MBRN-V4-RS485 Interface Control Document (ICD)
v0.3 – November 1, 2020

Table of Contents

Change History

Ver	Date	Summary
0.0	Sep 19, 2020	Initial release
0.1	Oct 5, 2020	Redefined Address 0 as reserved to guarantee a first byte of 0 is never valid Defined Address 30 as the special broadcast address for field upgrades Changed response timeout (at LS4) to 100ms Changed broadcast repeat window to [5ms - 20ms] (4-bits of entropy) Noted that drawer indexes of 15 (0xF) denote an unassigned drawer Added new interlock fields to MT02 (corresponding states added to MT83) Added MT51 (and MTD1 response) for reading DSB debug data Added global lock/unlock to the Discovery sample message flow. Added new error code definitions
0.2	Oct 14, 2020	Refined meaning of drawer **Open** field in MT83 and MT99 Added new **Event** field to MT99 to more definitively capture behavior Removed MT91 which had no real use in the end Added MT05/MT85 to applicability matrix (support BLM to poll for errors) Added polling of errors to DSB Field Upgrade message flow Added new error code definitions
0.3	Nov 1, 2020	Now permit 8-byte data lengths in any message (was only MT77 previously) Expanded drawer indexes from 4-bits to 5-bits for growth Added note to MT02 regarding deenergization of solenoids on mode change Added MT07 for global drawer proximity recalculation Added MT08 for drawer lock override Expanded MT22 drawer index to 5-bits to align with related changes Added new debug indexes to MT51 Expanded MT81 drawer identification scheme (now an 8-data-byte message) Expanded MT83 to include drawer indexes (now an 8-data-byte message) Expanded MT99 drawer index to 5-bits to align with related changes Added MT51/MTD1 to applicability matrix (for BLM debugging) Added new error code definitions

Introduction

This document defines a multi-drop protocol and relevant message content for a single half-duplex internal V4 RS485 serial bus. This link connects the **Landshark V4** (*LS4*) to a series of **Drawer Sensor Boards** (*DSB*) and (in the future) an evolved **ChillUPS V4** (*CUPS4*) that supports a more robust RS485 control interface (versus I2C used in V3).

Throughout this ICD, LS4 is considered the “master” of the RS485 link. All other elements on the bus are referred to as “nodes.” Most of the time, nodes speak (send bus messages) only when the master asks for information (via a read request). However, there are cases where nodes must broadcast messages for all recipients on the bus. Likewise, the master may broadcast messages for all nodes. Given the asynchronous nature of these use cases, this ICD has provisions for detecting and overcoming RS485 message collisions that may occur.

Terminology

The following terms and acronyms are used throughout this ICD.

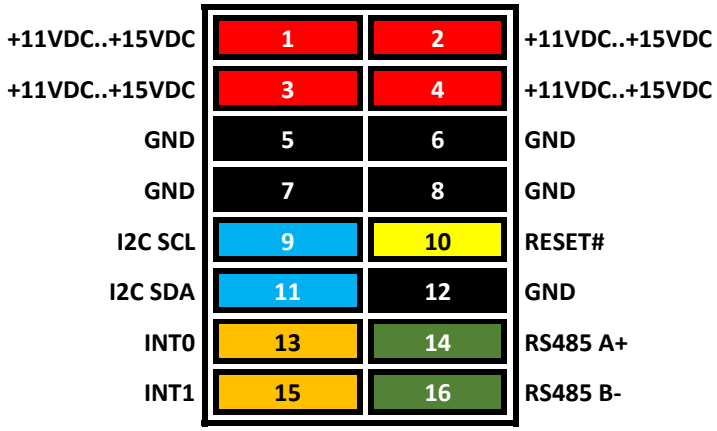
Term	Definition
BLM	Bootloader Mode

CUPS4	ChillUPS V4
DSB	Drawer Sensor Board
FTM	Factory Test Mode
GND	Ground
LS4	Landshark V4
Master	Landshark V4 on the RS485 bus (it owns the bus)
NC	No Connect
Node	Any end-point on the RS485 bus that is not the master
TBD	To Be Determined

Hardware Interfaces

Landshark V4

The internal V4 RS485 serial bus originates on LS4 at the 16-pin "backplane" connector (**J5**). The pin-out and signal allocation of **J5** are defined below.

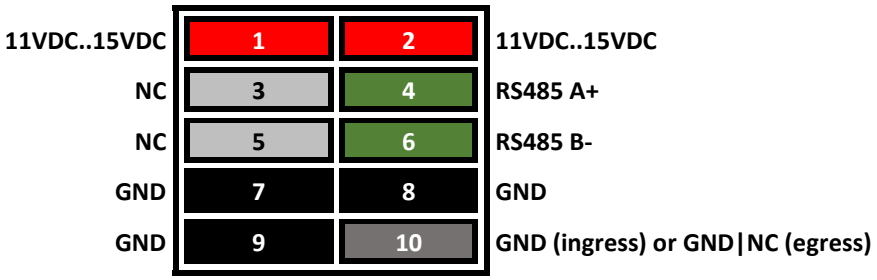


Name	Description
+11VDC..+15VDC	Supply voltage provided to downstream DSBs (ranges based on source which can be backup battery)
GND	Ground
RESET#	Open drain reset. Active low. Referenced to TBD. Usage is TBD (*not wired at this time*).
I2C	I2C link to CUPS4 for field upgrades. This link must support **5V logic**.
INTn	General purpose interrupt lines (or GPIOs perhaps); referenced to TBD. Usage is TBD (*not wired at this time*).
RS485	Serial bus pair for command/control of downstream nodes.

