F1 2018 UDP Specification

By Hoo, August 23, 2018 in Technical Assistance

1 2 3 4 5 6 NEXT » Page 1 of 12

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted August 23, 2018

OVERVIEW

The F1 series of games support the outputting of key game data via a UDP data stream. This data can be interpreted by external apps or connected peripherals for a range of different uses, including providing additional telemetry information, customised HUD displays, motion platform hardware support or providing force feedback data for custom steering wheels. The following information is a summary of the data that is outputted so that developers of supporting hardware or software are able to configure these to work with the F1 game correctly. If the information you require is not contained here, or if you have any issues with the UDP data itself, then please let us know and a member of the dev team will respond to your query as soon as possible.

Hoo

Codemasters Staff

Codemasters

163
1,199 posts

Posted August 23, 2018

PACKET TYPES

The main change for 2018 is the introduction of multiple packet types: each packet can now carry different types of data rather than having one packet which contains everything. A header has been added to each packet as well so that versioning can be tracked and it will be easier for applications to check they are interpreting the incoming data in the correct way.

Each packet has the following header:

```
struct PacketHeader
{
    uint16    m_packetFormat;         // 2018
    uint8     m_packetVersion;        // Version of this pack
    uint8     m_packetId;             // Identifier for the p
    uint64    m_sessionUID;           // Unique identifier fo
    float     m_sessionTime;          // Session timestamp
    uint      m_frameIdentifier;      // Identifier for the f
    uint8     m_playerCarIndex;       // Index of player's ca
};
```

Packet Name	ID	Description	Frequency	Size
Motion	0	Contains motion data for all cars	Menu setting	1341 bytes
Session	1	General data about the session	2 per second	147 bytes
Lap Data	2	Lap time info for all cars in the session	Menu setting	841 bytes
Event	3	Session start or session end	On event	25 bytes
Participants	4	List of participants in the session	Every 5 seconds	1082 bytes
Car Setups	5	Car setup info for cars in the race	2 per second	841 bytes
Car Telemetry	6	Telemetry data for all cars	Menu setting	1085 bytes
Car Status	7	General car status info for all cars	2 per second	1061 bytes

Hoo

Codemasters Staff

Codemasters

Posted August 23, 2018

MOTION PACKET

The motion packet gives physics data for all the cars being driven. There is additional data for the car being driven with the goal of being able to drive a motion platform setup.

N.B. For the normalised vectors below, to convert to float values divide by 32767.0f. 16-bit signed values are used to pack the data

163
1,199 posts

and on the assumption that direction values are always between -1.0f and 1.0f.

Frequency: Rate as specified in menus

Size: 1341 bytes

```
struct CarMotionData
{
    float         m_worldPositionX;           // World space
    float         m_worldPositionY;           // World space
    float         m_worldPositionZ;           // World space
    float         m_worldVelocityX;           // Velocity in
    float         m_worldVelocityY;           // Velocity in
    float         m_worldVelocityZ;           // Velocity in
    int16         m_worldForwardDirX;         // World space
    int16         m_worldForwardDirY;         // World space
    int16         m_worldForwardDirZ;         // World space
    int16         m_worldRightDirX;           // World space
    int16         m_worldRightDirY;           // World space
    int16         m_worldRightDirZ;           // World space
    float         m_gForceLateral;            // Lateral G-Fo
    float         m_gForceLongitudinal;       // Longitudinal
    float         m_gForceVertical;           // Vertical G-F
    float         m_yaw;                      // Yaw angle ir
    float         m_pitch;                    // Pitch angle
    float         m_roll;                     // Roll angle i
};



struct PacketMotionData
{
    PacketHeader    m_header;               // Header

    CarMotionData   m_carMotionData[20];    // Data for all c

    // Extra player car ONLY data
    float         m_suspensionPosition[4];       // Note: All
    float         m_suspensionVelocity[4];       // RL, RR, F
    float         m_suspensionAcceleration[4];   // RL, RR, F
    float         m_wheelSpeed[4];               // Speed of
    float         m_wheelSlip[4];                // Slip rati
    float         m_localVelocityX;              // Velocity
    float         m_localVelocityY;              // Velocity
    float         m_localVelocityZ;              // Velocity
    float         m_angularVelocityX;            // Angular v
    float         m_angularVelocityY;            // Angular v
    float         m_angularVelocityZ;            // Angular v
    float         m_angularAccelerationX;        // Angular v
    float         m_angularAccelerationY;        // Angular v
    float         m_angularAccelerationZ;        // Angular v
    float         m_frontWheelsAngle;            // Current f
};
```

SESSION PACKET

The session packet includes details about the current session in progress.

