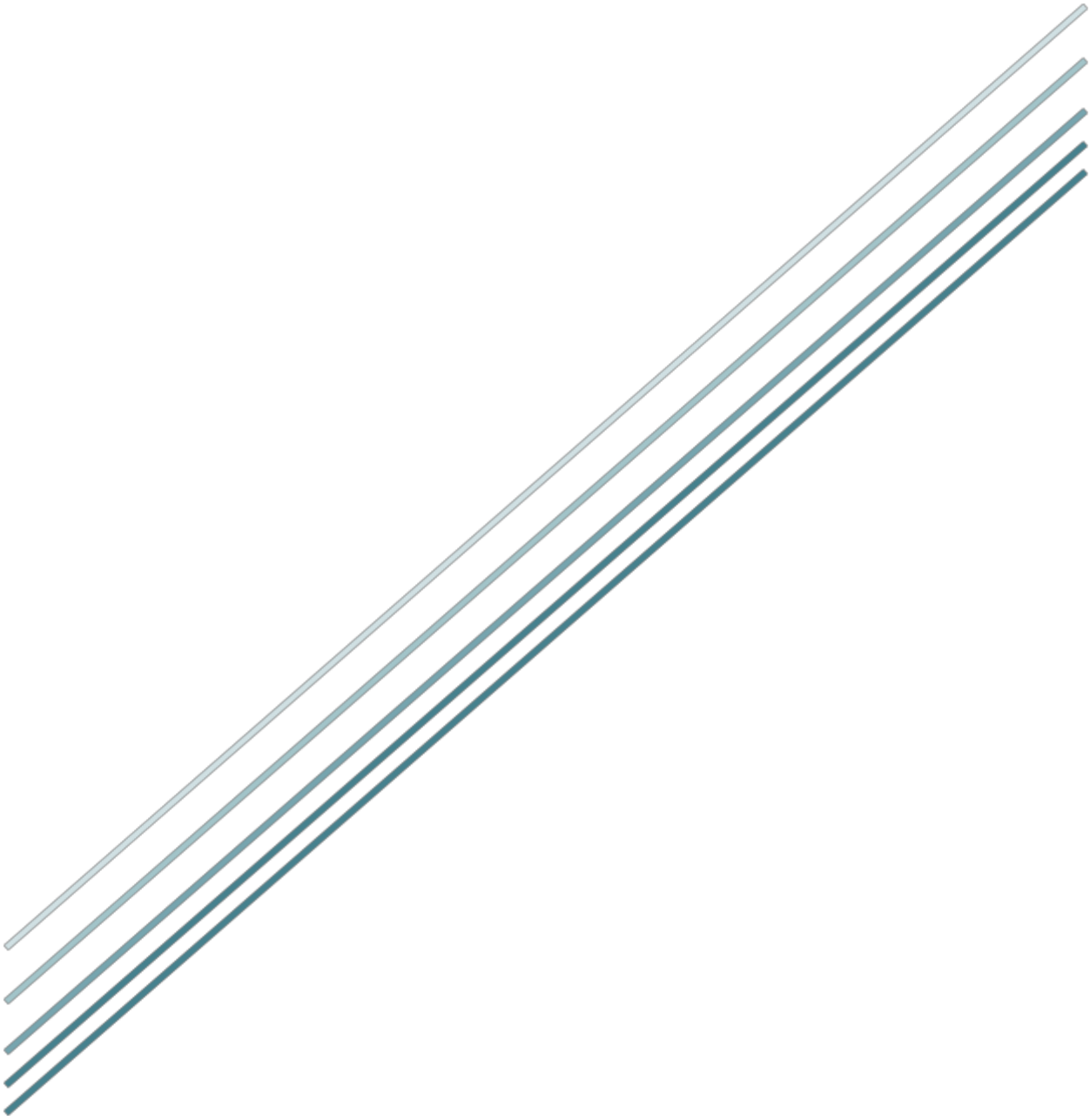
COMMUNICATION PROTOCOL

ME218C, SPRING 2014
Rev A



Revision History

Revision A	Initial Documentation
Revision B	1. Added tag out location as COACH required I/O. 2. Added tag out location to TAG_OUT message data. 3. Added tag out location to TAG_DETECTED message data. 4. Added STATUS message data definition.
Revision C	1. Added sequence diagrams for pairing process. 2. Added basic state chart for COACH / PLAYER communication system. 3. Enforced 5Hz update rate for STATUS. 4. RESPDATA Error bits now defined as zero in case of pairing accepted. 5. Added summary table of message bytes.
Revision D	1. Fixed typo in summary table of messages. 2. Explicitly require PLAYER to synthesize STATUS data from differer LiFKIM bytes. 3. Added note in XBee data frame description referring to how to obtain radi address for “destination address.” 4. Added message type to message definitions: SEND for messages sent to a specific, paired address, and BROADCAST for messages sent to all devices on the network. 5. Updated state charts.
Revision E	1. Fixed typo in message summary table, direction of REQ_PAIR now agrees with definition.

