DC-4E/GEDC-6E/AHRS-8 Inertial Systems -- Application Note 1001

Backwards Compatibility Considerations for the DC-4E, GEDC-6E, and AHRS-8 inertial systems.

Introduction

Sparton's new line of NorthTek™ enabled inertial systems, starting with the DC-4E, GEDC-6E, and AHRS-8, introduced several new interface options and inertial system features not present in the previous generation of products, the SP3002, SP3003 and SP3004. To make room for some of the new features some slight incompatibilities are present in the NorthTek™ enabled new products with respect to the communications interfaces, protocols and calibration. These differences are described in this document. Also workarounds for most differences are available. In cases where there is no workaround with the inertial system itself, the required workaround on the host system is described, if possible.

Interface compatibility issues between the NorthTek™ Enabled Systems and SP300X products

This is the summary of the differences between the previous generation and new generation products. Each difference will be addressed in detail further in this application note.

SP300X	DC-4E, GEDC-6E, AHRS-8	Notes
At startup the string BX is transmitted, where X is the current baud rate index.	By default, no output.	This feature may be programmed by the user, if desired.
The NMEA repeat option is stored so that the repeat continues after a power cycle.	By default, repeating options are not continued through a power cycle.	This feature may be programmed by the user, if desired.
NMEA repeat is stopped by a single "$.	Not implemented.	Other methods are provided to stop repeating output.
Default baud rate is 9600.	Default baud rate is 115200.	The baud rate, once stored will remain the same through a power cycle. The default baud rate is only an issue for the first time operation if 9600 is desired.
Serial signals are available at logic and TIA232 signal levels.	Serial data is logic level only.	

Field calibration process.	The field calibration process is different in the new products.	
Analog and Digital Inputs/Outputs	Not available.	
Serial Peripheral Interface (SPI)	Not available.	
Update of World Magnetic Model.	Also available, uses a different method.	
Selectable Digital Filter.	Not present.	The new products have more tuning options than the single filter choice.
Factory Code updates	In field code updates.	

Detailed Differences and Workarounds

BX Transmitted at startup

The legacy SP300X compasses output a string containing a capital letter B followed by a digit from 1 to 8 indicating the current baud rate. The DC-4E/ GECD-6E/AHRS-8 units do not output this information by default. However, there is a workaround. The NorthTek™ script below can be sent to the DC-4E/ GECD-6E/AHRS-8 via the user port using a dumb terminal emulator (At 115200 baud you must add 5msec delay per line to the terminal emulator program to avoid overrunning the NorthTek™ compiler). This script will store a program in the User EEPROM boot space. The inertial system will then output the B followed by the baud rate encoding, after every reset or resume from low power mode. This NorthTek™ script is described in detail in another application note.

```
// ************************************************************
// ************************************************************
// Program to enable legacy compass NMEA behavior.
// This file loads a set of forth words into the EEPROM
// that will execute at bootup.
// Needs svn revision 35 or later.
// This program does the following:
// 1) Defines a word call b4. This word prints out a B followed
//    by the baud rate as an index. B4 would be printed
//    if the baud rate is 9600.
// 2) Executes the b4. word so it is printed at startup.
//
// This program is sent to the compass over the serial port.
// Upon reboot the compass will output "BX"<CR><LF>
// Then it will echo the NMEA command and then an OK.
// Then the repeating output will occur at the rate specified.
// ************************************************************
// ************************************************************

// ************************************************************
// Open the user space file.
// ************************************************************
0x10000 userOpen
: put start: userWrite ;  // This makes writing a record easy.
// ************************************************************
// this record prints BX
// start a formatted string, format one character
// then drop the rest.
// Insert the B in front, type it out, then type cr,lf.
// ***********************************************************
put : b4. baud di@ <# # drop char B hold #> type ." \r\n"  ;
// ***********************************************************
// Run the word, note this runs at power up, not when loaded.
// ***********************************************************
put b4.

// ***********************************************************
// Close the user space file
// ***********************************************************
userClose
```

