)    0x2000
#define   GAS_4_INCREMENTAL_ADJUST_FAULT         (WORD)    0x1000
#define   STACK_PRESSURE_FAULT                   (WORD)    0x0800
#define   GAS_3_TOTAL_ADJUST_FAULT               (WORD)    0x0400
#define   GAS_3_INCREMENTAL_ADJUST_FAULT         (WORD)    0x0200
#define   STACK_TEMPERATURE_FAULT                (WORD)    0x0100
#define   GAS_2_TOTAL_ADJUST_FAULT               (WORD)    0x0080
#define   GAS_2_INCREMENTAL_ADJUST_FAULT         (WORD)    0x0040
#define   O2_ANALYZER_FAULT                      (WORD)    0x0020
#define   GAS_1_TOTAL_ADJUST_FAULT               (WORD)    0x0010
#define   GAS_1_INCREMENTAL_ADJUST_FAULT         (WORD)    0x0008
#define   X_3_FPGA_TIMEOUT                       (WORD)    0x0004
```

```
#define   FPGA_TIMEOUT                          (WORD)    0x0002
#define   OUT_OF_SERVICE                        (WORD)    0x0001
/* ***************************** ****************************** */
/* #15 = SM8200 Status Summary Definitions                      */
/* These definitions are use for the DAS Status Summary word     */

#define   STATUS_SUM_AVE_ZERO_VALUES            (WORD)    0x0001
#define   STATUS_SUM_AVE_SPAN_VALUES            (WORD)    0x0002
#define   STATUS_SUM_STACK_MEASURE_VALID        (WORD)    0x0004
#define   STATUS_SUM_AVE_MID_VALUES             (WORD)    0x0008

#define   STATUS_SUM_AVE_COMM_FAULT             (WORD)    0x0010
#define   STATUS_SUM_AVE_UNIT_DOWN              (WORD)    0x0020
#define   STATUS_SUM_AVE_OUT_OF_SERVICE         (WORD)    0x0040
#define   STATUS_SUM_AVE_ANALYZER_FAULT         (WORD)    0x0080

#define   STATUS_SUM_AVE_NON_FATAL_FAULT        (WORD)    0x0100
#define   STATUS_SUM_AVE_FATAL_FAULT            (WORD)    0x0200
#define   STATUS_SUM_AVE_IN_CALIBRATION         (WORD)    0x0400
#define   STATUS_SUM_AVE_IN_LINEARITY           (WORD)    0x0800

#define   STATUS_SUM_AVE_OVER_RANGE             (WORD)    0x1000
#define   STATUS_SUM_AVE_INTERFERENCE_FAIL      (WORD)    0x2000
#define   STATUS_SUM_AVE_INTERFERENCE_CHECK     (WORD)    0x4000
/*#define   STATUS_SUM_AVE_SPARE                  (WORD)    0x8000*/
/**** ***************************** ***************************** ****/
/* #16 = SM8200 Extended Status Summary Definitions             */
/* These definitions are use for the DAS Status Summary word     */

/* ***************************** ****************************** */
/* #17 = SM8200 Commands                                        */

#define   UNKNOWN_CMD                           (BYTE)    0
#define   NORMAL_CMD                            (BYTE)    1
#define   ZERO_CMD                              (BYTE)    2
#define   INITIAL_ZERO_CMD                      (BYTE)    3
#define   FINAL_ZERO_CMD                        (BYTE)    4
#define   SPAN_CMD                              (BYTE)    5
#define   EO_CAL_CMD                            (BYTE)    6
#define   CAL_CYCLE_CMD                         (BYTE)    7
#define   AUDIT_CMD                             (BYTE)    8
#define   OUT_OF_SERVICE_CMD                    (BYTE)    9
#define   PUT_IN_SERVICE_CMD                    (BYTE)    10
#define   DARK_CMD                              (BYTE)    11
#define   WAVELENGTH_CHECK_CMD                  (BYTE)    12
#define   CGA_LOW1_CMD                          (BYTE)    13
#define   CGA_LOW2_CMD                          (BYTE)    14
#define   CGA_LOW3_CMD                          (BYTE)    15
#define   CGA_LOW4_CMD                          (BYTE)    16
#define   CGA_LOW5_CMD                          (BYTE)    17
#define   CGA_LOW6_CMD                          (BYTE)    18
#define   CGA_MID1_CMD                          (BYTE)    19
#define   CGA_MID2_CMD                          (BYTE)    20
#define   CGA_MID3_CMD                          (BYTE)    21
#define   CGA_MID4_CMD                          (BYTE)    22
#define   CGA_MID5_CMD                          (BYTE)    23
#define   CGA_MID6_CMD                          (BYTE)    24
```

```
#define   CGA_HIGH1_CMD                         (BYTE)    25
#define   CGA_HIGH2_CMD                         (BYTE)    26
#define   CGA_HIGH3_CMD                         (BYTE)    27
#define   CGA_HIGH4_CMD                         (BYTE)    28
#define   CGA_HIGH5_CMD                         (BYTE)    29
#define   CGA_HIGH6_CMD                         (BYTE)    30
#define   RESET_CMD                             (BYTE)    31
#define   RECONFIG_CMD                          (BYTE)    32
#define   PAR_CRC_CHECK_CMD                     (BYTE)    33
#define   DIAGNOSTIC_CMD                        (BYTE)    34
#define   TRAIN_DARK_CMD                        (BYTE)    35
#define   TRAIN_CLEAR_CMD                       (BYTE)    36
#define   TRAIN_SPAN_CMD                        (BYTE)    37
#define   TRAIN_EO_CAL_CMD                      (BYTE)    38
#define   TRAIN_NORMAL_CMD                      (BYTE)    39
#define   UV_LAMP_FILAMENT_ON_CMD               (BYTE)    40
#define   UV_LAMP_FILAMENT_OFF_CMD              (BYTE)    41

/**** *************************** **************************** ****/
/* #18 = SM8200 Modes                                           */

#define   UNKNOWN                               (BYTE)    0
#define   DARK_CURRENT                          (BYTE)    1
#define   ZERO_ACQUIRE                          (BYTE)    2
#define   ZERO                                  (BYTE)    3
#define   WAVELENGTH_CHECK                      (BYTE)    4
#define   SPAN_ACQUIRE                          (BYTE)    5
#define   SPAN                                  (BYTE)    6
#define   NORMAL_ACQUIRE                        (BYTE)    7
#define   NORMAL                                (BYTE)    8
#define   EO_CAL_ACQUIRE                        (BYTE)    9
#define   EO_CAL                                (BYTE)    10
#define   MANUAL_DARK                           (BYTE)    11
#define   MANUAL_ZERO                           (BYTE)    12
#define   INITIAL_ZERO                          (BYTE)    13
#define   FINAL_ZERO                            (BYTE)    14
#define   MANUAL_WAVELENGTH_CHECK               (BYTE)    15
#define   MANUAL_SPAN                           (BYTE)    16
#define   MANUAL_EO_CAL                         (BYTE)    17
#define   CGA_LOW_ACQUIRE                       (BYTE)    18
#define   CGA_LOW1                              (BYTE)    19
#define   CGA_LOW2                              (BYTE)    20
#define   CGA_LOW3                              (BYTE)    21
#define   CGA_LOW4                              (BYTE)    22
#define   CGA_LOW5                              (BYTE)    23
#define   CGA_LOW6                              (BYTE)    24
#define   CGA_MID_ACQUIRE                       (BYTE)    25
#define   CGA_MID1                              (BYTE)    26
#define   CGA_MID2                              (BYTE)    27
#define   CGA_MID3                              (BYTE)    28
#define   CGA_MID4                              (BYTE)    29
#define   CGA_MID5                              (BYTE)    30
#define   CGA_MID6                              (BYTE)    31
#define   CGA_HIGH_ACQUIRE                      (BYTE)    32
#define   CGA_HIGH1                             (BYTE)    33
#define   CGA_HIGH2                             (BYTE)    34
#define   CGA_HIGH3                             (BYTE)    35
```

```
#define   CGA_HIGH4                              (BYTE)    36
#define   CGA_HIGH5                              (BYTE)    37
#define   CGA_HIGH6                              (BYTE)    38
#define   TRAIN_DARK                             (BYTE)    39
#define   TRAIN_CLEAR                            (BYTE)    40
#define   TRAIN_SPAN                             (BYTE)    41
#define   TRAIN_EO_CAL                           (BYTE)    42
#define   TRAIN_NORMAL                           (BYTE)    43
#define   TRAIN_DARK_ACQUIRE                     (BYTE)    44
#define   TRAIN_CLEAR_ACQUIRE                    (BYTE)    45
#define   TRAIN_SPAN_ACQUIRE                     (BYTE)    46
#define   TRAIN_EO_CAL_ACQUIRE                   (BYTE)    47
#define   TRAIN_NORMAL_ACQUIRE                   (BYTE)    48
#define   LAMP_ALIGNMENT                         (BYTE)    49
#define   UV_LAMP_FILAMENT_OFF                   (BYTE)    50
/* ****************************** ****************************** */
/* #19 = FACTORY RESERVED                                        */

/**** ************************** ************************** ****/
/* #20 = FACTORY RESERVED                                        */

/* ****************************** ****************************** */
/* #21 = FACTORY RESERVED                                        */

/**** ************************** ************************** ****/
/* #22 = FACTORY RESERVED                                        */

/* ****************************** ****************************** */
/* #23 = FACTORY RESERVED                                        */

/**** ************************** ************************** ****/
/* #24 = FACTORY RESERVED                                        */

/* ****************************** ****************************** */
/* #25 = FACTORY RESERVED                                        */

/* ****************************** ****************************** */
/* #26 = SM8200 Alarm Status Definitions, use with UNVT_AlarmState */
/*                                                               */
/* Section modified to make sure that redundant definitons do not  */
/* cause problems. These values are also referenced in sections    */
/* #41, #42 and #43.                                             */

#define   ALARM_INST_CONC_GAS1_LOW               (WORD)    0x0001
#define   ALARM_INST_CONC_GAS2_LOW               (WORD)    0x0002
#define   ALARM_INST_CONC_GAS3_LOW               (WORD)    0x0004
#define   ALARM_INST_CONC_GAS4_LOW               (WORD)    0x0008

#define   ALARM_AVG_CONC_GAS1_LOW                (WORD)    0x0010
#define   ALARM_AVG_CONC_GAS2_LOW                (WORD)    0x0020
#define   ALARM_AVG_CONC_GAS3_LOW                (WORD)    0x0040
#define   ALARM_AVG_CONC_GAS4_LOW                (WORD)    0x0080

#define   ALARM_INST_MBTU_GAS1_LOW               (WORD)    0x0001
#define   ALARM_INST_MBTU_GAS2_LOW               (WORD)    0x0002
#define   ALARM_INST_MBTU_GAS3_LOW               (WORD)    0x0004
#define   ALARM_INST_MBTU_GAS4_LOW               (WORD)    0x0008
```

```
#define   ALARM_AVG_MBTU_GAS1_LOW                (WORD)    0x0010
#define   ALARM_AVG_MBTU_GAS2_LOW                (WORD)    0x0020
#define   ALARM_AVG_MBTU_GAS3_LOW                (WORD)    0x0040
#define   ALARM_AVG_MBTU_GAS4_LOW                (WORD)    0x0080

#define   ALARM_INST_CONC_GAS1_HIGH              (WORD)
ALARM_INST_CONC_GAS1_LOW
#define   ALARM_INST_CONC_GAS2_HIGH              (WORD)
ALARM_INST_CONC_GAS2_LOW
#define   ALARM_INST_CONC_GAS3_HIGH              (WORD)
ALARM_INST_CONC_GAS3_LOW
#define   ALARM_INST_CONC_GAS4_HIGH              (WORD)
ALARM_INST_CONC_GAS4_LOW