Table of Contents

Definition of Terms

COACH: One of the controllers constructed by each team.
PLAYER: One of the hovercraft constructed by each team.
Tag Out: The process by which a PLAYER goes from being Active to being On Deck.
Broadcast: A message sent to all radio devices on the network/sending to all devices on the network. See TX Request Packet.
Send: Unless otherwise specified, this shall refer to sending a message to a single other device whose address is known.
Pairing error: When the PLAYER denies a pair request, but there is no communication breakdown.

Communications Overview

Broadly speaking, communication will be in two modes. The radios will first handshake to form a pairing, and then begin exchanging status and control data.

Pairing

COACHes may only control PLAYERs to whom they are paired. This pairing is generated whenever a COACH wishes to control an unpaired PLAYER on its team. The COACH and PLAYER will execute a handshaking protocol initiated by the COACH. If the PLAYER confirms, then that COACH and PLAYER will be paired, and will only communicate with each other until the pairing is broken. This can be accomplished by going through the Tag Out process, or by issuing a RESET command from the COACH.

Control

Once paired, the COACH and PLAYER will begin exchanging information. The COACH will be required to send control data at 5 Hz, and the PLAYER will be required to send status information at 5 Hz. PLAYERs and COACHes should ignore control messages that are erroneously sent to them by an unpaired device.

Retries

There will be no requirement in the protocol for retries of messages. In the case of the status and control packets, the desired response to a dropped packet is not to resend that packet, but rather the most current one. In the case of all other messages, they are triggered by physical events, and retrying will involve causing the event again. For example, if a COACH presses the pair button, and there is no indication of a successful pairing, then the user will press the pair button again to retry connecting.

Communications Failure

The combination of control and status information packets will serve as a communications heartbeat in each direction. If at any time a COACH or PLAYER fails to receive messages from their paired device for a period of 1 second, then that device is required to break the pairing and reinitialize to its boot-up state.

Hardware Requirements

In order to implement the minimum functionality described in this document, there are minimum I/O requirements imposed on the PLAYER and COACH.

PLAYER

The PLAYER

- **Shall** provide a visual indication when maintaining an active pairing with a COACH.
- **Shall** provide a means to set which team it is playing for.
- **Recommended** to provide a visual indication of which team it is playing for.

COACH

The COACH

- **Shall** provide a means to indicate a robot jersey number from 1-13, inclusive.
- **Shall** provide a means to select team color, red or green.
- **Shall** provide a visual indication when maintaining an active pairing to a PLAYER.
- **Shall** provide a means to trigger a message indicating the COACH wishes to tag out.
- **Shall** provide a means to select a tag out location from 0-3, inclusive.
- **Shall** provide a means to trigger a reset message.
- **Recommended** to provide a visual indication when a pairing error has occurred.
    - **May** also differentiate between types of pairing errors.
- **May** provide a visual indication when a general communication error has occurred.
- **May** provide a visual indication when a successful tag-out has occurred.

Pairing Protocol

A pairing must be initiated by a COACH, and only when the corresponding button is pressed. At no time may a COACH automatically attempt a pairing to a PLAYER.

To initiate a pairing, the COACH will broadcast a REQ_PAIR message. This message will contain the jersey number of the PLAYER to which the COACH wishes to pair, and the team color that the coach is playing for. A REQ_PAIR may only be broadcast if the COACH is not paired. The COACH will then enter a WAIT_PAIR state to monitor for pairing responses.

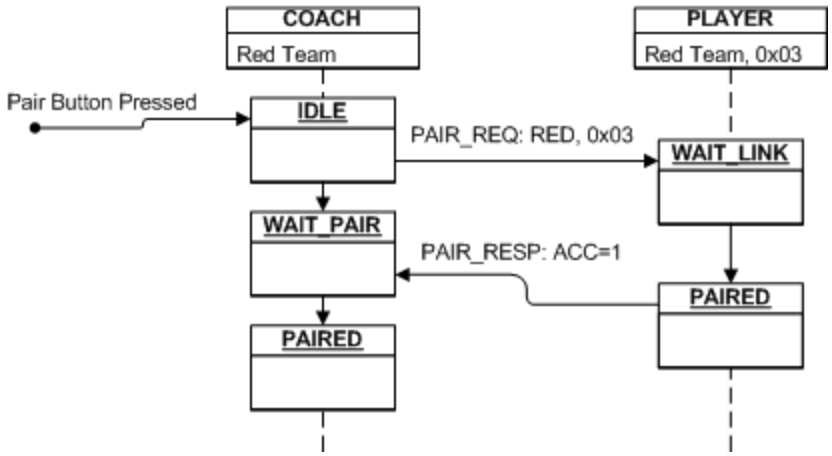
On receipt of a REQ_PAIR message, a PLAYER shall:

