



# FSAcars

## ACARS and PIREP System

### SDK Document

1	FSAcars SDK .....	3
1.1	Company Options.....	3
1.2	FSAcars VA ini .....	8
1.3	FSAcars pirep sintaxe (by Paulo Correia) .....	9
1.4	FSAcars Position syntax (by Paulo Correia).....	10
1.5	FSAcars VA automated Flight Plan (by Miguel Costa) .....	11
1.6	FSAcars Acars messages.....	12
1.6.1	Downlink (By José Lopes) .....	12
1.6.2	Uplink (by Pedro Sousa).....	12
1.7	Plane Configuration file (José Oliveira).....	13
1.8	FSAcars Plug-ins (José Oliveira) .....	14
2	FSAcars Flight Plan Format (José Oliveira).....	17
3	Fsacars Gauge SDK.....	17
3.1	Gauge characteristics .....	17
3.2	Gauge orientation .....	17
3.3	License .....	19
3.4	Establishing communication between Gauge and main program.....	19
3.5	Exchanging messages between the gauge and the program .....	19
3.5.1	Messages from Gauge to program .....	20
3.5.2	Messages from Program to Gauge .....	20
3.6	Program examples .....	21

# 1 FSAcars SDK

This document specifies all the information that is needed to operate FSacars from a VA point of view. These implementations aren't specified by me but from other authors that see the need to implement the functionalities. However in the project spirit the syntax is available to everyone that would like to implement the functionality. The acars messages are available to everyone thru the SATA website. The PIREP PHP skeleton is available thru the FSacars project.

## 1.1 Company Options

The following dialog will appear to help you creating a new company. If you want to change an already made ini file add to the ini: Change=yes under [fsacars] section. Next time you launch FSacars and choose the company the following menu will appear.

Company Options	
Company ICAO	
Company IATA	
Number used in Callsign	Flight
Company Complete name	
Units	Pounds
Use cargo plane	<input type="checkbox"/>
Remarks	
Local Time	<input type="checkbox"/>
Alternate Acars Site	

Number used in Callsign

Close

- **Company ICAO:** Company ICAO code. If the space is left blank it's assumed private flight.
- **Company IATA:** Company IATA code. Is it is left blank the company IATA is not used in the flight number pre fill.
- **Number Used in Callsign:** Help to establish a standard callsign in the VA. The callsign always starts with Company ICAO followed by a number. Possible values for

the number are:

- **Blank:** No prefill is provided for the callsign
  - **Flight:** The Flight Number is used in Callsign;
  - **Pilot:** The pilot number is used in Callsign;
- **Company full name:** Company full name that appears in the choose company dialog;
- **Company Site:** URL for the company site;
- **Units:** Self Explained;
- **Use Cargo Plane:** Use cargo instead of PAX in the Flight Data Dialog;
- **Remarks.** This will allow automated online flight remarks creation. This field is composed by tags separated by space that will be automatically filled in the flight plan. The allowed tags are:
  - **RMK** - Free text. Like RMK/Charts on Board
  - **OPR**- Will be composed with Company full name;
  - **REG** - Airplane registration
  - **SEL** – Airplane Selcal
- **Local Time:** The log hours is referenced to local time and not UTC.;
- **Alternate AcarsSite:** VA own FSAcars site URL (Syntax available as separate document).
- **Plane Status Site.** VA Plane position URL (Syntax available as separate document).
- **Get FP from VA Site.** VA automated FP filling URL (Syntax available as separate document).
- **Logged Events:** Events logged during Flight;

Logged event's	
Flaps	<input type="checkbox"/>
Touchdown	<input type="checkbox"/>
T/O LD Position	<input type="checkbox"/>
TOC TOD	<input type="checkbox"/>
Com1 Freq	<input type="checkbox"/>
Gear	<input type="checkbox"/>
Flight Length	<input type="checkbox"/>
Vr/V2	<input type="checkbox"/>
Flight Position	<input type="checkbox"/>