#define   ALARM_AVG_CONC_GAS1_HIGH               (WORD)
ALARM_AVG_CONC_GAS1_LOW
#define   ALARM_AVG_CONC_GAS2_HIGH               (WORD)
ALARM_AVG_CONC_GAS2_LOW
#define   ALARM_AVG_CONC_GAS3_HIGH               (WORD)
ALARM_AVG_CONC_GAS3_LOW
#define   ALARM_AVG_CONC_GAS4_HIGH               (WORD)
ALARM_AVG_CONC_GAS4_LOW

#define   ALARM_INST_MBTU_GAS1_HIGH              (WORD)
ALARM_INST_MBTU_GAS1_LOW
#define   ALARM_INST_MBTU_GAS2_HIGH              (WORD)
ALARM_INST_MBTU_GAS2_LOW
#define   ALARM_INST_MBTU_GAS3_HIGH              (WORD)
ALARM_INST_MBTU_GAS3_LOW
#define   ALARM_INST_MBTU_GAS4_HIGH              (WORD)
ALARM_INST_MBTU_GAS4_LOW

#define   ALARM_AVG_MBTU_GAS1_HIGH               (WORD)
ALARM_AVG_MBTU_GAS1_LOW
#define   ALARM_AVG_MBTU_GAS2_HIGH               (WORD)
ALARM_AVG_MBTU_GAS2_LOW
#define   ALARM_AVG_MBTU_GAS3_HIGH               (WORD)
ALARM_AVG_MBTU_GAS3_LOW
#define   ALARM_AVG_MBTU_GAS4_HIGH               (WORD)
ALARM_AVG_MBTU_GAS4_LOW

#define   ALARM_INST_CONC_O2                     (WORD)    0x0001
#define   ALARM_AVG_CONC_O2                      (WORD)    0x0002

/* ****************************** ****************************** */
/* #27 = FACTORY RESERVED                                        */

/* #28 = MIO COMMANDS                                            */

#define    MIO_ONLINE_CMD                        (BYTE)    1
#define    MIO_ZERO_SCALE_CMD                    (BYTE)    2
#define    MIO_MID_SCALE_CMD                     (BYTE)    3
#define    MIO_FULL_SCALE_CMD                    (BYTE)    4
#define    MIO_SEND_CONFIG_CMD                   (BYTE)    5

/* ****************************** ****************************** */
```

```
/* #29 = MIO Selection                                                  */

#define   MIO1                                  (BYTE)    0x01
#define   MIO2                                  (BYTE)    0x02

/* ****************************** ****************************** */
/* #30 = MIO instrument selection                                     */

#define    _NO_SELECTION                        (BYTE)    0
#define    _SM8200_INST                         (BYTE)    1
#define    _FACTORY_RESERVED_INST               (BYTE)    2

/* ****************************** ****************************** */
/* #31 = MIO ANALOG output_type_A or B when instrument_A = SM8200 */

// #define    _NO_SELECTION                     (WORD)    0
#define    _INST_NO                             (WORD)    1
#define    _AVE_NO                              (WORD)    2
#define    _INST_NO_MBTU                        (WORD)    3
#define    _AVE_NO_MBTU                         (WORD)    4
#define    _NO_CAL_ZERO                         (WORD)    5
#define    _NO_CAL_ZERO_MBTU                    (WORD)    6
#define    _NO_CAL_SPAN                         (WORD)    7
#define    _NO_CAL_SPAN_MBTU                    (WORD)    8
#define    _INST_SO2                            (WORD)    9
#define    _AVE_SO2                             (WORD)    10
#define    _INST_SO2_MBTU                       (WORD)    11
#define    _AVE_SO2_MBTU                        (WORD)    12
#define    _SO2_CAL_ZERO                        (WORD)    13
#define    _SO2_CAL_ZERO_MBTU                   (WORD)    14
#define    _SO2_CAL_SPAN                        (WORD)    15
#define    _SO2_CAL_SPAN_MBTU                   (WORD)    16
#define    _INST_NO2                            (WORD)    17
#define    _AVE_NO2                             (WORD)    18
#define    _INST_NO2_MBTU                       (WORD)    19
#define    _AVE_NO2_MBTU                        (WORD)    20
#define    _NO2_CAL_ZERO                        (WORD)    21
#define    _NO2_CAL_ZERO_MBTU                   (WORD)    22
#define    _NO2_CAL_SPAN                        (WORD)    23
#define    _NO2_CAL_SPAN_MBTU                   (WORD)    24
#define    _INST_NH3                            (WORD)    25
#define    _AVE_NH3                             (WORD)    26
#define    _INST_NH3_MBTU                       (WORD)    27
#define    _AVE_NH3_MBTU                        (WORD)    28
#define    _NH3_CAL_ZERO                        (WORD)    29
#define    _NH3_CAL_ZERO_MBTU                   (WORD)    30
#define    _NH3_CAL_SPAN                        (WORD)    31
#define    _NH3_CAL_SPAN_MBTU                   (WORD)    32
#define    _INST_O2_actual                      (WORD)    33
#define    _AVE_O2_actual                       (WORD)    34
#define    _O2_CAL_ZERO                         (WORD)    35
#define    _O2_CAL_SPAN                         (WORD)    36
#define    _PDA_TEMP                            (WORD)    37
#define    _BENCH_TEMP                          (WORD)    38
#define    _CAL_GAS_TEMP                        (WORD)    39
#define    _PROBE_TEMP                          (WORD)    40
#define    _STACK_TEMP                          (WORD)    41
```

```
#define    _STACK_PRESSURE                          (WORD)    42

#define    _MAX_SM8200_ANALOG                      (WORD)    42

/* ****************************** ****************************** */
/* #32 = FACTORY RESERVED                                        */

/* ****************************** ****************************** */
/* #33 = MIO ANALOG CalEnable_A or CalEnable_B                   */

// #define    _NO_SELECTION                         (BYTE)    0
#define    _WITHOUT_CAL                             (BYTE)    1
#define    _WITH_CAL                                (BYTE)    2

/* ****************************** ****************************** */
/* #34 = MIO RELAY STRUCTURE ELEMENTS for INSTR_SEL[8]           */

// #define _NO_SELECTION                             (BYTE)    0
#define    _SM8200_ALARM                            (BYTE)    1
#define    _SM8200_FAULT                            (BYTE)    2
#define    _SM8200_MODE                             (BYTE)    3
/*FACTORY RESERVED                                (BYTE)    4*/
/*FACTORY RESERVED                                (BYTE)    5*/
/*FACTORY RESERVED                                 (BYTE)    6*/
#define    _O2_RELAYS                               (BYTE)    7


#define    _MAX_RELAY_INSTRUMENT                    (BYTE)    7
/* ****************************** ****************************** */
/* #35 = MIO RELAY_SEL[x] when INSTR_SEL[x] = _SM8200_ALARM      */

// #define _NO_SELECTION                             (BYTE)    0
#define    _INST_NO_ALARM                           (BYTE)    1
#define    _AVE_NO_ALARM                            (BYTE)    2
#define    _INST_MBTU_NO_ALARM                      (BYTE)    3
#define    _AVE_MBTU_NO_ALARM                       (BYTE)    4
#define    _INST_SO2_ALARM                          (BYTE)    5
#define    _AVE_SO2_ALARM                           (BYTE)    6
#define    _INST_MBTU_SO2_ALARM                     (BYTE)    7
#define    _AVE_MBTU_SO2_ALARM                      (BYTE)    8
#define    _INST_NO2_ALARM                          (BYTE)    9
#define    _AVE_NO2_ALARM                           (BYTE)    10
#define    _INST_MBTU_NO2_ALARM                     (BYTE)    11
#define    _AVE_MBTU_NO2_ALARM                      (BYTE)    12
#define    _INST_NH3_ALARM                          (BYTE)    13
#define    _AVE_NH3_ALARM                           (BYTE)    14
#define    _INST_MBTU_NH3_ALARM                     (BYTE)    15
#define    _AVE_MBTU_NH3_ALARM                      (BYTE)    16
#define    _INST_O2_ACTUAL_ALARM                    (BYTE)    17
#define    _AVE_O2_ACTUAL_ALARM                     (BYTE)    18

#define    _MAX_SM8200_ALARM                        (BYTE)    18
/* ****************************** ****************************** */
/* #36 = MIO RELAY_SEL[x] when INSTR_SEL[x] = _SM8200_FAULT      */

// #define _NO_SELECTION                 (BYTE)    0
#define    _FATAL_FAULT                  (BYTE)    1  /* FAIL-SAFE */
```

```
#define    _NON_FATAL_FAULT              (BYTE)    2  /* FAIL-SAFE */
#define    _ANALYZER_FAULT               (BYTE)    3  /* FAIL-SAFE */
#define    _DATA_VALID                   (BYTE)    4  /* FAIL-SAFE */
#define    _CAL_FAIL_FAULT               (BYTE)    5
#define    _STACK_TEMP_FAULT             (BYTE)    6
#define    _STACK_PRESSURE_FAULT         (BYTE)    7
#define    _PDA_TEMP_FAULT               (BYTE)    8
#define    _BENCH_TEMP_FAULT             (BYTE)    9
#define    _CAL_GAS_TEMP_FAULT           (BYTE)    10
#define    _PROBE_TEMP_FAULT             (BYTE)    11
#define    _OUT_OF_SERVICE               (BYTE)    12

#define    _MAX_SM8200_FAULT           (BYTE)    12
/* ****************************** ****************************** */
/* #37 = MIO RELAY_SEL[x] when INSTR_SEL[x] = _SM8200_MODE        */

// #define _NO_SELECTION                          (BYTE)    0
#define    _NORM_OR_NORM_ACQ                     (BYTE)    1
#define    _NORMAL_MODE                          (BYTE)    2
#define    _CAL_OR_CAL_ACQ                       (BYTE)    3
#define    _CAL_ON_AOUT                          (BYTE)    4
#define    _ZERO_OR_ZERO_ACQ                     (BYTE)    5
#define    _ZERO_MODE                            (BYTE)    6
#define    _SPAN_OR_SPAN_ACQ                     (BYTE)    7
#define    _SPAN_MODE                            (BYTE)    8
#define    _CGA_MODE_OR_ACQ                      (BYTE)    9
#define    _CGA_MODE                             (BYTE)    10
#define    _CGA_LOW_MODE_OR_ACQ                  (BYTE)    11
#define    _CGA_LOW_MODE                         (BYTE)    12
#define    _CGA_MID_MODE_OR_ACQ                  (BYTE)    13
#define    _CGA_MID_MODE                         (BYTE)    14
#define    _CGA_HIGH_MODE_OR_ACQ                 (BYTE)    15
#define    _CGA_HIGH_MODE                        (BYTE)    16

#define    _MAX_SM8200_MODE                    (BYTE)    16
/* ****************************** ****************************** */
/* #38 = FACTORY RESERVED                                         */

/* ****************************** ****************************** */
/* #39 = FACTORY RESERVED                                         */

/* ****************************** ****************************** */
/* #40 = FACTORY RESERVED                                         */

/**** ************************** ************************** ****/

/* #41 = SM8200 Alarm (concentrations alarms)                    */

/*                                                                */
/* The following definitions are redundant with #26. We keep them  */
/* here for compatibillity issues.                                */

#define     SM8200_INST_CONC_GAS_1_ALARM        (WORD)
ALARM_INST_CONC_GAS1_LOW
#define     SM8200_INST_CONC_GAS_2_ALARM        (WORD)
ALARM_INST_CONC_GAS2_LOW
```

```
#define     SM8200_INST_CONC_GAS_3_ALARM        (WORD)
ALARM_INST_CONC_GAS3_LOW
#define     SM8200_INST_CONC_GAS_4_ALARM        (WORD)
ALARM_INST_CONC_GAS4_LOW
#define     SM8200_AVG_CONC_GAS_1_ALARM         (WORD)
ALARM_AVG_CONC_GAS1_LOW
#define     SM8200_AVG_CONC_GAS_2_ALARM         (WORD)
ALARM_AVG_CONC_GAS2_LOW
#define     SM8200_AVG_CONC_GAS_3_ALARM         (WORD)
ALARM_AVG_CONC_GAS3_LOW
#define     SM8200_AVG_CONC_GAS_4_ALARM         (WORD)
ALARM_AVG_CONC_GAS4_LOW