1. If the jersey number contained in the message **does not match** the PLAYER's stored jersey number, ignore the message.
2. If the PLAYER's jersey number **matches** that in the REQ_PAIR message, the PLAYER shall reply with a PAIR_RESP message:
    a. If already paired to a COACH, the PAIR_RESP message shall indicate that the pairing is denied, and that the reason for the error is that the PLAYER is already paired.
    b. If not already paired, but the team color in the REQ_PAIR message does not match the PLAYER's stored team color, the PAIR_RESP message shall indicate that the pairing is denied, and that the reason for the error is that the wrong team color was given.
    c. If not already paired, and the team color matches, the PAIR_RESP message shall indicate that the pairing is accepted.
3. At this point, the PLAYER will form a pairing with the COACH, before processing any further messages from serial, by:
    a. Leaving the WAIT_LINK state.
    b. Storing the source radio address of the REQ_PAIR message as the radio address of the paired COACH.
    c. Only respond to control commands from the paired COACH.
    d. Begin sending status messages at 5 Hz.
    e. Begin monitoring heartbeat timeouts to maintain the active pairing.
    f. The PLAYER is now permitted to hover.
4. If the COACH receives a PAIR_RESP message that indicated the pairing has been accepted, the COACH will form a pairing with the PLAYER, before processing any further messages from serial, by:
    a. Leaving the IDLE state.
    b. Storing the source radio address of the PAIR_RESP message as the radio address of the paired PLAYER.
    c. Only use status commands from the paired PLAYER.
    d. Begin sending control messages at 5 Hz.
    e. Begin monitoring heartbeat timeouts to maintain the active pairing.

5. If the COACH does not receive a pairing confirmation for 1 second while in the WAIT_PAIR state, then it will revert to the IDLE state and ignore incoming messages.

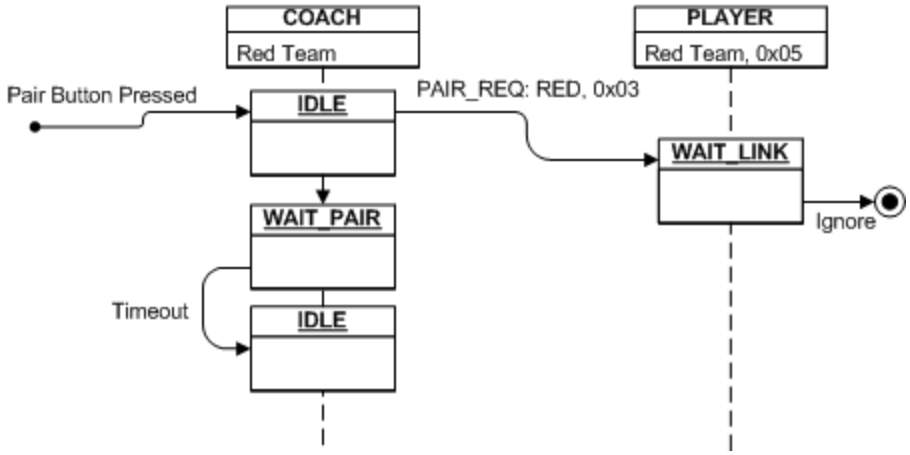
The PLAYER changes state atomically, and so does not need to implement timeouts specific to the pairing process. Sequence diagrams of possible pairing scenarios follow.

Successful Pairing



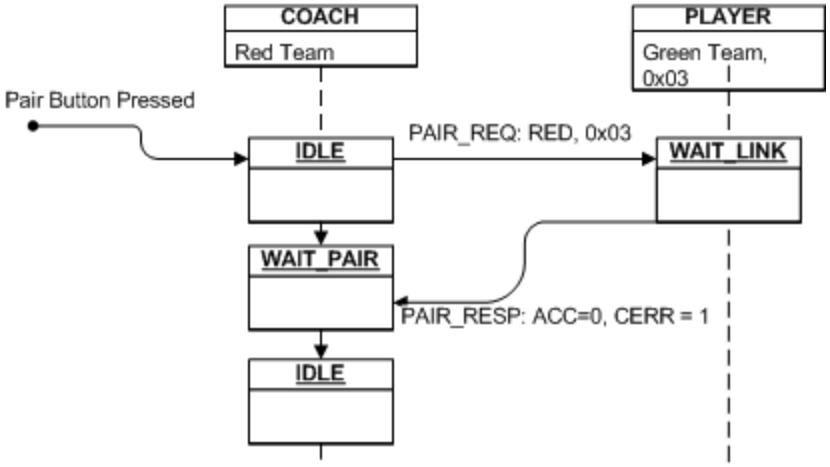
Once paired, the COACH and PLAYER begin exchanging CTRL and STATUS information.

