

GPS140HR GPS Positioning Module

User's Manual



RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

GPS140HR

GPS Positioning Module

User's Manual



RTD Embedded Technologies, INC.

103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087

FAX: +1-814-234-5218

E-mail

sales@rtd.com

techsupport@rtd.com

web site

<http://www.rtd.com>

Revision History

12/08/2000 HW Release 1.1, Preliminary version, released 18/08/2000
Rev. A New manual naming method

Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION	8
Features	8
Some of the key features of the GPS140HR include:.....	8
GPS receiver.....	8
16C550 compatible UART.....	9
I/O interfaces.....	9
Mechanical description.....	9
Connector description.....	9
What comes with your board.....	9
Using this manual	10
When you need help.....	10
CHAPTER 2 - BOARD SETTINGS	11
Factory-Configured Jumper Settings.....	12
Base address jumpers (Factory setting: 2E8h)	13
Host interrupt (Factory setting: IRQ5, G closed).....	15
GPS configuration (Factory setting: ROM disabled, NMEA).....	16
CHAPTER 3 BOARD INSTALLATION	18
Board installation	18
General installation guidelines:.....	18
GPS receiver connector	19
Utility I/O header connector J3.....	21
CHAPTER 4 - HARDWARE DESCRIPTION	22
Antenna.....	23
UART channel	24
Pulse outputs	24
1 PPS output and LED.....	24
10KHz output	24
Fuses	25
CHAPTER 5 BOARD OPERATION AND PROGRAMMING	26
Defining the Memory Map.....	26
INTERRUPTS	27
What is an interrupt?	27
Interrupt request lines	27
8259 Programmable Interrupt Controller.....	27
Interrupt Mask Register (IMR).....	28
End-of-Interrupt (EOI) Command.....	28
What exactly happens when an interrupt occurs?.....	28

Using Interrupts in your Program.....	28
Writing an Interrupt Service Routine (ISR).....	29
Your ISR should have the following structure:.....	30
Saving the Startup Interrupt Mask Register (IMR) and interrupt vector.....	30
Common Interrupt mistakes.....	31
Example on Interrupt vector table setup in C-code:.....	32
CHAPTER 6 - GPS140HR SPECIFICATIONS.....	33
Host interface.....	33
GPS Receiver specifications.....	33
Operational.....	33
RF Signal environment.....	33
Environmental.....	33
UART.....	33
GPS140HR Electromechanical.....	34
CHAPTER 7 - RETURN POLICY AND WARRANTY.....	35
Return Policy.....	35
CHAPTER 8 LIMITED WARRANTY.....	37

List of Illustrations & Tables

.....	Illustrations
Fig. 2-1 GPS140HR Board layout showing jumper locations	
Fig. 2-2 Base address jumpers illustrating address 3F8h	
Fig. 2-3 Interrupt jumpers from left to right: IRQ 2,5,7,10,11,12,15 and G	
Fig. 2-4 GPS receiver jumper blocks for configuration	
Fig. 2-5 Antenna selection jumper block set for +5V antenna supply	
Fig. 3-1 GPS140HR integrated in a RTD PC/104 cpuModule stack together with a HPWR104 and a CMM series cpuModule	
Fig. 3-2 DGPS data input header connector	
Fig. 3-3 Utility connector	
Fig. 4-1 Block diagram of the GPS140HR	
Fig. 4-2 1 PPS output indicator LED	
.....	Tables

Table 2-1 Factory configured jumper settings

Table 2-2 Base address jumper settings GPS140HR

Table 2-3 GPS receiver configuration jumpers

Table 2-3 GPS receiver configuration jumpers

Table 3-1 Pin outs of the GPS receiver interface connector

Table 3-2 Pin out of DGPS input header connector

Table 3-3 Utility connector pin out

Table 5-1 General I/O map of the GPS140HR

Chapter 1 - INTRODUCTION

This user's manual describes the operation of the RTD GPS140HR integrated global positioning system (GPS) positioning module ideal for mobile, marine, aviation and automotive applications.

Features

Some of the key features of the GPS140HR include:

- Direct connection to GPS Receiver module
- True NMEA-0183 data protocol version 2.01, also binary output
- 12-satellite parallel tracking GPS receiver with fast response
- Differential GPS support with external correction source
- Onboard UART with flexible I/O and IRQ selection
- Supports passive and active antennas with +5V or +12V supply
- OSX/MSX antenna connector
- 1 PPS time mark output as well as 10KHz clock
- Backup battery to store GPS receiver data while powered off
- Status LED indicating 1 PPS time mark and GPS activity
- Wide operating temperature range -40 to + 85C
- Fully PC/104 compliant

The following paragraphs briefly describe the major features of the GPS140HR. A more detailed discussion is included in Chapter 4 (Hardware description) The boards installation is described in Chapter 2 (Board Installation).

GPS receiver

The GPS140HR Global Positioning System (GPS) board uses for the satellite signal reception data processing a miniature OEM GPS receiver module specially designed for the most demanding applications. This low power receiver outputs ASCII-character based NMEA-0183 data protocol or alternatively a binary message protocol format. The used data protocol can be selected with a jumper. Special features of the GPS receiver include 12 channel parallel-tracking channels and a fast acquisition and reacquisition response. Powerful algorithms ensure reliable operation in dense vegetation or in urban canyon environments.

16C550 compatible UART

Communication to the GPS receiver module is performed through a standard UART channel. This onboard serial port leaves the other system serial ports free for the user. All operating systems will recognize and support this 16C550 standard UART, and therefore no special communication drivers are needed to receive data from your GPS receiver. The address and interrupt of your serial channel can be changed with the onboard jumpers.