NMEA Repeat Function continues after power cycle

The SP300X compasses retained the last NMEA repeat command if the power was removed and reconnected. The DC-4E/ GECD-6E/AHRS-8 inertial systems do not, by default, output any user data at power up. However the DC-4E/ GECD-6E/AHRS-8 may be programmed to output 1 or more NMEA commands, with or without auto-repeat at power up. The same technique employed in the previous section is employed with the NorthTek™ Program shown below.

```
// ***********************************************************
// ***********************************************************
// Program to enable legacy compass NMEA behavior.
// This file loads a set of forth words into the EEPROM
// that will execute at bootup.
// Needs svn revision 35 or later.
// This program does the following:
// 1) Defines a word call b4.  this word prints out a B followed
//    by the baud rate as an index. B4 would be printed
//    if the baud rate is 9600.
// 2) Executes the b4. word so it is printed at startup.
// 3) Defines a word %, that will stop NMEA repeats.
// 4) Issues a NMEA command with repeat at startup.
//    In this case the command is to output heading with
//    a repeat rate of 0.5 seconds.
//
// This program is sent to the compass over the serial port.
// Upon reboot the compass will output "BX"<CR><LF>
// Then it will echo the NMEA command and then an OK.
// Then the repeating output will occur at the rate specified.
// ***********************************************************
// ***********************************************************

// ***********************************************************
// Open the user space file.
// ***********************************************************
0x10000 userOpen
: put start: userWrite ;  // This makes writing a record easy.
// ***********************************************************
// this record prints BX
// start a formatted string, format one character
// then drop the rest.
// Insert the B in front, type it out, then type cr,lf.
// ***********************************************************
put : b4. baud di@ <# # drop char B hold #> type ." \r\n"  ;
// ***********************************************************
// Run the word
// ************************************************************
put b4.
// ************************************************************
// Define the % word that allow nmea repeat suppress.
// Type %<CR> to execute.
// ************************************************************
put : % " $xxHDM\r\n" count nmea! ;


// ************************************************************
// Issue the NMEA repeat command at startup
// ************************************************************
put " $xxHDM,RPT=0.5\r\n" count nmea!

// ************************************************************
// ************************************************************
// Close the user space file
// ************************************************************
userClose
```

NMEA Repeat Stops with “$” character

As discussed in the previous section, certain multiple repeating NMEA outputs are allowed in the DC-4E/GEDC-6E/AHRS-8. The use of the “$” sign to terminate a previous repeating command would therefore cancel any previous repeat commands before the additional repeat command could be processed. Therefore the DC-4E/GEDC-6E/AHRS-8 inertial systems do not terminate repeating output with the reception of another “$”. Instead, any valid NMEA command without the repeat option or the GLOM option will terminate a repeating output. Note that the DC-4E/GEDC-6E/AHRS-8 inertial systems accept software flow control on the input port to control the flow of data on the output port. Thus a <CTRL-S> character will suspend any repeating output and a <CTRL-Q> will resume the repeating output. The NorthTek™ program in the previous section also programmed a command to cancel NMEA repeat. The user would only need to enter a “%” followed by a carriage return to cancel the NMEA repeat function, if the above script had been stored in the user boot EEPROM.

Default Baud rate is 9600

The default baud rate for the DC-4E/GEDC-6E/AHRS-8 is 115200 bps. The user can change the default baud rate at power up by installing the inertial system in the NDS-1 evaluation kit and selecting the desired baud rate before installing in the end user equipment. Additionally the baud rate may be changed with NMEA (“$PSPA,BAUD=4<CR><LF>”) or NorthTek (“baud 4 set<CR>”).

TIA RS232 Signal Levels

The DC-4E/GEDC-6E/AHRS-8 inertial systems only receive and transmit logic level serial signals (3.3V). To use the new inertial systems the user will have to provide the appropriate level (and polarity) signals to the DC-4E/GEDC-6E/AHRS-8 for logic level serial communication. It should be remembered that logic level serial signals are at different signal levels and polarity than the same EIA232 level signals, e.g. an EIA232 “mark” is -12 volts, but a logic level “mark” is 3.3 volts. Similarly the EIA232 “space” signal level is +12 volts and the logic level “space” voltage is 0 volts.

