Offset Mapping for PMDG 737NGX

[Revised edition for 737NGX update SP1d and FSUIPC 4.95 and later, now including CDU screen data]

PLEASE READ THIS FIRST:

Developers using FSUIPC to interface with the PMDG line of products must be aware of and comply with certain restrictions designed to prevent the use of PMDG products in a for-hire or pilot training environment. Please see the PMDG EULA that accompanies the NGX, 777X and 747 line of products for details.

Subject to the above condition, the facilities for reading the PMDG 737NGX data direct from FSUIPC4 offsets are included with kind permission of PMDG.

To enable the data communication output from the PMDG aircraft, you will need to open the file 737NGX_Options.ini (located in the FSX folder PMDG\PMDG 737 NGX, and add the following lines to the end of the file:

```
[SDK]
EnableDataBroadcast=1
```

For CDU screen data you also need one or both of these lines:

```
EnableCDUBroadcast.0=1
EnableCDUBroadcast.1=1
```

Which enable the contents of the corresponding CDU screen to be sent to FSUIPC.

Please also note that the offsets are only populated with data whilst the PMDG 737NGX is running *and* SimConnect is supplying the "Client Data".

At the time of release it appears that there may be a problem, either with SimConnect or with the NGX, which stops the flow of data for either all re-loads of the NGX after the first, or every alternate load. Reports differ on this. Some say that re-connecting with SimConnect fixes it, though this doesn't work for me either. If you want to try this you can assign a button or keypress to FSUIPC's special re-connection control:

```
Re-simconnect
```

and use this after reloading the NGX.

Notes for programmers

All offsets are READ ONLY. To change values please use the Events (known as "controls" in FSUIPC) as listed in the "PMDG_NGX_SDK.h" file which you can find in the PMDG 737NGX SDK. The numerical values of those controls can be used directly in button and key assignments in the FSUIPC4.INI file, or from Lua plug-ins using the ipc.control function.

The list here is simply a version of the full list in the PMDG_NGX_SDK.h file with the hexadecimal offset, size in bytes, and type of value added. Programmers using C/C++ would be better off using the original header file directly and simply mapping the PMDG_NGX_Data structure direct to an offset area, but do note that the reserved area of 168 bytes at the end are NOT mapped to offsets.

The data is provided exactly as provided by the PMDG code

CDU Screen Data

This is provided the raw matrix form provided by PMDG, in offsets 0x5400-0x57FF (for CDU 0) and 0x5800-0x5BFF (for CDU1).

NOTE that these offsets are also used by Project Magenta. You cannot use the
PMDG and PM at the same time if you want to read this data!

For reference, I've included the format definition, copied from the PMDG SDK header file on the next page, *with my own notes added in italics:*

NGX CDU Screen Cell Structure

The Symbol is the ASCII code of the character to be drawn plus the following special symbols:
\xA1: left arrow
\xA2: right arrow

In fact there are also other special non-ASCII characters used -- the boxes indicating places where a value must be supplied by the pilot, for instance, are not ASCII.

```
Struct PMDG_NGX_CDU_Cell
{       unsigned char   Symbol;
        unsigned char   Color;   // any of PMDG_NGX_CDU_COLOR_ defines
        unsigned char   Flags;   // a combination of PMDG_NGX_CDU_FLAG_ bits
};

// NGX CDU Screen Data Structure
#define CDU_COLUMNS 24
#define CDU_ROWS 14

struct PMDG_NGX_CDU_Screen
{       PMDG_NGX_CDU_Cell Cells[CDU_COLUMNS][CDU_ROWS];
        Bool Powered;   // true if the CDU is powered
};
```

/ This structure does seem to be a little odd to me. The 'powered' flag is right at the end – i.e 3 x 24 x 14 bytes from the start of the data. Since the whole screen should be blank without power it would seem better at the beginning.*