I/O interfaces

The GPS140HR can be controlled and monitored from the software through the dedicated serial port of the module. A special I/O connector is available for the user to connect to the GPS 1 PPS time mark as well as a precise 10KHz timing clock signal. Also the active antenna supply of +12V can be fed through the I/O connector.

Mechanical description

The GPS140HR is designed on a PC/104 form factor. An easy mechanical interface to both PC/104 and RTD DAN systems can be achieved. Stack your GPS140HR directly on a PC/104 compatible CPU module using the onboard mounting holes and standoffs.

Connector description

The GPS receiver antenna interface is an OSX type miniature coaxial connector. Connect your antenna directly to the GPS140HR antenna connector, or use a short cable inside your enclosure to connect to a feed through connector to allow connection of the antenna to the wall of your enclosure. All I/O connections are made using header type terminals.

What comes with your board

Your GPS140HR package contains the following items:

- GPS140HR board
- Software disk with some example programs
- User's manual

Additional software and drivers can be downloaded from our website.

If any item is missing or damaged, please send an EMAIL to Real Time Devices Finland sales service department at <sales@rtdfinland.fi>.

Note that RTD embedded Technologies, Inc. also can offer a GPS104HR starter kit that will include an active antenna with ready cables for direct evaluation and testing of this module. The part number for this starter kit is **SK-GPS140HR**.

Using this manual

This manual is intended to help you install your new GPS140HR module and get it working quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in the most demanding applications.

When you need help

This manual and all the example programs will provide you with enough information to fully utilize all the features on this board. If you have any problems installing or using this board, contact our Technical support department at <techsupport@rtdfinland.fi>. When sending us an Email request please include the following information: Your company's name and address, your name, your telephone number, and a brief description of the problem.

Chapter 2 - BOARD SETTINGS

The GPS140HR board has jumper settings, which can be changed to suit your application and host computer configuration. The factory settings are listed and shown in the diagram at the beginning of this chapter. Make sure you completely study and understand this chapter before making changes to these settings.

Factory-Configured Jumper Settings

Table 2-1 below illustrates the factory jumper setting for the GPS140HR. Figure 2-1 shows the board layout of the GPS140HR and the locations of the jumpers. The following paragraphs explain how to change the factory jumper settings to suit your specific application.

Table 2-1 Factory configured jumper settings (Please see figure 2-1 below for more detailed locations)

JUMPER NAME	DESCRIPTION	NUMBER OF JUMPERS	FACTORY SETTING
BASE	Base Address	6	2E8
IRQ	Host interrupt	11+1	5, G – jumper closed
CONF	GPS configuration	3	NMEA output
X1	Antenna supply	3	5V antenna cable

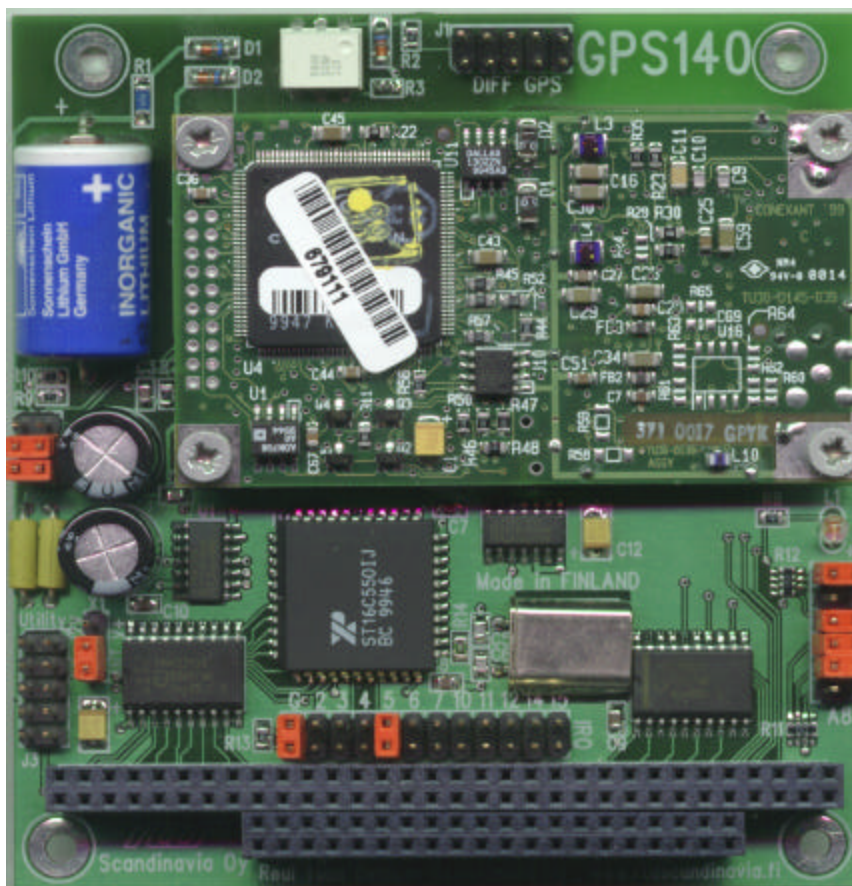


Fig. 2-1 GPS140HR Board layout showing jumper locations

Base address jumpers (Factory setting: 2E8h)

The GPS140HR is I/O mapped into the memory space of your host XT/AT. The board occupies a consecutive memory window of 8 bytes starting from the base address.