Field Calibration

Both the legacy and new products provide a method for field calibration. The procedures are similar, but different enough to deserve consideration. The new calibration procedure is described in the Software Interface User’s Manual. Calibration in the new inertial systems may be performed using NMEA commands but when interacting with the inertial system manually, it is generally easier using the NorthTek calibration program shown below. The NorthTek™ script is interactive and prompts the user at each stage and provides the magnetic error feedback during the calibration process. Either the NMEA or NorthTek™ method generates the same accuracy in heading readings. A separate application note describes this NorthTek™ program in detail. For users that have an automated procedure for calibration using NMEA commands, some changes will be required in both message syntax and algorithm.

```
// ***********************************************************
// NorthTek Script for 3d calibration.
// ***********************************************************
// This erases this script should it be reloaded.

forget cal3D


// ***********************************************************
// This is the actual calibration program.
// ***********************************************************
( -- )
: cal3D

  // ***********************************************************
  // Init the calibration process by setting calmode to 1.
  // ***********************************************************

  calmode 1 set


  // ***********************************************************
  // Give the user a heads up that we are starting.
  // ***********************************************************

  ." Calibration starting" cr


  // ***********************************************************
  // Tell the calibration logic to start calibration
  // ***********************************************************

  calCommand cal_start set        // cal start

  // ***********************************************************
  // Give the user some instructions
  // ***********************************************************

  ." Press any key to take next point, ESC to finish" cr
```

```
// **************************************************************
// The code now grabs a point, the user changes the
// compass position and repeats the cal3DState
// from 4-12 times total.
// sit in a loop, taking points until
// the user hits escape.
// User hits spacebar to take a point, ESC to quit.
// **************************************************************

begin
   key 27 = 0=                           // until user enters ESC
while
   // capture a point
   calCommand cal_capture set            // take another point
   200 delay                             // give compass time to capture
   // Print out the current point number
   calNumPoints di.
repeat


// **************************************************************
// Now the points are captured,
// Issue the command to start computing
// the real time cal values
//
// **************************************************************

calCommand cal_end_capture set          // cal computation

// **************************************************************
// Some more user instructions
// **************************************************************

." Starting error settling" cr
." Press any key to terminate" cr

// **************************************************************
// The user observes magErr to watch it settle
// at a minimum value (NDS-1 can display every sec or so):
// This runs until the user has decided that the value
// has converged. See Software Interface Users Manual
// regarding the calibration process.
// **************************************************************

begin                   // keep printing magErr at .250 sec intervals
  ?key 0=               // till user hits a keystroke
while
  magErr di.
  250 delay
repeat
key drop               // read and remove the key used
                       // to stop the loop

// **************************************************************
// Let the user know that we are all done.
// **************************************************************

." Calibration done!" cr

// **************************************************************
// Send the cal_end command.
// **************************************************************

calCommand cal_end set        // cal computation
```

```
  // ************************************************************
  // End calibration mode
  // ************************************************************

  calmode 0 set          // terminate calibration mode.
;
```