Drawer Sensor Boards

There are two configurations of DSBs in the V4 system. One configuration has three sets of drawer related sensors and locks. The other configuration has only one set for a single drawer. Thus, each DSB may control a variable number of drawers.

Each DSB has one "ingress" (**J2**) connector and one "egress" connector (**J1**) to allow chaining together with the next node. Both connectors are wired identically with the exception of the "egress" connector which has an open pin that either gets grounded by the next node or remains floating thus enabling RS485 termination on the last board in the chain. The pin-out and signal allocation of **J1** and **J2** are defined below.



Name	Description
+11VDC..+15VDC	Supply voltage provided from LS4 (ranges based on source which can be backup battery)
GND	Ground
GND|NC	Pin is always ground at **ingress** connector. Pin is ground at **egress** when connected to another board. Pin is floating at **egress** on the last board which enables internal 120-ohm termination of the bus on that board
RS485	Serial bus pair for command/control

Software Interfaces

Introduction

The internal V4 RS485 serial bus is seen by connected systems as a basic UART interface in software. The following basic functions are required:

- LS4 discovery of all nodes on the bus (who do not speak until discovered)
- LS4 read of DSB configuration information (type, drawer count, etc.)
- LS4 read of DSB current temperature
- LS4 read of DSB drawer states (opened/closed, proximity, etc.)
- LS4 read of DSB solenoid states (unlocking, unlocked, locked, etc.)
- LS4 read/clear node errors
- LS4 broadcast allow/disallow drawer opening to all DSBs
- DSB node broadcast drawer-opening event to LS4 and all other DSBs
- LS4 broadcast firmware upgrade to all DSBs

Serial Interface Characteristics

The required serial interface configuration is listed below:

Characteristic	Setting
Baud Rate	115200
Data Bits	8
Parity	None
Stop Bits	1

Serial Interface Protocol

Message Structure

Every serial message has the following form:

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	DATA							
N	CRC							

- **R/W** indicates the transaction type. **0**=**write** and **1**=**read**.
- **Data-Size** encodes the number of data bytes in the message.
    - The field is treated as a power of two (**0**=1-byte, **1**=2-bytes, **2**=4-bytes, **3**=8-bytes). **At least one byte of data is required in every message**.
    - ~~A field value of **3** is reserved for Hex Record messages (that include length as the first byte of data in the message).~~
- **Address** indicates the target of the message following this scheme:
    - **Address 0** is reserved
      Addresses 1..13 are nodes on the bus (other than CUPS4)
      Address 14 is CUPS4 (always at this address regardless of configuration)
      Address 15 is LS4 (the bus master)
      Addresses 16..29 are reserved
      Address 30 is used in *special broadcast* messages during field upgrades (**write** only)
      Address 31 is broadcast to all (**write** only)
    - Addresses for nodes in the 1..13 range are assigned during factory testing using a manual procedure of drawer opening and interaction with the system UI. This process makes use of factory-test specific messages define in this ICD. Review the Address Assignment sample message flow for further details.
    - Any recipient that is addressed directly or via a broadcast message must act on the message in accordance with this ICD.
- **Message-Type** is a binary value (0..255) that uniquely identifies the message. Message types are defined in hexadecimal in this ICD. Some message numbering conventions include:
    - Bit 7 is **0** in LS4-sourced messages and **1** in node-sourced messages.
    - Responses to LS4-sourced **read** requests echo the lower 7-bits of the Message-Type (e.g., an MT15 read would elicit an MT95 response from the addressed node).
- **Data** is additional information that is associated with the particular message and is further defined in each message definition that follows. At least one byte of data is present in every message.
- **CRC** is the 8-bit CRC of all preceding message bytes. The CRC polynomial is $\mathbf{X^8 + X^5 + X^4 + 1}$ with a seed value of **0**. See CRC Sample Code for details.

Message Processing Rules

The following general requirements apply to all messaging on the RS485 serial bus.

- Only LS4 may issue a read transaction. If LS4 receives a read transaction it shall ignore the message.
- Read transactions shall not be broadcast. If such a transaction occurs, all recipients shall ignore the message.
- Read transactions are always directed to a single node on the bus which must respond within **100ms** of receipt.
- With the exception of Discovery messages, response timeouts cause LS4 to reissue the failed read transaction a maximum of two additional times before abandoning the transaction and declaring a link/node failure.
- An LS4 timeout on response to a Discovery message is interpreted by LS4 as a missing node versus an error. No LS4 retry occurs in this case. LS4 uses Discovery to find all nodes on the bus at system initialization. The Discovery sample message flow provides additional detail.
- Read transactions may elicit an error response (instead of the expected response) from the addressed node if the message fails basic validation (not CRC, that's covered below) based on conditions at the node. Refer to the Error Response message for details.
- Any bus element may issue a write transaction. However, nodes generally do not issue write transactions unless first prompted to do so by receipt of an LS4-sourced read request. An exception to this occurs when a node must generate an asynchronous broadcast message to all end-points on the bus.
- Any bus element may issue a broadcast (write) transaction at any time. All subsystems in receipt of a valid broadcast shall act on it as in accordance with this ICD.
- To reduce the chance of bus collisions, all broadcast messages must be kept as short as possible.
- To reduce the impact of bus collisions, all broadcast messages shall be sent **three times** with a pseudo-random delay of **5ms** to **20ms** between each broadcast.
- With the exception of Hex Record messages (**Data-Size**=3), message lengths on the bus are fixed at 4-bytes, 5-bytes, and 7-bytes depending on **Data-Size**.
- Received messages with a CRC of **0x00** are always accepted and acted upon by the receiver of the message (as though the message had arrived with a valid CRC). This allows bypass of message veracity validation logic in the event that is necessary.
- Messages received with an invalid CRC are silently ignored by the recipient. Bad CRCs may be logged locally but no serial traffic is generated (i.e., there are no ACK/NAK actions on the bus).
- Reserved fields (marked as such in the message definitions to follow) shall be set to 0 in all outgoing messages. Message receivers shall not rely on the value of any reserved field (a value of 0 shall not be assumed).
- Message applicability to node types and in various modes is summarized in the Message Applicability Matrix. This information also appears where appropriate in each message definition that follows.
- Sample message flows for various system functions are provided to help clarify intended processing.