The most common cause of failure when you are first setting up your module is address contention: some of your computer's I/O space is already occupied by other devices and memory resident programs. When the GPS140HR attempts to use its own reserved memory addresses (which are being already used by another peripheral device) erratic performance may occur and the data read from the board may be corrupted.

To avoid this problem make sure you set up the base address by using the six jumpers on the right side of the board, this allows you to choose from a number of different addresses in your host computer's I/O map. Should the factory installed setting of 38fh be incompatible to your system configuration, you may change this setting to another using the options illustrated in Table 2-2 (overleaf). The table shows the jumper settings and their corresponding values in hexadecimal form. Ensure that you verify the correct location of the base address jumpers. When the jumper is removed it corresponds to a logical "0", connecting the jumper to a "1". When you set the base address of the module, record the setting inside the back cover of this manual.

BASE ADDRESS JUMPER SETTINGS GPS140HR

BASE (HEX)	A8 A7 A6 A5 A4 A3
218	0 0 0 0 1 1
238	0 0 0 1 1 1
258	0 0 1 0 1 1
278	0 0 1 1 1 1
298	0 1 0 0 1 1
2B8	0 1 0 1 1 1
2D8	0 1 1 0 1 1
2F8	0 1 1 1 1 1
318	1 0 0 0 1 1
338	1 0 0 1 1 1
358	1 0 1 0 1 1
378	1 0 1 1 1 1
398	1 1 0 0 1 1
3B8	1 1 0 1 1 1
3D8	1 1 1 0 1 1
3F8	1 1 1 1 1 1

Table 2-2 Base address jumper settings GPS140HR

0 = JUMPER OFF

1 = JUMPER CLOSED

Note that this table shows decoding only the 4 high bits of the address.

Address jumpers A4 and A3 can be used for addresses not listed, example:

Addr 2E8 = 011101.

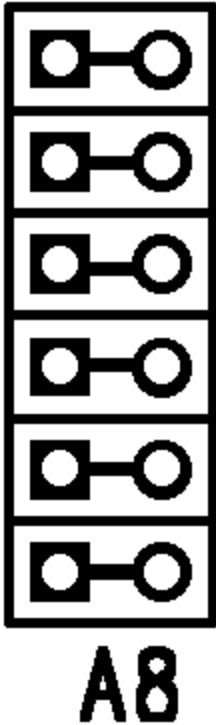


Fig. 2-2 Base address jumpers illustrating address 3F8h, A8 is to the bottom, A3 is located to the top of the jumper block

Host interrupt (Factory setting: IRQ5, G closed)

The header connector, shown in Figure 2-3 below, lets you connect the onboard control logic interrupt outputs to one of the interrupt channels available on the host computer XT/AT bus.

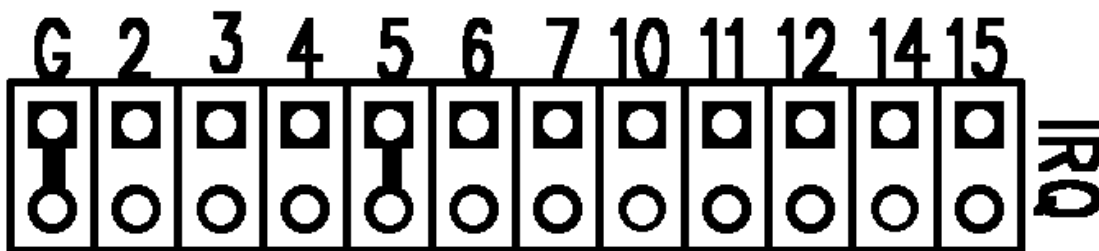


Fig. 2-3 Interrupt jumpers from left to right: IRQ 2,3,4,5,7,10,11,12,14,15 and G

Note: The GPS140HR hardware supports interrupt sharing! Jumper G must be closed on one module per used interrupt. For example if two boards share interrupt number 7 only one board may have the G jumper closed. The G jumper connects a 1KOhm resistor to ground while the shared interrupts are 3-stated pulling the line to an inactive level.

GPS configuration

(Factory setting: ROM disabled, NMEA)

The three jumper blocks illustrated below let you configure the mode of operation of your GPS receiver module. The topmost jumper is the GPS receiver hardware reset, the ROM jumper configures whether settings from the EEPROM are used during startup or not, NMEA jumper sets the protocol mode of the receiver. Table 2-3 describes in detail the operation of these jumpers. Note that if you change the jumper settings for ROM or NMEA you should perform a reset of the receiver by closing the contacts of the RST-jumper for the changes to take effect.

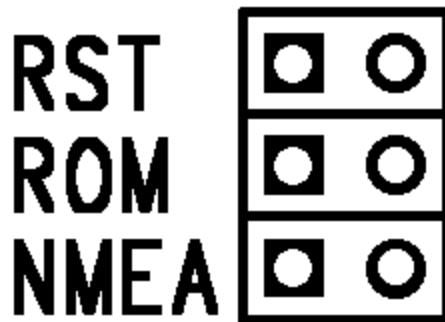


Fig. 2-4 GPS receiver jumper blocks for configuration

NMEA JUMPER	ROM JUMPER	RESULT
0	0	NMEA message format, communication at 4800bps, NO parity, 8 data, 1 stop. The receiver operates from default initialization values stored in ROM and will output default NMEA message set from ROM
0	1	NMEA message format, communication at 4800bps, NO parity, 8 data, 1 stop. The receiver selects the default NMEA output message set and uses initialization values from the data stored in SRAM or EEPROM
1	0	Binary message format, communication at 9600bps, NO parity, 8 data, 1 stop. The receiver operates from default initialization values stored in ROM
1	1	Data stored in SRAM or EEPROM determines message format, host port communication settings, and default message set (Zodiac NMEA,9600bps,N,1)

Table 2-3 GPS receiver configuration jumpers, 0 = jumper OFF, 1 = jumper closed

Antenna supply **(Factory setting: +5V for active antenna)**

Active GPS antennas require a supply to drive the antenna. Typical standard supply voltages are +5V and +12V. Note that in case you use a passive antenna the supply selection jumper must be removed. Driving a passive antenna may cause permanent damage to the antenna.