- **Flaps:** Log Flaps position and Indicated speed for each position. Position in 1...9 formats.
- **Touchdown:** Log Touchdown analysis, speed and rate of descend.
- **T/O LD Position:** Log Take off and Landing geographical position;
- **TOC TOD:** Logs Top of Climb and Top of Descend; This will include Fuel weight.
- **Com1 Freq.:** Log Com1 frequency
- **Gear:** Log Gear up and down time and indicated speed.
- **Flight Length:** Log flight length
- **Vr/V2:** Rotate speed and V2 speed;
- **Flight Position:** Airplane position during flight
- **T/O N1:** Take off N1 for both motors.
- **Duration:** Flight duration T/O Land and Block to Block
- **Fuel:** Spent fuel and Fuel on board;
- **Weight:** Airplane weight Block and T/O;
- **Metars:** Departing and arrival metars;
- **Dist. To Land:** Distance between TOD and Touchdown;
- **Realism:** Options to improve flight realism

Realism	
No Slew and time accel	<input type="checkbox"/>
No pause	<input type="checkbox"/>
Force Crash Detect	<input type="checkbox"/>
PIC in Cockpit	<input type="checkbox"/>
Minimum Reset Time	0
Maximum reset Time	0
Warning sound	
+ Send PIREP	
+ Send log options	

- **No Slew and time accel:** FSacars will disable slew and x2-x32 time compression.
- **No pause:** FSacars will disable pause.
- **Force crash detect:** FSacars will force crash detect. However crash with other planes isn't forced.
- **PIC in cockpit:** From time to time FSacars will ask for confirmation that pilot isn't away
- **Send PIREP:** Options for sending automated PIREP

Alternate Acars Site	
Plane status Site	
Get FP from VA Site	
+ Logged event's	
+ Realism	
- Send PIREP	
PHP/ASP script	
E-Mail	
Web Page	
+ Send log options	

- **PHP/ASP script:** URL for automated PIREP sending (Syntax available as separate document).
- **E-Mail.** E-Mail Address used to send log. The log is an attached file to an E-Mail and the file name is the pilot number. See the bellow text for the options.
- **Web Page.** The VA web page used to fill the flight results. FSacars will open this page after the flight.
- **Send Log Options:** Decides what kind of information will be sent after the flight.

Send log options	
Pilot Number	<input checked="" type="checkbox"/>
Password	<input type="checkbox"/>
Date	<input checked="" type="checkbox"/>
Hour	<input checked="" type="checkbox"/>
Callsign	<input checked="" type="checkbox"/>
IATAN	<input type="checkbox"/>
Regist	<input checked="" type="checkbox"/>
Depart	<input checked="" type="checkbox"/>
Arrival	<input checked="" type="checkbox"/>

- **Pilot Number:** VA Pilot Number
- **Password:** A password will be sent with the log
- **DATE:** Flight Date yyyy/mm/dd format
- **Hour:** Flight hour hh:mm
- **Callsign:** Flight Callsign
- **IATAN:** Flight number
- **Regist:** Airplane Registration
- **Depart:** Departure Airport ICAO
- **Arrival:** Arrival Airport ICAO
- **Alternate:** Alternate Airport ICAO
- **Plane Type:** Plane Type ICAO
- **Spent Fuel:** Spent Fuel in the units chosen above
- **Init Fuel:** Initial Fuel in the units chosen above
- **End Fuel:** Final Fuel in the units chosen above
- **Duration:** Flight Duration hh:mm
- **Length:** Flight length in nm
- **T/D:** Plane Touchdown analysis. Vertical speed and indicated speed
- **Complete Log:** Complete flight log in txt format.

- **Version:** FSacars version major+minor

## 1.2 FSacars VA ini

Each VA can have an ini that could change FSacars behavior. The above described options will create the ini file described here. The ini should be named ICAO.ini. For example for SATA International should be RZO.ini. Each VA could also distribute a bmp with the name ICAO.bmp. For example for SATA International should be RZO.bmp. The bmp dimensions should be 72x72 pixels. This bmp will be shown in FSacars during the flight. All the options could be changed via the company wizard except the password field.

