Global Navigation Satellite System Module

Every effort has been made to ensure the accuracy of the information contained herein. If you find errors or inconsistencies please bring them to our attention.

Notice:

"HP-41C", "HP-41CV", "HP-41CX" and "HP" are registered trademarks of Hewlett-Packard, Inc. All uses of these terms in this document are to be construed as adjectives, whether or not the noun "calculator", "CPU" or "device" are actually present.

Acknowledgements:

Mark Fleming provided the initial impetus for this 41C module and software, and provided guidance during the development. Nate Martin provided the design of the module housing. Sylvain Côté, Mark Fleming, Ángel Martin and Robert Prosperi reviewed the preliminary specification.

Disclaimer

The GNSS Module should not be used in such a manner that deficiencies, omissions, inaccuracies or errors could result in death, loss or injury.

WARRANTIES AND DISCLAIMER

The GNSS Software is supplied “as is” and without any warranty of any kind, either express, implied, or arising by statute, custom, course of dealing, or trade usage. Systemyde International Corporation specifically disclaims any and all implied warranties or conditions of title, non-infringement, accuracy or completeness, the presence or absence of errors, fitness for a particular purpose, merchantability, or otherwise. Some jurisdictions do not allow the exclusion of implied warranties, so this exclusion may not apply to you.

LIMITATION OF LIABILITY

Subject to any liability that may not be excluded or limited by law, Systemyde International Corporation is not liable for, and expressly excludes, all liability for loss or damage however and whenever caused to anyone by any use of the GNSS Module, whether by you or by anyone else, and whether caused by any fault on the part of Systemyde International Corporation or not. This exclusion of liability includes, but is not limited to, any special, incidental, consequential, punitive, or exemplary damages such as loss of revenue, data, anticipated profits, and lost business. This exclusion applies even if Systemyde International Corporation has been advised of the possibility of such damages.

If liability may not be excluded by law, it is limited to actual and direct financial loss to the extent it is caused by proved negligence on the part of Systemyde International Corporation.

It is particularly important to realize that the GNSS position information may not be accurate. You should make your own judgement about the accuracy of the GNSS position information. This is especially true for flying, inland navigation, marine navigation, and military applications. You are strongly advised not to use the GNSS Module as your primary or sole information source for these activities.

Table of Contents

1. Introduction 7

2. Batteries and Current Drain 9

3. GNSS Data Formats 11

4. Receiver Control Functions 14

GNAUTO (GNSS Receiver Automatic On-Off) 14
GNON (GNSS Receiver On) 14
GNAON (GNSS Receiver Always On) 15
GNSBY (GNSS Receiver in Standby) 15
GNOFF (GNSS Module Off) 15
GNACT? (GNSS Receiver Active?) 15
GNTRK? (GNSS Receiver Tracking?) 16

5. Individual GNSS Data Functions 17

DATUM (Generate GNSS Datum) 17
LAT (Retrieve Latitude) 18
LON (Retrieve Longitude) 19
ALTI (Retrieve MSL Altitude) 19
HEADING (Retrieve Heading) 20
SPEED (Retrieve Speed) 20
UTIME (Retrieve UTC Time) 21
UDATE (Retrieve UTC Date) 21
SATS (Retrieve GNSS Satellites Used) 21
SATIV (Retrieve GNSS Satellites In View) 22
GEOID (Retrieve Geoidal Separation) 22
HDOP (Retrieve Horizontal Dilution of Precision) 22
VDOP (Retrieve Vertical Dilution of Precision) 23

6. Waypoint Functions 24

AC (Altitude and Course) 25
ACT (Altitude, Course and Time) 25
ACTD (Altitude, Course, Time and Date) 26
CT (Course and Time) 26
CTD (Course, Time and Date) 26
LL (Position) 26
LLA (Position and Altitude) 27
LLAC (Position, Altitude and Course) 27
LLACT (Position, Altitude, Course and Time) 27
LLACTD (Position, Altitude, Course, Time and Date) 27

LLC (Position and Time) 28
LLCT (Position, Course and Time) 28
LLCTD (Position, Course, Time and Date)..... 28
LLT (Position and Time) 28
LLTD (Position, Time and Date)..... 29

7. Waypoint Housekeeping Functions 30

PTRP (Use Waypoint Pointer Register) 30
PTRX (Use X-Register as waypoint pointer) 30
RCLWBR (Recall Waypoint Base Register) 31
STOWBR (Store Waypoint Base Register) 31
RCLWPR (Recall Waypoint Pointer Register) 31
STOWPR (Store Waypoint Pointer Register) 31
WPR+X (Add X to Current Waypoint Register) 32
RCLRNG (Recall Waypoint Register Range) 32

8. Miscellaneous Functions 33

DMT (Convert to Degrees-Minutes-Tenths) 33
GMODE (Set GNSS Mode) 33
GMODE? (Query GNSS Modes) 33
GCLOCK (Continuously Display UTC Time) 36

9. ROM Option Functions 37

DISROM (Disable GNSS Module ROM) 37
ENROM (Enable GNSS Module ROM) 37
RELROM (Relocate GNSS Module ROM) 37
WRROM (Enable Writeable GNSS Module ROM) 37

10. Direct Communication Functions 38

GPEEK (Read GNSS Control Register) 38
GPOKE (Write GNSS Control Register) 38
GRDA (Read GNSS Inbound Buffer as Aplha) 38
GRDH (Read GNSS Inbound Buffer as Hex) 39
GWRA (Write GNSS Outbound Buffer as Alpha) 39
GWRH (Write GNSS Outbound Buffer as Hex) 39

11. Error Conditions 40

12. Internal Details 42

String Buffer Control 46
String Buffer Data 47
String Buffer Claim 47
String Buffer Release 47
GNSS Control/Status 48
ROM Control 51

Scratch 1 51
Scratch 2 51
Inbound Buffer Control/Status 52
Inbound Buffer Data 53
Outbound Buffer Control/Status 54
Outbound Buffer Data 55

13. Revision History 56

Introduction

The Global Navigation Satellite System (GNSS) Module adds functionality previously unavailable in a 41C system, and is compatible with the 41C, 41CV, 41CX and 41CL. This module provides position information, including latitude, longitude and altitude, course information, including speed and heading, as well as accurate time and date information.

The GNSS Module is housed in a 3-D printed double-length module housing that can be plugged into any Port of the calculator. When plugged into Port 3, the module allows the Card Reader to still be used with the calculator.

Like the Card Reader and Time Module, the GNSS Module uses a fixed page address, independent of its physical location in the calculator. Where the Card Reader uses page 14 and the Time Module uses page 5, the GNSS Module uses page 15 by default.

If the page 15 address is inconvenient the ROM in the GNSS Module can be relocated to any available page under software control. In addition, the ROM can be disabled and the software run from an external source, such as a QRAM device (MLDL2K, Clonix, NoV-64K, etc), or an internal image in the 41CL.

The ROM in the GNSS Module is implemented using a RAM that is loaded with the GNSS Software when the module is first plugged in. Normally this RAM is read-only, but writes can be enabled via a function in the GNSS Software. This allows the software to be patched or even completely replaced. However, any patching or replacement is only active as long as the GNSS Module is powered.

The GNSS receiver used in the GNSS Module supports 56 channels and has the following specifications:

Horizontal Position Accuracy	< 2.5m
Velocity Accuracy	< 0.1 m/s
Heading Accuracy	< 0.5'
Time to First Fix	29s
Sensitivity (Tracking)	-162dBm
Sensitivity (Acquisition)	-148 dBm
Maximum Altitude	< 50,000m
Maximum Velocity	< 1,000 knots
Operating/Storage Temperature	-40'C ~ +85'C

The Time to First Fix number assumes an open sky, so if you are in an urban environment or inside of a building the time may be significantly longer. Once the receiver has built its

almanac (containing the approximate orbital data for all satellites) the time required for a fix is usually significantly reduced.

The GNSS Software provides keyboard or program access to individual GNSS data as well as the means to store waypoints in 41C registers. The GNSS Software in the GNSS Module is grouped into seven categories:

1. The Receiver Control functions allow you to turn the GNSS receiver on or off and verify the current status of the GNSS receiver.

2. The Individual GNSS Data functions read the selected value from the latest position fix and write this information to the stack.

3. The Waypoint functions are a convenient way to store GNSS data directly to 41C registers. A waypoint requires a minimum of two and a maximum of eight registers.

4. The Waypoint Housekeeping functions allow you to control where the GNSS waypoints are stored in the 41C register memory, and provide a way to communicate this information to other programs that might use the waypoint data.

5. The Miscellaneous functions select the display format for some of the Individual GNSS Data functions and provide format conversions.