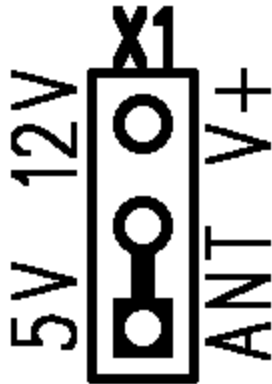


Fig. 2-5 Antenna supply selection jumper block set for +5V antenna supply

Chapter 3 **BOARD INSTALLATION**

The GPS140HR GPS module is designed to directly mount on top or under your RTD PC/104 cpuModule stack. This chapter tells you step-by-step how to install your GPS140HR into your system.

Board installation

Keep your board in its antistatic bag until you are ready to install it to your system! When removing it from the bag, hold the board at the edges and do not touch the components or connectors. Please handle the board in an antistatic environment and use a **grounded** workbench for testing and handling of your hardware. Before installing the board in your computer, check the power cabling. Failure to do so may cause the power supply unit to malfunction or even cause permanent damage.

General installation guidelines:

- Touch the grounded metal housing of your computer to discharge any antistatic buildup and then remove the board from its antistatic bag.
- Hold the board by the edges and install it in an enclosure or place it on the table on an antistatic surface
- Install your board in your system, and wire the power supply correctly.
- Failure to do so may cause the power supply unit to malfunction or even cause permanent damage to the device.
- Check all wiring connections once and then once more again
- Check the antenna supply voltage and configure X1 correctly
- Connect the GPS antenna to the OSX connector on the receiver module
- Apply power to your system, this will automatically initialize your GPS receiver, a blinking LED indicates activity of the GPS receiver and messages being sent

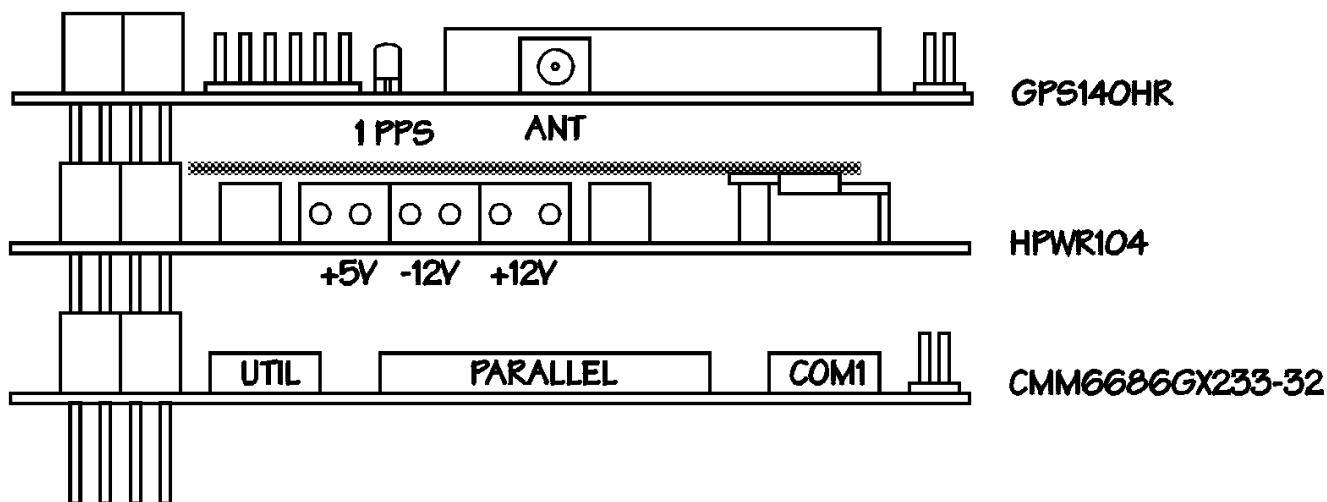


Fig. 3-1 GPS140HR integrated in a RTD PC/104 cpuModule stack together with a HPWR104 power supply module and a CMM series cpuModule

GPS receiver connector

The table 3-1 below shows the pin outs of the GPS receiver interface. This interface is compatible with the majority of GPS receivers in the same form factor. The OEM GPS receiver connects to this 2mm female socket connector with pins facing the PCB.

PIN	Description	PIN	Description
1	Antenna +V	2	GPS +5V supply
3	3,6V battery backup	4	No Connect
5	GPS RESET	6	No Connect
7	NMEA_SEL	8	ROM_SEL
9	No Connect	10	GND
11	SER_OUT	12	SER_IN
13	GND	14	No Connect
15	DIFF_GPS_IN	16	GND
17	GND	18	GND
19	1 PPS	20	10KHz

Table 3-1 Pin outs of the GPS receiver interface connector

Differential GPS signal input connector J1

Differential GPS correction signal can be connected to your GPS140HR for exact precise positioning using the RTCM SC-04 differential signal source. The signal levels on this input are RS232. The connector pinout is shown below in figure 3-2 and table 3-2.