[FSacars]		
UnitSystem	Metric or pounds.	IS or GB
PilotNumber	VA Pilot Number	Numeric value
CompanyICAO	VA ICAO value	4 letter ICAO
CompanyName	VA full name	Any string
CompanyIATA	Company IATA code	Any String
AcarsSite	VA ACARS messages site URL	Any string
UseLocal	Use Local time	0 deactivate 1 activate
Remarks	Flight Plan remarks	RMK/string - the string will be included OPR/ - The OPR/Company name will be included REG/ - Aircraft registration will be included
FPSite	URL that indicates the automated Flight Plan site	Any string
StatusSite	URL that indicates the status site	Any string
CompanySite	URL that indicates the VA site	Any string
CallsignUses	What is the company using in callsign	Pilot: Pilot number, Flight: Flight number , Blank: No pre fill
[Events]		
FlapsEvent	Flaps positions.	0 deactivate 1 activate
ToutchDownEvent.	Touch Down analysis.	0 deactivate 1 activate
TOCTODEvent.	Top of climb Top of descend.	0 deactivate 1 activate
ComFreqEvent	Com1 frequency.	0 deactivate 1 activate
GearEvent.	Gear up and down	0 deactivate 1 activate
FlightLengthEvent	Flight Length	0 deactivate 1 activate
VrV2Event.	Vr V2	0 deactivate 1 activate
PIREPEvent.	Using automated Pirep	0 deactivate 1 activate
FlightPosEvent.	Flight position on Take Off and Land	0 deactivate 1 activate
N1Event	Take Off N1	0 deactivate 1 activate
DurationEvent.	Flight duration	0 deactivate 1 activate
FuelEvent.	Spent fuel	0 deactivate 1 activate
WeightEvent.	Plane weight, Zero fuel weight	0 deactivate 1 activate
DistLandEvent.	Distance from Top of climb to land	0 deactivate 1 activate
UseCargo	Cargo instead of pax	0 deactivate 1 activate



MinReset	Interval for PIC in cockpit testing	1-... in minutes
MaxReset	Interval for LOG if the PIC didn't reset the test	
Wave	Path for the sound that alerts PIC for resetting	Default is =.\\alert
[Log]		
Mail	Mail address to send pirep	String
URL	Url to open after flight. For VA with web based filling	String
Log	Url to send pirep	String
passwd	Password to send with log	string
UplinkReset	URL for uplink reset	string
Uplink	URL for uplink	string
[Realism]		
NoSlew	No slew and time acceleration	0 deactivate 1 activate
NoPause	No pause	0 deactivate 1 activate
Crash	crash detect	0 deactivate 1 activate
PIC	PIC in cockpit.	0 deactivate 1 activate
MinReset	Interval for PIC in cockpit testing	1-... in minutes
MaxReset	Interval for LOG if the PIC didn't reset the test	1-... in minutes
Wave	Path for the sound that alerts PIC for resetting	Default is =.\\alert.wav
[SendLog]	Which fields to send in log	
Date	Log date	0 deactivate 1 activate
Hour	Log Hour	0 deactivate 1 activate
Callsign	Pilot Callsign	0 deactivate 1 activate
IATAN	Flight number	0 deactivate 1 activate
Regist	Aircraft registration	0 deactivate 1 activate
Depart	Departure Airport	0 deactivate 1 activate
Arrival	Arrival Airport	0 deactivate 1 activate
PlaneType	Plane Type ICAO	0 deactivate 1 activate
SpentFuel	Spent Fuel (Kg or Lb)	0 deactivate 1 activate
IniFuel	Initial Fuel (Kg or Lb)	0 deactivate 1 activate
EndFuel	Final Fuel (Kg or Lb)	0 deactivate 1 activate
Dur	Flight duration	0 deactivate 1 activate
Len	Flight length	0 deactivate 1 activate
TD	Touchdown analysis (2 lines in log)	0 deactivate 1 activate
TOW	Take off Weight	0 deactivate 1 activate
Log	Complete log	0 deactivate 1 activate
Password	Send password stored at	0 deactivate 1 activate
PilotNumber	Pilot number	0 deactivate 1 activate
ZFW	Airplane zero Fuel Weight	0 deactivate 1 activate
Version	FSacars version String in ascii like3400	0 deactivate 1 activate

### 1.3 FSAcars pirep sintaxe (by Paulo Correia)

FSacars send the pirep data by writing it in a URL with a PHP script. The data is composed by and header + complete log text. If you are writing your private script make sure it can handle null fields since the pilot could not have written them.

In the following table the header is the green text.

