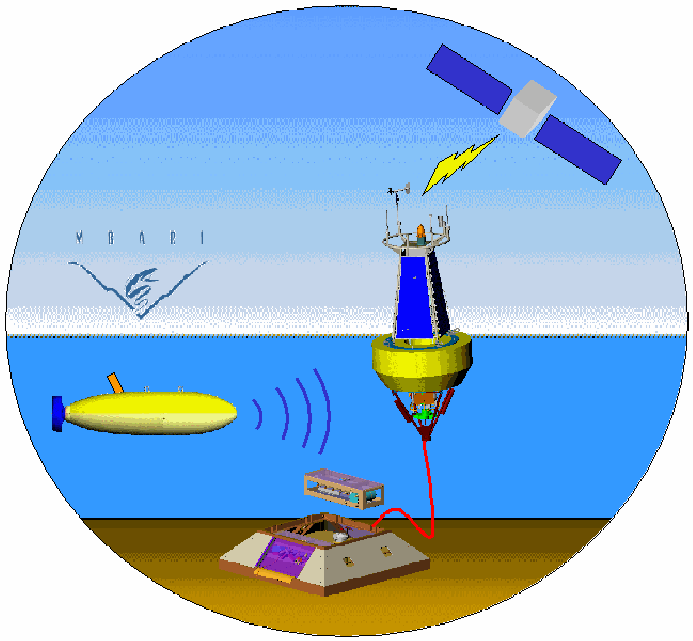
PUCK Changes



Michael Risi

March 28, 2005

“Ch-ch-ch-ch-Changes”

-David Bowie

- What we do now
- Proposed changes
- SSDS role?
- PUCK consortium role

What we do now

- Platform scans PUCK for header (A.K.A. MOOS memory map)
- PUCK uploads jar file based on information contained in header
  - Only PUCK commands used
  - PUCK has no knowledge of memory content or format
- SIAM starts the service located in the retrieved jar file

MOOS puck memory map

0x0000 0x0003	Organization (4 bytes)
0x0004 0x0007	Type\Version (4 bytes)
0x0008 0x0017	UUID (16 bytes)
0x0018 0x0023	payload start address (4 bytes) payload end address (4 bytes) payload checksum crc32 (4 bytes)
0x0024	sensor data sheet -driver name (256 bytes) -ISI ID (8 bytes) -max current required (4 bytes) -etc. The format will be determined by the version and newer version wills always be backward compatible with older versions.
start address	driver and meta-data in jar file

MBARI

PUCK command summary

- RM - Read from the PUCK's memory
- WM - Write to the PUCK's memory
- FL - End a writing session
- ER - Erase the PUCK's memory
- GA - Get the internal address pointer
- SA - Set the internal address pointer
- SZ - Return PUCK's memory size
- SM - Put the PUCK into sensor mode
- SB - Set the baud rate of the PUCK
- VB - See if another baud rate is supported
- VR - Get the PUCK's version string

Proposed changes(1)

- Changes are being made based on deployment experience and feedback from SIAM design review

- Reduced PUCK command set

Proposed changes(2)

- New more generic instrument datasheet included as part PUCK Specification
  - IEEE 1451.4 influenced
  - Possibly read only for embedded pucks

- New discovery processes

PUCK Command Changes

Proposed change	Reason
Prepend all PUCK protocol commands with PUCK_ to avoid collision with instrument commands	Comments from: John Graybeal, Mike Godin, Andrew Barnard, John Backes
PUCK enters instrument mode automatically after failed attempts. (What is a failed attempt? Should this be a time out? Start PUCK command?)	Comments from: Andrew Maffei, John Backes
Minimal PUCK command set that is ostensibly for read-only PUCKs	Comments from: Jeff Burch

PUCK Content Changes

Proposed change	Reason
Changed instrument datasheet (influenced by 1451.4)	Comments from: Dale Chayes, Jeff Burch, Scott McLean
Single unique ID per PUCK (should this be UUID? read only?)	Comments from: Andrew Maffei, Andrew Barnard
Make PUCK payload other than ID/header optional (ID only option)	Comments from: Scott McLean, Jeff Burch

New PUCK command summary

- PUCK_RM - Read from the PUCK's memory
- PUCK_ WM - Write to the PUCK's memory
- PUCK_ FL - End a writing session
- PUCK_ ER - Erase the PUCK's memory
- PUCK_ GA - Get the internal address pointer
- PUCK_ IM - Put the PUCK into instrument mode
- PUCK_ SA - Set the internal address pointer
- PUCK_ SZ - Return PUCK's memory size
- PUCK_ SB - Set the baud rate of the puck
- PUCK_TY - Query PUCK type
- PUCK_ VB - See if another baud rate is supported
- PUCK_ VR - Get the PUCK's version string

Instrument Datasheet

Description	Size (bytes)	Format
UUID	16	UUID
Version	2	U16
Datasheet Size	2	U16
Manufacture ID	4	U32
Manufacture Model	2	U16
Manufacture Version	2	U16
Serial Number	4	U32
Instrument Name	128	CHAR ARRAY
Optional Payload Start Address	4	U32
Optional Payload Size	4	U32
Optional Payload Checksum	4	U32
Datasheet checksum	4	U32
Total	**176**	

New PUCK Discovery Processes

- Lookup driver and metadata via UUID
- Extract driver and metadata from optional payload
- Lookup driver via manufacture ID and generate generic metadata dynamically

UUID Approach

- Drivers and metadata could be stored on the platform prior to deployment
- When an instrument is going to be deployed the driver and metadata can be queued in the portal for upload
- An interface could be used to load drivers and metadata onto a platform at sea

Optional Payload Approach

- The platform can scan the PUCK's instrument datasheet
- If an optional payload exists it can be uploaded
- If the payload is in a format that the platform understands it can use it
- Very similar to what we do now

Manufacture ID Approach

- Drivers could be stored on the platform by manufacture
  - ID
  - Model
  - Version
- Generic metadata could be generated or stored with the driver
- The instrument will still be uniquely identified via UUID

SSDS could provide

- Utility to read and write PUCKs
- Track UUIDs of instruments in database
- Provide API to access drivers and metadata for queuing in the portal

Marine Plug-and-Work Consortium

- Assign and track manufacture IDs
- Manufacture must provide model and version information to consortium
- Maintain the PUCK specification
- Provide forum to evaluate and adopt Plug-and-Work interfaces for different instrument interfaces

Discussion