Mismatched Jersey Number



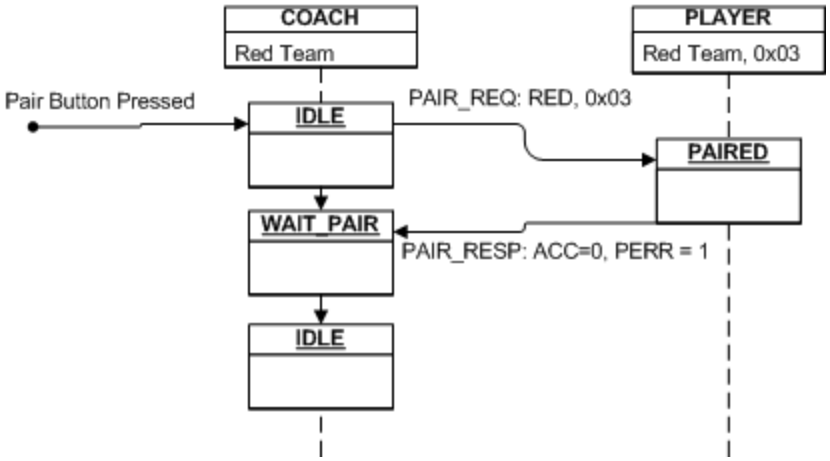
The PLAYER reads the message, and because the jersey number is not matched, ignores it. If there are no other PLAYERs with a matching jersey number, the COACH will timeout to IDLE.

Team Color Mismatch



The PLAYER and COACH are on opposite teams, so the PLAYER denies the pairing and remains in the WAIT_LINK state.

Already Paired



The PLAYER receives a PAIR_REQ while already paired; denies the pairing with the error, and remains paired to the previous COACH.

Communications Failure Detection

Once paired, both PLAYERs and COACHes shall monitor the communications link, and unpair and reset systems if they detect that the communications link has failed.

This will be implemented by monitoring heartbeat messages in both directions. The CTRL and STATUS packets will serve as heartbeats, in the COACH-PLAYER and PLAYER-COACH directions respectively.

When a device enters the PAIRED state, it will initialize a timer to trigger in one second. Upon receipt of a heartbeat message, the device will reinitialize the timer to trigger in one second from receipt of the heartbeat. If at any point the timer triggers (4 consecutive missed heartbeats), then the device will assume that communications have failed, and re-initialize as follows:

COACH:

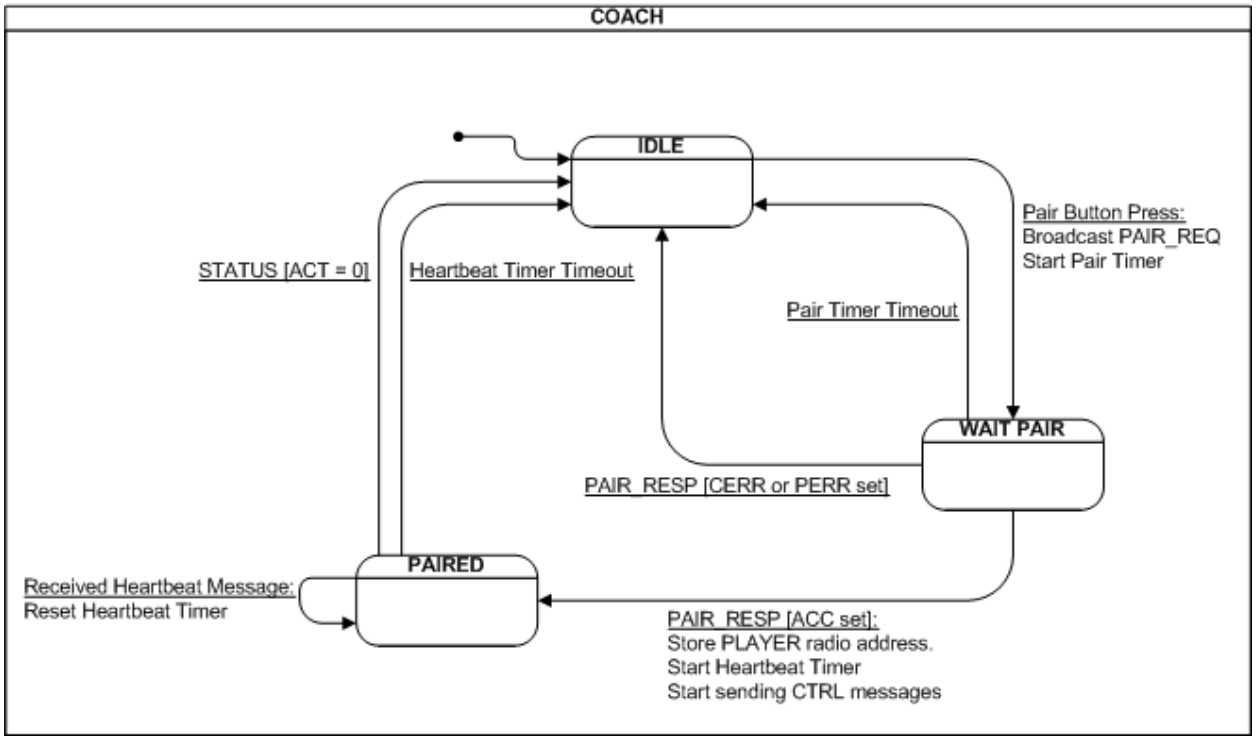
1. Clear any displays associated with the PLAYER status.
2. Clear the display of the active communications link.
3. Break the pairing with the PLAYER, and stop sending data
4. Move to the IDLE state.