Controlling field	URL?	The script URL+ the ? symbol
Password	passwd=xxxx	
PilotNumber	pilot=xxxxxxx	Company icao folowed by pilot number. For example for SATA Internacional and pilot 481 it will be RZO481
Date	&date=yyyy/mm/dd	Flight date
Hour	&time=hh:mm:ss	Flight start time
Callsign	&callsign=xxxxx	Flight callsign
Depart	&origin=xxxx	Depating airport ICAO
Arrival	&dest=xxxx	Destination airport ICAO
Alternate	&alt=xxxx	Alternate airport ICAO
PlaneType	&equipment=xxxx	Airplane ICAO code
SpentFuel	&fuel=xxxx	Spent fuel
Dur	&duration=hh:mm	This is block to block as standard
Len	&distance=xxxx	In NM
Version	&version=xxxx	Crosscheck to make sure pilots are using fsacars current version. Current value is at least 4080 in ASCII.
	&more=0	It will indicate that this is the first chunk of data
	&log=	Complete log text. The ‘\n’ is changed to *

If the URL text is bigger than 1000 bytes then the header data is sent again but with more=1 and the rest of the log text. This process is repeated as long as FSACars as data to send from the same log. The database should answer OK in each send in plain ASCII to inform FSACars that everything is correct. If something goes wrong it should return the error in HTML.

#### 1.4 FSACars Position syntax (by Paulo Correia)

This syntax allows a VA to implement a map/table that shows the aircraft position and state.

If the URL is present in the options it is activated. The data is sent each 2 min.

URL?	The script URL+ the ? symbol
Lat=xx.xxx	Plane Latitude (decimal degrees, - is South)
&long==xx.xxx	Plane Longitude (decimal degrees, - is West)
&GS=xxx	Plane Ground Speed Knots
&Alt=xxx	Plane Altitude in Feet
&IATA=xxx	Flight IATA number
&pnumber=xxx	Pilot number
&depaptICAO=xxx	Departing airport ICAO
&depapt=xxx	Departing airport name
&disdepapt=xxx	Departing airport distance in Nautical Miles
&timedepapt=hh:mm	Time from departing airport
&destaptICAO=xxx	Arrival airport ICAO

&destapt=xxx	Arrival airport name
&disdestapt=xxx	Arrival airport distance in Nautical Miles
&timedestapt=hh:mm	Time to Arrival airport
&Ph=	Flight Phase: 1 – Boarding; 2 – Departing 3 - Cruise; 4 - Arrived

### 1.5 FSAcars VA automated Flight Plan (by Miguel Costa)

This facility allows FSAcars to get all the flight data directly from the VA.

URL?	The script URL+ the ? symbol
pilot=xxxxxxx	Company icao followed by pilot number. For example for SATA International and pilot 481 it will be RZO481

The site must answer with a page in ASCII with the information. One item per line separated by carriage return:

OK (Tested by FSAcars to validate data)
Departing airport ICAO
Arrival airport ICAO
Cruise altitude in feet
Flight Plan
Airplane type
Flight Start time hh:mm in UTC
Flight Start Date yyyy:mm:dd
Flight Number
Aircraft Registration
Callsign
Route
Fuel (in the same coordinates as specified in the ini)
Selcal
PAX
Cargo (in the same coordinates as specified in the ini)

If one of the values is omitted the page should contain a carriage return only in the correspondent line.

## 1.6 FSacars Acars messages

### 1.6.1 Downlink (By José Lopes)

The messages are sent to the URL presented at fsacars.ini AcarsSite the default address is <http://www.satavirtual.org/fsacars/fsacars.asp> .

Packet description

?dtm=	Plus the first message line. Message date
&acftreg=	Plus the second line. Aircraft registration
&mlabel=	Plus the third line. Message type
&fltnum=	Plus the fourth line. Flight number
&mcontent=	Plus the message content. In the messages the \n are replaced by £.

Example: The message for Acars free text message for the aircraft B734 with registration CS-TGZ, SATA International (RZO) and flight number 7859. Should be:

```
http://www.satavirtual.org/fsacars/fsacars.asp?dtm=[18-02-2004
10:21]&acftreg=ACARS mode: C Aircraft reg: CS-TGZ [B734/M RZO]
&mlabel=Message Label: QD Block id: 1 MSg. no:
M22A&fltnum=Flight id: 7859&mcontent=Message content: £this is
a test
```

### 1.6.2 Uplink (by Pedro Sousa)

FSacars calls the UplinkReset URL this URL must return a number followed by \n that represents the last message number in the DB.

Then FSacars calls each 30 s the URL uplink

URL?	The script URL+ the ? symbol
acarsid=xxxxx	Last message number processed by FSacars
&FlightIATA==xx.xxx	Flight number of the plane requesting messages

The site should answer with a page in ASCII one item per line. The answer is the acars messages. Each message is separated by %

If the message is for the plane it must contain the SENDTO keyword + space + aircraft IATA number in the body to identify the message.