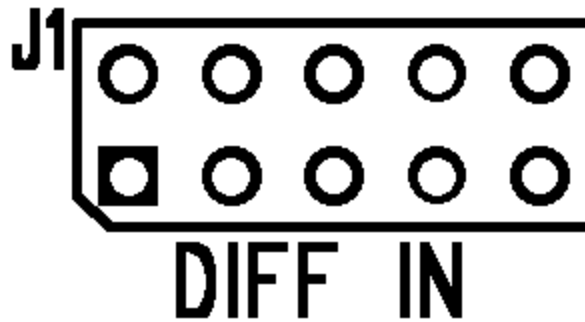


Fig 3-2 DGPS Data input header connector

PIN	DESC	PIN	DESC
1	N.C.	2	N.C.
3	DGPS_IN	4	N.C.
5	N.C.	6	N.C.
7	N.C.	8	N.C.
9	GND	10	GND

Table 3-2 Pin out of DGPS input header connector

Utility I/O header connector J3

External power and I/O signals connect to your GPS140HR through header connector J3. The connector pinout is shown below in figure 3-3 and table 3-3.

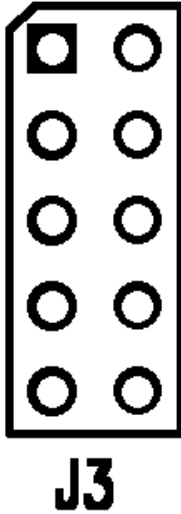


Fig 3-3 Utility connector

PIN	DESC	PIN	DESC
1	+5V	2	+12V for antenna
3	SER_OUT	4	1 PPS output
5	SER_IN	6	10KHz output
7	BATT in 3,6V	8	GND
9	GND	10	GND

Table 3-3 Utility connector pin out

In case you are using a +12V active antenna you must either have a +12V present from the PC/104 bus or you can supply it through pin # 2 of the utility connector. The +3,6V battery input pin # 7 may be used in case you do not use the onboard battery for GPS SRAM backup. Do not supply this pin if the battery is connected. The 1PPS and 10KHz outputs are buffered using HCT04 buffers. The SER_OUT signal present in pin #3 is the TTL-level output of the GSP receiver. You may use this to connect to other devices in your system that may need GPS information.

Chapter 4 - HARDWARE DESCRIPTION

This chapter describes the major hardware building blocks of the GPS140HR:

- ?? The GPS receiver module
- ?? Antenna
- ?? UART channel
- ?? Pulse outputs
- ?? Fuses

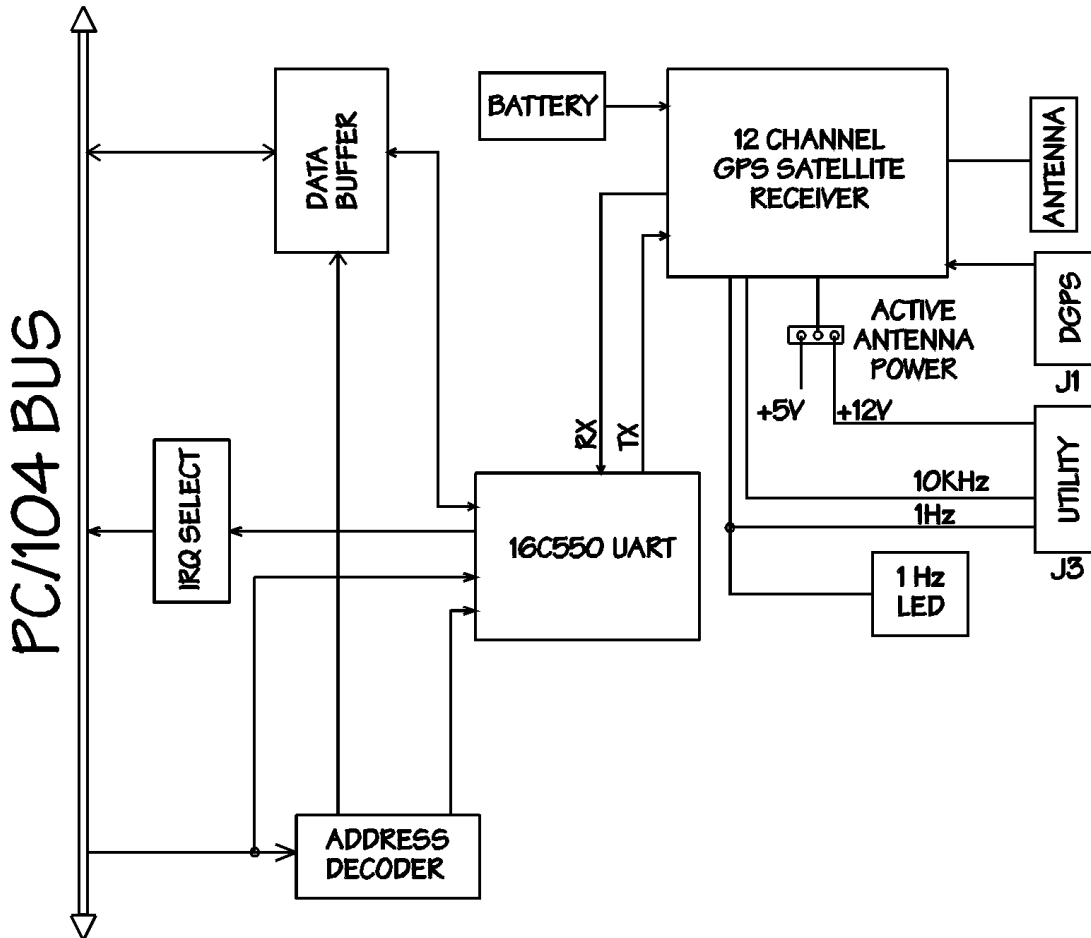


Fig. 4-1 Block diagram of the GPS140HR

The GPS receiver module

A 12-channel parallel-tracking GPS receiver provides fast Time-To-First-Fix (TTFF) under all startup conditions. While the best TTFF performance is achieved when the time of day and current position estimates are provided by the receiver, the flexible signal acquisitions system takes advantage of the available information to provide fast TTFF. Acquisition is guaranteed under all initialization conditions as long as visible satellites are not obscured.

