OceanServer Iver3-580 EP Autonomous Underwater Vehicle Remote Helm Functionality

Jeffery DeArruda
OceanServer Technology
Fall River, MA 02723

***Abstract*- The Iver3 Autonomous Underwater Vehicle remote helm functionality provides an open system interface, which allows users to easily integrate their own sensors, and take full control over the vehicle in real time. The remote helm option provides a separate CPU to communicate with the main vehicle control CPU. The remote helm API provides all the commands when interfacing the Iver3 to a user control program. This paper will describe the remote helm commands of this serial bus API and the software tools that can aid in implementing a system.**

I. INTRODUCTION

Historically most Autonomous Underwater Vehicles (AUVs) have been closed systems. Any changes require contracting the manufacturer for a custom design change, which are costly in time and money. In addition, the expanding role of AUVs has users looking for more reliable and versatile vehicle to perform various tasks. The Iver3 AUV from OceanServer is an open system, where the users can install their hardware and make software extensions to the vehicle without a custom design. OceanServer provides physical space inside the forward section for user electronics as well as hull penetrators to make connections to external sensors. The remote helm design model requires a separate CPU, which allows the users to install their own operating system that can connect to the added hardware. The second CPU communicates with the Iver3 control CPU through a serial port via a rich set of commands for remote helm control.

II. IVER3 SYSTEM ARCHITECTURE

The remote helm's second CPU is connected to the main CPU through a serial port and Ethernet connection. The serial port is used to transmit and receive remote helm commands. The Ethernet allows the user to remote login and transfer from the main CPU. The remote helm's CPU is a Dual Core Intel Atom 1.6GHz with 4GB of RAM, and the motherboard comes with sixteen serial ports and six USB ports. If the user would like to install their sensors inside the AUV different forward section space is provided depending upon the size selected. The different types and dimensions for the user's added electronics are shown in Table I. The AUV has six hull penetrators provided for any added external sensors, which has access to the vehicle's power.

TABLE I
IVER3 EP MODEL

Model	**Payload Tube** (Options) **5.8 Diameter Tube*	**Main Section**
EP	No Forward Section	28 - 30"
EP10	10" Forward Section	28 - 30"
EP16	16" Forward Section	28 - 30"

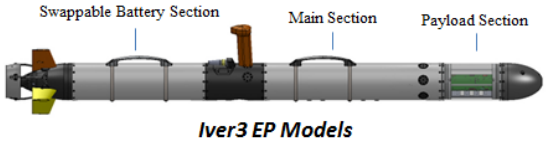

Figure 1. Illustration of the EP model

III. BASIC OPERATION OF THE IVER3

There are three steps in order to operate the Iver3. The first is to create and save a mission to the vehicle via Wi-Fi. To create a mission, the operator uses the OceanServer mission planning software called VectorMap (VM). To plan a mission, a standard chart is loaded and then the user can point and click to add waypoints. The vehicle's operating characteristics and sensor functions can be assigned to a waypoint. Once the mission is finished and saved to the vehicle, the next step is to log on to the vehicle using Windows Remote Desktop. The last step is to load and start the mission using the Underwater Vehicle Console (UVC) software on the Iver3.

IV. REMOTE HELM COMMUNICATION

The remote helm commands are an ASCII string with a NMEA checksum. On the main CPU resides the Underwater Vehicle Console (UVC) software, which is responsible for operating the vehicle and processing remote helm commands. After a remote helm command is sent to the main CPU, the UVC will send back an acknowledgment string, which verifies that the command was successfully received and processed (Figure 2). The remote helm commands provide full control of the vehicle's settings, behavior and sensors. There are five different control modes the remote helm has access to along

with the ability to read and write vehicle setting and sensors. The remote helm command modes include Servo, Primitive, Normal, Park, and Mission. Each control command is interrupt-driven; meaning any mode can interrupt another.