6. The ROM Option functions provide control over the ROM in the GNSS Module.

7. The Direct Communication functions allow direct access to the serial port that communicates with the GNSS receiver.

Batteries and Current Drain

While the GNSS Reciever is active, it requires about the same amount of current as a Wand during a barcode scan or a Card Reader during a card read or write. The table below shows the details.

GNSS receiver state	Module Current drain
Standby	750 µA
Active (Acquisition)	47 mA
Active (Tracking)	37 mA

While a fresh set of batteries should last for several hours, it is best not to leave the GNSS Receiver active for any longer than necessary. The GNSS software provides a number of power-control options for the GNSS receiver that will be detailed below.

The GNSS Receiver is in the Standby state when the module is first inserted into the calculator, and also when the calculator is off unless overridden by user command.

The GNSS Receiver enters the Active (Acquiring) state when first turned on. Typically the receiver will require about 30 seconds to acquire sufficient GNSS information to allow the automatic transition to the Active (Tracking) state.

In the Active (Tracking) state the GNSS Receiver updates the position information in the buffers internal to the GNSS Module once every second. This information can then be accessed at any time using functions in the GNSS software. Functions are providied to query the state of the GNSS receiver.

The GNSS Receiver supports other power-saving modes, which can further reduce the power requirements, but this requires special control messages to initiate. Refer to the Ublox documentation (UBX-13003704) for the details about this type of operation. These special messages can be issued using the the **GWRA** and **GWRH** functions.

To support the different use cases the GNSS software provides five different functions to control the power for the GNSS receiver.

Function	Operation	Low Batt	Normal Execute	Calc Off
GNAUTO	Automatic	Standby & beep	On	Standby
GNON	On	Standby & beep	On	On
GNAON	Always On	On	On	On
GNSBY	Standby	Standby	Standby	Standby
GNOFF	Off	Off	Off	Off

The Automatic mode is enabled by the **GNAUTO** function. In this mode the GNSS receiver is active until the calculator is turned off, at which point the GNSS receiver is automatically placed in Standby. Note that because of the way that automatic shut-down feature of the 41C calculator is designed, there is no way to notify a peripheral like the GNSS Module that an automatic shut-down has occurred. This means that in this case the GNSS receiver will remain powered after the shut-down.

The **GNAUTO** function turns the GNSS receiver on unless the ***BAT*** annunciator is already on. This function checks the battery status before, but not after, the GNSS receiver is turned on. This means that if the batteries are weak the GNSS receiver will still be turned on but the ***BAT*** annunciator may be activated at the same time. Then the next time the battery condition is tested (which normally happens after every keypress) the GNSS receiver will enter the Standby state and the calculator will beep to signal to the user that this state change has occurred.

The On mode is enabled by the **GNON** function. The mode is identical to the Automatic mode except that the GNSS receiver will remain on even when the calculator is turned off.

The Always On mode is enabled by the **GNAON** function. In this mode the GNSS receiver remains on even if a low-battery condition is detected.

The Standby mode is enabled by the **GNSBY** function. This is the default mode, selected when the GNSS Module is first inserted in the calculator and after a ***MEMORY LOST*** condition. In this mode the GNSS receiver is in Standy by mode but the remainder of the GNSS Module remains active.

The Off mode is enabled by the **GNOFF** function. In this mode the GNSS receiver is in Standy by mode and the remainder of the GNSS Module is disabled. In this mode the GNSS Module will still respond to reads and writes but no direct communication with the GNSS receiver is possible.

GNSS Data Formats

The GNSS Receiver in the GNSS Module outputs position information as serial strings of ASCII characters according to the NMEA-0183 (National Maritime Electronics Association) standard. Circuitry in the GNSS Module converts these serial strings into bytes and buffers the bytes prior to making them available to the Individual GNSS Data functions and Waypoint functions.

The NMEA protocol defines specific formats for the various data that make up a GNSS position according to the table below:

GNSS data	Native Data Format	Units	Range
Latitude	ddmm.mmmmm	dd: degrees mm.mmmmm: minutes	-90 ≤ degrees ≤ +90 00.00000 ≤ minutes ≤ 59.99999
North/South	character		N: North Latitude, S: South Latitude
Longitude	dddmm.mmmmm	ddd: degrees mm.mmmmm: minutes	-180 ≤ degrees ≤ +180 00.00000 ≤ minutes ≤ 59.99999
East/West	character		E: East Longitude, W: West Longitude
Altitude	decimal number	meters	< 50,000 m
Heading	decimal number	degrees	0 ≤ degrees < 360 clockwise from due North
Speed	decimal number	knots	0 ≤ knots < 1,000
UTC	hhmmss.ss	hh: hours mm: minutes ss.ss: seconds	00 ≤ hours ≤ 23 00 ≤ minutes ≤ 59 00.00 ≤ seconds ≤ 59.99
UTC Date	ddmmyy	dd: day mm: month yy:year	01 ≤ day ≤ 31 01 ≤ month ≤ 12 00 ≤ year ≤ 99
Satellites used	integer		0 ≤ satellites used ≤ 12
Satellites in View	integer		0 ≤ satellites in view ≤ 32
Geoidal Separation	decimal number	meters	refer to WGS-84 for details
HDOP	decimal number		see below
VDOP	decimal number		see below

While the GNSS Receiver is starting up not all GNSS position information becomes valid at the same time. For example, time and date require a signal from only one satellite. Horizontal position requires information from three satellites, while altitude requires four satellites. Even then, depending on the relative location of the satellites, the position information may be inaccurate. The table below shows progression of valid position information.

Satellites used	Fix type	Valid position	precision measurement
0	Invalid	none	
1-2	1D	time, date only	
3	2D	latitude, longitude	HDOP
≥ 4	3D	latitude, longitude, altitude	HDOP, VDOP

The HDOP (Horizontal Dilution of Precision) and VDOP (Vertical Dilution of Precision) numbers are measures of the precision of the position information:

DOP	precision
≤ 1	ideal
1 - 2	excellent
2 - 5	good
5 - 10	moderate
10 - 20	fair
≥ 20	poor

HDOP and VDOP can be combined to provide an overall Position Dilution of Precision (PDOP) using the formula:

$$PDOP = (HDOP^2 + VDOP^2)^{1/2}$$

To mitigate potential problems with other programs interpreting GNSS data, before storing the data the GNSS Software always reformats the latitude, longitude, heading, UTC Time and UTC Date into a format that makes it easy to use with existing 41C software.

The latitude, longitude and heading are stored as signed decimal degrees in the standard 41C floating point number format. Following accepted convention, North latitude is positive and South latitude is negative. Similarly, East longitude is positive and West longitude is negative.

The signed decimal degrees latitude, longitude and heading values that are stored can be converted to degrees-minutes-tenths (dd.mmmmmm) using the **DMT** function or to degrees-minutes-seconds (dd.mmssss) using the regular 41C **HMS** function.

The UTC Time and UTC Date values are reformated into the same formats used by the Time Module for time and date. UTC Time is always reported in 24 hour format, and UTC Date is always reported in the DMY format.

The table below shows the formats used for the stored GNSS data, including waypoints.

GNSS data	Stored Data Format	Stored Units	Stored Data Range
Latitude	standard 41C number	degrees	-90 ≤ degrees ≤ +90
Longitude	standard 41C number	degrees	-180 ≤ degrees ≤ +180
Altitude	standard 41C number	meters	< 50,000 m
Heading	standard 41C number	degrees	0 ≤ degrees < 360
Speed	standard 41C number	knots	0 ≤ speed < 1,000
UTC	HH.MMSShh	HH: hours MM: minutes SS: seconds hh:hundredths of seconds	00 ≤ hours ≤ 23 00 ≤ minutes ≤ 59 00 ≤ seconds ≤ 59 00 ≤ hundredths ≤ 99
UTC Date	dd.mmyyyy	dd: day mm: month yyyy:year	01 ≤ day ≤ 31 01 ≤ month ≤ 12 year ≤ 2099

With the exception of UTC Time and UTC Date, all values are stored as standard 41C floating point numbers that contain a 10-digit mantissa and a 2-digit exponent. However, the precision of individual GNSS data values is determined by the GNSS Receiver, which is based on a u-blox 7 receiver. Refer to the u-box 7 receiver specification for details about the precision of individual GNSS data values.

There is a boundary condition for longitude values (±180 degrees) that is not defined in the GNSS Receiver specification. If you ever cross the International Date Line with your GNSS Module active, please let us know how this boundary condition is handled!

Receiver Control Functions

The GNSS Receiver Control functions allow you to turn the GNSS receiver on or off and verify the current status of the GNSS receiver.

GNAUTO

Executing **GNAUTO** (*GNSS Receiver Automatic Mode*) places the GNSS Receiver in Active mode. In this mode the receiver will attempt to track satellites and update the position information.