PLAYER:

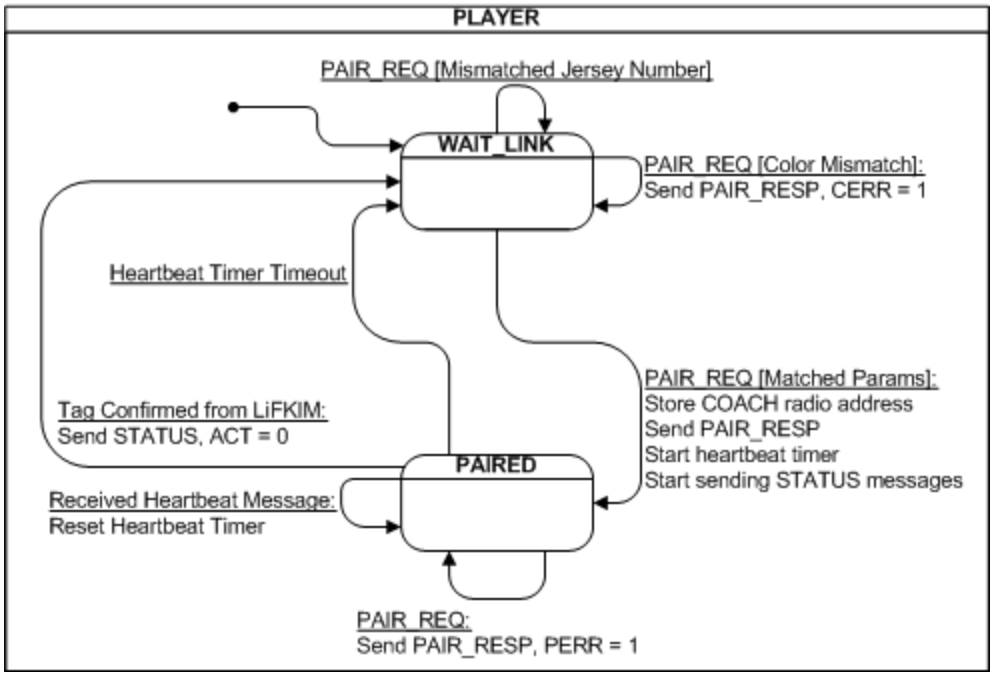
1. Reset all mechanical systems to their neutral positions.
2. Set all propulsion systems to zero thrust.
3. Clear the display of the active communications link.
4. Break the pairing with the COACH, and stop sending data.
5. Move to the WAIT_LINK state.

Communication System State Charts

COACH



PLAYER



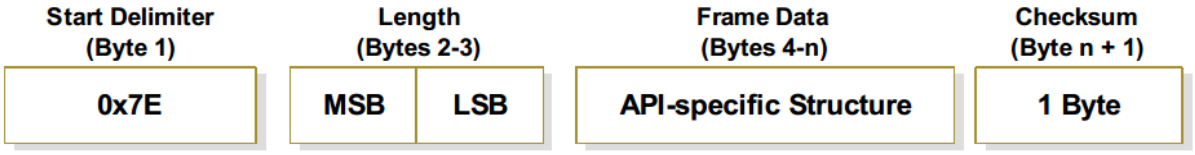
Data Structure Definition

The following sections define the data structures to be used for communications between the COACHes and PLAYERs. All communication between these classes of devices will take place over a Zigbee radio network operating at 9600 baud.