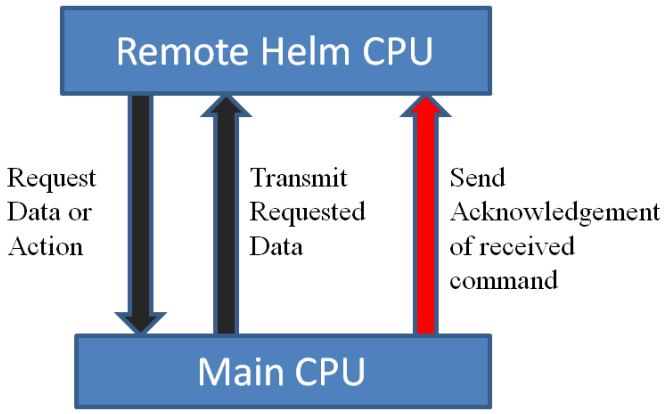

Figure 2. Remote helm architecture

A. *Servo Mode*

The servo mode is similar to an autopilot behavior; this mode allows the remote helm to provide the vehicle with its next heading, depth from surface, maximum pitch angle, speed and a timeout value. The timeout value is used to regain control of the vehicle if the UVC hasn't received a new command within the specified timeout value. The servo commands can be sent anytime after the start of the mission. Once control is released the vehicle will continue to the next waypoint, where it was interrupted. If the vehicle is parking, the remaining time the vehicle has to park will be saved, and once control is released the vehicle will park for the remaining time.

TABLE II
SERVO COMMAND

FORMAT:	$OMS,<Heading>, <Depth>,<Max Angle >,<Speed>,<TO> <*cc>
$OMS	Message type: Servo Control
<Heading>	Heading set point in degrees relative to magnetic north
<Depth>	Depth set point of the AUV referenced from the depth pressure sensor
<Max Angle>	Maximum pitch angle of the vehicle for changing depth (up or down) in degrees
<Speed>	Forward speed set point in water column
<TO>	Time Out: Command life
<*cc>	NMEA Check sum

B. *Primitive Mode*

The primitive mode control allows the remote helm to operate the vehicle as a controller. Primitive control overrides the settings of the control fins and propulsion, also a timeout value is specified. Primitive control can be entered before or after the start of a mission.

TABLE III
PRIMITIVE COMMAND

FORMAT:	$OMP,<FYT><FYB><FPL><FPR><MS>, <RP>,<TO><*cc>
$OMP	Message type: Primitive Control
<FYT>	Fin Control: Yaw Top Fin
<FYB>	Fin Control: Yaw Bottom Fin
<FPL>	Fin Control: Pitch Left Fin
<FPR>	Fin Control: Pitch Right Fin
<MS>	Motor Speed
<RP>	Reserved Parameter
<TO>	Time Out: Command life
<*cc>	NMEA Check sum

C. *Normal Mode*

In Normal mode the main CPU is in control of the vehicle and executing a user defined mission. Commands that are executed under the normal mode are:

- **Jump to waypoint**: Allows the backseat driver to jump to any waypoint during a mission. The jump to waypoint command also overrides the commands in Primitive, Servo and Park mode.

TABLE IV
JUMP TO WAYPOINT COMMAND

FORMAT:	$OJW,<WPN><*cc>
$OJW	Message type: Set next way point
<WPN>	The next waypoint value is this value
<*cc>	NMEA Check sum

- **Mission Control**: The remote helm is allowed to stop the current mission, load mission and start a mission.

TABLE V
STOP MISSION COMMAND

FORMAT:	$OMSTOP,<FL><*cc>
$OMSTOP	Message type: Stop Current Mission
<FL>	Message Flag – the null flag is always included
<*cc>	NMEA Check sum

TABLE VI
LOAD MISSION COMMAND

FORMAT:	$OMLOAD,<DR>,<FL><*cc>
$ OMLOAD	Message type: Load Mission
<DR>	Directory or name of mission file
<FL>	Message Flag
<*cc>	NMEA Check sum

TABLE VII
START MISSION COMMAND

FORMAT:	$OMSTART,<FL><*cc>
$OMSTART	Message type: Start Mission
<FL>	Message Flags: 0 : Null 1 : Ignore GPS (wait for $OPOS) 2 : Ignore Sounder 3 : Calibrate Pressure Transducer
<*cc>	NMEA Check sum

D. Read Write Mode

- **Set current position, speed and/or conductivity and temperature**: The conductivity and temperature are accepted every time the UVC receives the command, but setting the vehicle's current location and speed are only accepted when the vehicle is not receiving valid GPS position.