To minimize the TTFF when primary system power is off the GPS140HR maintains the SRAM and the Real Time Clock (RTC) using the onboard battery. In this case the shortest TTFF is achieved by using the RTC time data and prior position data stored in the receiver's SRAM.

The receiver supports 2D operation when less than three satellites are available or when required by operating conditions. Altitude information required for 2D operation is determined by the receiver or may be performed by your application software.

Communication to and from the receiver is performed through a serial channel that is connected to the onboard UART. The receiver's primary serial port outputs navigation data and accepts commands from the OEM application in NMEA-0183 format or Connexant binary message format. The secondary serial port is configured to accept DGPS corrections in the RTCM SC-104 format. A complete detailed description of the serial data interface is contained in the Connexant document: ZODIAC GPS Receiver Family Designer's Guide available on the web at <http://www.conexant.com/>.

The GPS140HR is factory preconfigured to operate in NMEA/0183 format at a datarate of 9600bps.

This manual is not designed to be a GPS handbook, for a complete description on the GPS receiver TU30-D140 visit the receiver manufacturers website, for a complete description on the NMEA protocol you can visit <http://www.nmea.org/>.

Antenna

The GPS receiver can operate from an active or passive GPS antenna to receive L1 band frequency GPS carrier signals. Matsushita Automation Controls produces a complete range of GPS antennae to meet the ultra-compact automotive, heavy-duty marine and extremely accurate survey and timing requirements. The starter kit SK-GPS140HR contains a VIC1-LP low profile antenna. This is a +5V active antenna with a power consumption of 25mA max and it will operate from -40 to +100C. Typical cable connections are made with OSX, SMA and SMB connections. If you connect the antenna directly to your GPS receiver without a feed through connector through your enclosure wall you can select an antenna with an OSX connector to plug into the OSX plug on your receiver module. Visit <http://www.mac-europe.com/> for information on GPS antennae.

UART channel

GPS data is sent to the host from the receiver through a standard 16C550 compatible UART. All today's operating systems will recognize and support this serial communication device. The GPS140HR uses its own onboard serial port and will not reserve serial port resources from the system. The I/O base address and interrupt for this serial port can be flexibly set as has been described in previous chapters of this manual. This user's manual will not wade into details of serial port programming, this information is commonly available today. You can use any communication software package or terminal program to connect to your GPS140HR UART. Just make sure you set up the I/O and IRQ right. ASCII data will start coming out every second from the GPS receiver onto the screen. The UART on the board is specified for full operation from -40 to $+85\text{C}$. The oscillator frequency is set 1.8432MHz.

The UART is connected as a NULL-MODEM device, only the TXD and RXD lines of the UART are connected to the GPS receiver chip.

Pulse outputs

1 PPS output and LED

A 1 PPS buffered TTL level output is generated by the GPS receiver module. This output can be used by your OEM application for timing purposes. When the receiver provides a valid navigation solution, the rising edge of this signal is synchronized with the UTC one-second epochs to within $+300\text{ns}$. The receiver software produces a data message containing the UTC time associated with each time mark pulse. This output signal is connected the 1S LED shown below in figure 4-2.



Fig. 4-2 1PPS output indicator LED

10KHz output

A 10KHz clock-output that is synchronized to the UTC 1PPS output is also generated by the GPS140HR. This signal can also be used for OEM timing purposes. This signal is buffered with a HCT04 buffer.

Fuses

A 2A fuse (yellow) protects the GPS receiver from error conditions. The active antenna is fused with a 125mA fuse (green). These fuses are on the left edge of the board.

Chapter 5 BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your GPS140HR. It provides a general description of the I/O map. Detailed serial port programming tips are not within the scope of this manual.

Defining the Memory Map

The memory map of the GPS140HR occupies eight bytes of host PC I/O space. This window is freely selectable by the user as described in *Chapter 2, Table 2-2*. After setting the base address you have access to the internal resources of the GPS140HR control logic. These resources are not described in detail, since they are mapped as a standard PC serial port. For more details on the EXAR ST16C550J44 UART chip programming please download the component specific data sheet from the manufacturers website: <http://www.exar.com/products/st16c550.html>

ADDR	REGISTER	DIR	COMMENTS
BASE	TXD	O	Only if control reg. Bit 7=0
	RXD	I	Only if control reg. Bit 7=0
	BAUD div. low		Only if control reg. Bit 7=1
BASE+1	BAUD div. high		Only if control reg. Bit 7=1
	IRQ enable		Only if control reg. Bit 7=0
BASE+2	IRQ ID		
BASE+3	Line control		
BASE+4	Modem control		
BASE+5	Line status		
BASE+6	Modem status		

Table 5-1 General I/O map of the GPS140HR

BASE = Base Address

INTERRUPTS

What is an interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are a very flexible way of dealing with asynchronous events. Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time to do nothing whilst waiting for a keystroke to occur. Thus the interrupt scheme is used and the processor proceeds with other tasks. When a keystroke finally occurs, the keyboard then 'interrupts' the processor so that it can get the keyboard data. It then places it into the memory, and then returns to what it was doing before the interrupt occurred. Other common devices that use interrupts are A/D boards, network boards, other used serial ports etc.