In the Active mode the current drain on the calculator batteries is significant, but a fresh set of alkaline batteries should last for several hours of continuous use. This function checks the battery condition before turning on the GNSS receiver, and if a low battery condition is present, the function will merely beep and exit, without turning on the GNSS receiver. If the GNSS receiver was already on when this function is executed and a low battery condition is detected the function will beep and the GNSS receiver will be placed in Standby mode to reduce current drain.

After entering Active mode, the GNSS Receiver typically requires about 30 seconds to acquire the satellite signals and update the position information. During this time invalid position information is reported using the Alpha string ***INV***.

In this mode the GNSS receiver is active until the calculator is turned off, at which point the GNSS receiver is automatically placed in Standby. Note that because of the way that automatic shut-down feature of the 41C calculator is designed, there is no way to notify a peripheral like the GNSS Module that an automatic shut-down has occurred. This means that in this case the GNSS receiver will remain powered after the shut-down.

GNON

Executing **GNON** (*GNSS Receiver On*) places the GNSS Receiver in Active mode. This mode is identical to the Automatic mode selected by the **GNAUTO** function, except that the GNSS receiver will not automatically be placed in Standby mode when the calculator is turned off.

GNAON

Executing **GNAON** (*GNSS Receiver Always On*) places the GNSS Receiver in Active mode, independent of the state of the battery. In this mode the GNSS receiver also remains in Active mode when the calculator is turned off.

GNSBY

Executing **GNSBY** (*GNSS Receiver in Standby*) places the GNSS Receiver in Standby mode. In this mode the current drain on the calculator batteries is significantly reduced, because the receiver is neither tracking satellites nor updating the position information.

The GNSS receiver is automatically placed in Standby mode when the module is first inserted into the calculator and by a ***MEMORY LOST*** condition. While the GNSS receiver is in Standby mode all position information is reported using the Alpha string ***INV***.

This command does not turn off the other logic in the GNSS Module.

GNOFF

Executing **GNOFF** (*GNSS Receiver Off*) places the GNSS Receiver in Standby mode, as well as disabling some of the logic in the GNSS Module itself. In this mode the GNSS Module will still respond to reads and writes but no direct communication with the GNSS receiver is possible because the oscillator that clocks the serial communication with the GNSS receiver is disabled.

GNACT?

Executing **GNACT?** (*Test GNSS Receiver Active Status*) tests the power state of the GNSS receiver, returning with ***NO*** in the display if the GNSS Receiver is in Standby mode and ***YES*** in the display when the GNSS Receiver is in Active mode. When used in a program, if the GNSS receiver is in Active mode the next program line will be executed; if the GNSS Receiver is in Standby mode the next line in the program is skipped.

GNTRK?

Executing **GNTRK?** (*Test GNSS Receiver Tracking Status*) tests the tracking state of the GNSS Receiver, returning with ***NO*** in the display if the GNSS Receiver is in Standby mode or not yet tracking and ***YES*** in the display when the GNSS Receiver is in Active mode and tracking. When used in a program, if the GNSS Receiver is tracking the next program line will be executed; if the GNSS Receiver is in Standby mode or not yet tracking the next line in the program is skipped.

The table below shows the various combinations for these two GNSS Receiver tests.

GNSS receiver state	**GNACT?**	**GNTRK?**
Standby	***NO***	***NO***
Active (Acquisition)	***YES***	***NO***
Active (Tracking)	***YES***	***YES***

The state of the GNSS Receiver is reported based on the status in the "Mode 2" field of the GPGSA NMEA string. Hardware in the GNSS Module automatically extracts the information in this field from the output of the GNSS Receiver.

Individual GNSS Data Functions

The Individual GNSS Data functions retrieve and display individual components of a GNSS position fix. The data displayed is not read directly from the GNSS output buffers, but must first be transferred to the Datum Buffer internal to the GNSS Module. This is because individual components of the GNSS position can change between successive function calls.

The Datum Buffer is only updated by command and retains its contents for as long as the GNSS Module is powered. The Datum Buffer consists of normal 41C registers, that exist outside of the normal 41C register address space. Like all other 41C registers, the registers in the Datum Buffer can hold a floating point number or a short Alpha string. Valid entries are always floating point numbers and invalid entries are marked with the Alpha string ***INV***.

All of the data retrieval functions return the data to the X-register, even if the data is the Alpha string ***INV***. In addition, the primary data retrieval functions (for latitude, longitude, altitude, heading, speed, time and date) can also return the data to the Alpha register and the display.

This ALPHA/display feature allows information about the units to be appended to the data. This feature is enabled by default, but can be disabled using the **GMODE** function with the ***NOA*** (No Alpha) mode identifier.

Writing data to the display does not occur in program mode, even if enabled, because in program mode the display is updated by the operating system to show program execution.

DATUM

Executing **DATUM** (*Generate GNSS Datum*) reads the latest position output from the GNSS Receiver, extracts all of the relevant information, and transfers it to the Datum Buffer in the GNSS Module. This information is then available for the Individual GNSS Data functions.

If the GNSS Receiver is in Standby mode, or is not supplying a valid fix, the **DATUM** function writes the Alpha string ***INV*** to every register in the Datum Buffer. This is a convenient way to initialize the Datum Buffer, and is also automatically done when the GNSS Module is first inserted into the calculator.

The GNSS Receiver has internal range checks on all of the position information that it outputs, and if the information does not pass these checks, nothing is output. The

DATUM function will mark any missing information with the ***INV*** Alpha string in the Datum Buffer.

If the GNSS Receiver is supplying a 1D fix, only the UTC Time and UTC Date are valid, and the registers holding the UTC Time and UTC Date are updated by this function. All other registers in the Datum Buffer are written with an ***INV*** Alpha string.

If the GNSS receiver is supplying a 2D fix, the Altitude and VDOP are not valid, so the corresponding registers in the Datum Buffer are written with an ***INV*** Alpha string.

If the GNSS receiver is supplying a 3D fix, all of the entries in the Datum buffer should be valid, although the accuracy of each entry depends on the number of satellites in view as well as the position of the satellites relative to the GNSS receiver.

The Speed and Heading components of a position fix may be marked with an ***INV*** Alpha string even when all other position information is available, because Heading can only be determined given some minimum amount of speed.

Note that the Waypoint functions also update the Datum Buffer before storing a waypoint, so these functions can also write an ***INV*** Alpha string as part of a waypoint.

LAT

Executing **LAT** (*Retrieve Latitude*) reads the latitude information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The latitude information is always stored in the Datum Buffer in decimal degree format, but the value returned by this function can be decimal degrees, degree-minute-second or degree-minute-tenths. These options are selected using the **GMODE** function. Latitude is positive for North and negative for South.

The latitude is also returned in the ALPHA register and the display (Run mode only) by default. This option can be disabled using the **GMODE** function. The table below shows the various cases.

	Latitude Format	
GNSS Mode	Display and ALPHA	X-register
DEG (default)	***±DD.DDDDDD***	***±DD.DDDDDD***
DMS	***DD:MM:SS.SS N/S***	***±DD.MMSSSS***
DMT	***DD:MM.MM N/S***	***±DD.MMMMMM***

LON

Executing **LON** (*Retrieve Longitude*) reads the longitude information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The longitude information is always stored in the Datum Buffer in decimal degree format, but the value returned by this function can be decimal degrees, degree-minute-second or degree-minute-tenths. These options are selected using the **GMODE** function. Latitude is positive for East and negative for West.

The longitude is also returned in the ALPHA register and the display (Run mode only) by default. This option can be disabled using the **GMODE** function. The table below shows the various cases.

	Longitude Format	
GNSS Mode	Display and ALPHA	X-register
DEG (default)	***±DDD.DDDDDD***	***±DDD.DDDDDD***
DMS	***DDD:MM:SS.SS E/W***	***±DDD.MMSSSS***
DMT	***DDD:MM.MM E/W***	***±DDD.MMMMMM***

ALTI

Executing **ALTI** (*Retrieve Altitude*) reads the MSL (Mean Sea Level) altitude information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The altitude information is always stored in the Datum Buffer in meters, but the value returned by this function can be meters or feet. These options are selected using the **GMODE** function.

The altitude is also returned in the ALPHA register and the display (Run mode only) by default. This option can be disabled using the **GMODE** function. The table below shows the two cases.

	Altitude Format	
GNSS Mode	Display and ALPHA	X-register
M (default)	***NNNNNN M***	***NNNNNN***
FT	***NNNNNN FT***	***NNNNNN***

HEADING

Executing **HEADING** (*Retrieve Heading*) reads the heading (direction of travel) information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The heading information is always stored in the Datum Buffer in decimal degree format, and this is the value returned by this function. In other words, the **GMODE** function for degree format does not affect the heading value.

The heading is also returned in the ALPHA register and the display (Run mode only) by default. This option can be disabled using the **GMODE** function. The table below shows the various cases.