/**** ************************** ************************** ****/

/* #42 = SM8200 Alarm (MBTU concentrations alarms)                */

/*                                                                 */
/* The following definitions are redundant with #26. We keep them  */
/* here for compatibillity issues.                                 */

#define     SM8200_INST_MBTU_GAS_1_ALARM        (WORD)
ALARM_INST_MBTU_GAS1_LOW
#define     SM8200_INST_MBTU_GAS_2_ALARM        (WORD)
ALARM_INST_MBTU_GAS2_LOW
#define     SM8200_INST_MBTU_GAS_3_ALARM        (WORD)
ALARM_INST_MBTU_GAS3_LOW
#define     SM8200_INST_MBTU_GAS_4_ALARM        (WORD)
ALARM_INST_MBTU_GAS4_LOW
#define     SM8200_AVG_MBTU_GAS_1_ALARM         (WORD)
ALARM_AVG_MBTU_GAS1_LOW
#define     SM8200_AVG_MBTU_GAS_2_ALARM         (WORD)
ALARM_AVG_MBTU_GAS2_LOW
#define     SM8200_AVG_MBTU_GAS_3_ALARM         (WORD)
ALARM_AVG_MBTU_GAS3_LOW
#define     SM8200_AVG_MBTU_GAS_4_ALARM         (WORD)
ALARM_AVG_MBTU_GAS4_LOW

/**** ************************** ************************** ****/

/* #43 = Oxygen Alarms (O2 alarms)                                 */

/*                                                                 */
/* The following definitions are redundant with #26. We keep them  */
/* here for compatibillity issues.                                 */

#define     INST_OXYGEN_ALARM                   (WORD)    ALARM_INST_CONC_O2
#define     AVG_OXYGEN_ALARM                    (WORD)    ALARM_AVG_CONC_O2

/**** ************************** ************************** ****/

/* #44 = Plant I/O Board Alarms                                    */


#define     PLANTIO_AVG_A2D_INPUT_1_ALARM       (WORD)    0x0001
#define     PLANTIO_AVG_A2D_INPUT_2_ALARM       (WORD)    0x0002
```

```
/* ****************************** ****************************** */
/* #45 = MIO MODES                                               */

//#define   UNKNOWN                                 (BYTE)    0
#define    MIO_ONLINE                             (BYTE)    1
#define    MIO_ZERO_SCALE                         (BYTE)    2
#define    MIO_MID_SCALE                          (BYTE)    3
#define    MIO_FULL_SCALE                         (BYTE)    4
#define    MIO_DIAGNOSTIC                         (BYTE)    5

/* ****************************** ****************************** */
/* #46 = MIO RELAY_SEL[x] when INSTR_SEL[x] = _O2_RELAYS         */

// #define _NO_SELECTION                 (BYTE)    0
#define    _O2_ANALYZER_FAULT            (BYTE)    1  /* FAIL-SAFE */
#define    _O2_DATA_VALID                (BYTE)    2  /* FAIL-SAFE */

#define    _O2_MAX_RELAY_SEL             (BYTE)    2

/* ****************************** ****************************** */
/* #47 = MIO Primary Status                                      */
//#define   NORMAL      (WORD)      0x0000
#define     INVALID_INSTRUMENT_SELECTION_ANALOG_1A    (WORD)      0x0001
#define     INVALID_OUTPUT_SELECTION_ANALOG_1A        (WORD)      0x0002
#define     INVALID_CALIBRATION_SELECTION_ANALOG_1A   (WORD)      0x0004
#define     INVALID_RANGE_SELECTION_ANALOG_1A         (WORD)      0x0008
#define     INVALID_INSTRUMENT_SELECTION_ANALOG_1B    (WORD)      0x0010
#define     INVALID_OUTPUT_SELECTION_ANALOG_1B        (WORD)      0x0020
#define     INVALID_CALIBRATION_SELECTION_ANALOG_1B   (WORD)      0x0040
#define     INVALID_RANGE_SELECTION_ANALOG_1B         (WORD)      0x0080
#define     INVALID_INSTRUMENT_SELECTION_ANALOG_2A    (WORD)      0x0100
#define     INVALID_OUTPUT_SELECTION_ANALOG_2A        (WORD)      0x0200
#define     INVALID_CALIBRATION_SELECTION_ANALOG_2A   (WORD)      0x0400
#define     INVALID_RANGE_SELECTION_ANALOG_2A         (WORD)      0x0800
#define     INVALID_INSTRUMENT_SELECTION_ANALOG_2B    (WORD)      0x1000
#define     INVALID_OUTPUT_SELECTION_ANALOG_2B        (WORD)      0x2000
#define     INVALID_CALIBRATION_SELECTION_ANALOG_2B   (WORD)      0x4000
#define     INVALID_RANGE_SELECTION_ANALOG_2B         (WORD)      0x8000

/* ****************************** ****************************** */
/* #48 = MIO Extended Status                                     */
#define     MIO_RELAY_1_ENERGIZED   (WORD)      0x0001
#define     MIO_RELAY_2_ENERGIZED   (WORD)      0x0002
#define     MIO_RELAY_3_ENERGIZED   (WORD)      0x0004
#define     MIO_RELAY_4_ENERGIZED   (WORD)      0x0008
#define     MIO_RELAY_5_ENERGIZED   (WORD)      0x0010
#define     MIO_RELAY_6_ENERGIZED   (WORD)      0x0020
#define     MIO_RELAY_7_ENERGIZED   (WORD)      0x0040
#define     MIO_RELAY_8_ENERGIZED   (WORD)      0x0080
#define     MIO_ISOLATOR_1_ENGAGED  (WORD)      0x0100
#define     MIO_ISOLATOR_2_ENGAGED  (WORD)      0x0200
#define     MIO_ISOLATOR_3_ENGAGED  (WORD)      0x0400
#define     MIO_ISOLATOR_4_ENGAGED  (WORD)      0x0800
#define     MIO_ISOLATOR_5_ENGAGED  (WORD)      0x1000
#define     MIO_ISOLATOR_6_ENGAGED  (WORD)      0x2000
#define     MIO_ISOLATOR_7_ENGAGED  (WORD)      0x4000
```

```
#define     MIO_ISOLATOR_8_ENGAGED  (WORD)      0x8000

/* ****************************** ****************************** */
/* #49 = FACTORY RESERVED                                        */

/**** ************************** ************************** ****/
/****  Author: IVICA EFTIMOVSKI                              ****/
/**** ************************** ************************** ****/
/**** Date of this file: DEC-15-2010                         ****/
/**** ************************** ************************** ****/
/**** NET Definitions for the SM8200 Analyzer                ****/
/**** Teledyne Monitor Labs Inc.                             ****/
/**** Copyright 2004  /  All rights reserved                 ****/
/**** ************************** ************************** ****/
/**** THIS FILE IS CONTROLLED BY THE MIO NEURON NODE AUTHOR(IE) ****/
/****             DO NOT EDIT WITHOUT APPROVAL FROM          ****/
/****             THE ENGINEERING MANAGER.                   ****/
/**** ************************** ************************** ****/

typedef struct {

     /* Current Time - 7 */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     /* Communication status - 1 */
     BYTE      com_flag;

     /* Current Status  - 4 */
     WORD      primary_status;
     WORD      extended_status;

     /* Current Mode - 1 */
     BYTE      mode;

     /* Command Ack - 1 */
     BYTE      command;

     /* Software NEURON - 2 */
     BYTE      major_ver_NC;
     BYTE      minor_ver_NC;

     /* Software 332 side - 2 */
     BYTE      major_ver_332;
     BYTE      minor_ver_332;

     /* Software FPGA - 2 */
     BYTE      major_ver_FPGA;
     BYTE      minor_ver_FPGA;

     /* Training Set - 2 */
```

```
     BYTE      major_ver_TS;
     BYTE      minor_ver_TS;

} UNVT_Status; /* Total 22 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     /* Array Control - 4 */
     BYTE      control;

     BYTE      mode;

     WORD      index;

     /* Current Status Summary  - 4 */
     WORD      status_summary;
     WORD      extended_summary;

     /* Concentrations - 16 */
     FLOAT     conc1;
     FLOAT     conc2;
     FLOAT     conc3;
     FLOAT     conc4;

     /* Current Time */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

} UNVT_DataBuffer; /* Total 31 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     /* Current Time - 7 */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     /* Current Mode field - 1 */
     BYTE      mode;

     /* Current Status Summary  - 4 */
     WORD      status_summary;
     WORD      extended_summary;

     /* Concentrations - 16 */
     FLOAT     conc1;
     FLOAT     conc2;
     FLOAT     conc3;
```

```
     FLOAT     conc4;

} UNVT_ConcData; /* Total 28 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     /* Current Time - 7 + 1 */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     BYTE      source;

     /* Current Mode field - 1 */
     BYTE      mode;

     BYTE      filler;

     /* Current Status - 4*/
     WORD      primary_status;
     WORD      extended_status;

     /* Current Status Summary  - 4 */
     WORD      status_summary;
     WORD      extended_summary;

     /* Concentrations */
     FLOAT     inst;
     FLOAT     avg;

} UNVT_OData; /* Total 26 bytes */
/**** *************************** *************************** ****/

typedef struct {

     BYTE      type;
     BYTE      num_io;

     /* Current Time - 8 (7 + 1) */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     BYTE      com_flag;

     WORD      status;

     /* Digital Status */
     WORD      digital;
```

```
     /* Analog input - 16*/
     FLOAT     analog1;
     FLOAT     analog2;
     FLOAT     analog3;
     FLOAT     analog4;

} FloatOutput; /* Total 30 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     BYTE      type;
     BYTE      num_io;

     /* Current Time - 8 (7 + 1) */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     BYTE      com_flag;

     WORD      status;

     /* Digital Status */
     WORD      digital;

     /* Analog input - 16*/
     WORD      analog1;
     WORD      analog2;
     WORD      analog3;
     WORD      analog4;

     WORD      filler[4];

} ScaledOutput; /* Total 30 bytes */
/* ------------------------------ ------------------------------ */

typedef union {

     FloatOutput    fltdata;
     ScaledOutput   scldata;

} UNVT_AUXData; /* Total 30 bytes */
/**** *************************** *************************** ****/

typedef struct {

     /* Current Time */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;
```

```
     BYTE      filler;

} UNVT_Time; /* Total 8 bytes */
/**** *************************** *************************** ****/

ypedef struct {

     BYTE      type;
     BYTE      control;

     WORD      index;

     FLOAT     float_data;

} FloatElement; /* Total 8 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     BYTE      type;
     BYTE      control;

     WORD      index;

     WORD      word_data;
     WORD      filler;

} WordElement; /* Total 8 bytes */
/* ------------------------------ ------------------------------ */

typedef union {

     FloatElement   flt_elem;
     WordElement    wrd_elem;

} UNVT_Array;     /* Total 8 bytes */
/**** *************************** *************************** ****/

typedef struct {

     /* Time of command */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

     BYTE      command;

} UNVT_Command; /* Total 8 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     BYTE      instrument_A;
```

```
     BYTE      CalEnable_A;

     WORD      output_type_A;
     FLOAT     zero_scale_A;
     FLOAT     full_scale_A;

     BYTE      instrument_B;
     BYTE      CalEnable_B;

     WORD      output_type_B;
     FLOAT     zero_scale_B;
     FLOAT     full_scale_B;

} UNVT_MIOAnalog; /* Total 24 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     BYTE      instr_sel[8];
     BYTE      relay_sel[8];