Zigbee Protocol Definition

ME218C will be using the radio devices in the non-beacon API mode of operation. The following API data frames will be used in the communication protocol specified in this document:

Common UART data frame structure in API mode:

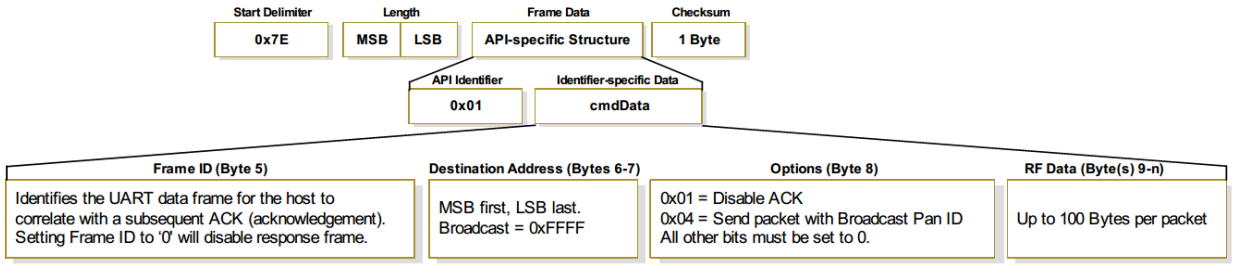


The data frame for all communication between the microprocessor and the XBee has a common basic structure. The checksum is defined as (0xFF - ∑ bytes in frame data).

The frame data contains a Frame ID byte. This can be set independently for each frame of data. However, in the 218C remote-control application, the response for a lost control or status packet will be to send the next control or status packet, so the Frame ID is not specified by this protocol, and is available for teams to use for troubleshooting. It is strongly recommended, however, that Frame ID not be set of 0x00, as this disables acknowledgement messages from the XBee back to the microcontroller.

TX Request Packet

This is the request-to-send frame sent from the microcontroller to the XBee radio.



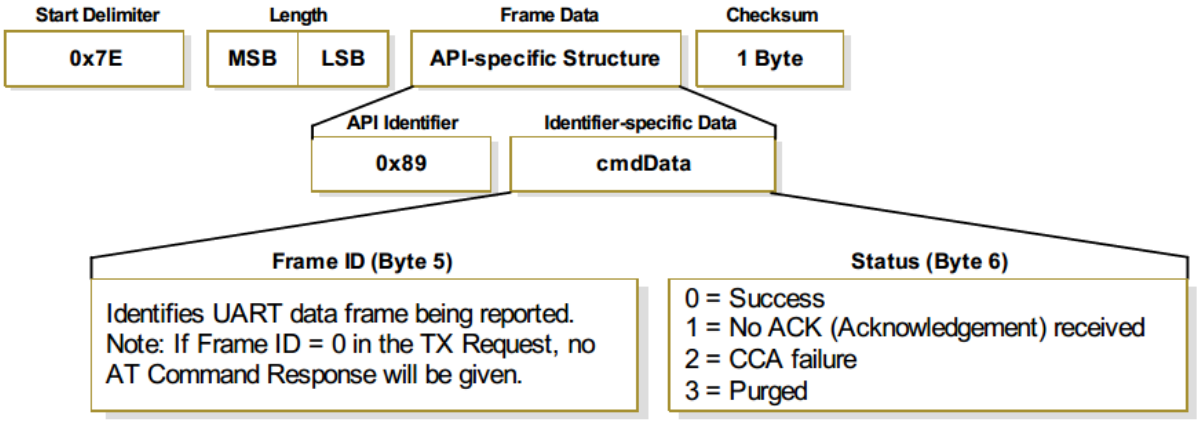
This will be used in two ways, either to send a broadcast message, or to send a message to a specific device.

To broadcast a message, the destination address should be set to 0xFFFF, and the options byte should be set to 0x04.

To send a message, the destination address should be set to the stored address of the paired device, and the options byte should be set to 0x00 so that ACK is enabled. Refer to Pairing Protocol for the process of obtaining the destination radio address.

TX Status Packet

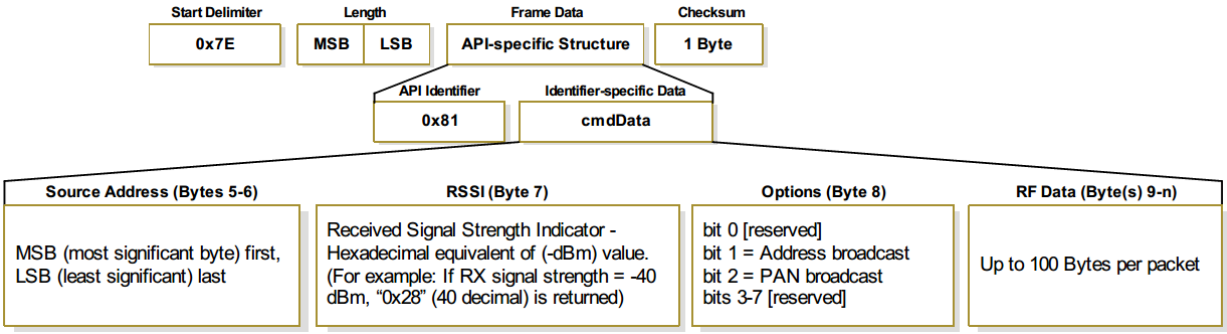
This is the frame sent from the XBee to the microcontroller on attempt to send a packet.