Frequency: 2 per second

Size: 147 bytes

```
struct MarshalZone
{
    float  m_zoneStart;   // Fraction (0..1) of way through t
    int8   m_zoneFlag;    // -1 = invalid/unknown, 0 = none,
};


struct PacketSessionData
{
    PacketHeader    m_header;                   // Header

    uint8           m_weather;                  // Weather -
                                                // 3 = light
    int8            m_trackTemperature;         // Track temp
    int8            m_airTemperature;           // Air temp.
    uint8           m_totalLaps;                // Total numb
    uint16          m_trackLength;              // Track leng
    uint8           m_sessionType;              // 0 = unknow
                                                // 5 = Q1, 6
                                                // 10 = R, 11
    int8            m_trackId;                  // -1 for unk
    uint8           m_era;                      // Era, 0 = m
    uint16          m_sessionTimeLeft;          // Time left
    uint16          m_sessionDuration;          // Session du
    uint8           m_pitSpeedLimit;            // Pit speed
    uint8           m_gamePaused;               // Whether th
    uint8           m_isSpectating;             // Whether th
    uint8           m_spectatorCarIndex;        // Index of t
    uint8           m_sliProNativeSupport;      // SLI Pro su
    uint8           m_numMarshalZones;          // Number of
    MarshalZone     m_marshalZones[21];         // List of ma
    uint8           m_safetyCarStatus;          // 0 = no saf
```

```
                                                   // 2 = virtua
    uint8          m_networkGame;                 // 0 = offline
};
```

LAP DATA PACKET

The lap data packet gives details of all the cars in the session.

Frequency: Rate as specified in menus

Size: 841 bytes

```
struct LapData
{
    float       m_lastLapTime;           // Last lap time in
    float       m_currentLapTime;        // Current time arou
    float       m_bestLapTime;           // Best lap time of
    float       m_sector1Time;           // Sector 1 time in
    float       m_sector2Time;           // Sector 2 time in
    float       m_lapDistance;           // Distance vehicle
                                         // be negative if li
    float       m_totalDistance;         // Total distance tr
                                         // be negative if li
    float       m_safetyCarDelta;        // Delta in seconds
    uint8       m_carPosition;           // Car race position
    uint8       m_currentLapNum;         // Current lap numbe
    uint8       m_pitStatus;             // 0 = none, 1 = pit
    uint8       m_sector;                // 0 = sector1, 1 =
    uint8       m_currentLapInvalid;     // Current lap inval
    uint8       m_penalties;             // Accumulated time
    uint8       m_gridPosition;          // Grid position the
    uint8       m_driverStatus;          // Status of driver
                                         // 2 = in lap, 3 = c
    uint8       m_resultStatus;          // Result status - 0
                                         // 3 = finished, 4 =
                                         // 6 = retired
};



struct PacketLapData
{
    PacketHeader    m_header;              // Header

    LapData         m_lapData[20];         // Lap data for al
};
```

EVENT PACKET

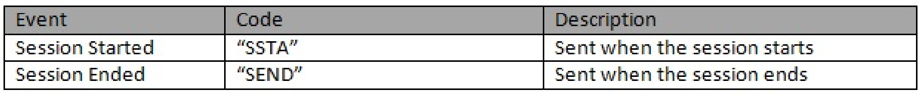
This packet gives details of events that happen during the course of the race.

Frequency: When the event occurs

Size: 25 bytes

```
struct PacketEventData
{
    PacketHeader    m_header;               // Header

    uint8           m_eventStringCode[4];   // Event string c
};
```



Event	Code	Description
Session Started	"SSTA"	Sent when the session starts
Session Ended	"SEND"	Sent when the session ends

PARTICIPANTS PACKET

This is a list of participants in the race. If the vehicle is controlled by AI, then the name will be the driver name. If this is a multiplayer game, the names will be the Steam Id on PC, or the LAN name if appropriate. On Xbox One, the names will always be the driver name, on PS4 the name will be the LAN name if playing a LAN game, otherwise it will be the driver name.

Frequency: Every 5 seconds

Size: 1082 bytes

```
struct ParticipantData
{
    uint8      m_aiControlled;           // Whether the vehic
    uint8      m_driverId;               // Driver id - see a
    uint8      m_teamId;                 // Team id - see app
    uint8      m_raceNumber;             // Race number of th
    uint8      m_nationality;            // Nationality of th
    char       m_name[48];               // Name of participa
                                         // Will be truncatec
};


struct PacketParticipantsData
{
    PacketHeader    m_header;            // Header
    uint8           m_numCars;           // Number of cars ir
    ParticipantData m_participants[20];
};
```

CAR SETUPS PACKET

This packet details the car setups for each vehicle in the session. Note that in multiplayer games, other player cars will appear as blank, you will only be able to see your car setup and AI cars.

Frequency: Every 5 seconds

Size: 841 bytes

```
struct CarSetupData
{
    uint8     m_frontWing;                // Front wing aero
    uint8     m_rearWing;                 // Rear wing aero
    uint8     m_onThrottle;               // Differential adj
    uint8     m_offThrottle;              // Differential adj
    float     m_frontCamber;              // Front camber ang
    float     m_rearCamber;               // Rear camber angl
    float     m_frontToe;                 // Front toe angle
    float     m_rearToe;                  // Rear toe angle (
    uint8     m_frontSuspension;          // Front suspensior
    uint8     m_rearSuspension;           // Rear suspension
    uint8     m_frontAntiRollBar;         // Front anti-roll
    uint8     m_rearAntiRollBar;          // Front anti-roll
    uint8     m_frontSuspensionHeight;    // Front ride heigh
    uint8     m_rearSuspensionHeight;     // Rear ride height
    uint8     m_brakePressure;            // Brake pressure (
    uint8     m_brakeBias;                // Brake bias (perc
    float     m_frontTyrePressure;        // Front tyre press
    float     m_rearTyrePressure;         // Rear tyre pressu
    uint8     m_ballast;                  // Ballast
    float     m_fuelLoad;                 // Fuel load
};


struct PacketCarSetupData
{
    PacketHeader    m_header;            // Header

    CarSetupData    m_carSetups[20];
};
```

CAR TELEMETRY PACKET

This packet details telemetry for all the cars in the race. It details various values that would be recorded on the car such as speed, throttle application, DRS etc.