TABLE VIII
SET CURRENT POSTION AND/OR
CONDUCTIVITY AND TEMPERATURE

FORMAT:	$OPOS,<LAT>,<LNG>,<SPD>,<COND>,<TMP><*cc>
$OPOS	Message type: Current position and/or Conductivity and Temperature
<LAT>	User Provided update to vehicle current position. The backseat may have a more accurate position estimate and want to update the Dead reckoned position to reduce position errors.
<LNG>	User provided longitude
<SPD>	Estimated vehicle speed (knots)
<COND>	Conductivity (mmhos/cm)
<TMP>	Temperature (c)
<*cc>	NMEA Check sum

- **Request vehicle information**. This command permits the backseat driver to request data from the vehicle, such as the vehicle's Current State, Compass, GPS, State, Power, YSI sensor and DVL data.

TABLE IX
REQUEST VEHICLE INFORMATION

FORMAT:	$OSD,<GPS>,<Compass>,<State>,<Power><YSI>,<DVL>,<CTD>,<TO>,<*cc>
$OSD	Message type: Send Sensor Data
<GPS>	UVC Outputs: Current coordinates
<Compass>	UVC Outputs: All compass data
<State>	UVC Outputs: The current vehcile Mode, Next Waypoint (WP), Coordinates, Speed, Distance to Next WP, Error State, Depth and Remaining park time.
<Power>	UVC Outputs: Current battery parameters
<YSI>	UVC Outputs: With outputs from all YSI sensors
<DVL>	UVC Outputs: X.Y and Z speed
<CTD>	UVC Outputs: Conductivity, Temperature and Depth
<*cc>	NMEA Check sum

- **Log user data**: The second CPU is allowed to log 12 parameters in the vehicle's main log file. The remote helm first sends column headers for the log file once, and then continuously passes the data over for the main CPU to log. To setup the column headers for the main log file the $OLOGL command is used, as shown in Table X. OLOGL command must be sent before is mission is started. The OLOGD command is used to log the user's data.

TABLE X
COLUMN HEADERS FOR LOG FILE

FORMAT:	$OLOGL,<ST1>…<ST12><*cc>
$OLOGL	Message type: Store column headers in the main log
<ST1>…<ST12)	Column header labels
<*cc>	NMEA Check sum

TABLE XI
DATA FOR LOG FILE

FORMAT:	$OLOGD,< Data1>,< Data2>, <Data3>,..< Data12><*cc>
$OLOGD	Message type: Store data in the main log
<Data1>… < Data12)	Data to be stored in the main log file.
<*cc>	NMEA Check sum

- **Read or Write vehicle settings**: The settings are corresponding to the vehicles behavior in the UVC and either read or written before the start of a mission.

TABLE XI
READ OR WRITE VEHICLE SETTINGS COMMAND

FORMAT:	$ORWSET,<REQ SET>, <R/W>,<STATE or

	VALUES>,<*cc>
$ ORWSET	Message type: Read and write current vehicle settings
< REQ SET >	Requested Setting: Select a setting in the UVC software
<R/W>	Read or Write the setting
<STATE or VALUES>	If writing include and state to enable or disable the setting and also any value associated with that setting
<*cc>	NMEA Check sum

- **Send X and Y velocities**: The DVL command, permits the backseat driver to maintain the correct position and speed by sending the X and Y velocity from another user installed DVL.

TABLE XII
X AND Y VELCOITY COMMAND

FORMAT:	$ODVL,<XSPEED>,<YSPEED >,<TO><*cc>
$ODVL	Message type: Send X and Y Velocities
< XSPEED >	X velocity
< YSPEED >	Y velocity
<TO>	Time Out: Command life
<*cc>	NMEA Check sum

E. Park Mode

Park mode is activated when the remote helm sends a park command, which informs the vehicle of the park location and the amount of time (minutes) the vehicle will station keep. If the park command is interrupted by another command then the vehicle will exit its park location and will not return to complete the remaining park time.