The last line must contain #number. Number is going to be used in the next request.

## 1.7 Plane Configuration file (José Oliveira)

FSacars supports a configuration file that customizes the planes. The file name must be ICAO+planes.txt (eg: RZOplanes.txt) or for private flights userplanes.txt. The file format is the following:

```
[Plane.i]
Name=Plane ICAO
Reg=xxx,yyy,zzz
Sel= ttt,vvv,lll
Flaps=a,b,c,d,
```

i – Number beginning at 0 and adding 1 for each plane;

xxx,yyy,zzz – Strings separated by , each one representing a airplane registration for this type;

ttt,vvv,lll – Strings separated by , each one representing a airplane Selcal for this type,

Note: The number of strings must be the same in sel and reg. Leave a space separated by , to make the same.if there aren't any string in sel= and reg= the FSacars will not supply a list;

a,b,c,d,– Strings separated by , each one representing a airplane flap value.

Example for SATA fleet:

```
[Plane.0]
Name=B733/M
Reg=CS-TGP
Sel=KS-GR
Flaps=Up,1°,2°,5°,10°,15°,25°,30°,40°
```

```
[Plane.1]
Name=B734/M
Reg=CS-TGZ
Sel=JK-FQ
Flaps=Up,1°,2°,5°,10°,15°,25°,30°,40°
```

```
[Plane.2]
Name=A322/M
Reg=CS-TKJ
Sel=HP-AF
Flaps=Up,1,1+F,2,3
```

```
[Plane.3]
Name=A333/H
Reg=CS-TGU,CS-TGV,CS-TKI
Sel=JL-DS,CS-JL,MR-KS
Flaps=Up,1,1+F,2,3
```

## 1.8 FSacars Plug-ins (José Oliveira)

Suppose that your VA want something special for it like a proprietary send log or a cargo loader or even a flight analysis package. This kind of additions isn't something that belongs to the FSacars program they belong to a plug-in. A Plug-in is nothing more than a dll that FSacars calls and performs some VA proprietary code. You will gain the entire fsacars package resources making your job a lot easier. The dll code is yours and doesn't belong to FSacars, of course you can make it public to all VA if you want to or make it restrict.

Instead of defining a complete plug-in API we decided to arrange FSacars to call your dll after we both agree what is the function (s) name what kind of var that we should pass and get etc. So if you are interested in developing a plug-in for FSacars please contact me using the E-Mail [jcboliveira@netcabo.pt](mailto:jcboliveira@netcabo.pt)

FSacars available struts

These are the main variables available for plug-in. Of course others can be arranged

```
struct SPrivateOptions {
    CStringA    PilotNumber;
    CStringA    CompanyICAO;
    CStringA    CompanyName;
    CStringA    Wave;
    CStringA    MailLog;
    CStringA    URLLog;
    CStringA    LogLog;
    CStringA    AcarsSite;
    CStringA    Remarks;
    CStringA    Passwd;
    BOOL    Units;
    BOOL    UseCargo;
    BOOL    FlapsEvent;
    BOOL    TouchDownEvent;
    BOOL    TOLDPosEvent;
    BOOL    TOCTODEvent;
    BOOL    ComFreqEvent;
    BOOL    GearEvent;
    BOOL    ArmTouchDownEvent;
    BOOL    FlightLengthEvent;
    BOOL    VrV2Event;
    BOOL    PIREPEvent;
    BOOL    FlightPosEvent;
    BOOL    N1Event;
    BOOL    DurationEvent;
    BOOL    FuelEvent;
    BOOL    WeightEvent;
    BOOL    MetarsEvent;
    BOOL    DistLandEvent;
    WORD    MaxReset;
    WORD    MinReset;
```

```

        BOOL    NoPause;
        BOOL    NoSlew;
        BOOL    Crash;
        BOOL    PIC;
        BOOL    UseLocal;
        CStringA    PlaneList;
        CStringA    RegList;
        CStringA    StatusSite;
        CStringA    FPSite;
    }PrivateOptions;

    struct SFlightPlan {
        CStringA    Route;
        CStringA    Remarks;
        CStringA    DepAirport;
        CStringA    DestAirport;
        CStringA    Date;
        CStringA    Hour;
        CStringA    CallSign;
        CStringA    Regist;
        CStringA    PlaneType;
        CStringA    FL;
        CStringA    FlightIATA;
        CStringA    Pax;
        CStringA    Cargo;
        CStringA    AircraftRegist;
        CStringA    AltApt;
        CStringA    Online;
        SCoordinates DepAptCoor;
        SCoordinates ArrAptCoor;
        CStringA    DepAirportName;
        CStringA    DestAirportName;
        CStringA    SelCal;
    } FlightPlan;

    struct SFlightData {
        BYTE dummy;
    } FlightData;

    struct SFlightResults {
        CStringA Date;
        CStringA Hour;
        CStringA CallSign;
        CStringA FlightIATA;
        CStringA AircraftRegist;
        CStringA DepAirport;
        CStringA DestAirport;
        CStringA PlaneType;
        CStringA UsedFuel;
        CStringA InitialFuel;
        CStringA FuelEnd;
        CStringA FlightDur;
        CStringA FlightLength;
        CStringA RateTD;
        CStringA speedTD;
        CStringA ZFW;
        CStringA Reg;
        CStringA Log;
    } FlightResults;

    struct SSendLog {