Frequency: Rate as specified in menus

Size: 1085 bytes

```
struct CarTelemetryData
{
    uint16    m_speed;                      // Speed of car i
    uint8     m_throttle;                   // Amount of thro
    int8      m_steer;                      // Steering (-100
    uint8     m_brake;                      // Amount of brak
    uint8     m_clutch;                     // Amount of clut
    int8      m_gear;                       // Gear selected
    uint16    m_engineRPM;                  // Engine RPM
    uint8     m_drs;                        // 0 = off, 1 = o
    uint8     m_revLightsPercent;           // Rev lights inc
    uint16    m_brakesTemperature[4];       // Brakes tempera
    uint16    m_tyresSurfaceTemperature[4]; // Tyres surface
    uint16    m_tyresInnerTemperature[4];   // Tyres inner te
    uint16    m_engineTemperature;          // Engine tempera
    float     m_tyresPressure[4];           // Tyres pressure
};


struct PacketCarTelemetryData
{
    PacketHeader        m_header;                // Header

    CarTelemetryData    m_carTelemetryData[20];

    uint32              m_buttonStatus;         // Bit flags
                                                // pressed cu
};
```

CAR STATUS PACKET

This packet details car statuses for all the cars in the race. It includes values such as the damage readings on the car.

Frequency: 2 per second

Size: 1061 bytes

```
struct CarStatusData
{
    uint8       m_tractionControl;          // 0 (off) - 2 (h
    uint8       m_antiLockBrakes;           // 0 (off) - 1 (c
    uint8       m_fuelMix;                  // Fuel mix - 0 =
    uint8       m_frontBrakeBias;           // Front brake bi
    uint8       m_pitLimiterStatus;         // Pit limiter st
    float       m_fuelInTank;               // Current fuel m
    float       m_fuelCapacity;             // Fuel capacity
    uint16      m_maxRPM;                   // Cars max RPM,
    uint16      m_idleRPM;                  // Cars idle RPM
    uint8       m_maxGears;                 // Maximum number
    uint8       m_drsAllowed;               // 0 = not allowe
    uint8       m_tyresWear[4];             // Tyre wear perc
    uint8       m_tyreCompound;             // Modern - 0 = h
                                            // 2 = super soft
                                            // 6 = super harc
                                            // Classic - 0-6
    uint8       m_tyresDamage[4];           // Tyre damage (p
    uint8       m_frontLeftWingDamage;      // Front left wir
    uint8       m_frontRightWingDamage;     // Front right wi
    uint8       m_rearWingDamage;           // Rear wing dama
    uint8       m_engineDamage;             // Engine damage
    uint8       m_gearBoxDamage;            // Gear box damag
    uint8       m_exhaustDamage;            // Exhaust damage
    int8        m_vehicleFiaFlags;          // -1 = invalid/u
                                            // 2 = blue, 3 =
    float       m_ersStoreEnergy;           // ERS energy stc
    uint8       m_ersDeployMode;            // ERS deployment
                                            // 3 = high, 4 =
    float       m_ersHarvestedThisLapMGUK;  // ERS energy har
    float       m_ersHarvestedThisLapMGUH;  // ERS energy har
    float       m_ersDeployedThisLap;       // ERS energy dep
};


struct PacketCarStatusData
{
    PacketHeader       m_header;            // Header

    CarStatusData      m_carStatusData[20];
};
```