	Heading Format	
GNSS Mode	Display and ALPHA	X-register
any	***DDD.DDDDDD***	***DDD.DDDDDD***

SPEED

Executing **SPEED** (*Retrieve Speed*) reads the speed information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The speed information is always stored in the Datum Buffer in knots, but the value returned by this function can be knots, miles-per-hour, kilometers-per-hour, feet-per-second or meters-per-second. These options are selected using the **GMODE** function.

The speed is also returned in the ALPHA register and the display (Run mode only) by default. This option can be disabled using the **GMODE** function. The table below shows the various cases.

	Speed Format	
GNSS Mode	Display and ALPHA	X-register
KT (default)	***NNNNNN KT***	***NNNNNN***
MPH	***NNNNNN MPH***	***NNNNNN***
KM/H	***NNNNNN KM/H***	***NNNNNN***
M/S	***NNNNNN M/S***	***NNNNNN***
FT/S	***NNNNNN FT/S***	***NNNNNN***

UTIME

Executing **UTIME** (*Retrieve UTC Time*) reads the time information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The UTC Time in the Datum Buffer is always in the form of hh.mmssss where hh is hours, mm is minutes and ssss is seconds and hundredths of a second. UTC Time is always in the same format used by the Time Module in the **CLK24** format.

In Run mode, the UTC Time is also returned in the display and the ALPHA register by default. The table below shows the various cases.

	UTC Time Format	
GNSS Mode	Display and ALPHA	X-register
any	***HH:MM:SS UTC***	***HH.MMSSSS***

UDATE

Executing **UDATE** (*Retrieve UTC Date*) reads the date information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The UTC Date in the X-register in the Datum Buffer is always in the form of dd.mmyyyy where dd is day, mm is month and yyyy is year. This is the same format used by the Time Module when the **DMY** format is used.

In Run mode, the UTC Date is also returned in the display and the ALPHA register by default. The table below shows the various cases.

	UTC Date Format	
GNSS Mode	Display and ALPHA	X-register
any	***DD/MM/YYYY***	***DD.MMYYYY***

SATS

Executing **SATS** (*Retrieve Count of Satellites Used*) reads the "satellites used in positioning calculation" information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The number of satellites used will range from 0 to 12. In general, the higher the number of satellites used, the higher the quality of the position information, but the best measure of the quality of the position information are the HDOP and VDOP values.

SATIV

Executing **SATIV** (*Retrieve Count of Satellites In View*) reads the "satellites in view" information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

The number of satellites in view will range from 0 to 32. In general, the higher the number of satellites in view, the higher the quality of the position information, but the best measure of the quality of the position information are the HDOP and VDOP values.

GEOID

Executing **GEOID** (*Retrieve Geoidal Separation*) reads the geodial separation information from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

Geoidal separation is always reported in meters. Refer to WGS-84 for detailed information about what geoidal separation means.

HDOP

Executing **HDOP** (*Retrieve Horizontal Dilution of Precision*) reads the HDOP value from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

HDOP is a dimensionless measure of the quality of the reported latitude and longitude. Refer to the "GNSS Data Formats" section for an explanation of what a particular HDOP value means.

VDOP

Executing **VDOP** (*Retrieve Vertical Dilution of Precision*) reads the VDOP value from the Datum Buffer and writes it to the X-register. The stack is lifted if Stack Lift is enabled.

VDOP is a dimensionless measure of the quality of the reported altitude. Refer to the "GNSS Data Formats" section for an explanation of what a particular VDOP value means.

Waypoint Functions

The Waypoint functions are a convenient way to store a snapshot of GNSS data directly to 41C registers. The 41C registers used for waypoint data are selected either by the Waypoint Pointer Register (WPR) or the contents of the X register, depending on the mode selected. The default mode is to use the WPR.

A waypoint requires a minimum of three, and a maximum of eight, registers. The number of registers used for a waypoint is determined by the function used to store the waypoint, and the waypoint is tagged with an alpha string that identifies the contents of the waypoint unless you disable the tagging feature. Waypoint data is just normal floating point numbers or an ***INV*** Alpha string.

The format of Waypoint data is always identical to the format used in the Datum Buffer: Decimal degrees for latitude, longitude and heading; meters for altitude; knots for speed.

Waypoint data is always stored in a specific order, as shown in the table below. The starting register for a waypoint is either the contents of the WPR or the register address (*bbb*) in the X-register when waypoint function is executed. The default operation includes the alpha tag in the first register, but this can be disabled to save register space if you are always storing the same type of waypoint. This option is completely separate from the Waypoint functions.

	Register address, relative to WPR or *bbb*							
function	+0	+1	+2	+3	+4	+5	+6	+7
AC	"AC"	Altitude	Heading	Speed				
ACT	"ACT"	Altitude	Heading	Speed	Time			
ACTD	"ACTD"	Altitude	Heading	Speed	Time	Date		
CT	"CT"	Heading	Speed	Time				
CTD	"CTD"	Heading	Speed	Time	Date			
LL	"L"	Latitude	Long.					
LLA	"LA"	Latitude	Long.	Altitude				
LLAC	"LAC"	Latitude	Long.	Altitude	Heading	Speed		
LLACT	"LACT"	Latitude	Long.	Altitude	Heading	Speed	Time	
LLACTD	"LACTD"	Latitude	Long.	Altitude	Heading	Speed	Time	Date
LLC	"LC"	Latitude	Long.	Heading	Speed			
LLCT	"LCT"	Latitude	Long.	Heading	Speed	Time		
LLCTD	"LCTD"	Latitude	Long.	Heading	Speed	Time	Date	
LLT	"LT"	Latitude	Long.	Time				
LLTD	"LTD"	Latitude	Long.	Time	Date			

The different options for the data in a waypoint were selected to support different use cases. For example, if you're on a boat (and not in a lock) you likely don't care about altitude. But if you're in an aircraft you probably should. And the date is only marginally useful in many cases.

Certain GNSS information is not available individually in a waypoint. Latitude and longitude are always reported together, as are heading and speed. Altitude, time and date are reported individually.

Waypoints can be stored in 41C data memory as long as sufficient registers are available. In addition, waypoint data can be easily transferred from data memory to X-Memory or Mass Storage as either individual files or as a combined file.

Executing a Waypoint function automatically updates the Datum Buffer before writing the waypoint data to 41C registers, so that the individual pieces of position information are then available for Individual GNSS Data functions without needing to execute the **DATUM** function.

When using the WPR to hold the register address, after the waypoint is written the WPR is automatically incremented to point at the next available register for a waypoint. You can also use the **WPR+X** function to insert or remove data from a waypoint.

AC (optional register number *bbb* in X-register)

The **AC** (*Store Altitude/Course Waypoint*) function stores the alpha waypoint identifier "AC", plus the GNSS Altitude, Heading and Speed information into four consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by four.

ACT (optional register number *bbb* in X-register)

The **ACT** (*Store Altitude/Course/Time Waypoint*) function stores the alpha waypoint identifier "ACT", plus the GNSS Altitude, Heading, Speed and Time information into five consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by five.

ACTD (optional register number *bbb* in X-register)

The **ACTD** (*Store Altitude/Course/Time/Date Waypoint*) function stores the alpha waypoint identifier "ACTD", plus the GNSS Altitude, Heading, Speed, Time and Date information into six consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by six.

CT (optional register number *bbb* in X-register)

The **CT** (*Store Course/Time Waypoint*) function stores the alpha waypoint identifier "CT", plus the GNSS Heading, Speed and Time information into four consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by four.

CTD (optional register number *bbb* in X-register)

The **CTD** (*Store Course/Time/Date Waypoint*) function stores the alpha waypoint identifier "CTD", plus the GNSS Heading, Speed, Time and Date information into five consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by five.

LL (optional register number *bbb* in X-register)

The **LL** (*Store Position Waypoint*) function stores the alpha waypoint identifier "L", plus the GNSS Latitude and Longitude information into three consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by three.

LLA (optional register number *bbb* in X-register)

The **LLA** (*Store Position/Altitude Waypoint*) function stores the alpha waypoint identifier "LA", plus the GNSS Latitude, Longitude and Altitude information into four consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by four.

LLAC (optional register number *bbb* in X-register)

The **LLAC** (*Store Position/Altitude/Course Waypoint*) function stores the alpha waypoint identifier "LAC", plus the GNSS Latitude, Longitude, Altitude, Heading and Speed information into six consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by six.

LLACT (optional register number *bbb* in X-register)

The **LLACT** (*Store Position/Altitude/Course/Time Waypoint*) function stores the alpha waypoint identifier "LACT", plus the GNSS Latitude, Longitude, Altitude, Heading, Speed and Time information into seven consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by seven.

LLACTD (optional register number *bbb* in X-register)

The **LLACTD** (*Store Position/Altitude/Course/Time/Date Waypoint*) function stores the alpha waypoint identifier "LACTD", plus the GNSS Latitude, Longitude, Altitude, Heading, Speed, Time and Date information into eight consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by eight.