```

```

        BOOL Password;
        BOOL PilotNumber;
        BOOL Date;
        BOOL Hour;
        BOOL Callsign;
        BOOL IATAN;
        BOOL Regist;
        BOOL Depart;
        BOOL Arrival;
        BOOL PlaneType;
        BOOL SpentFuel;
        BOOL IniFuel;
        BOOL EndFuel;
        BOOL Dur;
        BOOL Len;
        BOOL TD;
        BOOL TOW;
        BOOL Log;
        BOOL Version;
    } SendLog;

    struct SProgramOptions {
        CStringA    SiteAcars;
        CStringA    ServinfoPath;
        BOOL    UseGauge;
        BOOL    UseDB;
        CStringA    FSFPPath;
        CStringA    LogFile;
        CSMTTPConnection::LoginMethod m_Authenticate;
        CString    m_sPassword;
        CString    m_sUsername;
        CString    m_sAddress;
        CString    m_sHost;
        CString    m_sName;
        CStringA    Machine;
        BOOL    AcarsEvent;
        CStringA    provider;
        BOOL    StartWarning;
        BOOL    FuelTest;
        BOOL    TimeSync;
        int        BeforePause;
        CString    UplinkReset;
        CString    Uplink;
    } ProgramOptions;

    struct SFlightInformation {
        BOOL        LogAltitude;
        BOOL        LogGroundSpeed;
        BOOL        LogIndicatedSpeed;
        BOOL        LogTotalSpeed;
        BOOL        LogWinds;
        BOOL        LogHeading;
        BOOL        LogPlaneWeight;
        BOOL        LogFuelOnBoard;
        BOOL        LogFuelFlow;
        BOOL        LogEndurance;
        BOOL        LogFlightTime;
        BOOL        LogFlightLength;
        CStringA    LogDumpPath;
        WORD        LogUpdate;
    } FlightInformation;

```



## 2 FSacars Flight Plan Format (José Oliveira)

[Fsacars]	
Arricao	Arrival Airport ICAO
Depicao	Departing Airport ICAO
AltIicao	Alternate Airport ICAO
FP	Flight Plan
Planeicao	Plane ICAO
FL	Flight Level: Fxxx, Axxx, FLxxx, xxxxx
FlightIATA	Flight number
AircraftRegist	Aircraft Registration
PAX	Passengers
Cargo	Cargo
Fuel	Fuel
Units	LBS, KGS

## 3 Fsacars Gauge SDK

### 3.1 Gauge characteristics

The gauge has 3 internal pages.

1 full screen page with 9 lines and 45 characters, this page is written in one shot. Called Menu from this point forward. Each shot has up to 9 lines, each line delimited by \n. The page must be limited by \0.

2 pages with line up line down word wrap and internal buffer capable of 500 lines. Called Page1 and Page 2 from this point forward. Each line is buffered in the gauge should be terminated \n the shot should be terminated with /0. There is no limit to the amount of simultaneous written lines except the data block buffer

Command line with 45 characters.