Hoo

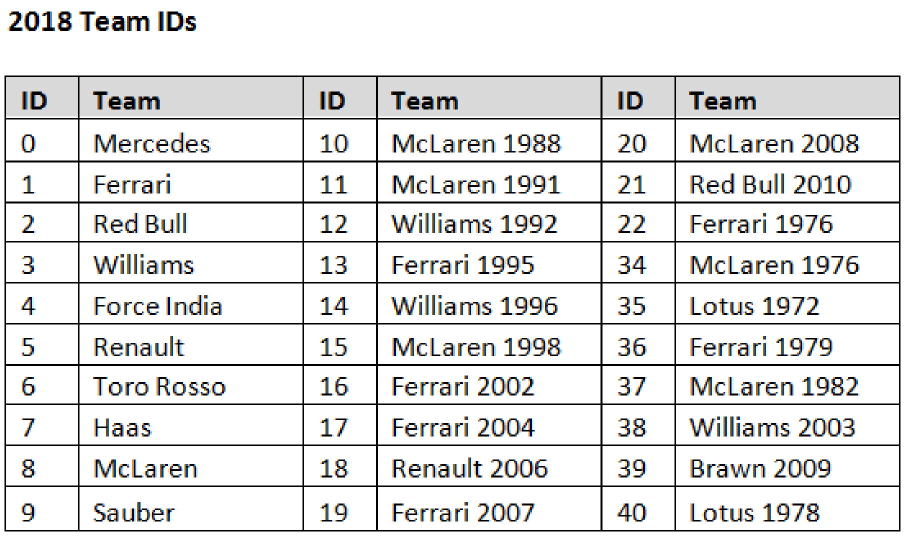
Posted August 23, 2018

Codemasters Staff

Codemasters

163
1,199 posts

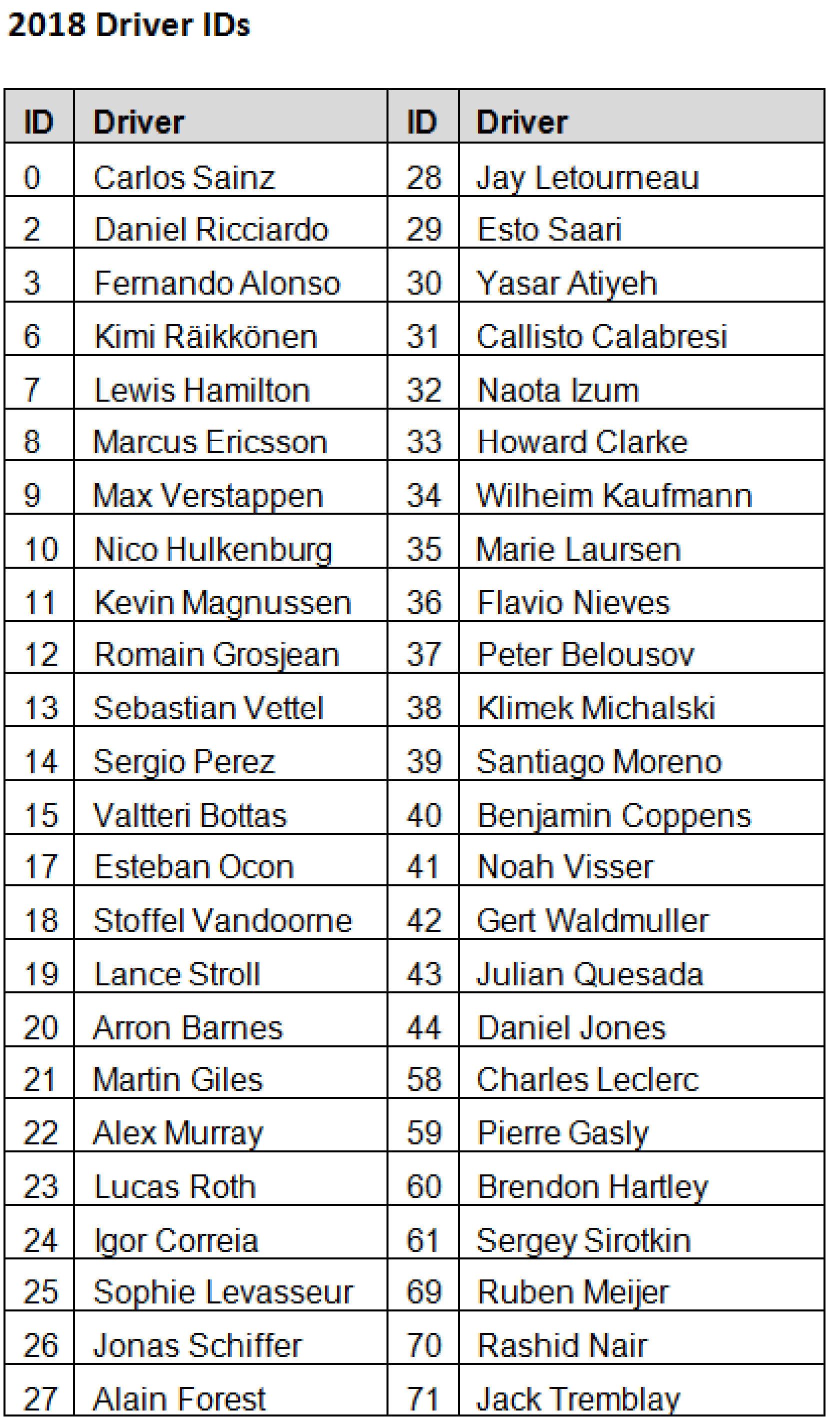
Appendices for the various IDs used in the UDP output:



2018 Team IDs

ID	Team	ID	Team	ID	Team
0	Mercedes	10	McLaren 1988	20	McLaren 2008
1	Ferrari	11	McLaren 1991	21	Red Bull 2010
2	Red Bull	12	Williams 1992	22	Ferrari 1976
3	Williams	13	Ferrari 1995	34	McLaren 1976
4	Force India	14	Williams 1996	35	Lotus 1972
5	Renault	15	McLaren 1998	36	Ferrari 1979
6	Toro Rosso	16	Ferrari 2002	37	McLaren 1982
7	Haas	17	Ferrari 2004	38	Williams 2003
8	McLaren	18	Renault 2006	39	Brawn 2009
9	Sauber	19	Ferrari 2007	40	Lotus 1978