LLC (optional register number bbb in X-register)

The **LLC** (*Store Position/Course Waypoint*) function stores the alpha waypoint identifier "LC", plus the GNSS Latitude, Longitude, Heading and Speed information into five consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by five.

LLCT (optional register number *bbb* in X-register)

The **LLCT** (*Store Position/Course/Time Waypoint*) function stores the alpha waypoint identifier "LCT", plus the GNSS Latitude, Longitude, Heading, Speed and Time information into six consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by six.

LLCTD (optional register number *bbb* in X-register)

The **LLCTD** (*Store Position/Course/Time/Date Waypoint*) function stores the alpha waypoint identifier "LCTD", plus the GNSS Latitude, Longitude, Heading, Speed, Time and Date information into seven consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by seven.

LLT (optional register number *bbb* in X-register)

The **LLT** (*Store Position/Time Waypoint*) function stores the alpha waypoint identifier "LT", plus the GNSS Latitude, Longitude, and Time information into four consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by four.

LLTD (optional register number *bbb* in X-register)

The **LLTD** (*Store Position/Time/Date Waypoint*) function stores the alpha waypoint identifier "LTD", plus the GNSS Latitude, Longitude, Time and Date information into five consecutive data registers, starting with the register selected either by the WPR or the X-Register. When using the WPR the register number in the WPR is then incremented by five.

Waypoint Housekeeping Functions

The Waypoint Housekeeping functions allow you to control where the GNSS waypoints are stored in the 41C register memory, and provide a way to communicate this information to other programs that might use the waypoint data.

Two dedicated registers are implemented inside the 41C GNSS Module to hold pointers to the registers involved in waypoints. The first is called the *Waypoint Base Register* (WBR), which is used to point to the start of the very first waypoint. The second is called the *Waypoint Pointer Register* (WPR), which is used to point to the start of the **next** waypoint to be stored.

When the 41C GNSS Module is first inserted both the WBR and WPR are initialized to zero, selecting R_{00} as the first register to be used for waypoint data. Normally the WBR and WPR will be initialized to point at the same register.

If you plan to use the Navigation module to perform calculations using waypoint data, remember that this module uses R_{00} through R_{53} to hold intermediate data, so in this case both the WBR and WPR should be manually initialized to point at R_{54} or higher.

If you choose not to use the WBR and WPR, you can also specify the starting register for waypoint data with a three-digit number in the X-Register. But this requires you to do any buffer management in software.

PTRP

Executing **PTRP** (*Use Waypoint Pointer Register*) enables the WPR to be used as the register pointer by subsequent Waypoint functions. This selection is stored in the GNSS Module as long as the module is inserted in the calculator. This is the default selection when the GNSS Module is first inserted into the calculator.

PTRX

Executing **PTRX** (*Use X-Register as pointer*) enables the X-Register to be used as the register pointer by the Waypoint functions. This selection is stored in the GNSS Module as long as the module is inserted in the calculator.

RCLWBR

Executing **RCLWBR** (*Recall Waypoint Base Register*) copies the WBR contents to the X-register. The stack is lifted if Stack Lift is enabled.

The contents of the X-register can then be used directly, using stack-indirect addressing, to fetch the first stored waypoint register. There is no check that waypoint data has ever been written, so that should be done before attempting to access waypoint data.

STOWBR (register number *rrr* in X-register)

Executing **STOWBR** (*Store Waypoint Base Register*) writes the contents of the X-register to the WBR. The contents of the X-register are unaffected by this operation.

The WBR is initialized with 0 when the 41C GNSS Module is first inserted in the calculator. The WBR is used only for reference to the start of the waypoint data, and is never modified by the GNSS Software.

RCLWPR

Executing **RCLWPR** (*Recall Waypoint Pointer Register*) copies the WPR contents to the X-register. The stack is lifted if Stack Lift is enabled.

Because the WPR points at the start of the **next** waypoint, the contents of the X-register will need to be modified before it can be used, using stack-indirect addressing, to fetch any stored waypoint data. There is no check that waypoint data has ever been written, so that should be done before attempting to access waypoint data.

STOWPR (register number *rrr* in X-register)

Executing **STOWPR** (*Store Waypoint Pointer Register*) writes the contents of the X-register to the WPR. The contents of the X-register are unaffected by this operation.

The WPR is initialized with 0 when the 41C GNSS Module is first inserted in the calculator. The WPR is automatically incremented after a waypoint is stored, to point at

the next available register.

WPR+X (offset *bbb* in X-register)

Executing **WPR+X** (*Add X to Waypoint Pointer Register*) adds the contents of the X-register to the WPR. The contents of the X-register are unaffected by this operation.

If the value in the X-register is positive, this creates available slots, either at the beginning of the **next** waypoint, or at the end of the **current** waypoint, that can be used to hold auxiliary information in the waypoint. If the value in the X-register is negative, this removes entries at the end of the **current** waypoint. No range checks are done before or after this addition.

RCLRNG

Executing **RCLRNG** (*Recall Waypoint Register Range*) reads the WBR and WPR and forms a number of the form *bbb.eee* which is written to the X-register. The stack is lifted if Stack Lift is enabled.

This number can be used directly by the **SAVERX** function (in the *41C Extended Functions*) to write all of the waypoint data to X-Memory or by the **WRTRX** function (in the *HP-IL Module*) to write all of the waypoint data to mass storage.

If there is no valid waypoint data, because integer part of WPR is less than or equal to the integer part of WBR, the X-register is written with 999.999. This value will then generate an error if you attempt to use it with either the **SAVERX** function or the **WRTRX** function.

Miscellaneous Functions

The Miscellaneous functions provide a conversion from decimal degrees and set and query the mode control options for the GNSS Software.

DMT

Executing **DMT** (*Convert to Degrees-Minutes-Tenths*) converts the decimal degrees in the X-register to dd.mmmmmm format in the X-register. The sign of the value is preserved. No range check is performed prior to this conversion. There is an implicit decimal point after the first two digits of the minutes field.

To convert from decimal degrees to degree-minute-second format, use the built-in 41C **HMS** function.

GMODE (mode selects in ALPHA register)

Executing **GMODE** (*Set GNSS Mode*) sets the GNSS operating mode according to the mode identifiers in the ALPHA register. Mode identifiers must be separated by a space and can be in any order. The mode identifiers are scanned from right to left in the ALPHA register, and the ALPHA register is cleared by this function. These modes are stored within the GNSS Module and are retained as long as the module is powered.

The ***DEF*** mode identifier can be used to reset the GNSS Module to the default state. Placing this identifier in the rightmost position in the ALPHA register will guarantee that only the other modes you specify will be modified from the default.

All Individual GNSS Data functions return data to the X-register, but a number of these functions can also return the data, with a units identifier, to the display and the ALPHA register by default. This feature can be disabled using the ***NOA*** mode identifier.

Functions	Identifier	Meaning	default
Individual GNSS Data Functions	***ALP*** ***NOA***	Written to ALPHA/display Not written to ALPHA/display	***ALP***

All Waypoint functions normally include an alpha identifier stored with each waypoint. This option can be disabled using the ***RAW*** mode identifier, which saves space, but means that there will be no indication of exactly what information is contained in the waypoint or how many registers the waypoint is using.

Functions	Identifier	Meaning	default
Waypoint Functions	***TAG*** ***RAW***	Tag each waypoint No waypoint tag	***TAG***

None of the modes below that select units affect waypoint data. The format of Waypoint data is ALWAYS identical to the format used in the Datum Buffer: Decimal degrees for latitude, longitude and heading; meters for altitude; knots for speed. This is a concious design decision. Remember the Mars Climate Orbiter? Be extremely careful calculating with data from Individual GNSS Data functions if you ever change the units for those functions! The tables below show the mode identifiers available for the different cases.

Individual GNSS functions that return latitude or longitude default to decimal degrees. Heading is always reported in decimal degrees, independent of the degrees identifier.

Functions	Identifier	Meaning	default
LAT **LON**	***DEG*** ***DMS*** ***DMT***	decimal degrees degree-minute-second degree-minute-tenths	***DEG***

The **ALTI** function returns the altitude information in meters by default.

Functions	Identifier	Meaning	default
ALTI	***M*** ***FT***	meters feet	***M***

The **SPEED** function returns the speed information in knots by default.

Function	Identifier	Meaning	default
SPEED	***KT*** ***MPH*** ***KM/H*** ***M/S*** ***F/S***	knots miles/hour kilometers/hour meters/second feet/second	***KT***

Different use cases for the GNSS Module will need different mode settings. For example, if you are in an automobile in the US, you will probably want to know your speed in MPH and your altitude in feet:

"MPH FT"
XEQ "GMODE"

In an automobile elsewhere in the world, you will probably want your speed in KM/H:

"KM/H"
XEQ "GMODE"

An aircraft will usually want altitude in feet and latitude/longitude in DMT:

"FT DMT"
XEQ "GMODE"

GMODE?