Mounting Orientation

The SP300X compasses only allowed two mounting configurations, horizontal and vertical. The DC-4E/GEDC-6E/AHRS-8 inertial systems allow any arbitrary orientation in the user's product.   Standard orthogonal configurations may be chosen. In addition, there is a command called InvokeTare that implements an algorithm to calculate a Rotation Matrix between the User's system orientation and the inertial system's reference orientation. There is a third method which we call sat_tare which gives the user the ability to point there host system towards a known Azimuth at a known elevation (with zero roll) and calculate the Rotation Matrix. As with other NorthTek™ programs in this document, this program is also described in detail in another application note. It is important to note that this method requires the user to have an absolute zero roll in their environment. This is a one-time mechanical alignment procedure and therefore any errors associated with its use will show up as errors in the final solution.

```
// ************************************************************
// If the macro gets reloaded in the same session
// Forget the previous version
// Needs revision 2.1.1 or later.
// ************************************************************
// This forgets the following from a previous load of the macro
// this is standard practice with NorthTek for debugging
// so as to not overflow the wordlist space.
// When this file is reloaded with a terminal a second time
// this removes the previous variables and program.
forget comp_rot_matrix

// ************************************************************
// Declare a few working variables.
// ************************************************************
variable comp_rot_matrix 9 allot
variable sat_rot_matrix 9 allot
variable tare_rot_matrix 9 allot
variable azimuth
variable pitch1
variable cosine_theta
variable sine_theta
variable cosine_psi
variable sine_psi
variable copyarray 5 allot

// ************************************************************
// Function (aka NorthTek word): index!
// Shorthand to compute an array index into the variable
// copyarray and write the given value to that index position
// Inputs:
```

```
//   TOS-1: value to be written
//     TOS: index into copyarray
// Modifies:
//    NorthTek variable: copyarray
// This word takes a value and an index and stores that element
// in that position in the copyarray.
// ************************************************************
( value index -- )
: index! 4 * copyarray + ! ;


// ************************************************************
// Function (aka NorthTek word): cp
( array_ptr -- )
// copies a 3 element array into the copyarray
// copyarray ends up with 0 2 V1 V2 V3 which is
// the right form for setting in the database
// Inputs:
//    TOS: ptr to source array
// Modifies:
//    NorthTek variable: copyarray
// ************************************************************
: cp
  0 0 index!              // store 0 in position 0 of copyarray
  2 1 index!              // 2 in position 1, therefore we set items 0..2
  dup @ 2 index!          // make copy of pointer,for next index, store 1st element.
  4 + dup @ 3 index!      // move to second element, make copy, store value
  4 + @ 4 index!          // move to third element, store it.
;
: compute_vars
       pitch1 @ d>r cos cosine_theta !
       pitch1 @ d>r sin sine_theta !
       azimuth @ d>r cos cosine_psi !
       azimuth @ d>r sin sine_psi !
;

// ************************************************************
// Functions (aka NorthTek words): row0, row1, row2
// some convenient shorthand for matrices
// Inputs:
//  TOS: Ptr to start of a matrix
// Outputs:
//  TOS: Ptr. to start of a row within the given matrix
// ************************************************************
// Since in Forth you must do matrix/array index calculations
// explicitly, these operators just make it easy to
// get to rows 1 and 2 of a 3x3 matrix.
: row0 0 + ;
: row1 12 + ;
: row2 24 + ;

// ************************************************************
// Function (aka NorthTek word): compute
// The actual computation
( -- )
// This stack diagram indicates that there are no params
// and no explicit results.
// This function uses the global variables comp_rot_matrix and sat_rot_matrix
// Uses:
//   Database variables: cp2, cp1, accelEst
// Modifies:
//   NorthTek variable: comp_rot_matrix
// ************************************************************
: compute
  compute_vars
  cp2 di@ cp1 di@ accelEst di@  // get the three desired columns
  comp_rot_matrix buildMatrix            // build the comp_rot_matrix with rows

  cosine_theta @ cosine_psi @ f* sat_rot_matrix row0 !
  cosine_theta @ sine_psi @ f* sat_rot_matrix row0 4 + !
  f0.0 sine_theta @ f- sat_rot_matrix row0 8 + !

  f0.0 sine_psi @ f- sat_rot_matrix row1 !
  cosine_psi @ sat_rot_matrix row1 4 + !
  f0.0 sat_rot_matrix row1 8 + !

  cosine_psi @ sine_theta @ f* sat_rot_matrix row2 !
  sine_theta @ sine_psi @ f* sat_rot_matrix row2 4 + !
  cosine_theta @ sat_rot_matrix row2 8 + !

  sat_rot_matrix comp_rot_matrix tare_rot_matrix m*m>r
;

//

// ************************************************************
// Function (aka NorthTek word): copyit
// Copy each row of the matrix into the
// The corresponding row of the boresight matrix.
// Uses:
//    NorthTek variable: matrix
// Modifies:
//    Database variables:
//         boresightMatrixX
//         boresightMatrixY
//         boresightMatrixZ
// ************************************************************
: copyit
  tare_rot_matrix row0 cp                    // Copy row 0 of matrix to
                                    //   the copyarray
  boresightMatrixX copyarray        // Set the database with the row
  set drop                          //   and then drop the result
                                    //   from the stack
  tare_rot_matrix row1 cp                    // same as before, Y row
  boresightMatrixY copyarray
  set drop
  tare_rot_matrix row2 cp                    // same as before Z row
  boresightMatrixZ copyarray
  set drop
;

// ************************************************************
// Function (aka NorthTek word): printit
// Printout the computed tare_rot_matrix and the current boresight
// matrix.
// ************************************************************
: printit
  tare_rot_matrix cr m.                  // Use the matrix print function
  boresightMatrixX  di.         // Use the database print function
  boresightMatrixY  di.
  boresightMatrixZ  di.
;

// ************************************************************
// Function (aka NorthTek word): tare
// Create a small program to perform the Tare function
// Uses:
//   Database variables (used by compute): cp2, cp1, accelEst
// Modifies:
//   NorthTek variable: tare_rot_matrix
//   Database variable: orientation
//   Database variables (set by copyit):
//         boresightMatrixX
//         boresightMatrixY
//         boresightMatrixZ
// ************************************************************
: sat_tare
  pitch1 !
  azimuth !
  tare_rot_matrix clear(m)       // clear the matrix we declared
  orientation 0 set      // setup to default orientation, required.
  1000 delay             // Wait 5 seconds for this to settle in computation
  compute copyit         // compute the matrix and copy to the boresight matrix.
  printit                // Print it for verification.
;

// Function (aka NorthTek word): bye
: bye
." Boresight adjustment complete!\r\n"
;

// Input format is azimuth pitch1 sat_tare CRLF
//    ex. f191.4 f49.5 sat_tare [ENTER]
```