(Hoo: IDs have been updated on 10th Sept 2018 as several of them were missing)



2018 Driver IDs

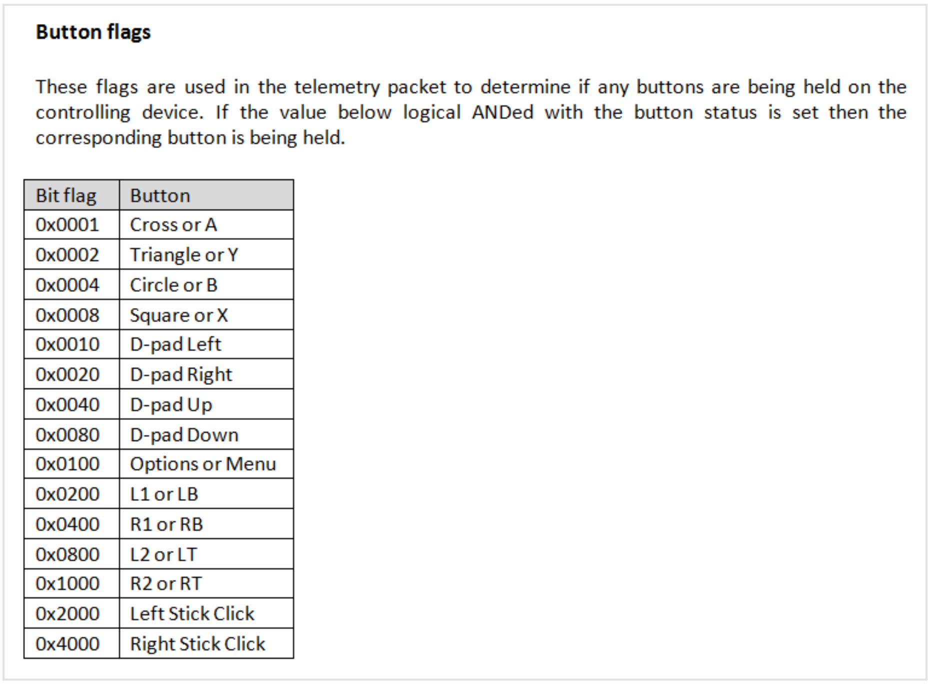
ID	Driver	ID	Driver
0	Carlos Sainz	28	Jay Letourneau
2	Daniel Ricciardo	29	Esto Saari
3	Fernando Alonso	30	Yasar Atiyeh
6	Kimi Räikkönen	31	Callisto Calabresi
7	Lewis Hamilton	32	Naota Izum
8	Marcus Ericsson	33	Howard Clarke
9	Max Verstappen	34	Wilheim Kaufmann
10	Nico Hulkenburg	35	Marie Laursen
11	Kevin Magnussen	36	Flavio Nieves
12	Romain Grosjean	37	Peter Belousov
13	Sebastian Vettel	38	Klimek Michalski
14	Sergio Perez	39	Santiago Moreno
15	Valtteri Bottas	40	Benjamin Coppens
17	Esteban Ocon	41	Noah Visser
18	Stoffel Vandoorne	42	Gert Waldmuller
19	Lance Stroll	43	Julian Quesada
20	Arron Barnes	44	Daniel Jones
21	Martin Giles	58	Charles Leclerc
22	Alex Murray	59	Pierre Gasly
23	Lucas Roth	60	Brendon Hartley
24	Igor Correia	61	Sergey Sirotkin
25	Sophie Levasseur	69	Ruben Meijer
26	Jonas Schiffer	70	Rashid Nair
27	Alain Forest	71	Jack Tremblay

2018 Track IDs

ID	Track	ID	Track
0	Melbourne	13	Suzuka
1	Paul Ricard	14	Abu Dhabi
2	Shanghai	15	Texas
3	Sakhir (Bahrain)	16	Brazil
4	Catalunya	17	Austria
5	Monaco	18	Sochi
6	Montreal	19	Mexico
7	Silverstone	20	Baku (Azerbaijan)
8	Hockenheim	21	Sakhir Short
9	Hungaroring	22	Silverstone Short
10	Spa	23	Texas Short
11	Monza	24	Suzuka Short
12	Singapore		

Nationality IDs

ID	Nationality	ID	Nationality	ID	Nationality	ID	Nationality
1	American	26	Estonian	51	Maltese	76	South Korean
2	Argentinean	27	Finnish	52	Mexican	77	South African
3	Australian	28	French	53	Monegasque	78	Spanish
4	Austrian	29	German	54	New Zealander	79	Swedish
5	Azerbaijani	30	Ghanaian	55	Nicaraguan	80	Swiss
6	Bahraini	31	Greek	56	North Korean	81	Taiwanese
7	Belgian	32	Guatemalan	57	Northern Irish	82	Thai
8	Bolivian	33	Honduran	58	Norwegian	83	Turkish
9	Brazilian	34	Hong Konger	59	Omani	84	Uruguayan
10	British	35	Hungarian	60	Pakistani	85	Ukrainian
11	Bulgarian	36	Icelander	61	Panamanian	86	Venezuelan
12	Cameroonian	37	Indian	62	Paraguayan	87	Welsh
13	Canadian	38	Indonesian	63	Peruvian		
14	Chilean	39	Irish	64	Polish		
15	Chinese	40	Israeli	65	Portuguese		
16	Colombian	41	Italian	66	Qatari		
17	Costa Rican	42	Jamaican	67	Romanian		
18	Croatian	43	Japanese	68	Russian		
19	Cypriot	44	Jordanian	69	Salvadoran		
20	Czech	45	Kuwaiti	70	Saudi		
21	Danish	46	Latvian	71	Scottish		
22	Dutch	47	Lebanese	72	Serbian		
23	Ecuadorian	48	Lithuanian	73	Singaporean		
24	English	49	Luxembourger	74	Slovakian		
25	Emirian	50	Malaysian	75	Slovenian		