Executing **GMODE?** (*Query GNSS Modes*) returns the current set of GNSS modes in the ALPHA register. Modes are returned in a specific order, with each identifier separated by a space.

For example, the default modes will be returned as ***KT DEG M TAG ALP***. Some combinations of modes are too long for the display, and will scroll left.

If you have changed the mode as shown in the **GMODE** examples above, **GMODE?** will return the following:

US automobile: ***MPH DEG FT TAG ALP***

Rest of world automobile: ***KM/H DEG M TAG ALP***

Aircraft: ***KT DMT FT TAG ALP***

GCLOCK

Executing **GCLOCK** (*Enable GNSS Clock Display*) enables the continuous display of the GNSS time, just like the **CLOCK** function does for the Time Module.

This function automatically enables the GNSS receiver. If the GNSS receiver was not already enabled the calculator will enter deep sleep mode and the clock display will not start until a 1-D fix is available.

The clock display is cancelled by pressing any key. Pressing any key except for ON will cause the GNSS receiver to enter Standby mode. Pressing the ON key will cancel the time display but the GNSS receiver will remain active.

If a low battery condition is detected, either while waiting for a 1-D fix, or during the clock display, the GNSS receiver will be disabled and the calculator will beep and turn off.

This function does not update the DATUM buffer.

ROM Option Functions

The ROM Option functions control the operation of the ROM in the GNSS Module.

DISROM

Executing **DISROM** (*Disable GNSS Module ROM*) disables the software internal to the GNSS Module. This function should only be used in cases where some other source of GNSS software is available to be executed in place of the code in the GNSS Module.

ENROM

Executing **ENROM** (*Enable GNSS Module ROM*) enables the software internal to the GNSS Module. This function obviously can only be used in cases when the current source of GNSS Software is external to the GNSS Module itself.

This function is also used to return the internal ROM to read-only mode after having been enabled for writing with the **WRROM** function.

RELROM (page number *p* in X-register)

The **RELROM** (*Relocate GNSS Module ROM*) modifies the address decoder in the GNSS Module to relocate the GNSS Software. Only pages 6-15 are valid relocation targets. This function does check for occupied pages before attempting the relocation, and returns a ***DATA ERROR*** error message if a collision is detected.

WRROM

Executing **WRROM** (*Enable Writeable GNSS Module ROM*) enables the ROM in the GNSS Module for writes. This feature allows patching or even complete replacement of the GNSS Software. **This patching or replacement is only stored as long as power is applied to the module.**

Direct Communication Functions

The Direct Communication functions allow you to directly access the registers in the GNSS Module, including the 512-byte FIFOs connected to the internal serial port.

GPEEK (register number *r* in X-register)

Executing **GPEEK** (*Read GNSS Control Register*) reads the GNSS module register selected by the number in the X-register. The register read is in the 0xF1 peripheral page and the register number can range from 0 through 15.

The read data is returned as 14 hexadecimal digits in the ALPHA register, which replaces any data present in the ALPHA register.

GPOKE (register number *r* in X-register, hex data in ALPHA)

Executing **GPOKE** (*Write GNSS Control Register*) writes the GNSS module register selected by the number in the X-register with the hexadecimal number in the ALPHA register. The register written is in the 0xF1 page and the register number can range from 0 through 15.

The fourteen least-significant digits in the ALPHA register are used, and the ALPHA register is not disturbed by this function. Leading zeros do not need to be present in the hex number in ALPHA.

GRDA

Executing **GRDA** (*Read GNSS Receive Buffer as Alpha*) reads the inbound buffer one time and appends this data as alpha characters, in received byte order, to the ALPHA register. If the ALPHA register is full appended characters push characters off of the most-significant end of the ALPHA register.

The number of characters appended to the ALPHA register ranges from 1 to 4. The number of characters appended to the ALPHA register is added to the contents of the X-register.

GRDH

Executing **GRDH** (*Read GNSS Receive Buffer as Hex*) reads the inbound buffer one time and appends the hex data, in received byte order, to the ALPHA register. If the ALPHA register is full appended bytes push characters off of the most-significant end of the ALPHA register.

Each received character requires two bytes to display in the ALPHA register, so the number of bytes appended to the ALPHA register is 2, 4, 6 or 8. The number of bytes (divided by two, for the number of characters) appended to the ALPHA register is added to the contents of the X-register.

GWRA (character data in ALPHA)

Executing **GWRA** (*Write GNSS Transmit Buffer as Alpha*) writes to the outbound buffer using the contents of the ALPHA register directly. Up to 24 characters (the width of the ALPHA register) will be written by this function.

The characters in the ALPHA register are written so that they will be transmitted starting with the most-significant character first. The characters are removed from the ALPHA register as they are written to the buffer.

The ALPHA register treats the byte 0x00 as special. If you need to write 0x00 bytes to the buffer, it is best to use the **GWRH** function for those bytes.

GWRH (hex data in ALPHA)

Executing **GWRH** (*Write GNSS Transmit Buffer as Hex*) writes to the outbound buffer using the contents of the ALPHA register, which is assumed to hold hexadecimal data. Each transmitted byte requires two characters in the ALPHA register to represent a byte. Up to twelve bytes can be written to the buffer by this function.

The hex data in the ALPHA register is written so that it will be transmitted starting with the most-significant byte first. The hex data is removed, two characters at a time, from the ALPHA register. In the case of an odd number of characters in the ALPHA register, the final character will be written as the most-significant nibble of a byte, with the least-significant nibble of 0.

Error Conditions

The functions in the GNSS Module almost all deal with simply reading data from the GNSS Receiver, and as a result there are not a lot of possible error conditions. The table below lists all possible error messages returned by functions in the GNSS Module, along with the meaning of the message.

Error Message	Function	Meaning
ALPHA DATA	**STOWBR** **STOWPR** **WPR+X** **DMT**	Operation attempted on Alpha data
BFR EMPT	**GRDA** **GRDH**	Inbound buffer is empty
BFR FULL	**GWRA** **GWRH**	Outbound Buffer is full
BFR OV	**GRDA** **GRDH**	Inbound buffer overflowed
DATA ERROR	**STOWBR** **STOWPR** **WPR+X**	Number larger than 999
	GPOKE **GWRH**	Invalid hex number
	GWRA **GWRH**	ALPHA register empty
OUT OF RANGE	**STOWBR** **STOWPR** **WPR+X**	Register does not exist
	GPOKE **GPEEK**	Invalid register select
	All Waypoint functions	Required registers exceed highest numbered data storage register
REL ERROR	**RELROM**	Attempt to relocate ROM to an occupied page
GNSS OFF	**DATUM** All Waypoint functions	GNSS Module in Deep Sleep state

While it is not an error condition, invalid or missing data in a NMEA string will result in an Alpha string ***INV*** being written to the Datum Buffer in place of a number. This Alpha string will also be transferred to a register as part of a Waypoint function. Certain invalid data will affect more than just the individual GNSS data, as will a too-long NMEA string, along with other conditions, according to the table below.

Condition	Datum entries marked as ***INV***
GNSS receiver off	all
RMC string marked Invalid	
Invalid **UTIME**	
Invalid **UDATE**	
RMC string too long	**LAT LON SPEED HEADING**
Invalid **LAT**	
Invalid **LON**	
GGA string too long or missing	**ALTI SATS GEOID**
GSA string too long or missing	**HDOP VDOP**
GSV string too long or missing	**SATIV**

Internal Details

The GNSS Module appears to the MCODE programmer as a set of peripheral registers that can be accessed after the correct peripheral address is selected. The table below shows the peripheral addresses currently used in the 41C ecosystem.

Peripheral Address	Usage
0xF0	41CL on-chip
0xF1	GNSS Module
0xF2	
0xFB	Time Module
0xFC	Card Reader
0xFD	LCD Display
0xFE	Barcode Wand

The table below shows the peripheral registers in page 0xF1 in the GNSS Module that are visible to the MCODE programmer. Writes to read-only registers are ignored. Reads from write-only registers return all zeros. The unimplemented registers always return all zeros on read, and writes to these unimplemented registers are ignored.

Register	Usage	Read/Write	comments
0	String Buffer Control	yes	
1	String Buffer Data	read-only	dual 1536-byte RAMs
2	String Buffer Claim	write-only	
3	String Buffer Release	write-only	
4	GNSS Control/Status	yes	
5	ROM Control	yes	
6	Scratch 1	yes	
7	Scratch 2	yes	
8	Inbound Buffer Control/Status	yes	
9	Inbound Buffer Data	read-only	512-byte FIFO
10	reserved		
11	reserved		
12	Outbound Buffer Control/Status	yes	
13	Outbound Buffer Data	write-only	512-byte FIFO
14	reserved		
15	reserved		

Page 0xF2 contains sixteen general-purpose read/write registers that are used to hold the DATUM information as well as the Waypoint pointer registers, as shown in the table below.