TABLE XIII
GO TO PARK LOCATION COMMAND

FORMAT:	$OPK,<LAT>,<LNG>,<TM>,<SP><*cc>
$ OPK	Message type: Go to park location
< LAT>	Latitude
< LNG>	Longitude
<TM>	Time
<SP>	Speed
<*cc>	NMEA Check sum

F. Mission Mode

The Mission command allows users to create a mission on the fly by issuing each waypoint's parameters. The mission command is sent after the vehicle has started its pre-loaded mission. The OMW commands are stored in a queue and the vehicle will perform the waypoint's behaviors. The user can clear the mission at anytime by sending a clear signal ($OMW,Clr*cc) or interrupting the mission with another remote helm command. At any point while the vehicle is running the mission, a new waypoint can be added to the mission. Once the vehicle has completed the mission it will continue where it left off during the original mission.

TABLE XIV
CREATE MISSION COMMAND

FORMAT:	$OMW,<LAT>, <LNG>, <Depth1>, <Depth2>,< Max Angle > <SPD>,<Park>, <Sensors> <*cc>
$OMP	Message type: Primitive Control
<LAT>	Latitude
<LNG >	Longitude
<Depth1>	Enter Depth: DFS or HFB
<Depth2>	Enter Depth for undulation: DFS or HFB
<Max Angle>	Maximum pitch angle of the vehicle for changing depth (up or down) in degrees
<SPD>	Speed
<Park>	Park Time
< Sensors>	Send Sensor Configuration
<*cc>	NMEA Check sum

V. COMMAND OPERATION DURING A SAFETY VIOLATION

The UVC software has a set of safety rules to prevent the vehicle from becoming disabled. Each rule is user configurable, some of them, including Maximum Operation Time, Leak Detection, No Forward Progress and Maximum Depth. In the event a safety rule is triggered, the remote helm control will be ignored for twenty seconds, except for requesting data. During those twenty seconds an assessment is made of the vehicle's sensors in order to ensure that the compass and GPS are functioning. After the twenty seconds, Primitive mode is released. If the vehicle was underwater the depth is tracked to see if the vehicle is making upward progress to the surface. If the vehicle is stuck underwater the UVC will pulse the vehicle's motor in reverse in order to bring the vehicle to the surface. The remote helm could use primitive mode to spend more time freeing the vehicle. Once at the surface the user is given an option of parking at the present location until the batteries expire or execute another mission called the SRP (Safety Return Path). The SRP is another mission that will be used to bring the vehicle back to a safe location for retrieval. The SRP is provided to UVC at the beginning of a mission.

VI. EMULATING REMOTE HELM COMMANDS

An open source program called SubTester is provided to aid users in developing a working remote helm interface. SubTester demonstrates transmitting and receiving proper remote helm sentences. The program is also used by

OceanServer development to test and debug the backseat driver interface, as shown in Figure 3.

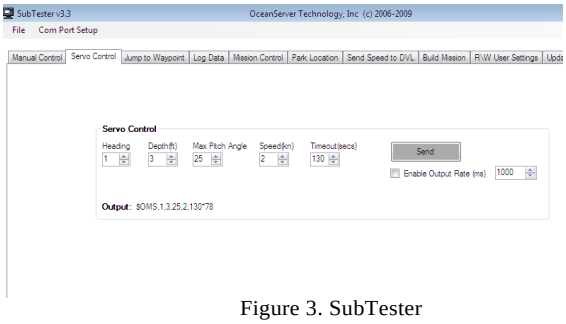

Figure 3. SubTester

To observe the backseat driver commands the UVC has a debug facility, in which the user can view all incoming serial data. Figure 4 shows the debug window running on a normal system. The user can see the system parsing all of the NMEA sentences that it is receiving. The area to the right of the screen shows the state of the control surfaces and motor as well as the present controller state. Some commands are given a timeout life, and the user can see the life of the command count down next to the *Timeout* label.