Button flags

These flags are used in the telemetry packet to determine if any buttons are being held on the controlling device. If the value below logical ANDed with the button status is set then the corresponding button is being held.

Bit flag	Button
0x0001	Cross or A
0x0002	Triangle or Y
0x0004	Circle or B
0x0008	Square or X
0x0010	D-pad Left
0x0020	D-pad Right
0x0040	D-pad Up
0x0080	D-pad Down
0x0100	Options or Menu
0x0200	L1 or LB
0x0400	R1 or RB
0x0800	L2 or LT
0x1000	R2 or RT
0x2000	Left Stick Click
0x4000	Right Stick Click

Hoo

Codemasters Staff

Codemasters

163

1,199 posts

Posted August 23, 2018

FAQS

How do I enable the UDP Telemetry Output?

In F1 2018, UDP telemetry output is controlled via the menus. To enable this, enter the options menu from the main menu (triangle / Y), then enter the settings menu - the UDP option will be at the bottom of the list. From there you will be able to enable / disable the UDP output, configure the IP address and port for the receiving application, toggle broadcast mode and set the send rate. Broadcast mode transmits the data across the network subnet to allow multiple devices on the same subnet to be able to receive this information. When using broadcast mode it is not necessary to set a target IP address, just a target port for applications to listen on.

Can I configure the UDP output using an XML File?

PC users can edit the game's configuration XML file to configure UDP output. The file is located here (after an initial boot of the game):

...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml

You should see the tag:

```
<motion>

  ...

  <udp enabled="false" broadcast="false" ip="127.0.0.1" port="20777" sendRate="20" format="2018" />

  ...

</motion>
```

Here you can set the values manually. Note that any changes made within the game when it is running will overwrite any changes made manually.

What is the order of the wheel arrays?

All wheel arrays are in the following order:

0 – Rear Left (RL)

1 – Rear Right (RR)

2 – Front Left (FL)

3 – Front Right (FR)

Do the vehicle indices change?

During a session, each car is assigned a vehicle index. This will not change throughout the session and all the arrays that are sent use this vehicle index to dereference the correct piece of data.

What encoding format is used?

All values are encoded using Little Endian format.

Is the data packed?

Yes, all data is packed.

Will my F1 2017 app still work with F1 2018?

F1 2018 uses a new format for the UDP data. However, the F1 2017 implementation is still supported by the game and is referred to as the “legacy” format. This should allow most apps implemented using the previous data format to work with little or no change from the developer. To use the old format, please enter the UDP options menu and set “UDP Format” to “legacy”. Specifications for the legacy format can be seen here: http://forums.codemasters.com/discussion/53139/f1-2017-d-box-and-udp-output-specification/p1.

How do I enable D-BOX output?

D-BOX output is currently supported on the PC platform. In F1 2018, the D-BOX activation can be controlled via the menus. Navigate to Game Options->Settings->UDP Telemetry Settings->D-BOX to activate this on your system.

Advanced PC Users: It is possible to control D-BOX by editing the games’ configuration XML file. The file is located here (after an initial boot of the game):

...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml

You should see the tag:

```
<motion>

  <dbox enabled="false" />

  ...

</motion>
```

Set the “enabled” value to “true” to allow the game to output to your D-BOX motion platform. Note that any changes made within the game when it is running will overwrite any changes made manually.

How can I disable in-game support for LED device?

The F1 game has native support for some of the basic features supported by some external LED devices, such as the *Leo Bodnar SLI Pro* and the *Fanatec* steering wheels. To avoid conflicts between Codemasters’ implementation and any third-party device managers on the PC platform it may be necessary to disable the native support. This is done using the following led_display flags in the hardware_settings_config.xml. The file is located here (after an initial boot of the game):

...\Documents\My Games\<game_folder>\hardwaresettings\hardware_settings_config.xml

The flags to enabled/disable LED output are:

```
<led_display fanatecNativeSupport="true" sliProNativeSupport="true" />
```

The sliProNativeSupport flag controls the output to SLI Pro devices. The fanatecNativeSupport flag controls the output to Fanatec (and some related) steering wheel LEDs. Set the values for any of these to “false” to disable them and avoid conflicts with your own device manager.

Please note there is an additional flag to manually control the LED brightness on the SLI Pro:

```
<led_display sliProForceBrightness="127" />
```

This option (using value in the range 0-255) will be ignored when setting the sliProNativeSupport flag to “false”.