### 3.2 Gauge orientation

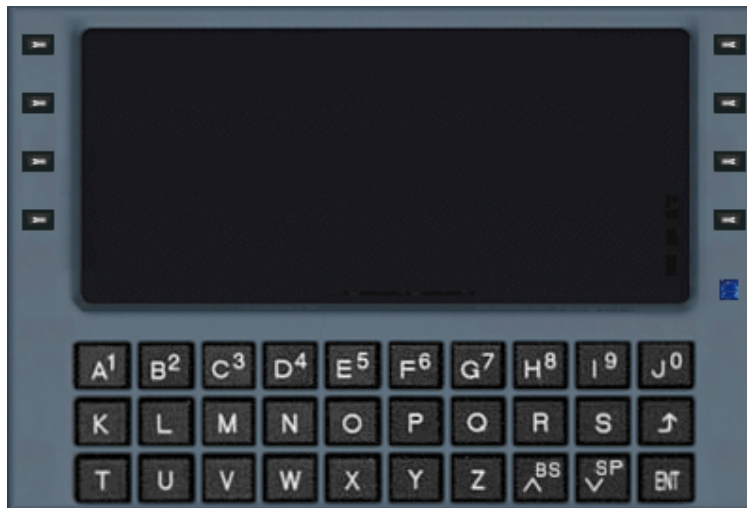
The FSacars gauge is a two parts gauge:

1. A hotspot 


This will allow showing or hiding the main gauge. This hotspot can be invisible and not displaying the FSacars logo. The panel with the hotspot must be visible before launching


FSAcars.


## 2. The gauge itself




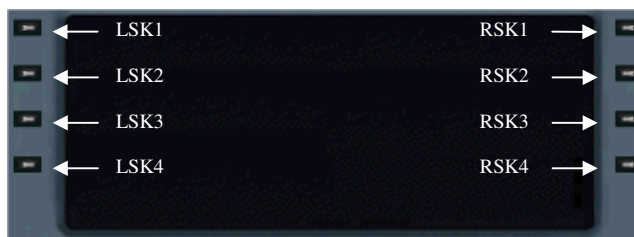
The keyboard is operated by mouse. Clicking the main screen will hide the gauge. The gauge will show the ACARS and LOG messages and will accept commands using the keyboard. The gauge will operate if the batteries are on. There are special keys:

Shift Key . This key will allow to move one level up in the menus.

Enter Key . This key will validate a command. Instructing the gauge to send the command to FSAcars program

Scroll up, Backspace Key . This key will scroll up the gauge messages. When issuing a command the key will delete the last character.

Scroll down, Space, key . This key will scroll down the gauge messages. When issuing a command the key will insert a space.



LSK1,LSK2,LSK3,LSK4, RSK1,RSK2,RSK3,RSK4. These keys allow choosing

one option in the menu. The closest key will choose the correspondent menu entry.

### **3.3 License**

Freeware use: You can change bitmaps and the source code is available on request.

Commercial program use: You can change Bitmaps code changing must be requested to me.

In both cases the program readme or manual must state very clearly:

**This program interfaces with FSAcars Gauge by Jose Oliveira.**

Although this product has been intensively tested, the authors accept no responsibility for any damages caused by the use or misuse of this Software. This software is distributed 'as is' with no warranty expressed or implied. The authors will not be responsible for any losses incurred, either directly or indirectly, by the use of this software.

### **3.4 Establishing communication between Gauge and main program**

The gauge has a sockets based server. This server is launched when the hotspot becomes visible in the panel. The server listens to TCP connection requests in port 1147. By request this can be changed.

The main program can connect to the gauge using the example N°1. The fsacarsgauge.ini in the Gauges directory can indicate a new port to prevent conflicts the syntax is:

[TCPIP]

Port=xxxx

Where xxxx is the port number. If you have FS9 and FS8 installed the ini file must be in the FS9 directory.

### **3.5 Exchanging messages between the gauge and the program**

The messages between the program and the gauge use the following data structure:

```
typedef struct DataBlock {  
    WORD    MessageType;  
    char    Message [2000];  
};
```

```
} DataBlock;
```

MessageType define the Message context.

### 3.5.1 Messages from Gauge to program

MessageType=2

Message: "LSK1", "LSK2", "LSK3", "LSK4", "RSK1", "RSK2", "RSK3", "RSK4"

Indicates that one of the lateral buttons was choosen.

Message: "Shift"

Indicates that the shift key was pressed.

Message: Any string

Different from above indicates a free text input in the gauge command line. Of course if the user writes LSK1 or any of the above messages the effect is the same as above.