Your GPS140HR can interrupt the main processor when a satellite navigation data message is received. You can write powerful code to interface to your GPS140HR when you use interrupts.

Interrupt request lines

To allow different peripheral devices to generate interrupts on the same computer, the PC AT bus has interrupt request channels (IRQ's). A rising edge transition on one of these lines will be latched into the interrupt controller. The interrupt controller checks to see if the interrupts are to be acknowledged from that IRQ and, if another interrupt is being processed, it decides if the new request should supercede the one in progress or if it has to wait until the one in progress has been completed. The priority level of the interrupt is determined by the number of the IRQ as follows; IRQ0 has the highest priority whilst IRQ15 has the lowest. Many of the IRQ's are already used by the standard system resources, IRQ0 is dedicated to the internal timer, IRQ1 is dedicated to the keyboard input, IRQ3 for the serial port COM2, and IRQ4 for the serial port COM1. Often interrupts 2,5,7,10,11 and 15 are free for the user.

8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in a PC is the 8259 Interrupt Controller. To use interrupts you will need to know how to read and set the 8259's internal interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to acknowledge the 8259 interrupt controller.

Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of the interrupt line. If a bit is set (equal to 1), then the corresponding IRQ is masked, and it will not generate an interrupt. If a bit is cleared (equal to 0), then the corresponding IRQ is not masked, and it can then generate an interrupt. The interrupt mask register is programmed through **port 21h**.

End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 Interrupt Controller must be acknowledged by writing the value 20h to port 20h.

What exactly happens when an interrupt occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to correctly write interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the GPS140HR), the interrupt controller checks to see if interrupts are enabled for that IRQ. It then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed onto the system stack, and a new set of CS and IP are loaded from the lowest 1024 bytes of memory.

This table is referred to as the interrupt vector table and each entry to this table is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor starts to execute code from the new Code Segment (CS) and from the new Instruction Pointer (IP). When the interrupt routine is completed, the old CS and IP are popped from the system stack and the program execution continues from the point where interruption occurred.

Using Interrupts in your Program

Adding interrupt support to your program is not as difficult as it may seem especially when programming under DOS. The following discussion will cover programming under DOS. Note that even the smallest mistake in your interrupt program may cause the computer to hang up and will only restart after a reboot. This can be frustrating and time-consuming.

Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write an interrupt service routine (ISR). This is the routine that will be executed automatically each time an interrupt request occurs for the specified IRQ. An ISR is different from other sub-routines or procedures. First on entrance the processor registers must be pushed onto the stack before anything else! Second, just before exiting the routine, you must clear the interrupt on the GPS140HR by writing to the Status register, and write the EOI command to the interrupt controller. Finally, when exiting the interrupt routine the processor registers must be popped from the system stack and you must execute the IRET assembly instruction. This instruction pops the CS, IP and processor flags from the system stack. These were pushed onto the stack when entering the ISR.

Most compilers allow you to identify a function as an interrupt type and will automatically add these instructions to your ISR with one exception: most compilers do not automatically add the EOI command to the function, you must do it yourself. Other than this and a few exceptions discussed below, you can write your ISR as any code routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend you stick to the basics; just something that enables you to verify you have entered the ISR and executed it successfully. For example: set a flag in your ISR and in your main program check for the flag.

Note: If you choose to write your ISR in in-line Assembly, you must push and pop registers correctly and exit the routine with the IRET instruction instead of the RET instruction.

There are a few precautions you must consider when writing ISR's. The most important is, **do not use any DOS functions or functions that call DOS functions from an interrupt routine.** DOS is not re-entrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about using interrupts? Consider then the following situation in your program: If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to the same DOS function X, then function X is essentially being called while active. Such cases will cause the computer to crash. DOS does not support such operations. The general rule is that do not call any functions that use the screen, read keyboard input or any file I/O routines, these should not be used in ISR's.

The same problem of re-entrancy also exists for many floating point emulators. This effectively means that you should also avoid floating point mathematical operations in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you use. Even, if you are writing your ISR in Assembly language, DOS and many floating point emulators are not re-entrant. Of course there are ways to avoid this problem, such as those which activate when your ISR is called. Such solutions are, however, beyond the scope of this manual.

The second major concern when writing ISR's is to make them as short as possible in term of execution time. Spending long times in interrupt service routines may mean that other important interrupts are not serviced. Also, if you spend too long in your ISR, it may be called again before you have exited. This will lead to your computer hanging up and will require a reboot.

Your ISR should have the following structure:

- Push any processor registers used in your ISR. Most C compiler do this automatically
- Put the body of your routine here
- Clear the interrupt bit by reading GPS140HR RXD register
- Issue the EOI command to the 8259 by writing 20h to address 20h
- Pop all registers. Most C compilers do this automatically

The following C example shows what the shell of your ISR should be like:

```

/*-----
| Function:   new_IRQ_handler
| Inputs:    Nothing
| Returns:   Nothing
|-----*/
void interrupt far new_IRQ_handler(void)
{
    IRQ_flag = 1;          // Indicate to main process interrupt has occurred

    {
        // Your program code to read UART data put here
        // read to a data buffer for example:
        Guc_buffer[Gi_bufpos++] = inp(gi_SERIAL_DATA);
    }

    outp(0x20, 0x20);     // Acknowledge the interrupt controller
}

```

Saving the Startup Interrupt Mask Register (IMR) and interrupt vector

The next step after writing the ISR is to save the startup state of the interrupt mask register (IMR) and the original interrupt vector you are using. The IMR is located in address **21h**. The interrupt vector you will be using is located in the interrupt vector table which is an array of pointers (addresses) and it is locate din the first 1024 bytes of the memory (Segment 0 offset 0). You can read this value directly, but it is better practice to use DOS function 35h (get interrupt vector) to do this. Most C compilers have a special function available for doing this. The vectors for the hardware interrupts on the XT - bus are vectors 8-15., where IRQ0 uses vector 8 and IRQ7 uses vector 15. Thus if your GPS140HR is using IRQ5 it corresponds to vector number 13.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read the current IMR at I/O port 21h, and set the bit that corresponds to the IRQ. The IMR is arranged so that bit 0 is for IRQ0 and bit 7 is for IRQ7. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h.