Analog/Digital I/O Functions

The DC-4E/GECD-6E/AHRS-8 inertial systems do not have the extra input/output functions found in the SP300X family parts. There are no known workarounds for this.

SPI Interface

There are no known workarounds for the lack of a SPI interface for the DC-4E/GECD-6E/AHRS-8. The end user should select one of the asynchronous serial protocols to utilize the DC-4E/GECD-6E/AHRS-8 inertial systems.

World Magnetic Model Update

The DC-4E/GECD-6E/AHRS-8 inertial systems are shipped with a World Magnetic Model that is current at time of manufacture. The World Magnetic Model may be updated in the field. This was also true of the SP300X products. However the previous generation products required the use of the evaluation kit software to download the World Magnetic Model. The DC-4E/GECD-6E/AHRS-8 inertial systems allow World Magnetic Model updates via a downloadable NorthTek™ program that can be sent to the inertial system over the user port in printable ASCII form. The user should provide some type of user port bypass to an external PC to facilitate this update. The user should also consider adding the necessary circuitry to perform field firmware updates as well. A firmware update will inherently contain a new World Magnetic Model. The World Magnetic Model may also be updated with a file transfer using the Remote Function Select (RFS) protocol. A separate application note (AN1004) describes the minimal circuitry required to have both firmware and World Magnetic Model update capability in the user's product.

Digital Filter

The new DC-4E/GECD-6E/AHRS-8 inertial systems contain the AdaptNav™ or other system fusion algorithms that supersede the digital filter in the SP300X products. The user should consult the Software Interface User's Manual or other application notes regarding the settings required to get similar performance to the filtered output from the previous generation products.

Code Update

The firmware in the DC-4E/GECD-6E/AHRS-8 inertial systems is field upgradeable if the proper circuitry is built into the user product. Consult the appropriate application note (AN1004) to obtain the necessary information to design this capability into the product. This same circuitry can be used to update the World Magnetic Model should the case arise. The firmware upgrade procedure is also described in a application note (AN1006).

Want to know more?

- Check it out here: www.spartonnavex.com