} UNVT_MIORelays; /* Total 16 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     BYTE      control;
     BYTE      DHCP_enable;

     BYTE      IP[4];
     BYTE      gateway[4];
     BYTE      mask[4];

     BYTE      MAC[6];

     BYTE      major_ver_eth; // Version of ethernet ARM Processor
     BYTE      minor_ver_eth; // Version of ethernet ARM Processor

     BYTE      major_ver_REM; // Version of Remote 332
     BYTE      minor_ver_REM; // Version of Remote 332

     BYTE      major_ver_NEU; // See NOTE below ***
     BYTE      minor_ver_NEU; // See NOTE below ***

// *** NOTE ***
// transfer from REM to ETHERNET this is the REM NEURON version
// transfer from ETHERNET to REM this is the ETHERNET NEURON version

} UNVT_EtherCfg; /* Total 26 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     FLOAT     detect_temp;
     FLOAT     bench_temp;

     FLOAT     calgas_temp;
```

```
     FLOAT     probe_temp;

     FLOAT     stack_temp;
     FLOAT     stack_press;

     /* Current Time - 7 */
     WORD      year;
     BYTE      month;
     BYTE      day;
     BYTE      hour;
     BYTE      minute;
     BYTE      sec;

} UNVT_TempPress; /* Total 31 bytes */
/* ------------------------------ ------------------------------ */

typedef struct {

     WORD     conc_alarms_low;
     WORD     conc_alarms_high;

     WORD     MBTU_alarms_low;
     WORD     MBTU_alarms_high;

     WORD     O2_alarms_low;
     WORD     O2_alarms_high;

     WORD     AUX_alarms_low;
     WORD     AUX_alarms_high;

     WORD     reserved1;
     WORD     reserved2;

} UNVT_AlarmState; /* Total 20 bytes */

/**** *************************** *************************** ****/
/****  Author: IVICA EFTIMOVSKI                               ****/
/**** *************************** *************************** ****/
/**** Date of this file: DEC-15-2010                          ****/
/**** *************************** *************************** ****/
/**** NET Definitions for the SM8200 Analyzer                 ****/
/**** Teledyne Monitor Labs Inc.                              ****/
/**** Copyright 2004  /  All rights reserved                  ****/
/**** *************************** *************************** ****/
/**** THIS FILE IS CONTROLLED BY THE MIO NEURON NODE AUTHOR(IE) ****/
/****             DO NOT EDIT WITHOUT APPROVAL FROM           ****/
/****             THE ENGINEERING MANAGER.                    ****/
/**** *************************** *************************** ****/

/* ETHERNET ARM (40-INPUT 21-OUTPUT = 61 VAR) */

extern UNVT_Status                                  /* 4 Variables */
          *nviStatusSM8200, *nviStatus_RESERVED,
              *nviStatusMIO1, *nviStatusMIO2;

extern  UNVT_ConcData                              /* 12 Variables */
          *nviInstValSM8200, *nviInstVal_RESERVED, *nviInstValMBTU,
```

```
          *nviConcAvgSM8200, *nviConcAvg_RESERVED, *nviConcAvgMBTU,
          *nviZeroAvgSM8200, *nviZeroAvg_RESERVED, *nviZeroAvgMBTU,
          *nviSpanAvgSM8200, *nviSpanAvg_RESERVED, *nviSpanAvgMBTU;

extern  UNVT_OData                                  /* 3 Variables */
          *nviConcO, *nviZeroO, *nviSpanO;

extern  UNVT_AUXData     *nviDataX;                  /* 1 Variable */

extern  UNVT_Time        *nvoRTimeSet;               /* 1 Variable */

extern  UNVT_MIOAnalog                              /* 8 Variables */
          *nvoMIO1Analog1, *nvoMIO1Analog2,
          *nvoMIO2Analog1, *nvoMIO2Analog2,
          *nviMIO1Analog1, *nviMIO1Analog2,
          *nviMIO2Analog1, *nviMIO2Analog2;

extern  UNVT_MIORelays                              /* 4 Variables */
          *nviMIO1Relays, *nvoMIO1Relays,
          *nviMIO2Relays, *nvoMIO2Relays;

extern  UNVT_Command                                /* 4 Variables */
          *nvoCmdSM8200, *nvoCmd_RESERVED, *nvoCmdMIO1, *nvoCmdMIO2;

extern  UNVT_Array                                 /* 10 Variables */
          *nviDataSM8200, *nviData_RESERVED,
              *nviCfgSM8200, *nviCfg_RESERVED, *nviCfgX,
          *nvoDataSM8200, *nvoData_RESERVED,
              *nvoCfgSM8200, *nvoCfg_RESERVED, *nvoCfgX;

extern  UNVT_DataBuffer                             /* 8 Variables */
          *nvoBufferSM8200, *nvoBuffer_RESERVED,
              *nvoBufferO, *nvoBufferX,
          *nviBufferSM8200, *nviBuffer_RESERVED,
              *nviBufferO, *nviBufferX;

extern  UNVT_TempPress                              /* 2 Variables */
          *nviTempPressSM8200, *nviTempPress_RESERVED;

extern  UNVT_EtherCfg                               /* 2 Variables */
          *nviEtherCfg, *nvoEtherCfg;

extern  UNVT_AlarmState                             /* 2 Variables */
          *nviAlarmStateSM8200, *nviAlarmState_RESERVED;