With the startup IMR saved and the interrupts temporarily disabled, you can assign the interrupt vector to point to your ISR. Again you can overwrite the appropriate entry in the vector table with a direct memory write, but this is not recommended. Instead use the DOS function 25h (Set Interrupt Vector) or, if your compiler provides it, the library routine for setting up interrupt vectors. Remember that interrupt vector 8 corresponds to IRQ0, vector 9 for IRQ1 etc.

If you need to program the source of your interrupts, do that next. For example, if you are using transmitted or received messages as an interrupt source program it to do that. Finally, clear the mask bit for your IRQ in the IMR. This will enable your IRQ.

Common Interrupt mistakes

Remember hardware interrupts are from 8-15, XT IRQ's are numbered 0-7. Do not forget to clear the IRQ mask bit in the IMR Forgetting to send the EOI command after ISR code. Disables further interrupts.

Example on Interrupt vector table setup in C-code:

```
void far _interrupt new_IRQ1_handler(void );    /* ISR function prototype */
#define IRQ1_VECTOR    3                      /* Name for IRQ */
void (interrupt far *old_IRQ1_dispatcher)
    (es,ds,di,si,bp,sp,bx,dx,cx,ax,ip,cs,flags);    /* Variable to store old
IRQ_Vector */
void far _interrupt new_IRQ1_handler(void );
```

```
/*-----
| Function:    init_irq_handlers
| Inputs:     Nothing
| Returns:    Nothing
| Purpose:    Set the pointers in the interrupt table to point to
|             our funtions ie. setup for ISR's.
|-----*/
```

```
void init_irq_handlers(void)
{
    _disable();
    old_IRQ1_handler = _dos_getvect(IRQ1_VECTOR + 8);
    _dos_setvect(IRQ1_VECTOR + 8, new_IRQ1_handler);
    Gi_old_mask = inp(0x21);
    outp(0x21,Gi_old_mask & ~(1 << IRQ1_VECTOR));
    _enable();
}
```

```
/*-----
| Function:    restore, do this before exiting program
| Inputs:     Nothing
| Returns:    Nothing
| Purpose:    Restore the interrupt vector table.
|-----*/
```

```
void restore(void)
{
    /* Restore the old vectors */
    _disable();

    _dos_setvect(IRQ1_VECTOR + 8, old_IRQ1_handler);
    outp(0x21,Gi_old_mask);

    _enable();
}
```


Chapter 6 - GPS140HR SPECIFICATIONS

Host interface

16-bit PC/104 bus, XT-bus used

GPS Receiver specifications

Operational

GPS receiver	Connexant Jupiter TU-D140
Update rate	Once per second
Reacquisition	2s typical with a 10s blockage
RTCM SC-104 differential compatibility	Direct connection to RS232 input
Serial data output protocols	NMEA-0183, Connexant binary
Power requirement	+5V, preamplifier passthrough up to +12V for active antenna

RF Signal environment

RF input	1575.42MHz (L1 band) at levels -130dBW and -163dBW
Connector	OSX high retention female conn.
Burnout protection	-10dBW signal with a bandwidth of 10MHz centered about the L1 carrier frequency.

Environmental

Cooling	Convection
Operating temperature	-40 to +85C
Humidity	RH up to 95% non condensing
Altitude	-1000 to 60.000 ft
Maximum Vehicle Dynamic	500m/s acquisition and navigation
Vibration	Survival 18G peak

UART

UART compatibility	16C550
Oscillator frequency	1.8432MHz
Connection	Null Modem
Base addresses	32
Interrupts	2,3,4,5,7,10,11,12,14 and 15

Fuses

GPS receiver
Active antenna

2A
125mA

GPS140HR Electromechanical

Operating temperature range
Humidity
Altitude

-40 to +85 C
RH up to 95% non condensing
-1000 to 30.000 ft

Chapter 7 - RETURN POLICY AND WARRANTY

Return Policy

If you wish to return a product to the factory for service, please follow this procedure:

Read the Limited Warranty to familiarize yourself with our warranty policy.

Contact the factory for a Return Merchandise Authorization (RMA) number.

Please have the following available:

- Complete board name
- Board serial number
- A detailed description of the board's behavior

List the name of a contact person, familiar with technical details of the problem or situation, **along with their phone and fax numbers, address, and e-mail address** (if available).

List your shipping address!!

Indicate the shipping method you would like used to return the product to you.

We will not ship by next-day service without your pre-approval.

Carefully package the product, using proper anti-static packaging.

Write the RMA number in large (1") letters on the outside of the package.

Return the package to:

RTD Embedded Technologies, Inc.

103 Innovation Blvd.

State College PA 16803-0906

USA

Chapter 8 LIMITED WARRANTY

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.

103 Innovation Blvd.

State College PA 16803-0906

USA

Our website: www.rtd.com