The status should be confirmed to be 0.

RX Packet

This is the frame sent from the XBee to the microcontroller on receipt of a data frame from a different radio.



Note that bytes 5-6 contain the source address, which is used in the handshaking protocol to create pairings between devices. Byte 8 should be checked to determine if the message was broadcast.

ME218C Data Byte Definition

The ME218C data is contained in the RF data section of each of the above Zigbee API data frames. The RF data will consist of an initial message header byte specifying the type of the message, followed by anywhere from 0 to 5 additional bytes of data.

ME218C Message Specifications

The ME218C communication protocol defines the following types of messages to be placed in RF data:

Summary

Name	DIR	HDR	Data [BIT7 ... BIT0]
RESET	C → P	0x04	NONE
TAG_OUT	C → P	0x02	TAGINFO: [- - - - - POS1 POS0 COL]
TAG_DETECTED	F → A	0x07	TAGINFO: [- - - - - POS1 POS0 COL]
STATUS	P → C	0x06	SDATA: [ACT - - LIFT1 LIFT0 ENR2 ENR1 ENR0]
REQ_PAIR	C → A	0x03	PAIRDATA: [- - - COL RID3 RID2 RID1 RID0]
PAIR_RESP	P → C	0x05	RESPDATA: [- - - - - PERR CERR ACC]
CTRL	C → P	0x01	SPD: [SPD7 SPD6 SPD5 SPD4 SPD3 SPD2 SPD1 SPD0] DIR: [DIR7 DIR6 DIR5 DIR4 DIR3 DIR2 DIR1 DIR0] AUX: [BRK3 BRK2 BRK1 BRK0 KICK3 KICK2 KICK1 KICK0] WHIM: [WHIM7 WHIM6 WHIM5 WHIM4 WHIM3 WHIM2 WHIM1 WHIM0] TRANS: [TR7 TR6 TR5 TR4 TR3 TR2 TR1 TR0]

RESET

Header: **0x04**
Direction: *COACH to PLAYER*
Type: *SEND*
Additional Data: NONE

The RESET message may be sent from a COACH to a paired PLAYER. A PLAYER receiving this message should first reset all of its hardware systems to the idle/inactive state, and then break the pairing with its COACH and reinitialize to the WAIT_LINK state. The PLAYER should reset hardware as for a communication failure.

TAG_OUT

Header: **0x02**
Direction: *COACH to PLAYER*
TYPE *SEND*
Additional Data: TAGINFO

TAGINFO							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
-	-	-	-	-	**POS1**	**POS0**	**COL**
Bit 7							*Bit 0*

The TAG_OUT message may be sent from a COACH to a paired PLAYER. A PLAYER receiving this message should translate and relay this message to its on-board LiFKIM. If a PLAYER is informed by its LiFKIM that is has been tagged out, it should immediately send a STATUS message to its COACH before unpairing.

COL is the team color of the COACH and PLAYER.

COL = 0: Red team
COL = 1: Green team

POS<0:1> is the tag-out location.

POS = 00: Position 0
POS = 01: Position 1
POS = 10: Position 2
POS = 11: Position 3

TAG_DETECTED

Header: **0x07**
Direction: *FIELD to ALL*
TYPE *BROADCAST*
Additional Data: TAGINFO

TAGINFO							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
-	-	-	-	-	**POS1**	**POS0**	**COL**
Bit 7							*Bit 0*

The TAG_DETECTED message will be broadcast from the FIELD to all devices. A PLAYER receiving this message should translate and relay this message to its on-board LiFKIM.

COL is the team color of the tagged side.

COL = 0: Red team
COL = 1: Green team

POS<0:1> is the tag-out location.
POS = 00: Position 0
POS = 01: Position 1
POS = 10: Position 2
POS = 11: Position 3

STATUS

Header: **0x06**
Direction: *PLAYER to COACH*
Type *SEND*
Additional Data: SDATA

SDATA							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
ACT	**-**	**-**	**LIFT1**	**LIFT0**	**ENR2**	**ENR1**	**ENR0**
Bit 7							*Bit 0*

The STATUS message will be sent from a PLAYER to a paired COACH. This message will be used to send status information to the COACH, and will also function as a heartbeat to monitor the communications link from PLAYER to COACH. The PLAYER will need to synthesize this information from the various status bytes sent from the LiFKIM.