Serial Interface Messages

LS4-to-Node Messages

MT01 – Discovery

This message causes the addressed node to send a response containing static configuration information. This is the first message sent by LS4 to each node at system initialization. LS4 sends this to each possible node address on the bus to discover the valid address map that's supported by the hardware present in the system. Refer to MT81 for the response. Review the Discovery sample message flow to better understand the process.

When addressed to CUPS4 (Address = 14), LS4 must always expect a response. CUPS4 is at a fixed address on the bus and thus should always answer with an MT81. If it does not, this is considered a link failure.

Note: LS4 does not retry this message on a response timeout given its use in discovering the valid addresses on the bus. A single response timeout is interpreted by LS4 as an invalid address that will not be used again after system initialization.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	1 (read)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	1..14 (node address)	
Message-Type	N/A	0x01	
CRC	N/A	Variable	0=CRC not in use (optional)

MT02 – GLOBAL INTERLOCKS

This **broadcast message** allows LS4 to globally allow/disallow V4 drawer openings, enable/disable solenoid locks, and enable/disable proximity sensors. For example, the system may want to disallow drawer opening until a user logs into the UI. The interlock states are included in MT83. As a broadcast message, nodes must not reply. This message applies only to DSB nodes, other nodes may ignore it entirely.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	RESERVED				SOLENOIDS		PROX	UNLOCK
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x02	
Unlock	N/A	0=disallow drawer open, 1=allow drawer open	DSBs default to disallowing drawer opening at initialization
Solenoids	N/A	0=disable, 1=enable, auto mode, 2=enable, manual mode, 3=reserved	DSB solenoids default to **disabled** at initialization; solenoids must be enabled to allow drawers to open; **warning** any change in solenoid mode causes all energized solenoids to deenergize across the system; if this occurs whilst a drawer is open (or missing), the solenoid pin will become a projectile and it (along with the corresponding spring) will be lost inside the V4 enclosure
Prox	N/A	0=disable, 1=enable	DSB proximity sensors default to enabled at initialization
CRC	N/A	Variable	0=CRC not in use (optional)

MT03 – Get Drawer States

This message requests DSB drawer state information. It is applicable to DSB nodes only. Refer to MT83 for the response message. This message is supported in both normal and FTM.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	1 (read)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	1..13 (node address)	
Message-Type	N/A	0x03	
CRC	N/A	Variable	0=CRC not in use (optional)

MT04 – Get Drawer Temperature

This message requests DSB temperature. It is applicable to DSB nodes only. Refer to MT84 for the response message.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	1 (read)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	1..13 (node address)	
Message-Type	N/A	0x04	
CRC	N/A	Variable	0=CRC not in use (optional)

MT05 – Get and Clear Error Log

This message requests any errors logged at the addressed node. This request would generally be sent to a node that has reported errors are present (as seen in MT83 from DSB nodes). Refer to MT85 for the response message.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	1 (read)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	1..14 (node address)	
Message-Type	N/A	0x05	
CRC	N/A	Variable	0=CRC not in use (optional)

MT06 – Global Reset

This **broadcast message** allows LS4 to globally reset all nodes on the bus. After sending this, LS4 must wait TBD seconds before reestablishing communications with the nodes (to allow them time to reboot).

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing. This may induce multiple reset events in some nodes.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x06	
CRC	N/A	Variable	0=CRC not in use (optional)

MT07 – Global Drawer Recalibration

This **broadcast message** allows LS4 to globally control drawer home position proximity calibration for all DSB nodes on the bus. **It is important that calibration is initiated only when all drawers in the system are known to be in their home position**. If a drawer is open or ajar, the computed values will be incorrect and the V4 will not operate properly.

After starting calibration, LS4 must wait a few seconds before reissuing this command to end and/or save (to EEPROM) the newly computed home position proximity values. Recalibration cycles run in approximately 60ms, so 5-seconds or less is recommended to get a wide enough average basis. This message is applicable to DSB nodes only, other node types ignore the message.

This message always requests recalibration operations for all V4 drawers, there is no means by which this operation can be initiated for individual drawers or DSB modules.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved						CalOp	
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x07	
CalOp	N/A	0=reserved, 1=start, 2=stop (no save), 3=save (stop w/save)	Controls the proximity calibration operation; if a ‘start’ (1) is not followed by a ‘stop’ (2) or ‘save’ (3) within 1023 calibration loops (approximately 60ms each, so around 1-minute total), the DSB will force an end to calibration without saving the values to EEPROM; if a ‘stop’ (2) or ‘save’ (3) is requested without first requesting a ‘start’ (1) the request is silently ignored
CRC	N/A	Variable	0=CRC not in use (optional)

MT08 – DRAWER OVERRIDE

This **broadcast message** allows LS4 to force any V4 drawer into a locked or unlocked state. This message is provided as a fallback that allows individualized drawer control during early development (e.g., to exit from bad states such as a drawer being open but locked). This message is applicable only to DSB nodes and is ignored by other node types on the bus. It is acted upon only by the DSB that owns the requested system-wide drawer index.