Register	Usage	Source
0	WBR	software
1	WPR	software
2	Time	RMC
3	Date	RMC
4	Latitude	RMC
5	Longitude	RMC
6	Speed	RMC
7	Heading	RMC
8	Position Fix Indicator	GGA
9	Satellites Used	GGA
10	MSL Altitude	GGA
11	Geodial Separation	GGA
12	Mode 2	GSA
13	HDOP	GSA
14	VDOP	GSA
15	Satellites In View	GSV

The receiver in the GNSS Module outputs position information as serial strings of ASCII characters according to the NMEA-0183 standard. All NMEA strings start with the "$" character and end with "<CR><LF>" so they are easy to recognize in hardware. Hardware in the GNSS Module routes the NMEA strings required by the GNSS software to a dedicated string buffer to make it easier for software to find the necessary position information.

The string buffer is 1536 bytes deep, and is divided into twelve 128-byte segments. Each NMEA string is written to a separate segment so that software does not need to search through NMEA strings but can access the desired string directly.

There are actually two string buffers, and the hardware automatically alternates between buffers, which allows the receiver to continue writing position information to a buffer while the software is reading the position information from the other buffer.

When the software wants to read a buffer it "claims" the buffer so that the receiver cannot overwrite the position information, and then "releases" the buffer after all of the data has been read. In the event that the receiver needs to start another buffer while it is locked out of one buffer, the unlocked buffer is automatically reused. This buffer management is completely transparent to the user. There is no need to select which buffer to claim or read.

The hardware switches string buffers when the first NMEA message in a sequence is recognized. For the receiver used in the GNSS Module this is the Recommended Minimum Specific GNSS Data (RMC) string. This string contains the UTC time, status

indicator, latitude, longitude, speed, course, UTC date, a couple of unused fields and a checksum. The GNSS Module software uses all of the fields in this string.

The Global Positioning System Fix Data (GGA) string contains the UTC time, latitude, longitude, position indicator, satellites used, horizontal dilution of precision, MSL altitude, geoidal separation, a couple of unused fields and a checksum. The GNSS Module software only uses the satellites used, MSL altitude and geoidal separation information in this string.

The GNSS Satellites in View (GSV) string contains information about the number and identity of satellites in view by the receiver. Only the information about the number of satellites in view is used by the GNSS module. The receiver can output up to four GSV strings, which are buffered separately in the string buffer.

The GNSS DOP and Active Satellites (GSA) string contains information about the number and identity of satellites used in the position fix, along with information about the dilution of precision for all three directions. This string is used by the hardware in the GNSS module to report the type of fix (2D or 3D) being reported. The receiver can output multiple GSA strings, but only the first and second are buffered in the string buffer.

If there are any other NMEA strings output by the receiver, they are automatically buffered in two uncommitted segments in the string buffer. If there are more than two of these extra strings, they are ignored.

The string buffers are linear (not recirculating) and are random-access as far as the software is concerned. The table below shows how the segments are organized. Note that the string headers (the first six bytes of a string, "$GPRMC" for example) of recognized strings are not written to the string buffer. Only the first bytes (those needed to distinguish an unrecognized header) of an unrecognized string are not written to the string buffer.

segment	Usage
0x0	RMC string
0x1	VTG string
0x2	GLL string
0x3	GGA string
0x4	other #1
0x5	other #2
0x6	GSA string #1
0x7	GSA string #2
0x8	GSV string #1
0x9	GSV string #2
0xA	GSV string #3
0xB	GSV string #4

The string buffer being filled by the receiver is automatically released (so that it can then be claimed by the GNSS Software) when an idle line is detected on the receiver output. For the GNSS Module an idle line is defined as 32 consecutive "1" bits. Since the GNSS Receiver synchronizes its NMEA output strings to an internal one-second tick, the receiver output is guaranteed to go idle after all of the NMEA strings for one update have been sent by the GNSS Receiver. With the default serial data rate of 38400 bits/sec, eighty bytes per string, and ten output strings, the GNSS receiver output will be active for only about 210mS out of each second.

There are two error conditions that will also cause a release of the string buffer. These error conditions should never occur in the GNSS Module, simply because the connection between the GNSS Receiver and the remainder of the logic in the GNSS Module is short and in a benign environment. But if a NMEA string exceeds 80 bytes in length the string buffer will be released and the logic will wait for the next RMC string to again claim a string buffer and start writing the NMEA information. The string buffer will also be released if a second RMC string is received before an idle line has been detected.

A separate buffer, called the inbound buffer, holds all messages from the receiver that are not NMEA strings. This buffer is implemented as a FIFO for both receiver and the software. This buffer can optionally be enabled for NMEA strings, although this option is not recommended because it is difficult for the 41C processor to keep up with the data rate output by the receiver.

There is also an outbound buffer, used for control messages destined for the receiver. This buffer is implemented as a FIFO for both the receiver and the software.

A loopback option is available for testing the GNSS module. In this mode the output of the transmitter is fed back to the receiver, completely bypassing the GNSS receiver. This allows arbitrary NMEA strings to be sent through the receive path.

The register interface to these buffers and the associated control and status are described below. All unused register bits and nibbles will return zero when read, and writes to the unused bits and nibbles are ignored.

String Buffer Control

Register 0 is the String Buffer Control register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
									rw		rw		
0	0	0	0	0	0	0	0	0	**saddr**		**seg**	0	0

Nibbles 4 and 3 are used to address the locations in the selected segment of the String Buffer to be read. These nibbles are read/write. The address of the first received byte in a string is 0x00. Since NMEA strings are limited to eighty bytes, the last byte of a string should be at or before address 0x4F and the remainder of the segment should not contain any valid NMEA data. Bit 7 of the address byte is ignored. The address byte is automatically incremented by seven whenever the String Buffer Data register is read, so that it only needs to be written once, when starting to access a segment in the buffer.

saddr byte	**Usage**
0x00	First received byte (the "," after the string ID)
0x01-0x4F	Subsequent received bytes, in order
0x50-0x7F	Should be empty

Nibble 2 is used to select the segment of the String Buffer to be read according to the table below. This nibble is read/write. Segment numbers greater than 11 (0xB) are invalid and will result in all zeros being read from the buffer.

seg nibble	**Usage**
0x0	RMC string
0x1	VTG string
0x2	GLL string
0x3	GGA string
0x4	unrecognized #1
0x5	unrecognized #2
0x6	GSA string #1
0x7	GSA string #2
0x8	GSV string #1
0x9	GSV string #2
0xA	GSV string #3
0xB	GSV string #4
0xC-0xF	Invalid selection

The two segments reserved for "unrecognized" strings will be used for the first two received strings that do not have a dedicated slot in the string buffer.

String Buffer Data

Register 1 is the String Buffer Data register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
r		r		r		r		r		r		r	
byte 6		**byte 5**		**byte 4**		**byte 3**		**byte 2**		**byte 1**		**byte 0**	

Byte 0 is the first received byte, followed in order by **byte 1**, **byte 2**, **byte 3** and so on. NMEA strings to not have a guaranteed length, but are delimited by <CR><LF> so that software can determine the end of the string. Data returned after <CR><LF> is invalid, because the string buffer is never initialized.

The string headers (the first six bytes of a string, ("$GPRMC" for example) of recognized strings are not written to the string buffer. All header bytes up to the first unrecognized byte of an unrecognized string are not written to the string buffer. For example, a string that starts with "$GPTXT" will buffer the "TXT" portion of the header to the string buffer. This allows software to fully decode any "unrecognized" strings.

String Buffer Claim

Register 2 is the String Buffer Claim trigger. Writing any value to this register claims the most-recently-filled String Buffer for the software. Once a string buffer is claimed the String Buffer Valid bits in the GNSS Control/Status register are valid and the String Buffer Data register can be accessed.

String Buffer Release

Register 3 is the String Buffer Release trigger. Writing any value to this register removes the claim on the String Buffer by the software.