MessageType=0 or 1 or 3

Not allowed

### 3.5.2 Messages from Program to Gauge

MessageType=0

Indicates to the gauge that the Message is for Page1. It doesn't display the page.

MessageType=1

Indicates to the gauge that the Message is for Page2. It doesn't display the page.

MessageType=2

Indicates to the gauge that the Message is a command.

"Page1" Forces the Page 1 display. The display pointer is the last line in the buffer.

"Page2" Forces the Page 2 display. The display pointer is the last line in the buffer.

"Page3" Forces the Menu page display.

"ClearPage1" Forces the Page 1 to clear. The display pointer is the first line in the buffer. The internal buffer is cleared. Doesn't show the page.

"ClearPage2" Forces the Page 2 to clear. The display pointer is the first line in the buffer. The internal buffer is cleared. Doesn't show the page.

"ClearPage3" Forces the Page 3 to clear. Doesn't show the page.

"Page1Top" Forces the page1 display from the beginning.

"Page2Top" Forces the page1 display from the beginning.

MessageType=3

Indicates to the gauge that the Message is for Menu Page. It doesn't display the page.

## 3.6 Program examples

Example 1: Connection to the server using non-blocking sockets

```
int ConnectServer (SOCKET * SocketClient)
{
    WORD version;
    WSADATA wsaData;
    int rVal=0;

    // init the socket
    version = MAKEWORD(2,0);
    rVal=WSAStartup(version, (LPWSADATA)&wsaData);
    if(rVal) {
        rVal=WSAGetLastError ();
        Return (1);
    }

    // store information about the server
    struct addrinfo *res;
    rVal=getaddrinfo ("127.0.0.1", "1147", NULL, (addrinfo**) &res);
    if(rVal) {
        rVal=WSAGetLastError ();
        WSACleanup();
        Return (2);
    }

    // create the socket
    theSocket = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(theSocket == SOCKET_ERROR) {
        rVal=WSAGetLastError ();
        WSACleanup();
        Return (3);
    }
}
```

```

    }

    // connect to server
    rVal=connect (SocketClient,res->ai_addr, sizeof(*res->ai_addr));
    if(rVal!=0) {
        rVal=WSAGetLastError ();
        closesocket(SocketClient);
        SocketClient=INVALID_SOCKET;
        WSACleanup();
        Return (4);
    }

    // make socket non blocking
    unsigned long dummy=10;
    rVal=ioctlsocket (SocketClient,FIONBIO ,& dummy);
    if(rVal) {
        rVal=WSAGetLastError ();
        closesocket(SocketClient);
        SocketClient=INVALID_SOCKET;
        WSACleanup();
        Return (5);
    }
    return (0);
}

```

#### Example 2: Sending data to the server, socket as global

```

if (SocketClient!=INVALID_SOCKET) {
    ACARSDataBlock.MessageType=ACARS;
    strcpy (ACARSDataBlock.Message,message1);
    send(SocketClient, (char*)&ACARSDataBlock, sizeof(DataBlock), 0);
}

```

#### Example 3: Receiving data from the server, implemented using a timer callback, socket as global

```

VOID CALLBACK TimerCommand( HWND , UINT , UINT , DWORD )
{
    int nbytes,i=2,j;
    char *buffer;
    unsigned long Error;
    nbytes=recv(SocketClient, (char*)&ACARSDataBlock, sizeof(DataBlock),
0);

    switch (nbytes) {
    case SOCKET_ERROR:
        switch ((Error=WSAGetLastError ())) {
            case WSAEMSGSIZE:
                do {
                    buffer=(char*)malloc(sizeof(DataBlock)*i);
                    ++i;
                    nbytes=recv(SocketClient, (char*)buffer,
sizeof(buffer), 0);
                }while (nbytes!=WSAEMSGSIZE);
                for (j=0;j<i-2;++j){
                    memcpy (&ACARSDataBlock,
buffer+sizeof(DataBlock)*j,
sizeof(DataBlock));
                }
                break;
            case WSAEWOULDBLOCK:
                break;
            default:
                closesocket(SocketClient);
                SocketClient =INVALID_SOCKET;
                break;
        }
        break;
    default:
        if (nbytes!=0){
            strcpy (CommentMetar,ACARSDataBlock.Message);

```

```
CommandGauge (CommentMetar);  
    }  
    break;  
}
```