*Before issuing this message, the system solenoids (that control drawer locks) must be placed into **manual mode** via MT02. It is worth noting here that simply transitioning the system from the (normally active) automatic solenoid mode to manual will cause any energized solenoid (unlocked drawer) to deenergize (locked drawer). **This message is silently ignored if received when the solenoids are not in manual mode**.*

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	RESERVED		UNLOCK	INDEX				
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x08	
Unlock	N/A	0=lock the drawer, 1=unlock the drawer	Requested override operation
Index	N/A	0=reserved, 1..30, 31=unassigned	1-based system-wide index of the single drawer that is being overridden; generally, there is no use case where LS4 uses drawer indexes 0 or 31 in this message
CRC	N/A	Variable	0=CRC not in use (optional)

MT20 – SET FACTORY MODE

This **broadcast message** allows LS4 to globally drive all nodes into (or out of) Factory Test Mode (FTM). When in FTM, additional messages are enabled to support a variety of factory test and configuration operations. These messages are identified where appropriate. When a node is in normal mode, it silently ignores any FTM-specific message it receives. At initialization, all nodes are in normal mode (i.e., FTM does not persist through reboot/power cycle). DSB nodes include their mode in MT83 responses. CUPS4 will support a similar feature as needed when it is implemented.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	RESERVED							FTM
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x20	
FTM	N/A	0=normal mode, 1=factory test mode	Requested mode for all system nodes; all nodes start in normal mode at power on
CRC	N/A	Variable	0=CRC not in use (optional)

MT21 – Clear Drawer Indexes

This **broadcast message** allows LS4 to globally clear any previously-assigned drawer indexes for each DSB node. It is used during factory testing as a precursor to assigning drawer indexes (and bus addresses as a side effect) to each DSB node. The drawer index assignment operation is performed once for each V4 system and is thereafter retained by each node (i.e., the settings are retained in NV at each node). This message is applicable to DSB nodes only; all other node types may ignore it. Review the Address Assignment sample message flow for additional details.

This message is not accepted nor acted upon by any node that is not in Factory Test Mode (set via MT20). Refer to the Message Applicability Matrix for additional details.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x21	
CRC	N/A	Variable	0=CRC not in use (optional)

MT22 – ASSIGN DRAWER INDEX

This **broadcast message** allows LS4 to assign a drawer index to its associated DSB node. It is used during factory testing to populate DSB nodes with drawer-specific indices while also setting the node bus address. This assignment operation is performed once for each V4 system and is thereafter retained by each node (i.e., the settings are retained in NV at each node). This message is applicable to DSB nodes only; all other node types may ignore it. Review the Address Assignment sample message flow for additional details.

During drawer assignment, each V4 drawer is opened manually, one at a time, and an action is made in the UI to inform all DSB nodes of the drawer index assignment using this message. Given this is a broadcast message, each DSB module must evaluate its inventory of drawers and, finding one fully open, associate the received index with the open drawer. From that point forward, the DSB node "owns" the assigned drawer. As new drawers are added, each DSB node adopts its lowest numbered drawer index as its bus address. MT21 may be used to clear all of the drawer assignments (should be done before starting a new assignment procedure).

LS4 (and thus the host application) may poll drawer information while the DSB nodes are in FTM via the MT03 message. This provides a feedback path that can support confirmation as each drawer is added.

This message is not accepted nor acted upon by any node that is not in FTM (set via MT20). The message is also ignored by any node that is in BLM. Refer to the Message Applicability Matrix for additional details.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	RESERVED			INDEX				
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	31 (broadcast)	
Message-Type	N/A	0x22	
Index	N/A	0=~~invalid~~reserved, 1..~~13~~30, ~~14..15~~31=unassigned	1-based index of the single drawer that is currently open (all other drawers must remain closed during this assignment); generally, there is no use case where LS4 uses drawer indexes 0 or 31 in this message
CRC	N/A	Variable	0=CRC not in use (optional)

MT51 – GET DEBUG DATA

This message requests DSB debug data. It is generally not used in operational systems unless troubleshooting an problem. It is applicable to DSB nodes only. Refer to MTD1 for the response message.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	DATA-ID							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	1 (read)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	1..13 (node address)	
Message-Type	N/A	0x51	
Data-ID	N/A	0=reserved, 1=RS485 RX timeouts, 2=RS485 TX timeouts, 3=RS485 CRC errors, 4=RS485 framing errors, 5=RS485 RX overflows, 6=Sensor0 OSC Offset, 7=Sensor1 OSC Offset, 8=Sensor2 OSC Offset, 9=Sensor0 OSC Value, 10=Sensor1 OSC Value, 11=Sensor2 OSC Value, ~~6~~12..255=reserved	Identifies the type of debug data to be returned
CRC	N/A	Variable	0=CRC not in use (optional)

MT70 – SET BOOTLOADER MODE

This **broadcast message** allows LS4 to globally set nodes into the desired mode (usually BLM) ahead of a field upgrade operation. Given the significant differences in field upgrade for DSB nodes and CUPS4, there are some unique messaging characteristics associated with this message (discussed in the field definitions below). After sending this message, each node acting on it will store the requested mode in NV and automatically reboot itself so as to end up in the requested configuration (generally ahead of the field upgrade). Review the DSB Field Upgrade message flow for additional information on the process.

LS4 may use MT01 to confirm each node is in BLM before moving ahead with an upgrade. All DSB nodes are upgraded at the same time. CUPS4 is upgraded separately.

When broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing. This may induce multiple reset events in some nodes. This message may also be sent via direct address to CUPS4 in preparation for a field upgrade of that node.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	RESERVED							BLM
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	14 (CUPS4), 31 (broadcast)	When CUPS4 receives this message as a broadcast it ignores it; CUPS4 must be directly addressed by this message before it will act on it; conversely, all DSB nodes act on the broadcast message when received
Message-Type	N/A	0x70	
BLM	N/A	0=normal mode, 1=bootloader mode	In most field upgrade processes this field is set to 1 to drive the receiving nodes into BLM (the nodes automatically transition out of BLM on successful completion of an upgrade); the 0 value is provided as a means of *forcing* all nodes into normal mode to correct failures or debug the field upgrade process
CRC	N/A	Variable	0=CRC not in use (optional)

MT77 – DSB HEX RECORD

This **broadcast message** is used to send HEX records to all DSB nodes in the system as part of a field upgrade. The embedded DSB bootloader expects data in this form for writing to application flash during an update. This message is accepted and processed by DSB nodes in BLM only. It is ignored by CUPS4. **To give DSB nodes time to receive and process each HEX record, a 100ms delay is required between successive MT77s.** Review the DSB Field Upgrade message flow for additional information on the process.

To upgrade DSB nodes, LS4 reads the selected HEX file one line at a time constructing binary messages of this type that encode the HEX contents. HEX records in the file are textual and have the following standardized form:

:llaaaatt[dd...]cc

- :=record start
- **ll**=number of data bytes (see **dd**, DSB generally uses **ll**=0x10/16)
- **aaaa**=start address for data within the record
- **tt**=record type (**00**=data, **01**=EOF, **04**=MS16b of address, others)
- **dd**=data bytes (count specified in **ll** field, can be 0)
- **cc**=2's complement of 8-bit wide sum of all hex bytes ahead of **cc**)

For each record, LS4 confirms then skips "**:**" and converts ASCII to binary byte-by-byte (not including the EOL CR/LF characters) and populates the **Length** field and **HEX-Data** portion of this message accordingly. Each record received is written into application flash on the DSB. There is no feedback from DSB during this process. A CRC mechanism on the DSB processor will leave the node in BLM if a partial or corrupted update occurs.

On receipt of the final (EOF, **tt=01**) HEX record, DSB nodes update their mode in NV to normal and automatically reboot. After allowing for the DSBs to reboot, LS4 should confirm all are in normal mode using MT01 which may also be used to confirm the new firmware version.

As a special broadcast message (Address is 30), this message is sent only one time. It is treated as a broadcast message by all DSB nodes in BLM. DSB nodes in BLM will not generate asynchronous messages (e.g., MT99) thus there is far less risk of bus collisions in this mode.

	7	6	5	4	3	2	1	0
0	R/W	DATA-SIZE		ADDRESS				
1	MESSAGE-TYPE							
2	LENGTH							
3	HEX-DATA$_0$							
N+3	HEX-DATA$_N$							
N+4	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	3 (~~hex-record~~8-bytes)	HEX length is in **Length** field so this is placeholder value only
Address	N/A	30 (special broadcast)	
Message-Type	N/A	0x77	
Length	Bytes	Variable	Length of HEX record data that follows this byte
HEX-Data	N/A	HEX-record content	Binary representation of the HEX record to send
CRC	N/A	Variable	0=CRC not in use (optional)

Node-to-LS4 and Node-to-Node Messages

MT81 – Discovery Response

This message is sent in response to an MT01 request from LS4. It contains information about the node present at the requested bus address. LS4 uses this message and each valid response to map valid addresses to the hardware that lives at each address while also gathering information about each node. Review the Discovery sample message flow to better understand the process.

This message is supported when the node is in FTM or BLM. Review the Message Applicability Matrix for more information.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved				Node-Type			
DSB	*DSB-Specific Fields*							
3	Reserved			BLM	Drawer-Count			
4	Reserved			Drawer-Index$_0$				
5	Reserved			Drawer-Index$_1$				
6	Reserved			Drawer-Index$_2$				
7	Reserved							
8	Reserved							
CUPS4	*CUPS4-Specific Fields*							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
~~5~~9	Major				Minor			
~~6~~10	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	~~2~~ 3 (~~4~~8-bytes)	
Address	N/A	15 (LS4)	
Message-Type	N/A	0x81	
Node-Type	N/A	0..2=reserved, ~~1=DSB-1-Drawer,~~ ~~2=reserved,~~ 3=DSB-3-Drawer, 4..6=reserved, 7=CUPS4, 8..15=reserved	Node type at this address
DSB-Specific Fields			
BLM	N/A	0=normal mode, 1=bootloader mode	When in BLM, only a subset of messages (including this one) are supported (these messages are identified as such in this ICD); generally, a node in BLM at initialization has failed an upgrade which must be run again to completion before normal operations are possible
Drawer-Count	Drawers	0=reserved, 1, 3=number of drawers 2, 4..15=invalid	Defines the number of drawers managed by this node; valid for DSB nodes only
~~Last-Index~~	~~N/A~~	~~0=invalid,~~ ~~1..13,~~ ~~14=invalid,~~ ~~15=unassigned~~	~~Highest numbered drawer index tied to this node (same as First-Index if only one drawer supported); indexes are 1-based; valid for DSB nodes only~~
~~First~~Drawer-Index$_N$	N/A	0=invalid, 1..~~13~~30, ~~14=invalid,~~ ~~15~~31=unassigned	~~Lowest numbered~~System-wide drawer index tied to this node ~~(same as Last-Index if only one drawer supported)~~; indexes are 1-based; valid for DSB nodes only
CUPS4-Specific Fields			
TBD	TBD	TBD	TBD
Major	N/A	0..15	Major firmware version (version is read as Major.Minor) on the node; when in BLM the bootloader version is returned
Minor	N/A	0..15	Minor firmware version (version is read as Major.Minor) on the node; when in BLM, the bootloader version is returned
CRC	N/A	Variable	0=CRC not in use (optional)

MT83 – Drawer States Response

This message is sent in response to an MT03 request. It reports DSB drawer state information for each available DSB drawer. The validity of drawer information is based on the number of drawers supported (as reported in MT81). This message is generated by DSB nodes only.