Also note it is now possible to edit these values on the fly via the Game Options->Settings->UDP Telemetry Settings menu.

Alex35zombi

A

Members
● 0
19 posts

Posted August 23, 2018

I have one question is the data streamed on replay mode and spectator?

gaetanomatonti

G

Posted August 23, 2018

Wow that's a big load of data being transmitted, good job!

Members
● 0
11 posts

Hoo

Codemasters Staff

163
1,199 posts

Posted August 24, 2018

> Alex35zombi said:
>
> I have one question is the data streamed on replay mode and spectator?

Yes. :)

Let us know if encounter any problems with this.

carlucio24

Members
● 0
12 posts

Posted August 24, 2018

Hi, great job with this new telemetry protocol.
What are MarshalZones ?

trenamax

Members
● 0
22 posts

Posted August 24, 2018

Hi Hoo -
I wish I'd spotted this issue during the Beta so apologies about that. I play the game with fully functional replica F1 wheels (see here: https://www.youtube.com/watch?v=hrq2cmSdp04). I have an LED which illuminated when DRS is allowed, and an additional one which illuminates when DRS is active.

I've noticed whilst playing this evening there seems to be a delay in the DRS Legal LED illuminating when entering a DRS zone, and now I realise it's because m_drsAllowed is in the Car Status packet (only sent twice a second), whereas m_drs is in the Car Telemetery Data packet (sent at game setting which is 60hz in my case). My DRS LED works instantly I should add. Both worked perfectly in F1 2015-2017.

Is there any chance we can move m_drsAllowed into the Car

Telemetery Data packet so it's sent at a descent rate please? It seems like the two should be sent in the same packet.

Many thanks, and loving the full version of the game :)

Mike

LonelyRacer

Members
8
60 posts

Posted August 26, 2018

> Hoo said:
>
> CAR TELEMETRY PACKET
>
> This packet details telemetry for all the cars in the race. It details various values that would be recorded on the car such as speed, throttle application, DRS etc.
>
> Frequency: Rate as specified in menus
>
> Size: 1085 bytes

```
struct CarTelemetryData
{
    uint16    m_speed;                      // Speed of c
    uint8     m_throttle;                   // Amount of
    int8      m_steer;                      // Steering (
    uint8     m_brake;                      // Amount of
    uint8     m_clutch;                     // Amount of
    int8      m_gear;                       // Gear selec
    uint16    m_engineRPM;                  // Engine RPM
    uint8     m_drs;                        // 0 = off, 1
    uint8     m_revLightsPercent;           // Rev lights
    uint16    m_brakesTemperature[4];       // Brakes tem
    uint16    m_tyresSurfaceTemperature[4]; // Tyres surf
    uint16    m_tyresInnerTemperature[4];   // Tyres inne
    uint16    m_engineTemperature;          // Engine tem
    float     m_tyresPressure[4];           // Tyres pres
};
```

Thank you for the data.

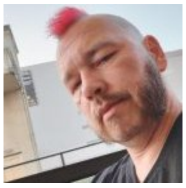
Been converting my Telemetry Tool to accept the new data. I haven't looked into the data too deeply yet, but will do that during the coming days.

Classic drivers
One note. When you do the race in Classic era, there are drivers (e.g. m_driverId 41), which are not listed in the table above. Any change to send the updated list here?

Speed etc in Floats instead of Ints?
Would it be possible to provide the m_speed, m_throttle, m_steer and m_brake as floats instead of ints? It would only add few bytes to the CarTelemetryData, but for people doing the telemetry for Wheel users, the more granular data would be super helpful.

Thanks.

EnsiFerrum



Members
84
730 posts

Posted August 27, 2018

Found a bug:
ParticipantData.m_teamId returns 35, no matter in which historic car I am.
Tested in Time-Trial mode.
This makes any per car setting useless.

Edit: It occurs only in Time Trial mode. After a quick look in my notes I can say we had this bug already in F1-2017. Never got fixed. Hope it will be adressed in the next patch, please.

cjorgens79

C

Members
3
168 posts

Posted August 28, 2018

@Hoo - what is the difference between m_tyresWear and m_tyresDamage in the Car Status packet? From what i can tell they always seem to have the exact same value. I thought that perhaps m_tyresDamage was related to the suspension or rim damage, however it just seems to be the same as the tyre wear percentage.

cjorgens79

Posted August 28, 2018

C

Members
3
168 posts

There appears to be some issues with the timing data for Time Trial, it seems to contain data for a number of additional non-existant players. The m_numCars is 1, however there are multiple entries in the lap data record that appear to be active. They show as lap 1, position 1 and for all intensive purposes appear to be an active player (albeit with overlapping race positions). They even have valid x/y/z world co-ordinates.

At the moment I have had to work around it by doing a check for m_numCars = 1, in which case i know there is only the "player", so i can use the m_playerCarIndex to make sure i grab the right entry, as they all overlap each other in terms of their result (race position).

ReddishTheGreat

R

Members
0
9 posts

Posted August 28, 2018

> Is there any chance we can move m_drsAllowed into the Car Telemetery Data packet so it's sent at a descent rate please? It seems like the two should be sent in the same packet.