/* ***************************** ***************************** */
/* ----------------------------- ----------------------------- */
#ifdef __cplusplus
}
#endif
/* ----------------------------- ----------------------------- */
#endif  // END for #ifndef COMPOSITE_DEFINITIONS_HEADER
/* ***************************** ***************************** */
```

Modbus Input Registers for SM8200

Variable	Element	Type	Address Offset
nviStatusSM8200	year	WORD	1
	month	BYTE	2
	day	BYTE	
	hour	BYTE	3
	min	BYTE	
	sec	BYTE	4
	filler	BYTE	
	primary_status	WORD	5
	extended_status	WORD	6
	mode	BYTE	7
	command	BYTE	
	major_ver_NC	BYTE	8
	minor_ver_NC	BYTE	
	major_ver_332	BYTE	9
	minor_ver_332	BYTE	
	major_ver_FPGA	BYTE	10
	minor_ver_FPGA	BYTE	
	major_ver_TS	BYTE	11
	minor_ver_TS	BYTE	
nviStatus_RESERVED	year	WORD	12
	month	BYTE	13
	day	BYTE	
	hour	BYTE	14
	min	BYTE	
	sec	BYTE	15
	filler	BYTE	
	primary_status	WORD	16
	extended_status	WORD	17
	mode	BYTE	18
	command	BYTE	
	major_ver_NC	BYTE	19
	minor_ver_NC	BYTE	
	major_ver_332	BYTE	20
	minor_ver_332	BYTE	
	major_ver_FPGA	BYTE	21
	minor_ver_FPGA	BYTE	
	major_ver_TS	BYTE	22
	minor_ver_TS	BYTE	
nviStatusMIO1	year	WORD	23
	month	BYTE	24

Variable	Element	Type	Address Offset
	day	BYTE	
	hour	BYTE	25
	min	BYTE	
	sec	BYTE	26
	filler	BYTE	
	primary_status	WORD	27
	extended_status	WORD	28
	mode	BYTE	29
	command	BYTE	
	major_ver_NC	BYTE	30
	minor_ver_NC	BYTE	
	major_ver_332	BYTE	31
	minor_ver_332	BYTE	
	major_ver_FPGA	BYTE	32
	minor_ver_FPGA	BYTE	
	major_ver_TS	BYTE	33
	minor_ver_TS	BYTE	
nviStatusMIO2	year	WORD	34
	month	BYTE	35
	day	BYTE	
	hour	BYTE	36
	min	BYTE	
	sec	BYTE	37
	filler	BYTE	
	primary_status	WORD	38
	extended_status	WORD	39
	mode	BYTE	40
	command	BYTE	
	major_ver_NC	BYTE	41
	minor_ver_NC	BYTE	
	major_ver_332	BYTE	42
	minor_ver_332	BYTE	
	major_ver_FPGA	BYTE	43
	minor_ver_FPGA	BYTE	
	major_ver_TS	BYTE	44
	minor_ver_TS	BYTE	
nviInstValSM8200	year	WORD	45
	month	BYTE	46
	day	BYTE	
	hour	BYTE	47
	min	BYTE	
	sec	BYTE	48
	mode	BYTE	

Variable	Element	Type	Address Offset
	status_summary	WORD	49
	extended_summary	WORD	50
	conc1	FLOAT	51
	conc2	FLOAT	53
	conc3	FLOAT	55
	conc4	FLOAT	57
nviInstVal_RESERVED	year	WORD	59
	month	BYTE	60
	day	BYTE	
	hour	BYTE	61
	min	BYTE	
	sec	BYTE	62
	mode	BYTE	
	status_summary	WORD	63
	extended_summary	WORD	64
	conc1	FLOAT	65
	conc2	FLOAT	67
	conc3	FLOAT	69
	conc4	FLOAT	71
nviInstValMBTU	year	WORD	73
	month	BYTE	74
	day	BYTE	
	hour	BYTE	75
	min	BYTE	
	sec	BYTE	76
	mode	BYTE	
	status_summary	WORD	77
	extended_summary	WORD	78
	conc1	FLOAT	79
	conc2	FLOAT	81
	conc3	FLOAT	83
	conc4	FLOAT	85
nviConcAvgSM8200	year	WORD	87
	month	BYTE	88
	day	BYTE	
	hour	BYTE	89
	min	BYTE	
	sec	BYTE	90
	mode	BYTE	
	status_summary	WORD	91
	extended_summary	WORD	92
	conc1	FLOAT	93
	conc2	FLOAT	95

Variable	Element	Type	Address Offset
	conc3	FLOAT	97
	conc4	FLOAT	99
nviConcAvg_RESERVED	year	WORD	101
	month	BYTE	102
	day	BYTE	
	hour	BYTE	103
	min	BYTE	
	sec	BYTE	104
	mode	BYTE	
	status_summary	WORD	105
	extended_summary	WORD	106
	conc1	FLOAT	107
	conc2	FLOAT	109
	conc3	FLOAT	111
	conc4	FLOAT	113
nviConcAvgMBTU	year	WORD	115
	month	BYTE	116
	day	BYTE	
	hour	BYTE	117
	min	BYTE	
	sec	BYTE	118
	mode	BYTE	
	status_summary	WORD	119
	extended_summary	WORD	120
	conc1	FLOAT	121
	conc2	FLOAT	123
	conc3	FLOAT	125
	conc4	FLOAT	127
nviZeroAvgSM8200	year	WORD	129
	month	BYTE	130
	day	BYTE	
	hour	BYTE	131
	min	BYTE	
	sec	BYTE	132
	mode	BYTE	
	status_summary	WORD	133
	extended_summary	WORD	134
	conc1	FLOAT	135
	conc2	FLOAT	137
	conc3	FLOAT	139
	conc4	FLOAT	141
nviZeroAvg_RESERVED	year	WORD	143
	month	BYTE	144

Variable	Element	Type	Address Offset
	day	BYTE	
	hour	BYTE	145
	min	BYTE	
	sec	BYTE	146
	mode	BYTE	
	status_summary	WORD	147
	extended_summary	WORD	148
	conc1	FLOAT	149
	conc2	FLOAT	151
	conc3	FLOAT	153
	conc4	FLOAT	155
nviZeroAvgMBTU	year	WORD	157
	month	BYTE	158
	day	BYTE	
	hour	BYTE	159
	min	BYTE	
	sec	BYTE	160
	mode	BYTE	
	status_summary	WORD	161
	extended_summary	WORD	162
	conc1	FLOAT	163
	conc2	FLOAT	165
	conc3	FLOAT	167
	conc4	FLOAT	169
nviSpanAvgSM8200	year	WORD	171
	month	BYTE	172
	day	BYTE	
	hour	BYTE	173
	min	BYTE	
	sec	BYTE	174
	mode	BYTE	
	status_summary	WORD	175
	extended_summary	WORD	176
	conc1	FLOAT	177
	conc2	FLOAT	179
	conc3	FLOAT	181
	conc4	FLOAT	183
nviSpanAvg_RESERVED	year	WORD	185
	month	BYTE	186
	day	BYTE	
	hour	BYTE	187
	min	BYTE	
	sec	BYTE	188

Variable	Element	Type	Address Offset
	mode	BYTE	
	status_summary	WORD	189
	extended_summary	WORD	190
	conc1	FLOAT	191
	conc2	FLOAT	193
	conc3	FLOAT	195
	conc4	FLOAT	197
nviSpanAvgMBTU	year	WORD	199
	month	BYTE	200
	day	BYTE	
	hour	BYTE	201
	min	BYTE	
	sec	BYTE	202
	mode	BYTE	
	status_summary	WORD	203
	extended_summary	WORD	204
	conc1	FLOAT	205
	conc2	FLOAT	207
	conc3	FLOAT	209
	conc4	FLOAT	211
nviConcO	year	WORD	213
	month	BYTE	214
	day	BYTE	
	hour	BYTE	215
	min	BYTE	
	sec	BYTE	216
	source	BYTE	
	mode	BYTE	217
	filler	BYTE	
	primary_statys	WORD	218
	extended_status	WORD	219
	status_summary	WORD	220
	extended_summary	WORD	221
	inst	FLOAT	222
	avg	FLOAT	224
nviZeroO	year	WORD	226
	month	BYTE	227
	day	BYTE	
	hour	BYTE	228
	min	BYTE	
	sec	BYTE	229
	source	BYTE	
	mode	BYTE	230

Variable	Element	Type	Address Offset
	filler	BYTE	
	primary_statys	WORD	231
	extended_status	WORD	232
	status_summary	WORD	233
	extended_summary	WORD	234
	inst	FLOAT	235
	avg	FLOAT	237
nviSpanO	year	WORD	239
	month	BYTE	240
	day	BYTE	
	hour	BYTE	241
	min	BYTE	
	sec	BYTE	242
	source	BYTE	
	mode	BYTE	243
	filler	BYTE	
	primary_statys	WORD	244
	extended_status	WORD	245
	status_summary	WORD	246
	extended_summary	WORD	247
	inst	FLOAT	248
	avg	FLOAT	250
nviDataX	type	BYTE	252
	num_io	BYTE	
	year	WORD	253
	month	BYTE	254
	day	BYTE	
	hour	BYTE	255
	min	BYTE	
	sec	BYTE	256
	com_flag	BYTE	
	status	WORD	257
	digital	WORD	258
	analog1	FLOAT	259
	analog2	FLOAT	261
	analog3	FLOAT	263
	analog4	FLOAT	265
nviMIO1Analog1	instrument_A	BYTE	267
	CalEnable_A	BYTE	
	output_type_A	WORD	268
	zero_scale_A	FLOAT	269
	full_scale_A	FLOAT	271
	instrument_B	BYTE	273

Variable	Element	Type	Address Offset
	CalEnable_B	BYTE	
	output_type_B	WORD	274
	zero_scale_B	FLOAT	275
	full_scale_B	FLOAT	277
nviMIO1Analog2	instrument_A	BYTE	279
	CalEnable_A	BYTE	
	output_type_A	WORD	280
	zero_scale_A	FLOAT	281
	full_scale_A	FLOAT	283
	instrument_B	BYTE	285
	CalEnable_B	BYTE	
	output_type_B	WORD	286
	zero_scale_B	FLOAT	287
	full_scale_B	FLOAT	289
nviMIO2Analog1	instrument_A	BYTE	291
	CalEnable_A	BYTE	
	output_type_A	WORD	292
	zero_scale_A	FLOAT	293
	full_scale_A	FLOAT	295
	instrument_B	BYTE	297
	CalEnable_B	BYTE	
	output_type_B	WORD	298
	zero_scale_B	FLOAT	299
	full_scale_B	FLOAT	301
nviMIO2Analog2	instrument_A	BYTE	303
	CalEnable_A	BYTE	
	output_type_A	WORD	304
	zero_scale_A	FLOAT	305
	full_scale_A	FLOAT	307
	instrument_B	BYTE	309
	CalEnable_B	BYTE	
	output_type_B	WORD	310
	zero_scale_B	FLOAT	311
	full_scale_B	FLOAT	313
nviMIO1Relays	instr_sel[0]	BYTE	315
	instr_sel[1]	BYTE	
	instr_sel[2]	BYTE	316
	instr_sel[3]	BYTE	
	instr_sel[4]	BYTE	317
	instr_sel[5]	BYTE	
	instr_sel[6]	BYTE	318
	instr_sel[7]	BYTE	
	relay_sel[0]	BYTE	319

Variable	Element	Type	Address Offset
	relay_sel[1]	BYTE	
	relay_sel[2]	BYTE	320
	relay_sel[3]	BYTE	
	relay_sel[4]	BYTE	321
	relay_sel[5]	BYTE	
	relay_sel[6]	BYTE	322
	relay_sel[7]	BYTE	
nviMIO2Relays	instr_sel[0]	BYTE	323
	instr_sel[1]	BYTE	
	instr_sel[2]	BYTE	324
	instr_sel[3]	BYTE	
	instr_sel[4]	BYTE	325
	instr_sel[5]	BYTE	
	instr_sel[6]	BYTE	326
	instr_sel[7]	BYTE	
	relay_sel[0]	BYTE	327
	relay_sel[1]	BYTE	
	relay_sel[2]	BYTE	328
	relay_sel[3]	BYTE	
	relay_sel[4]	BYTE	329
	relay_sel[5]	BYTE	
	relay_sel[6]	BYTE	330
	relay_sel[7]	BYTE	
nviDataSM8200	type	BYTE	331
	control	BYTE	
	index	WORD	332
	float_data	FLOAT	333
nviData_RESERVED	type	BYTE	335
	control	BYTE	
	index	WORD	336
	float_data	FLOAT	337
nviCfgSM8200	type	BYTE	339
	control	BYTE	
	index	WORD	340
	float_data	FLOAT	341
nviCfg_RESERVED	type	BYTE	343
	control	BYTE	
	index	WORD	344
	float_data	FLOAT	345
nviCfgX	type	BYTE	347
	control	BYTE	
	index	WORD	348
	float_data	FLOAT	349

Variable	Element	Type	Address Offset
nviBufferSM8200	control	BYTE	350
	mode	BYTE	
	index	WORD	351