It is anticipated that this data will be polled reasonably often (e.g., on the order of seconds) by LS4 to maintain states and statuses that are needed by the HAL API.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved			Drawer-Index$_0$				
~~2~~3	Sol0		Open0	Reserved	Position0			
4	Reserved			Drawer-Index$_1$				
~~3~~5	Sol1		Open1	Reserved	Position1			
6	Reserved			Drawer-Index$_2$				
~~4~~7	Sol2		Open2	Reserved	Position2			
8	Reserved							
~~5~~9	GUnlock	LUnlock	Reserved	Solenoids		Prox	FTM	Err
~~6~~10	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	~~2~~ 3 (~~4~~8-bytes)	
Address	N/A	15 (LS4)	
Message-Type	N/A	0x83	
Drawer-Index$_N$	N/A	0=invalid, 1..30, 31=unassigned	System-wide drawer index associated with local index N
SolN	N/A	0=not energized (locked), 1=holding (unlocked), 2=picking (unlocking), 3=failed (over/under cur.)	Solenoid (lock) status for drawer at local index N
OpenN	N/A	0=closed 1=open	Drawer local index N "openness" based on a windowed proximity range around the default "home" position of the drawer; if position is within the "home" window, *closed* is returned, otherwise *open* is returned
PositionN	mm	0..15 (or ~0.6in) *Accuracy is ± TBDmm*	Position of drawer local index N from the proximity sensor; a value of 15 may indicate the drawer is more than 15mm away
GUnlock	N/A	0=global lock active, 1=global unlock active	State of global unlock; can be set with MT02
LUnlock	N/A	0=local lock active 1=local unlock active	State of local unlock (changes based MT99s from other DSBs)
Solenoids	N/A	0=disable, 1=enable, auto mode, 2=enable, manual mode, 3=reserved	State of global solenoid enable; can be set with MT02
Prox	N/A	0=disable, 1=enable	State of global proximity enable; can be set with MT02
FTM	N/A	0=normal mode, 1=factory test mode	In order to accept Factory Test messages, the node must be in Factory Test Mode; this flag indicates the node's present mode
Err	N/A	0=no errors present, 1=errors present	Errors present flag; when present LS4 may use MT05 to read/clear them
CRC	N/A	Variable	0=CRC not in use (optional)

MT84 – Drawer Temperature Response

This message is sent in response to an MT04 request. It returns the most recent DSB temperature reading. The returned temperature is not drawer-specific.

It is anticipated that this data will be polled infrequently (e.g., on the order of minutes) by LS4 to maintain temperature reporting needed by the HAL API.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Temperature							
3	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	0 (1-byte)	
Address	N/A	15 (LS4)	
Message-Type	N/A	0x84	
Temperature	°C	-40..125	DSB temperature reading (as a signed 8-bit integer)
CRC	N/A	Variable	0=CRC not in use (optional)

MT85 – Error Log Response

This message is sent in response to an MT05 request. It returns the most recently logged node errors (if any) and automatically clears the entire node error log (after sending the message). An error count is included to identify which of the error codes contain useful data (i.e., if the count is 1, only the first code [Error0] is valid).

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Error0				Error-Count			
3	Error2				Error1			
4	Error4				Error3			
5	Error6				Error5			
6	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	2 (4-bytes)	
Address	N/A	15 (LS4)	
Message-Type	N/A	0x85	
Error-Count	N/A	0..7=count, 8..15=invalid	Number of error codes included in this message
ErrorN	N/A	0=no error, 1..15=error code	The Nth logged error code where N is defined by Error-Count field; error codes are defined in Node Error Codes
CRC	N/A	Variable	0=CRC not in use (optional)

MT99 – Drawer State Change

This is an asynchronous **broadcast message** that conveys a change in state for one DSB drawer. DSB nodes send this message on first detecting inward motion of a closed drawer (the "push-to-open" event) and again when the open drawer is closed. This message originates from DSB nodes only and is ignored by other node types.

At drawer opening, this message is sent only if the global lock is **unlocked** (see MT02). All other drawers in the system (regardless of their DSB node association) remain locked while the one reporting the event is allowed to open. If the global lock is **locked**, this message is not sent and the drawer is not permitted to open.

All other DSB nodes, on receipt of this message indicating a drawer is opening, disallow opening of any other drawer regardless of global lock status until a corresponding close event is received for the same drawer number (or until a TBD timeout occurs). Conversely, if the message indicates the drawer is now closed, all DSB nodes return to using the global lock state to determine their actions.

This message is not generated nor processed by any DSB node in FTM or BLM. In FTM, LS4 may poll this information via MT03 as needed. Review the Message Applicability Matrix for more information.