A possible alternative would be to change the m_drs field in the Car Telemetry Packet to be a bitfield:

0 == no DRS allowed;
2 == DRS allowed but inactive;
3 == DRS allowed and active.

(We should never see 1, normally -- well perhaps if there is a 'DRS stuck open' fault mode).

ReddishTheGreat

R

Members
0
9 posts

Posted August 28, 2018

Bug report (F1 2018 TELEMETRY):

At the beginning of a new session, some UDP packets are transmitted with the sessionUID field set to the previous session's UID.

In the first session after starting the game, a few packets with sessionUID equal to zero are transmitted.

Seems that the sessionUID field is set after enabling UDP telemetry, where it should be set before. So this may well be easy to fix.

EDIT: bug reported as top-level post, see:

http://forums.codemasters.com/discussion/138130/bug-f1-2018-pc-v1-0-4-udp-telemetry-bad-session-uid-in-first-few-packets-of-a-session

ReddishTheGreat

Members
● 0
9 posts

Posted August 28, 2018

Suggested improvement to telemetry:

It would be very useful to get a telemetry value for terrain type (perhaps one per wheel, perhaps one for the car center). It could be as simple as an enumeration: 0==ASPHALT, 1==CURB, 2==GRASS, 3==GRAVEL.

willgarling

Members
● 0
6 posts

Posted August 28, 2018

Can anyone tell me which platforms would be compatible with this game?

HassanKLD

Members
● 0
2 posts

Posted August 30, 2018

Hello!

Firstly thanks for posting this @Hoo. I'm currently creating something for myself, but found the forum a little tedious to keep coming back to for the info, so I created a quick reference doc for myself. Others might find it useful. Link here. It's not fully complete I still need to port of the full appendices. @Hoo If you or anyone at Codemasters would like to take ownership of this I'm more than happy to. It's just a github repo, and the docs are written in Markdown and auto generated.

Ok so on the actual UDP spec side, I picked up a few things and just wanted to clarify/point few things out.

1. ~~the buffer size for Motion packet is 1341 Bytes, after accounting for everything in the structs I get a total of 1221 Bytes. Is this correct? If so whats in the remaining 120 Bytes? anything useful or can it be simply ignored.~~
2. The m_eventStringCode in the PacketEventData Struct are mentioned to be UInt8[4], but in the table below it shows code as 'SSTA' and 'SEND'. is the UInt8 an error as they should be Chars
3. I agree with @trenamax to move the m_drsAllowed to the carTelemetryPacket
4. I would really like to have the track limits in the data stream, maybe m_trackLimitLeft and m_tackLimitRight maybe on the MotionData Packet. *Edit: thinking more about this, I think it would be useful to get the world x,y, z of the left and right limit*

thanks! Hass

trenamax

T

Members
● 0
22 posts

Posted August 30, 2018

Hi @HassanKLD -

I wrote an app during the beta, and can confirm that the motion data packet does indeed total to 1341 bytes all used. So you must have missed something somewhere.

You might find this mapping spreadsheet I put together useful:

https://btcloud.bt.com/web/app/share/invite/cCKudhtU11

Pleased you agree about DRS Legal moving to the car telemetry packet, its really noticeable for me whilst driving.

HassanKLD

H

Posted August 30, 2018

Hi @trenamax

Thanks for that spreadsheet, that helped. You are correct, was missing all the m_worldRgihtDirection entries in my app.

Members
● 0
2 posts

Alex35zombi

Members
● 0
19 posts

Posted August 30, 2018

I'm using C# and my UDPClient is only able to access to the bytes from the PacketCarTelemetryData, any way I could make my client to receive the bytes from all structures?

```
UdpClient client = new UdpClient(20777);
IPEndPoint ep = new IPEndPoint(IPAddress.Any, 0);


void UpdateData()
{
    while (true)
    {
        byte[] arr = client.Receive(ref ep);
        Console.WriteLine(arr.Length.ToString()); // Output:
    }
}
```

LetMeThrashU

Members
⊕ 38
353 posts

Posted August 30, 2018

What is the total value that the ERS battery can store and harvest? Looking at the rules, the MGU-K can recover unto 2MJ/lap and the MGU-H is unlimited. What values do the steering wheels in your game use for max harvest and total energy?

Drospy

Members

Posted August 30, 2018

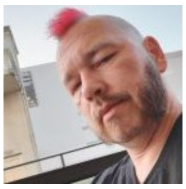
Hi all,

please Codemaster change the frequency for "Car Status Packet", 2 second is too high.

5
34 posts

m_drsAllowed, m_vehicleFiaFlags, m_pitLimiterStatus and m_fuelMix are very important.

EnsiFerrum



Members
84
730 posts

Posted August 30, 2018

> LetMeThrashU said:
>
> What is the total value that the ERS battery can store and harvest? Looking at the rules, the MGU-K can recover unto 2MJ/lap and the MGU-H is unlimited. What values do the steering wheels in your game use for max harvest and total energy?

Battery is 4MJ
Harvest is 2MJ

1 2 3 4 5 6 NEXT » Page 1 of 12

GO TO TOPIC LISTING
Technical Assistance

Home > F1 Games > F1® 2019 Game Forum > Technical Assistance > All Activity

CONTACT US