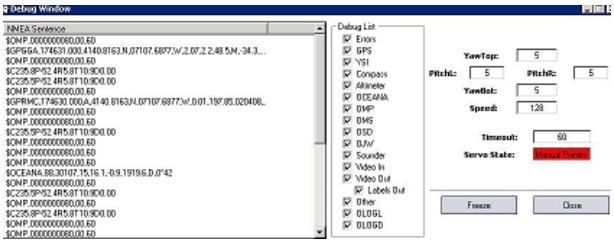

Figure 4. UVC Remote helm debug window

VII. Examples of Remote Helm Implementations

Many users that have utilized the remote helm functionally of the Iver3 have used the MOOS-IvP (Mission Oriented Operating Suite – Interval Programming) software which was developed by MIT [1]. MOOS-IvP is open source software and platform independent. The Naval Undersea Warfare Center (NUWC), Division Newport developed the MOOS module iOceanServerComms, which provides the interface between the main CPU and the backseat. Using the iOceanServerComms the user can utilize pre-defined behaviors in the IvP Helm or a custom behavior can be developed.

SeeByte's SeeTrack Neptune was integrated with the Iver3 through the remote helm functionality. Neptune provides a payload control architecture and real time autonomy engine for unmanned systems to plan and execute autonomous behavior that expedite and optimize single vehicle and multi-vehicle operations [3]. Neptune's hardware abstraction layer was mapped to the remote helm interface to access vehicle status, sensor data and navigation information from the vehicle and to provide waypoints back to the vehicle.

Neptune also allows third-party development of functions and behaviors to extend and enhance capabilities. This key ability means that autonomy behaviors written for other Neptune-enabled platforms can now be run on any Iver3 Neptune-enabled vehicle. Neptune is used by numerous Navies and has been most recently demonstrated running cooperative missions with multiple types of platforms from different domains (UUV & USV) from multiple countries (UK, Canada) at the ONR sponsored TTCP 2015.

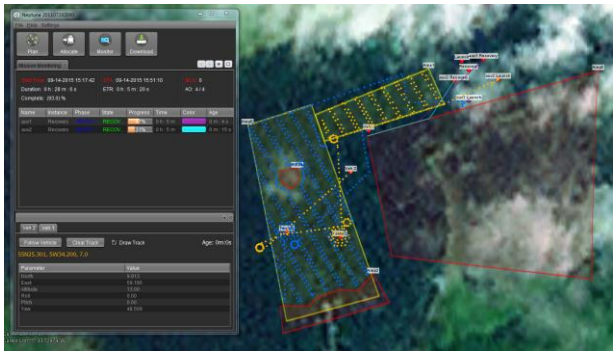

Figure 5. Neptune Image

VIII. CONCLUSION

In this paper the remote helm functionality of the hardware and software was described. The combination of an intuitive API along with open payload space opens the door to a wider range of AUV research. The remote helm option, with five different control mode options, gives users a versatile and rapid path for both custom vehicle behavior and sensor integration.

ACKNOWLEDGMENT

I would like to acknowledged the staff at Naval Undersea Warfare Center (NUWC), Division Newport RI, for all their work with the developing the OceanServer Iver interface with MOOS-IvP software. I would also like to acknowledge the Defence Research and Development Canada (DRDC), for sponsoring the developing the OceanServer Iver interface with SeeTrack Neptune.

REFERENCES

[1] M. Benjamin, P. Newman, H. Schmidt,, and J. Leonard. *An Overview of MOOS-IvP and a Brief Users Guide to the IvP Helm Autonomy Software*. Tech. Cambridge: MIT, 2009.

[2] D. Eickstedt and S. Sideleau. *The Backseat Control Architecture for Autonomous Robotic Vehicles: A Case Study with the Iver3 AUV*. Tech. Newport: Naval Undersea Warfare Center,2009. [3] P. Patron, *Semantic-based adaptive mission planning for unmanned underwater vehicles*, Heriot-Watt University, 2010