As a broadcast, this message is sent multiple times to ensure it gets through given there is a risk of bus collision. Refer to Message Processing Rules for details/timing.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	Reserved			Index				
3	Sol		Open	Event	Position			
4	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	1 (2-bytes)	
Address	N/A	31 (broadcast)	LS4 and all other DSB nodes in the system receive and must act on this message
Message-Type	N/A	0x99	
Index	N/A	0=~~invalid~~reserved, 1..~~13,~~ ~~14=invalid,~~ ~~15~~31=unassigned	System-wide ~~D~~drawer index that's generating the event; indexes are 1-based; the indexes are defined for each DSB node in MT81
Sol	N/A	0=not energized (locked), 1=holding (unlocked), 2=picking (unlocking), 3=failed (over/under cur.)	Solenoid (lock) status for the drawer
Open	N/A	0=closed 1=open	Drawer "openness" based on a windowed proximity range around the default "home" position of the drawer; if position is within the "home" window, *closed* is returned, otherwise *open* is returned
Event	N/A	0=unlock event, 1=lock event	Type of drawer event that's being reported; any given drawer generally transitions from unlock to lock as it is opened/closed by a user so both events are generated and important
Position	mm	0..15 (or ~0.6in) *Accuracy is ± TBDmm*	Position of drawer from the proximity sensor; a value of 15 may indicate the drawer is more than 15mm away
CRC	N/A	Variable	0=CRC not in use (optional)

MTD1 – Debug Data Response

This message is sent in response to an MT51 request. It always returns the requested debug data as a 32-bit unsigned integer.

	7	6	5	4	3	2	1	0
0	R/W	Data-Size		Address				
1	Message-Type							
2	$Data_{MSB}$							
3	Data							
4	Data							
5	$Data_{LSB}$							
6	CRC							

Field	Units	Range/Format	Notes
R/W	N/A	0 (write)	
Data-Size	Bytes	2 (4-bytes)	
Address	N/A	15 (LS4)	
Message-Type	N/A	0xD1	
Data	N/A	Variable	32-bit unsigned integer corresponding with the Data-ID requested in MT51
CRC	N/A	Variable	0=CRC not in use (optional)

Sample Message Flows

Discovery

During discovery, LS4 learns the mapping of nodes to bus addresses while also gathering static configuration information from each node found. This flow walks through the basic steps that occur during discovery.

LS4	Address	Node	Notes
MT02	**Broadcast**		LS4 issues a global **lock**, global solenoid **disable**, and global proximity **disable** to all DSB nodes to ensure they do not report MT99 (on drawer change) during discovery
MT01	**1**	**MT81**	LS4 gets response, maps Address 1 as valid and gathers the associated node information (a single drawer DSB)
MT01	**2**	**MT81**	LS4 gets response, maps Address 2 as valid and gathers the associated node information (a triple drawer DSB)
MT01	**3**		LS4 gets no response (it does not retry), assumes no node at Address 3, updates its address mapping accordingly
MT01	**4**		LS4 gets no response (it does not retry), assumes no node at Address 4, updates its address mapping accordingly
MT01	**5**	**MT81**	LS4 gets response, maps Address 5 as valid and gathers the associated node information (a triple drawer DSB)
• • •	• • •	• • •	Process continues in the same manner for all node addresses
MT01	**13**		LS4 gets no response (it does not retry), assumes no node at Address 13, updates its address mapping accordingly
MT01	**14**	**MT81**	LS4 gets response from CUPS4 and gathers its node information; **a missing CUPS4 response is an error**
MT02	**Broadcast**		LS4 issues global **unlock**, global solenoid **enable**, and global proximity **enable** to all DSB nodes after completion of discovery.

Address Assignment

Bus addresses and drawer number assignments occur in a factory test setting. This flow walks through a typical procedure and associated message sequence.

LS4	Address	Node	Notes
MT20	**Broadcast**		LS4 broadcasts message to put all nodes into FTM
MT21	**Broadcast**		LS4 broadcasts message to clear addresses and drawer assignments for all (DSB) nodes; CUPS4 ignores this
User Action			User instructed to ensure all drawers are **fully** closed
User Action			User instructed to **fully** open **Drawer 1**; one DSB node takes note it has an open drawer but does not send MT99 given the node is in FTM
MT22	**Broadcast**		On user confirmation that **Drawer 1** is open, LS4 broadcasts message that informs all nodes **Drawer 1** is open; the single DSB with an open drawer assumes ownership of **Drawer 1** and records this in NV while also adopting **Address 1** as its bus address; if this DSB is subsequently assigned a *lower-numbered* drawer (not applicable in this example, Drawer 1 is lowest index), it would adopt that value as its bus address; CUPS4 ignores this message
User Action			User instructed to ensure all drawers are **fully** closed
• • •	• • •	• • •	Process continues for each drawer in the system until completion after which the Discovery message flow is run to confirm all drawer indexes are assigned as intended
MT20	**Broadcast**		LS4 broadcasts message to put all nodes in normal mode

DSB Field Upgrade

This flow documents a typical DSB firmware upgrade operation. It is worth noting in the flow below that DSB nodes never send messages on the bus during the upgrade. All messaging is from LS4. Confirmation of success occurs only at the end of the upgrade process after all DSB nodes reboot (presumably with the new firmware version reflected in the Discovery process at the end).