	status_summary	WORD	352
	extended_status	WORD	353
	conc1	FLOAT	354
	conc2	FLOAT	356
	conc3	FLOAT	358
	conc4	FLOAT	360
	year	WORD	362
	month	BYTE	363
	day	BYTE	
	hour	BYTE	364
	min	BYTE	
	sec	BYTE	365
	N/A	BYTE	
nviBuffer_RESERVED	control	BYTE	366
	mode	BYTE	
	index	WORD	367
	status_summary	WORD	368
	extended_status	WORD	369
	conc1	FLOAT	370
	conc2	FLOAT	372
	conc3	FLOAT	374
	conc4	FLOAT	376
	year	WORD	378
	month	BYTE	379
	day	BYTE	
	hour	BYTE	380
	min	BYTE	
	sec	BYTE	381
	N/A	BYTE	
nviBuffer0	control	BYTE	382
	mode	BYTE	
	index	WORD	383
	status_summary	WORD	384
	extended_status	WORD	385
	conc1	FLOAT	386
	conc2	FLOAT	388
	conc3	FLOAT	390
	conc4	FLOAT	392
	year	WORD	394
	month	BYTE	395

Variable	Element	Type	Address Offset
	day	BYTE	
	hour	BYTE	396
	min	BYTE	
	sec	BYTE	397
	N/A	BYTE	
nviBufferX	control	BYTE	398
	mode	BYTE	
	index	WORD	399
	status_summary	WORD	400
	extended_status	WORD	401
	conc1	FLOAT	402
	conc2	FLOAT	404
	conc3	FLOAT	406
	conc4	FLOAT	408
	year	WORD	410
	month	BYTE	411
	day	BYTE	
	hour	BYTE	412
	min	BYTE	
	sec	BYTE	413
	N/A	BYTE	
nviTempPressSM8200	detect_temp	FLOAT	414
	bench_temp	FLOAT	416
	calgas_temp	FLOAT	418
	probe_temp	FLOAT	420
	stack_temp	FLOAT	422
	stack_press	FLOAT	424
	year	WORD	426
	month	BYTE	427
	day	BYTE	
	hour	BYTE	428
	min	BYTE	
	sec	BYTE	429
	N/A	BYTE	
nviTempPress_RESERVED	detect_temp	FLOAT	430
	bench_temp	FLOAT	432
	calgas_temp	FLOAT	434
	probe_temp	FLOAT	436
	stack_temp	FLOAT	438
	stack_press	FLOAT	440
	year	WORD	442
	month	BYTE	443
	day	BYTE	

Variable	Element	Type	Address Offset
	hour	BYTE	444
	min	BYTE	
	sec	BYTE	445
	N/A	BYTE	
nviAlarmStateSM8200	conc_alarms_low	WORD	446
	conc_alarms_high	WORD	447
	MBTU_alarms_low	WORD	448
	MBTU_alarms_high	WORD	449
	O2_alarms_low	WORD	450
	O2_alarms_high	WORD	451
	AUX_alarms_low	WORD	452
	AUX_alarms_high	WORD	453
	reserved1	WORD	454
	reserved2	WORD	455
nviAlarmState_RESERVED	conc_alarms_low	WORD	456
	conc_alarms_high	WORD	457
	MBTU_alarms_low	WORD	458
	MBTU_alarms_high	WORD	459
	O2_alarms_low	WORD	460
	O2_alarms_high	WORD	461
	AUX_alarms_low	WORD	462
	AUX_alarms_high	WORD	463
	reserved1	WORD	464
	reserved2	WORD	465
nviEtherCfg	enable_eth	BYTE	466
	DHCP	BYTE	
	IP[0]	BYTE	467
	IP[1]	BYTE	
	IP[2]	BYTE	468
	IP[3]	BYTE	
	gateway[0]	BYTE	469
	gateway[1]	BYTE	
	gateway[2]	BYTE	470
	gateway[3]	BYTE	
	mask[0]	BYTE	471
	mask[1]	BYTE	
	mask[2]	BYTE	472
	mask[3]	BYTE	
	MAC[0]	BYTE	473
	MAC[1]	BYTE	
	MAC[2]	BYTE	474
	MAC[3]	BYTE	
	MAC[4]	BYTE	475

Variable	Element	Type	Address Offset
	MAC[5]	BYTE	
	major_ver_eth	BYTE	476
	minor_ver_eth	BYTE	
	major_ver_REM	BYTE	477
	minor_ver_REM	BYTE	
	major_ver_NEU	BYTE	478
	minor_ver_NEU	BYTE	

Modbus Holding Registers for SM8200

Variable	Element	Type	Address Offset
Issue Set Time		WORD	1
nvoRTimeSet	year	WORD	2
	month	BYTE	3
	day	BYTE	
	hour	BYTE	4
	min	BYTE	
	sec	BYTE	5
	filler	BYTE	
Issue SM8200 Command		WORD	6
nvoCmdSM8200	year	WORD	7
	month	BYTE	8
	day	BYTE	
	hour	BYTE	9
	min	BYTE	
	sec	BYTE	10
	command	BYTE	
Issue RESERVED Command		WORD	11
nvoCmd_RESERVED	year	WORD	12
	month	BYTE	13
	day	BYTE	
	hour	BYTE	14
	min	BYTE	
	sec	BYTE	15
	command	BYTE	
Issue MIO1 Command		WORD	16
nvoCmdMIO1	year	WORD	17
	month	BYTE	18
	day	BYTE	
	hour	BYTE	19
	min	BYTE	
	sec	BYTE	20
	command	BYTE	
Issue MIO2 Command		WORD	21
nvoCmdMIO2	year	WORD	22
	month	BYTE	23
	day	BYTE	
	hour	BYTE	24
	min	BYTE	
	sec	BYTE	25
	command	BYTE	

Variable	Element	Type	Address Offset
Issue MIO1 Analog1		WORD	26
nvoMIO1Analog1	instrument_A	BYTE	27
	CalEnable_A	BYTE	
	output_type_A	WORD	28
	zero_scale_A	FLOAT	29
	full_scale_A	FLOAT	31
	instrument_B	BYTE	33
	CalEnable_B	BYTE	
	output_type_B	WORD	34
	zero_scale_B	FLOAT	35
	full_scale_B	FLOAT	37
Issue MIO1 Analog2		WORD	39
nvoMIO1Analog2	instrument_A	BYTE	40
	CalEnable_A	BYTE	
	output_type_A	WORD	41
	zero_scale_A	FLOAT	42
	full_scale_A	FLOAT	44
	instrument_B	BYTE	46
	CalEnable_B	BYTE	
	output_type_B	WORD	47
	zero_scale_B	FLOAT	48
	full_scale_B	FLOAT	50
Issue MIO2 Analog1		WORD	52
nvoMIO2Analog1	instrument_A	BYTE	53
	CalEnable_A	BYTE	
	output_type_A	WORD	54
	zero_scale_A	FLOAT	55
	full_scale_A	FLOAT	57
	instrument_B	BYTE	59
	CalEnable_B	BYTE	
	output_type_B	WORD	60
	zero_scale_B	FLOAT	61
	full_scale_B	FLOAT	63
Issue MIO2 Analog2		WORD	65
nvoMIO2Analog2	instrument_A	BYTE	66
	CalEnable_A	BYTE	
	output_type_A	WORD	67
	zero_scale_A	FLOAT	68
	full_scale_A	FLOAT	70
	instrument_B	BYTE	72
	CalEnable_B	BYTE	
	output_type_B	WORD	73
	zero_scale_B	FLOAT	74

Variable	Element	Type	Address Offset
	full_scale_B	FLOAT	76
Issue MIO1 Relays		WORD	78
nvoMIO1Relays	instr_sel[0]	BYTE	79
	instr_sel[1]	BYTE	
	instr_sel[2]	BYTE	80
	instr_sel[3]	BYTE	
	instr_sel[4]	BYTE	81
	instr_sel[5]	BYTE	
	instr_sel[6]	BYTE	82
	instr_sel[7]	BYTE	
	relay_sel[0]	BYTE	83
	relay_sel[1]	BYTE	
	relay_sel[2]	BYTE	84
	relay_sel[3]	BYTE	
	relay_sel[4]	BYTE	85
	relay_sel[5]	BYTE	
	relay_sel[6]	BYTE	86
	relay_sel[7]	BYTE	
Issue MIO2 Relays		WORD	87
nvoMIO2Relays	instr_sel[0]	BYTE	88
	instr_sel[1]	BYTE	
	instr_sel[2]	BYTE	89
	instr_sel[3]	BYTE	
	instr_sel[4]	BYTE	90
	instr_sel[5]	BYTE	
	instr_sel[6]	BYTE	91
	instr_sel[7]	BYTE	
	relay_sel[0]	BYTE	92
	relay_sel[1]	BYTE	
	relay_sel[2]	BYTE	93
	relay_sel[3]	BYTE	
	relay_sel[4]	BYTE	94
	relay_sel[5]	BYTE	
	relay_sel[6]	BYTE	95
	relay_sel[7]	BYTE	
Issue Data SM8200		WORD	96
nvoDataSM8200	type	BYTE	97
	control	BYTE	
	index	WORD	98
	float_data	FLOAT	99
Issue Data RESERVED		WORD	101
nvoData_RESERVED	type	BYTE	102
	control	BYTE	

Variable	Element	Type	Address Offset
	index	WORD	103
	float_data	FLOAT	104
Issue Config SM8200		WORD	106
nvoCfgSM8200	type	BYTE	107
	control	BYTE	
	index	WORD	108
	float_data	FLOAT	109
Issue Config RESERVED		WORD	111
nvoCfg_RESERVED	type	BYTE	112
	control	BYTE	
	index	WORD	113
	float_data	FLOAT	114
Issue Config Auxiliary		WORD	116
nvoCfgX	type	BYTE	117
	control	BYTE	
	index	WORD	118
	float_data	FLOAT	119
Issue Buffer SM8200		WORD	121
nvoBufferSM8200	control	BYTE	122
	mode	BYTE	
	index	WORD	123
	primary_status	WORD	124
	extended_status	WORD	125
	conc1	FLOAT	126
	conc2	FLOAT	128
	conc3	FLOAT	130
	conc4	FLOAT	132
	year	WORD	134
	month	BYTE	135
	day	BYTE	
	hour	BYTE	136
	min	BYTE	
	sec	BYTE	137
	N/A	BYTE	
Issue Buffer RESERVED		WORD	138
nvoBuffer_RESERVED	control	BYTE	139
	mode	BYTE	
	index	WORD	140
	primary_status	WORD	141
	extended_status	WORD	142
	conc1	FLOAT	143
	conc2	FLOAT	145
	conc3	FLOAT	147

Variable	Element	Type	Address Offset
	conc4	FLOAT	149
	year	WORD	151
	month	BYTE	152
	day	BYTE	
	hour	BYTE	153
	min	BYTE	
	sec	BYTE	154
	N/A	BYTE	
Issue Buffer Oxigen		WORD	155
nvoBuffer0	control	BYTE	156
	mode	BYTE	
	index	WORD	157
	primary_status	WORD	158
	extended_status	WORD	159
	conc1	FLOAT	160
	conc2	FLOAT	162
	conc3	FLOAT	164
	conc4	FLOAT	166
	year	WORD	168
	month	BYTE	169
	day	BYTE	
	hour	BYTE	170
	min	BYTE	
	sec	BYTE	171
	N/A	BYTE	
Issue Buffer Auxiliary		WORD	172
nviBufferX	control	BYTE	173
	mode	BYTE	
	index	WORD	174
	primary_status	WORD	175
	extended_status	WORD	176
	conc1	FLOAT	177
	conc2	FLOAT	179
	conc3	FLOAT	181
	conc4	FLOAT	183
	year	WORD	185
	month	BYTE	186
	day	BYTE	
	hour	BYTE	187
	min	BYTE	
	sec	BYTE	188
	N/A	BYTE	
Issue Ethernet Config		WORD	189

Variable	Element	Type	Address Offset
nvoEtherCfg	enable_eth	BYTE	190
	DHCP	BYTE	
	IP[0]	BYTE	191
	IP[1]	BYTE	
	IP[2]	BYTE	192
	IP[3]	BYTE	
	gateway[0]	BYTE	193
	gateway[1]	BYTE	
	gateway[2]	BYTE	194
	gateway[3]	BYTE	
	mask[0]	BYTE	195
	mask[1]	BYTE	
	mask[2]	BYTE	196
	mask[3]	BYTE	
	MAC[0]	BYTE	197
	MAC[1]	BYTE	
	MAC[2]	BYTE	198
	MAC[3]	BYTE	
	MAC[4]	BYTE	199
	MAC[5]	BYTE	
	major_ver_eth	BYTE	200
	minor_ver_eth	BYTE	
	major_ver_REM	BYTE	201
	minor_ver_REM	BYTE	
	major_ver_NEU	BYTE	202
	minor_ver_NEU	BYTE	

Word Configuration Data Descriptions for SM8200 Analyzer

nvoCfgP

Word (16 bit) SM8200 INPUT/OUTPUT configuration variables, unstamped.

Index #	Element Name	Default	Range, Unit Code
0	(this variable not-used)		
1-13	FACTORY RESERVED	N/A	0
14	Enable Gas 1 measurement	2	1 – DISABLED 2 -- ENABLED
15	Enable Gas 2 measurement	2	1 – DISABLED 2 -- ENABLED
16	Enable Gas 3 measurement	2	1 – DISABLED 2 -- ENABLED
17	Enable Gas 4 measurement	2	1 – DISABLED 2 -- ENABLED