GNSS Control/Status

Register 4 is the GNSS Control/Status register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
						r			r	rw	r	rw	
0	0	0	0	0	0	**segval**			**tid**	**pwr**	**rcv**	**hmd**	0

Nibbles 7-5 are the "valid" status bits for the segments in the claimed string buffer. All twelve of these bits are cleared when the receiver claims a string buffer and then individual valid bits are set when the corresponding string is received. This allows software to handle the case where not every string is received during a GNSS output message. A set of these bits exists for each string buffer, and claiming the string buffer automatically selects the appropriate set of bits. The table below shows which bit corresponds to which segment.

segval bit	Usage
0	RMC string valid
1	VTG string valid
2	GLL string valid
3	GGA string valid
4	unrecognized #1 valid
5	unrecognized #2 valid
6	GSA string #1 valid
7	GSA string #2 valid
8	GSV string #1 valid
9	GSV string #2 valid
10	GSV string #3 valid
11	GSV string #4 valid

Nibble 4 contains the talker ID for the current string buffer. This information is latched during the RMC string for each string buffer and the appropriate set of status is returned after the software claims the string buffer. The table below shows the encoding of this nibble. Only the bit combinations shown will ever be returned.

tid	talker
0100	GPS ($GP string)
0010	other ($GN string)
0001	GLONASS ($GL string)

Nibble 3 controls the various power options for the GNSS Module. Normally the power state of the GNSS Module follows the state of the calculator. When the calculator is off (in the Deep Sleep state) the GNSS Module is also in the Deep Sleep state, to maximize battery life. When the calculator is turned on the GNSS Module is also turned on, although the GNSS receiver remains in Standby until commanded to enter the Active state. This operation can be modified by the two control bits in this nibble.

The first option is to keep the GNSS Module always powered, even when the calculator is turned off. This will ensure that the GNSS receiver continues to track position information, at the expense of battery life. This option is necessary for the GNSS Module interrupt (to display the time) to work properly.

If the GNSS Module is not being used, but will remain inserted in a Port, an option is available to keep the GNSS Module in the Deep Sleep state independent of the state of the calculator. In this state the **DATUM** function and all Waypoint functions will return an error when executed, but Individual GNSS Data functions will continue to return the (stale) contents of the Datum Buffer.

The final option keeps the GNSS Module always powered, independent of the Low Battery warning.

pwr[3:2]	Usage	Function
00	Auto power-down (default)	***GNAUTO*** ***GNSLP***
01	On	***GNON***
10	Disable Clock (Deep Sleep)	***GNOFF***
11	Always On	***GNAON***

Bits [1:0] are used as shown below. These are command bits which can initiate action; they are always read as 0x00. By default, the receiver is automatically placed in Standby mode while the calculator is in the Deep Sleep state, unless the "Always Powered" option is selected as described above.

pwr[1:0]	Usage (write)
00	No effect
01	No effect
10	Enter Active State
11	Enter Standby State

Nibble 2 reports the current state of the GNSS receiver as shown below. After having been commanded to enter the active state the receiver will automatically transition from the Active (Acquiring) state to the Active (Tracking) state once sufficient satellite signals have been received. As mentioned previously, the GSA string Mode 2 field is extracted

by the hardware to report the receiver state in this field.

rcv	Usage (read)
0000	Standby (default)
0001	Active (Acquiring)
0010	Active (2D fix)
0100	Active (3D fix)

Nibble 1 contains hardware mode controls. Bits [3:1] control the serial data rate to and from the GNSS Receiver. **This selection does not program the GNSS Receiver, only the hardware in the GNSS Module.** A special control message to the GNSS Receiver is required to change the serial data rate used by the receiver.

hmd[3:1]	Usage	GMODE identifier
000	38400 (default)	***384***
001	38400 with loopback	***LP***
100	19200	***192***
101	9600	***96***
110	4800	***48***

Bit 0 of nibble 1 controls the GNSS Module interrupt. When enabled, the GNSS Module will wake up the calculator from Deep Sleep or Light Sleep when the following three conditions are all met: the RMC string Status field is "A" signifying data valid, the GSA string Mode 2 field is "3" signifying a 3D fix, and the receiver has released the string buffer. The interrupt is cleared by disabling the interrupt.

The GNSS Module interrupt is reported using the Flag input to the CPU during the nibble 3 time slot on the 41C bus.

ROM Control

Register 5 is the ROM Control register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
											rw	rw	
0	0	0	0	0	0	0	0	0	0	0	**rel**	**rm**	0

Nibble 2 relocates the ROM from its default page address of 0xF. Only 0x6 through 0xF are valid page addresses.

rel	**Usage**
0x0-0x5	Illegal
0x6-0xE	Page 6 - E
0xF	Page F (default)

Nibble 1 controls the operation of the ROM. Only bits [3:2] are used. By default the ROM is read-only, but this control nibble allows the GNSS Software to be patched or completely replaced, if necessary. Use the WROM instruction to write to the ROM.

rm[3:2]	**Enable**	**Mode**
00	enabled	read-only
01	enabled	read/write
1x	disabled	n/a

Scratch 1

Register 6 is a general-purpose read-write register.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Scratch 2

Register 7 is a general-purpose read-write register.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Inbound Buffer Control/Status

Register 8 is the Inbound Buffer Control register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
								r			r	r/w	w
0	0	0	0	0	0	0	0	**ifill**			**ist**	**im**	**cli**

Nibbles 5, 4 and 3 report the fill level of the inbound buffer, which is in the range of zero (0x000) through 512 (0x200). The reported fill level is frozen at the start of a read of this register, so there may actually be one more byte in the buffer by the time the fill level is reported.

ifill	Usage (rd-only)
0x000-0x200	Bytes available
0x201-0xFFF	illegal

Nibble 2 reports the empty/overflow status of the inbound buffer. If the buffer overflows only the last byte in the buffer is overwritten, but once software starts reading data anything after 511 bytes cannot be considered valid and the buffer should be purged.

ist	Usage (rd-only)
0000	Buffer empty
0001	Buffer not empty
1000	Buffer overflowed, is now empty
1001	Buffer overflowed, is not empty

Nibble 1 controls what is routed to the inbound buffer. Only bit [3] is used.

im[3]	Usage
0	No NMEA data (default)
1	All data to inbound buffer

Nibble 0 is used to purge the inbound buffer. Only bit [0] is used. A purge should only be necessary if the buffer overflows because the CPU is not able to keep up with incoming data. This nibble is always read as 0x0.

cli[0]	Usage (wr-only)
0	No Effect
1	Purge Inbound Buffer

Inbound Buffer Data

Register 9 is the Inbound Buffer Data register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
			r		r		r		r		r		
0	0	0	**byte 3**		**byte 2**		**byte 1**		**byte 0**		**val**	0	0

Byte 0 is the first received byte, followed in order by **byte 1**, **byte 2** and **byte 3**.

Nibble 2 reports the number of valid bytes being transferred. Only bits [2:0] are used. A maximum of four bytes are transferred with each read of this register so there are only five possible bit combinations returned for this status. This status is read-only.

val[2:0]	**Usage (rd-only)**
000	Buffer empty
001	Only byte 0 valid
010	Only bytes 1-0 valid
011	Only bytes 2-0 valid
100	Four bytes valid

Outbound Buffer Control/Status

Register 12 is the Outbound Buffer Control register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
								r			r	rw	w
0	0	0	0	0	0	0	0	**ofill**			**ost**	**hld**	**clo**

Nibbles 5, 4 and 3 report the empty level of the inbound buffer, which is in the range of zero (0x000) through 512 (0x200). The reported empty level is frozen at the start of a read of this register, so there may actually be one byte less in the buffer by the time the empty level is reported.

ofill	Usage (rd-only)
0x000-0x200	Bytes empty
0x201-0xFFF	illegal

Nibble 2 reports the status of the outbound buffer. Software is responsible for not overflowing the buffer.

ost	Usage (rd-only)
0000	Buffer Empty
0001	Buffer Not Empty
1000	Buffer Full

Nibble 1 is used to hold off sending data from the outbound buffer to the receiver, if a complete message needs to be assembled in the buffer. Only bit [3] is used.

hld[3]	Usage
0	No hold-off (default)
1	Hold transmission

Nibble 0 is used to purge the outbound buffer. Only bit [0] is used. A purge of the outbound buffer should never be necessary in practice. This nibble is always read as 0x0.

clo[0]	Usage (wr-only)
0	No Effect
1	Purge Outbound Buffer

Outbound Buffer Data

Register 13 is the Outbound Buffer Data register:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
			w		w		w		w		w		
0	0	0	**byte 3**		**byte 2**		**byte 1**		**byte 0**		**val**	0	0

Byte 0 is the first transmitted byte, followed in order by **byte 1**, **byte 2** and **byte 3**.

Nibble 2 controls the number of valid bytes being transferred. Only bits [2:0] are used. A maximum of four bytes are transferred with each write of this register so there are only four possible bit combinations allowed. This nibble is always read as 0x0.

val[2:0]	**Usage (wr-only)**
000	No data written
001	Only byte 0 valid
010	Only bytes 1-0 valid
011	Only bytes 2-0 valid
100	Four bytes valid

Revision History

05/14/2018	Preliminary #5
11/14/2018	Preliminary #6
02/21/2019	Added Direct Communication Functions and GCLOCK. Changes to low-level interface (valid string buffer block bits, loopback for testing.)
03/01/2019	Added low-battery operation explanations
03/27/2019	Modified GCS register.