LS4	Address	Node	Notes
MT70	**Special Broadcast**		LS4 broadcasts message to put all DSB nodes into BLM; CUPS4 ignores this message when broadcast, it must be addressed directly to CUPS4 to drive it into BLM ahead of a field upgrade for that node type
Node Action			On receipt, each DSB node sets BLM in NV and reboots itself (coming up in BLM)
LS4 Action			After a **1-second** delay, the Discovery message flow is run to confirm all DSB nodes are in BLM
LS4 Action			LS4 opens the selected DSB upgrade HEX file. The file is read and parsed one line at a time following the guidelines provided in the MT77 description. LS4 needs little awareness of the HEX record contents with the exception of tracking the EOF HEX record (generally the last record in the file)
MT77	**Special Broadcast**		LS4 broadcasts a binary HEX record message for each line in the HEX file; LS4 must maintain **100ms** spacing between sequential MT77s to give time for message processing at each DSB node
⋮	⋮	⋮	Process continues until all HEX records (except EOF) have been sent; each DSB node validates every HEX record it receives and will not write a failed HEX record to application flash. Failures generally lead to the DSB staying in BLM (or, if the node reboots, returning to BLM due to application flash CRC failure).
MT05	**Each DSB**	**MT85**	Just before sending the EOF HEX record, LS4 polls for errors from *each DSB node*; any errors will likely cause upgrade at the reporting node to fail so the most that can be done is logging/sharing upstream before sending the EOF and closing out the process even if errors exist.
Node Action			On receipt of the EOF HEX record, each DSB node sets normal mode in NV and reboots itself (coming up in normal); if a node reboots with corrupted application flash, it will automatically revert to BLM until a good code load can be accomplished.
LS4 Action			On sending the EOF HEX record or the last HEX record in the file, LS4 delays **1-second** then runs the Discovery message flow to confirm all DSB nodes are in normal mode and reporting the expected (new) version

Message Applicability Matrix

The table below summarizes message applicability for each message on the bus. An open cell indicates *not applicable* and a green cell indicates *applicable*. **Normal**, **FTM**, and **BLM** are possible node modes. **B-Cast** is applicable for any message this may be broadcast. **DSB** and **CUPS4** show applicability for those node types.

Message	Normal	FTM	BLM	B-Cast	DSB	CUPS4
MT01 – Discovery	■	■	■		■	■
MT02 – Global	■	■		■	■	
MT03 – Get Drawer States	■	■			■	
MT04 – Get Drawer Temperature	■	■			■	
MT05 – Get and Clear Error Log	■	■	■		■	■
MT06 – Global Reset	■	■	■	■	■	■
MT07 – Global Drawer Recalibration	■	■		■	■	
MT08 – Drawer Override	■	■		■	■	
MT20 – Set Factory Mode	■	■		■	■	■
MT21 – Clear Drawer Indexes		■		■	■	
MT22 – Assign Drawer Index		■		■	■	
MT51 – Get Debug Data	■	■	■		■	
MT70 – Set Bootloader Mode	■	■	■	■	■	■
MT77 – DSB Hex Record			■	■	■	
MT81 – Discovery Response	■	■	■		■	■
MT83 – Drawer States Response	■	■			■	
MT84 – Drawer Temperature Response	■	■			■	
MT85 – Error Log Response	■	■	■		■	■
MT99 – Drawer State Change	■			■	■	
MTD1 – Debug Data Response	■	■	■		■	

Node Error Codes

The following error codes may be reported within messages defined in this ICD.

Code	Definition
0	No error
1	**Proximity Sensor Failure** – (applicable to DSB only) a proximity sensor had more than 5 consecutive CRC or conversion-done timeout failures; the specific sensor is not identified (could be any one of up to three); *error is reported only one time per sensor*
2	**Solenoid Failure** – (applicable to DSB only) an over-current or under-current state occurred when a solenoid was energized (during the 'pick' phase of unlocking a drawer); the specific solenoid is not identified (could be any one of up to three); *once failed, the solenoid is not used again unless the corresponding DSB is restarted*
3	**Bad RS485 Message Type** – an invalid or undefined message type was received by the node.
4	**I2C Timeout** – transaction timeout occurred when attempting communication with an I2C end-point on the node (e.g., temperature sensor on DSB)
5	TBD
6	TBD
7	TBD
8	TBD
9	TBD
10	**HEX Record Invalid** – received HEX record with invalid format or length
11	**HEX Record Bad Checksum** – received HEX record with an invalid embedded checksum
12	**HEX Record Misaligned** – received HEX record that was not on a HEX record or flash page boundary
13	**HEX Record Discontinuity** – received HEX record that was not adjacent to the last HEX records' address
14	**HEX Record Bad EOF** – received end-of-file HEX record with unwritten data or no HEX records received
15	**HEX Record Write Failed** – write to flash memory failed during a field upgrade

CRC8 Sample Code

This section provides a sample implementation (in C) for computation of the CRC used in serial messages on the bus. This code is used verbatim on DSB and CUPS4 nodes.

```
/*****************************************************************************
* FUNCTION:         CRC8
*
* DESCRIPTION:      Computes 8-bit CRC.  Polynomial is X^8 + X^5 + X^4 + 1
*                   and seed is 0.
*
* PARAMS:           a_u8DataPtr - pointer to first byte of data
*                   a_u8Count - number of bytes to include
*
* RETURN:           8-bit CRC value
*
*****************************************************************************/
static uint8_t CRC8(uint8_t* a_u8DataPtr, uint8_t a_u8Count)
{
  uint8_t           u8CRC = 0;
  uint8_t           u8Byte;
  uint8_t           u8Bit;
  uint8_t           u8Worker;
  uint8_t           u8Mix;

  for (u8Byte = 0; u8Byte < a_u8Count; u8Byte++)
  {
    //get a byte to work with
    u8Worker = a_u8DataPtr[u8Byte];

    //roll over its bits making CRC magic
    for (u8Bit = 0; u8Bit < 8; u8Bit++)
    {
      u8Mix = (u8CRC ^ u8Worker) & 0x01;
      u8CRC >>= 1;
      if (u8Mix)
      {
        u8CRC ^= 0x8C;
      }

      u8Worker >>= 1;
    }
  }

  return u8CRC;
}
```