18	Enable Gas 1 auto-correction	1	1 – DISABLED 2 -- ENABLED
19	Enable Gas 2 auto-correction	1	1 – DISABLED 2 -- ENABLED
20	Enable Gas 3 auto-correction	1	1 – DISABLED 2 -- ENABLED
21	Enable Gas 4 auto-correction	1	1 – DISABLED 2 -- ENABLED
22	Gas 1 Linearizer Type	1	1 - Polynomial 2 -- LUT
23	Gas 2 Linearizer Type	1	1 - Polynomial 2 -- LUT
24	Gas 3 Linearizer Type	1	1 - Polynomial 2 -- LUT
25	Gas 4 Linearizer Type	1	1 - Polynomial 2 -- LUT
26	PDA Temp Command	64	0 to 128
27	Preamp Clock 1 Cancellation	0	0 to 128
28	Preamp Clock 2 Cancellation	0	0 to 128
29	Preamp Offset	64	0 to 128
30	PDA Video Gain	0	0 to 255
31	PDA Temp Gain	0	0 to 255
32	Cal Gas Temp Gain	0	0 to 255

Index #	Element Name	Default	Range, Unit Code
33	Bench Temp Gain	0	0 to 255
34	Cal Gas Heater Temp Command	0	0 to 4095
35	Cal Gas Heater Auto Adjust	1	1 – DISABLED 2 -- ENABLED
36	Probe Heater Temp Command	0	0 to 4095
37	Probe Heater Auto Adjust	1	1 – DISABLED 2 -- ENABLED
38	Diluent Type	1	1 - O2 2 -- CO2
39	Bottled Gas Calibration	2	1 – DISABLED 2 -- ENABLED
40	Acquire Mode Averaging	0	0
41	Stack Temp RTD Polynomial	2	1 – DISABLED 2 -- ENABLED
42	Zero Acquire Integration Periods	4	0 to 100
43	Zero Integration Periods	4	0 to 100
44	Dark Current Integration Periods	4	0 to 100
45	Wavelength Check Integration Periods	4	0 to 100
46	Span Acquire Integration Periods	4	0 to 100
47	Span or EO Integration Periods	4	0 to 100
48	Normal Acquire Integration Periods	4	0 to 100
49	Integration Periods in Average	4	0 to 100
50	English or Metric Units	1	1 - English 2 -- Metric
51	Hour of Auto-Cal	0	0 to 25 0=Midnight 25=DISABLED
52	Minute of Auto-Cal	0	0 to 59
53	Interval Hour of Auto-Cal	25	0 to 25
54	Manual Cal Timeout Minutes	0	(TBD)

Index #	Element Name	Default	Range, Unit Code
55	Wavelength Check	1	1 – DISABLED 2 -- ENABLED
56	Wavelength SG Used	1	0 - None 1 - SG set 1 2 -- SG set 2
57	ERP Screen Code (request many variables)	N/A	N/A
58	PDA Charge Time Counts	5250	1667 to 65535 / 24uS
59	CGA Cycles	N/A	1 to 6
60	Daily Cal Gas	2	1 - CGA_MID 2 – CGA_HIGH
61	Enable Standardization for SG1	1	1 – DISABLED 2 – ENABLED
62	Enable Standardization for SG2	1	1 – DISABLED 2 – ENABLED

Float Configuration Data Descriptions for SM8200 Analyzer

nvoCfgP

Floating point Procyon INPUT/OUTPUT configuration variables, unstamped.

Index #	Element Name	Default	Range, Unit Code
0	(this variable not-used)		
1	Gas 1 Zero Setpoint	0	(TBD)
2	Gas 2 Zero Setpoint	0	(TBD)
3	Gas 3 Zero Setpoint	0	(TBD)
4	Gas 4 Zero Setpoint	0	(TBD)
5	Gas 1 Span Setpoint	0	(TBD)
6	Gas 2 Span Setpoint	0	(TBD)
7	Gas 3 Span Setpoint	0	(TBD)
8	Gas 4 Span Setpoint	0	(TBD)
9	Gas 1 EO Cal Setpoint	0	(TBD)
10	Gas 2 EO Cal Setpoint	0	(TBD)
11	Gas 3 EO Cal Setpoint	0	(TBD)
12	Gas 4 EO Cal Setpoint	0	(TBD)
13	Gas 1 Full Scale	0	(TBD)
14	Gas 2 Full Scale	0	(TBD)
15	Gas 3 Full Scale	0	(TBD)
16	Gas 4 Full Scale	0	(TBD)
17	Gas 1 Cal Tolerance	0	% Full Scale
18	Gas 2 Cal Tolerance	0	% Full Scale
19	Gas 3 Cal Tolerance	0	% Full Scale
20	Gas 4 Cal Tolerance	0	% Full Scale
21	Gas 1 A0 or X1	0	(TBD)
22	Gas 1 A1 or Y1	1	(TBD)
23	Gas 1 A2 or X2	0	(TBD)
24	Gas 1 A3 or Y2	0	(TBD)
25	Gas 1 A4 or X3	0	(TBD)
26	Gas 1 A5 or Y3	0	(TBD)
27	Gas 2 A0 or X1	0	(TBD)
28	Gas 2 A1 or Y1	1	(TBD)
29	Gas 2 A2 or X2	0	(TBD)

Index #	Element Name	Default	Range, Unit Code
30	Gas 2 A3 or Y2	0	(TBD)
31	Gas 2 A4 or X3	0	(TBD)
32	Gas 2 A5 or Y3	0	(TBD)
33	Gas 3 A0 or X1	0	(TBD)
34	Gas 3 A1 or Y1	1	(TBD)
35	Gas 3 A2 or X2	0	(TBD)
36	Gas 3 A3 or Y2	0	(TBD)
37	Gas 3 A4 or X3	0	(TBD)
38	Gas 3 A5 or Y3	0	(TBD)
39	Gas 4 A0 or X1	0	(TBD)
40	Gas 4 A1 or Y1	1	(TBD)
41	Gas 4 A2 or X2	0	(TBD)
42	Gas 4 A3 or Y2	0	(TBD)
43	Gas 4 A4 or X3	0	(TBD)
44	Gas 4 A5 or Y3	0	(TBD)
45	Stack Press Low Counts	0	(TBD)
46	Stack Press High Counts	1	(TBD)
47	Stack Press Low Value	0	inch Hg/kPa
48	Stack Press High Value	1	inch Hg/kPa
49	Stack Press Fault Min	0	inch Hg/kPa
50	Stack Press Fault Max	0	inch Hg/kPa
51	Stack Temp Low Linearized Counts	0	(TBD)
52	Stack Temp High Linearized Counts	0	(TBD)
53	Stack Temp Low Value	0	Deg. F/C
54	Stack Temp High Value	0	Deg. F/C
55	Stack Temp Fault Min	0	Deg. F/C
56	Stack Temp Fault Max	0	Deg. F/C
57	Bench Temp Low Counts	0	(TBD)
58	Bench Temp High Counts	0	(TBD)
59	Bench Temp Low Value	0	Deg. F/C
60	Bench Temp High Value	0	Deg. F/C
61	Bench Temp Fault Min	0	Deg. F/C
62	Bench Temp Fault Max	0	Deg. F/C
63	Probe Temp Low Counts	0	(TBD)
64	Probe Temp High Counts	0	(TBD)
65	Probe Temp Low Value	0	Deg. F/C

Index #	Element Name	Default	Range, Unit Code
66	Probe Temp High Value	0	Deg. F/C
67	Probe Temp Fault Min	0	Deg. F/C
68	Probe Temp Fault Max	0	Deg. F/C
69	Cal Gas Temp Low Counts	0	(TBD)
70	Cal Gas Temp High Counts	0	(TBD)
71	Cal Gas Temp Low Value	0	Deg. F/C
72	Cal Gas Temp High Value	0	Deg. F/C
73	Cal Gas Temp Max Fault Delta	0	Deg. F/C
74	PDA Temp R2	0	(TBD)
75	PDA Temp Low Linearized 4th Root Rth	0	(TBD)
76	PDA Temp Low Temperature Value	0	(TBD)
77	PDA Temp High Linearized 4th Root Rth	0	(TBD)
78	PDA Temp High Temperature Value	0	(TBD)
79	PDA Thermistor Resistance Poly A0	1	(TBD)
80	PDA Thermistor Resistance Poly A1	0	(TBD)
81	PDA Thermistor Resistance Poly A2	0	(TBD)
82	PDA Thermistor Resistance Poly A3	0	(TBD)
83	PDA Thermistor Resistance Poly A4	0	(TBD)
84	PDA Thermistor Resistance Poly A5	0	(TBD)
85	Fuel Constant	0	(TBD)
86	Gas 1 Zero Incremental Adjust Limit	0	(TBD)
87	Gas 1 Zero Total Adjust Limit	0	(TBD)
88	Gas 1 Span Incremental Adjust Limit	0	(TBD)
89	Gas 1 Span Total Adjust Limit	0	(TBD)
90	Gas 2 Zero Incremental Adjust Limit	0	(TBD)
91	Gas 2 Zero Total Adjust Limit	0	(TBD)
92	Gas 2 Span Incremental Adjust Limit	0	(TBD)
93	Gas 2 Span Total Adjust Limit	0	(TBD)
94	Gas 3 Zero Incremental Adjust Limit	0	(TBD)
95	Gas 3 Zero Total Adjust Limit	0	(TBD)
96	Gas 3 Span Incremental Adjust Limit	0	(TBD)
97	Gas 3 Span Total Adjust Limit	0	(TBD)
98	Gas 4 Zero Incremental Adjust Limit	0	(TBD)
99	Gas 4 Zero Total Adjust Limit	0	(TBD)
100	Gas 4 Span Incremental Adjust Limit	0	(TBD)
101	Gas 4 Span Total Adjust Limit	0	(TBD)

Index #	Element Name	Default	Range, Unit Code
102	Gas 1 Slope Adjust Limit	0	(TBD)
103	Gas 2 Slope Adjust Limit	0	(TBD)
104	Gas 3 Slope Adjust Limit	0	(TBD)
105	Gas 4 Slope Adjust Limit	0	(TBD)
106	Gas 1 Density (for lbs/mmBTU)	0	(TBD)
107	Gas 2 Density (for lbs/mmBTU)	0	(TBD)
108	Gas 3 Density (for lbs/mmBTU)	0	(TBD)
109	Gas 4 Density (for lbs/mmBTU)	0	(TBD)
110	Pressure Conv. Gas 1 Poly A0	1	(TBD)
111	Pressure Conv. Gas 1 Poly A1	0	(TBD)
112	Pressure Conv. Gas 1 Poly A2	0	(TBD)
113	Pressure Conv. Gas 1 Poly A3	0	(TBD)
114	Pressure Conv. Gas 1 Poly A4	0	(TBD)
115	Pressure Conv. Gas 1 Poly A5	0	(TBD)
116	Pressure Conv. Gas 2 Poly A0	1	(TBD)
117	Pressure Conv. Gas 2 Poly A1	0	(TBD)
118	Pressure Conv. Gas 2 Poly A2	0	(TBD)
119	Pressure Conv. Gas 2 Poly A3	0	(TBD)
120	Pressure Conv. Gas 2 Poly A4	0	(TBD)
121	Pressure Conv. Gas 2 Poly A5	0	(TBD)
122	Pressure Conv. Gas 3 Poly A0	1	(TBD)
123	Pressure Conv. Gas 3 Poly A1	0	(TBD)
124	Pressure Conv. Gas 3 Poly A2	0	(TBD)
125	Pressure Conv. Gas 3 Poly A3	0	(TBD)
126	Pressure Conv. Gas 3 Poly A4	0	(TBD)
127	Pressure Conv. Gas 3 Poly A5	0	(TBD)
128	Pressure Conv. Gas 4 Poly A0	1	(TBD)
129	Pressure Conv. Gas 4 Poly A1	0	(TBD)
130	Pressure Conv. Gas 4 Poly A2	0	(TBD)
131	Pressure Conv. Gas 4 Poly A3	0	(TBD)
132	Pressure Conv. Gas 4 Poly A4	0	(TBD)
133	Pressure Conv. Gas 4 Poly A5	0	(TBD)
134	Reference Temperature	68	Deg. F/C
135	Reference Pressure	29.92	inch Hg/kPa
136	Integration Period	0	(TBD)
137	Lamp Adjust IP Seconds	1	1 - 10 Seconds

Index #	Element Name	Default	Range, Unit Code
138	Stack Temp RTD Counts Poly A0	0	(TBD)
139	Stack Temp RTD Counts Poly A1	1	(TBD)
140	Stack Temp RTD Counts Poly A2	0	(TBD)
141	Stack Temp RTD Counts Poly A3	0	(TBD)
142	Stack Temp RTD Counts Poly A4	0	(TBD)
143	Stack Temp RTD Counts Poly A5	0	(TBD)
144	Inst. Conc Gas 1 Alarm Threshold Low	0	(TBD)
145	Inst. Conc Gas 2 Alarm Threshold Low	0	(TBD)
146	Inst. Conc Gas 3 Alarm Threshold Low	0	(TBD)
147	Inst. Conc Gas 4 Alarm Threshold Low	0	(TBD)
148	Inst. Conc Gas 1 Alarm Threshold High	0	(TBD)
149	Inst. Conc Gas 2 Alarm Threshold High	0	(TBD)