ENR<0:2> is the last reported energy level from the LiFKIM.
ENR = 001: Energy level 1
:
ENR = 111: Energy level 7
Note: The LiFKIM will not report an energy level of zero.

LIFT<0:1> is the lift level.
LIFT = 00: Lift level 0
:
LIFT = 11: Lift level 3

ACT is the current state of the LiFKIM
ACT = 0: LiFKIM is currently inactive / tagged out.
ACT = 1: LiFKIM is currently active.

REQ_PAIR

Header: **0x03**
Direction: *COACH to ALL*
Type *BROADCAST*
Additional Data: PAIRDATA

PAIRDATA							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
-	-	-	**COL**	**RID3**	**RID2**	**RID1**	**RID0**
Bit							*Bit 0*

The REQ_PAIR message may be broadcast from a COACH looking to pair to a PLAYER.
RID<0:3> is the jersey number of the PLAYER to which the COACH is attempting to pair.
COL is the team color of the COACH.

COL = 0: Red team
COL = 1: Green team

PAIR_RESP

Header: **0x05**
Direction: *PLAYER to COACH*
Type *SEND*
Additional Data: RESPDATA

RESPDATA							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
-	-	-	-	-	**PERR**	**CERR**	**ACC**
Bit 7							*Bit 0*

The PAIR_RESP message is sent from a PLAYER in response to a COACH's REQ_PAIR message. The error bits should be zero if the pairing is accepted.
ACC indicates whether the pairing is accepted.

ACC = 0: Pairing is denied
ACC = 1: Pairing is accepted

CERR Indicates whether a color mismatch error occurred.
CERR = 0: No error
CERR = 1: Team color mismatch
PERR Indicates whether a pairing error occurred.
PERR = 0: No error
PERR = 1: PLAYER already paired

CTRL

Header: **0x01**
Direction: *COACH to PLAYER*
Type *SEND*
Additional Data: SPD DIR AUX WHIM TRANS

SPD							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
SPD7	**SPD6**	**SPD5**	**SPD4**	**SPD3**	**SPD2**	**SPD1**	**SPD0**
Bit 7							*Bit 0*

DIR							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
DIR7	**DIR6**	**DIR5**	**DIR4**	**DIR3**	**DIR2**	**DIR1**	**DIR0**
Bit 7							*Bit 0*

AUX							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
BRK3	**BRK2**	**BRK1**	**BRK0**	**KICK3**	**KICK2**	**KICK1**	**KICK0**
Bit 7							*Bit 0*

WHIM							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
WHIM7	**WHIM6**	**WHIM5**	**WHIM4**	**WHIM3**	**WHIM2**	**WHIM1**	**WHIM0**
Bit 7							*Bit 0*

TRANS							
R/W – 0	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*	*R/W – 0*
TR7	**TR6**	**TR5**	**TR4**	**TR3**	**TR2**	**TR1**	**TR0**
Bit 7							*Bit 0*

The CTRL message will be sent from a COACH to a paired PLAYER. This message will be used to send movement commands to the PLAYER, and will also function as a heartbeat to monitor the communications link from COACH to PLAYER.

SPD<0:7> is the commanded speed of the PLAYER. The speed is encoded as a signed char ranging from -128 to 127.

SPD = 127: Full speed forward
SPD = 0: Stop
SPD = -128: Full speed reverse

DIR<0:7> is the commanded rotation rate of the PLAYER. The direction is encoded as a signed char ranging from -128 to 127.

DIR = 127: Maximum left turn.
DIR = 0: Straight
DIR = -128: Maximum right turn.

AUX<4:7> is the commanded braking force. This value is encoded as an unsigned value ranging from 0-15. COACHes implementing digital braking should send all ones (0xF) for BRK. PLAYERs implementing digital braking should interpret nonzero values as braking, and zero values as no braking.

BRK = 0: No braking
BRK = 15: Maximum braking

AUX<0:3> is the commanded kicking force. This value is encoded as an unsigned char ranging from 0-15. COACHes implementing digital kicking should send all ones (0xF) for KICK. PLAYERs implementing digital kicking should interpret nonzero values as kicking, and zero values as not kicking.

KICK = 0: No kicking
KICK = 15: Maximum kicking

WHIM<0:7> is the commanded whimsy. COACHes should set this to zero unless they are paired to their own PLAYER. Teams are allowed complete freedom in the design of the whimsy byte, with the exception that a value of WHIM = 0x00 should prompt default behavior.

TR<0:7> is the commanded translation rate of the PLAYER. The translation is encoded as a signed char ranging from -128 to 127.

DIR = 127: Maximum left translation.
DIR = 0: No translation
DIR = -128: Maximum right translation.