150	Inst. Conc Gas 3 Alarm Threshold High	0	(TBD)
151	Inst. Conc Gas 4 Alarm Threshold High	0	(TBD)
152	Avg. Conc Gas 1 Alarm Threshold Low	0	(TBD)
153	Avg. Conc Gas 2 Alarm Threshold Low	0	(TBD)
154	Avg. Conc Gas 3 Alarm Threshold Low	0	(TBD)
155	Avg. Conc Gas 4 Alarm Threshold Low	0	(TBD)
156	Avg. Conc Gas 1 Alarm Threshold High	0	(TBD)
157	Avg. Conc Gas 2 Alarm Threshold High	0	(TBD)
158	Avg. Conc Gas 3 Alarm Threshold High	0	(TBD)
159	Avg. Conc Gas 4 Alarm Threshold High	0	(TBD)
160	Inst. MBTU Gas 1 Alarm Threshold Low	0	(TBD)
161	Inst. MBTU Gas 2 Alarm Threshold Low	0	(TBD)
162	Inst. MBTU Gas 3 Alarm Threshold Low	0	(TBD)
163	Inst. MBTU Gas 4 Alarm Threshold Low	0	(TBD)
164	Inst. MBTU Gas 1 Alarm Threshold High	0	(TBD)
165	Inst. MBTU Gas 2 Alarm Threshold High	0	(TBD)
166	Inst. MBTU Gas 3 Alarm Threshold High	0	(TBD)
167	Inst. MBTU Gas 4 Alarm Threshold High	0	(TBD)
168	Avg. MBTU Gas 1 Alarm Threshold Low	0	(TBD)
169	Avg. MBTU Gas 2 Alarm Threshold Low	0	(TBD)
170	Avg. MBTU Gas 3 Alarm Threshold Low	0	(TBD)
171	Avg. MBTU Gas 4 Alarm Threshold Low	0	(TBD)
172	Avg. MBTU Gas 1 Alarm Threshold High	0	(TBD)
173	Avg. MBTU Gas 2 Alarm Threshold High	0	(TBD)

Index #	Element Name	Default	Range, Unit Code
174	Avg. MBTU Gas 3 Alarm Threshold High	0	(TBD)
175	Avg. MBTU Gas 4 Alarm Threshold High	0	(TBD)
176	AUX Input 1 Low Counts (O2)	0	(TBD)
177	AUX Input 1 High Counts (O2)	0	(TBD)
178	AUX Input 1 Low Value (O2)	0	(TBD)
179	AUX Input 1 High Value (O2)	0	(TBD)
180	AUX Input 1 Fault Min (O2)	0	(TBD)
181	AUX Input 1 Fault Max (O2)	0	(TBD)
182	AUX Input 2 Low Counts (Undefined)	0	(TBD)
183	AUX Input 2 High Counts (Undefined)	0	(TBD)
184	AUX Input 2 Low Value (Undefined)	0	(TBD)
185	AUX Input 2 High Value (Undefined)	0	(TBD)
186	AUX Input 2 Fault Min (Undefined)	0	(TBD)
187	AUX Input 2 Fault Max (Undefined)	0	(TBD)
188	Mean Dark Current Fault Threshold Min	0	(TBD)
189	Mean Dark Current Fault Threshold Max	0	(TBD)
190	Mean Ref Current Fault Threshold Min	0	(TBD)
191	Mean Ref Current Fault Threshold Max	0	(TBD)
192	Wavelength Check Failure Threshold	0	(TBD)
193	Cell Length	1	centimeters
194	Moisture Fraction	0.027	(TBD)
195	Metric Conversion Gas 1	1.24596	(TBD)
196	Metric Conversion Gas 2	2.66	(TBD)
197	Metric Conversion Gas 3	1.91031	(TBD)
198	Metric Conversion Gas 4	0.70715	(TBD)
199	Inter Comp NO Gain U11	1	(TBD)
200	Inter Comp SO2 Gain U22	1	(TBD)
201	Inter Comp NO2 Gain U33	1	(TBD)
202	Inter Comp NH3 Gain U44	1	(TBD)
203	Inter Comp SO2 NO U12	1	(TBD)
204	Inter Comp NO2 NO U13	1	(TBD)
205	Inter Comp NH3 NO U14	1	(TBD)
206	Inter Comp NO SO2 U21	1	(TBD)
207	Inter Comp NO2 SO2 U23	1	(TBD)
208	Inter Comp NH3 SO2 U24	1	(TBD)
209	Inter Comp NO NO2 U31	1	(TBD)

Index #	Element Name	Default	Range, Unit Code
210	Inter Comp SO2 NO2 U32	1	(TBD)
211	Inter Comp NH3 NO2 U34	1	(TBD)
212	Inter Comp NO NH3 U41	1	(TBD)
213	Inter Comp SO2 NH3 U42	1	(TBD)
214	Inter Comp NO2 NH3 U43	1	(TBD)
215-727	Wavelength Check Stored Spectra (512)	0	N/A

Word Data Descriptions for Procyon Analyzer

nvoDataP

Word (16 bit) Procyon OUTPUT data variables, unstamped.

Index #	Element Name	Default	Range, Unit Code
0	(this variable not-used)	0	N/A
1	Cal Gas Temp Counts	0	0
2	PDA Temp Counts	0	0
3	Stack Temp Counts	0	0
4	Bench Temp Counts	0	0
5	Probe Temp Counts	0	0
6	AUX Input 1 Counts (O2)	0	0
7	AUX Input 2 Counts (Undefined)	0	0
8	Stack Pressure Count	0	0
9	Manual Pixel Offset	0	0
10	User Set CRC Check Result	0	0x0000-0xFFFF
11	Factory Cal Set CRC Check Result	0	0x0000-0xFFFF
12	Peak Pixel Number	0	0

Float Data Descriptions for Procyon Analyzer

nvoDataP

Floating point Procyon OUTPUT data variables, unstamped.

Index #	Element Name	Default	Range, Unit Code
0	(this variable not-used)	0	0
1	Gas 1 Zero Incremental Adjust	0	0
2	Gas 1 Zero Total Adjust	0	0
3	Gas 1 Span Incremental Adjust	0	0
4	Gas 1 Span Total Adjust	0	0
5	Gas 2 Zero Incremental Adjust	0	0
6	Gas 2 Zero Total Adjust	0	0
7	Gas 2 Span Incremental Adjust	0	0
8	Gas 2 Span Total Adjust	0	0
9	Gas 3 Zero Incremental Adjust	0	0
10	Gas 3 Zero Total Adjust	0	0
11	Gas 3 Span Incremental Adjust	0	0
12	Gas 3 Span Total Adjust	0	0
13	Gas 4 Zero Incremental Adjust	0	0
14	Gas 4 Zero Total Adjust	0	0
15	Gas 4 Span Incremental Adjust	0	0
16	Gas 4 Span Total Adjust	0	0
17	Gas 1 Auto-correct Offset	0	0
18	Gas 1 Auto-correct Slope	0	0
19	Gas 2 Auto-correct Offset	0	0
20	Gas 2 Auto-correct Slope	0	0
21	Gas 3 Auto-correct Offset	0	0
22	Gas 3 Auto-correct Slope	0	0
23	Gas 4 Auto-correct Offset	0	0
24	Gas 4 Auto-correct Slope	0	0
25	Mean Dark Current	0	0
26	Mean Reference Current	0	0
27	Gas 1 Conc 1 to 10*	0	0
37	Gas 2 Conc 1 to 10*	0	0
47	Gas 3 Conc 1 to 10*	0	0
57	Gas 4 Conc 1 to 10*	0	0
67	Gas 1 Zero Conc 1 to 10*	0	0

Index #	Element Name	Default	Range, Unit Code
77	Gas 2 Zero Conc 1 to 10*	0	0
87	Gas 3 Zero Conc 1 to 10*	0	0
97	Gas 4 Zero Conc 1 to 10*	0	0
107	Gas 1 Span Conc 1 to 10*	0	0
117	Gas 2 Span Conc 1 to 10*	0	0
127	Gas 3 Span Conc 1 to 10*	0	0
137	Gas 4 Span Conc 1 to 10*	0	0
147	Stack Temperature 1 to 10*	0	0
157	Stack Pressure 1 to 10*	0	0
167	Wavelength Check Offsets 1 to 10*	0	0
177	Wavelength Check SSE	0	0
178	RTD Linearized Counts	0	0
179	4th Root Rth	0	0
180	Linearized 4th Root Rth	0	0
181	Vth (PDA Temp Thermistor Voltage)	0	0
182	Rth (PDA Temp Thermistor Resistance)	0	0
183	PDA Charge Time Seconds	0	0
184	Normal Raw Gas 1 Conc	0	0
185	Normal Raw Gas 2 Conc	0	0
186	Normal Raw Gas 3 Conc	0	0
187	Normal Raw Gas 4 Conc	0	0
188	Zero Raw Gas 1 Conc	0	0
189	Zero Raw Gas 2 Conc	0	0
190	Zero Raw Gas 3 Conc	0	0
191	Zero Raw Gas 4 Conc	0	0
192	Span Raw Gas 1 Conc	0	0
193	Span Raw Gas 2 Conc	0	0
194	Span Raw Gas 3 Conc	0	0
195	Span Raw Gas 4 Conc	0	0
196	Signal Peak Wavelength	0	0
197	CGA Gas 1 Initial Zero	0	0
198	CGA Gas 1 Run 1 Low	0	0
199	CGA Gas 1 Run 1 Mid	0	0
200	CGA Gas 1 Run 1 High	0	0
201	CGA Gas 1 Run 2 Low	0	0
202	CGA Gas 1 Run 2 Mid	0	0

Index #	Element Name	Default	Range, Unit Code
203	CGA Gas 1 Run 2 High	0	0
204	CGA Gas 1 Run 3 Low	0	0
205	CGA Gas 1 Run 3 Mid	0	0
206	CGA Gas 1 Run 3 High	0	0
207	CGA Gas 1 Run 4 Low	0	0
208	CGA Gas 1 Run 4 Mid	0	0
209	CGA Gas 1 Run 4 High	0	0
210	CGA Gas 1 Run 5 Low	0	0
211	CGA Gas 1 Run 5 Mid	0	0
212	CGA Gas 1 Run 5 High	0	0
213	CGA Gas 1 Run 6 Low	0	0
214	CGA Gas 1 Run 6 Mid	0	0
215	CGA Gas 1 Run 6 High	0	0
216	CGA Gas 1 Final Zero	0	0
217	CGA Gas 2 Initial Zero	0	0
218	CGA Gas 2 Run 1 Low	0	0
219	CGA Gas 2 Run 1 Mid	0	0
220	CGA Gas 2 Run 1 High	0	0
221	CGA Gas 2 Run 2 Low	0	0
222	CGA Gas 2 Run 2 Mid	0	0
223	CGA Gas 2 Run 2 High	0	0
224	CGA Gas 2 Run 3 Low	0	0
225	CGA Gas 2 Run 3 Mid	0	0
226	CGA Gas 2 Run 3 High	0	0
227	CGA Gas 2 Run 4 Low	0	0
228	CGA Gas 2 Run 4 Mid	0	0
229	CGA Gas 2 Run 4 High	0	0
230	CGA Gas 2 Run 5 Low	0	0
231	CGA Gas 2 Run 5 Mid	0	0
232	CGA Gas 2 Run 5 High	0	0
233	CGA Gas 2 Run 6 Low	0	0
234	CGA Gas 2 Run 6 Mid	0	0
235	CGA Gas 2 Run 6 High	0	0
236	CGA Gas 2 Final Zero	0	0
237	CGA Gas 3 Initial Zero	0	0
238	CGA Gas 3 Run 1 Low	0	0

Index #	Element Name	Default	Range, Unit Code
239	CGA Gas 3 Run 1 Mid	0	0
240	CGA Gas 3 Run 1 High	0	0
241	CGA Gas 3 Run 2 Low	0	0
242	CGA Gas 3 Run 2 Mid	0	0
243	CGA Gas 3 Run 2 High	0	0
244	CGA Gas 3 Run 3 Low	0	0
245	CGA Gas 3 Run 3 Mid	0	0
246	CGA Gas 3 Run 3 High	0	0
247	CGA Gas 3 Run 4 Low	0	0
248	CGA Gas 3 Run 4 Mid	0	0
249	CGA Gas 3 Run 4 High	0	0
250	CGA Gas 3 Run 5 Low	0	0
251	CGA Gas 3 Run 5 Mid	0	0
252	CGA Gas 3 Run 5 High	0	0
253	CGA Gas 3 Run 6 Low	0	0
254	CGA Gas 3 Run 6 Mid	0	0
255	CGA Gas 3 Run 6 High	0	0
256	CGA Gas 3 Final Zero	0	0
257	CGA Gas 4 Initial Zero	0	0
258	CGA Gas 4 Run 1 Low	0	0
259	CGA Gas 4 Run 1 Mid	0	0
260	CGA Gas 4 Run 1 High	0	0
261	CGA Gas 4 Run 2 Low	0	0
262	CGA Gas 4 Run 2 Mid	0	0
263	CGA Gas 4 Run 2 High	0	0
264	CGA Gas 4 Run 3 Low	0	0
265	CGA Gas 4 Run 3 Mid	0	0
266	CGA Gas 4 Run 3 High	0	0
267	CGA Gas 4 Run 4 Low	0	0
268	CGA Gas 4 Run 4 Mid	0	0
269	CGA Gas 4 Run 4 High	0	0
270	CGA Gas 4 Run 5 Low	0	0
271	CGA Gas 4 Run 5 Mid	0	0
272	CGA Gas 4 Run 5 High	0	0
273	CGA Gas 4 Run 6 Low	0	0
274	CGA Gas 4 Run 6 Mid	0	0

Index #	Element Name	Default	Range, Unit Code
275	CGA Gas 4 Run 6 High	0	0
276	CGA Gas 4 Final Zero	0	0