*However, even more odd is having the data ordered in terms of columns first. This means, for example, that the first 14 sets of 3-byte values represent the left-most column from top to bottom. This had me puzzled a while during testing, so take care! */*

```
// NGX CDU Screen Cell Colors
#define PMDG_NGX_CDU_COLOR_WHITE            0
#define PMDG_NGX_CDU_COLOR_CYAN             1
#define PMDG_NGX_CDU_COLOR_GREEN            2
#define PMDG_NGX_CDU_COLOR_MAGENTA          3
#define PMDG_NGX_CDU_COLOR_AMBER            4
#define PMDG_NGX_CDU_COLOR_RED              5

// NGX CDU Screen Cell flags
#define PMDG_NGX_CDU_FLAG_SMALL_FONT  0x01    // small font,e.g. used for line headers
#define PMDG_NGX_CDU_FLAG_REVERSE     0x02    // highlighted in reverse video
#define PMDG_NGX_CDU_FLAG_UNUSED      0x04    // dimmed character color
```

Offset	Size	Data type	Name	Notes
Aft overhead				
ADIRU				
6420	1	BYTE	IRS_DisplaySelector	Positions 0..4
6421	1	BYTE	IRS_SysDisplay_R	Boolean: false: L true: R
6422	1	BYTE	IRS_annunGPS	Boolean
6423	2	BYTE x 2	IRS_annunALIGN[2]	Booleans
6425	2	BYTE	IRS_annunON_DC[2]	Booleans
6427	2	BYTE x 2	IRS_annunFAULT[2]	Booleans
6429	2	BYTE x 2	IRS_annunDC_FAIL[2]	Booleans
642B	2	BYTE x 2	IRS_ModeSelector[2]	0: OFF 1: ALIGN 2: NAV 3: ATT
PSEU				
642D	1	BYTE	WARN_annunPSEU	Boolean
SERVICE INTERPHONE				
642E	1	BYTE	COMM_ServiceInterphoneSw	Boolean
LIGHTS				
642F	1	BYTE	LTS_DomeWhiteSw	0: DIM 1: OFF 2: BRIGHT
ENGINE				
6430	2	BYTE x 2	ENG_EECSwitch[2]	Boolean
6432	2	BYTE x 2	ENG_annunREVERSER[2]	Boolean
6434	2	BYTE x 2	ENG _annunENGINE_CONTROL[2]	Boolean
6436	2	BYTE x 2	ENG_annunALTN[2]	Boolean
OXYGEN				
6438	1	BYTE	OXY_Needle	Position 0...240
6439	1	BYTE	OXY_SwNormal	Boolean
643A	1	BYTE	OXY_annunPASS_OXY_ON	Boolean true: NORMAL false: ON
GEAR				
643B	1	BYTE	GEAR_annunOvhdLEFT	Boolean
643C	1	BYTE	GEAR_annunOvhdNOSE	Boolean
643D	1	BYTE	GEAR_annunOvhdRIGHT	Boolean
FLIGHT RECORDER				
643E	1	BYTE	FLTREC_SwNormal	Boolean true: NORMAL false: TEST
643F	1	BYTE	FLTREC_annunOFF	Boolean

Forward overhead

FLIGHT CONTROLS				
6440	2		FCTL_FltControl_Sw[2]	0: STBY/RUD 1: OFF 2: ON
6442	2	BYTE x 2	FCTL_Spoiler_Sw[2]	Boolean true: ON false: OFF
6444	1	BYTE	FCTL_YawDamper_Sw	Boolean
6445	1	BYTE	FCTL_AltnFlaps_Sw_ARM	Boolean true: ARM false: OFF
6446	1	BYTE	FCTL_AltnFlaps_Control_Sw	0: UP 1: OFF 2: DOWN
6447	2	BYTE x 2	FCTL_annunFC_LOW_PRESSURE[2]	Boolean
6449	1	BYTE	FCTL_annunYAW_DAMPER	Boolean
644A	1	BYTE	FCTL_annunLOW_QUANTITY	Boolean
644B	1	BYTE	FCTL_annunLOW_PRESSURE	Boolean
644C	1	BYTE	FCTL_annunLOW_STBY_RUD_ON;	Boolean
644D	1	BYTE	FCTL_annunFEEL_DIFF_PRESS	Boolean
644E	1	BYTE	FCTL_annunSPEED_TRIM_FAIL	Boolean
644F	1	BYTE	FCTL_annunMACH_TRIM_FAIL	Boolean
6450	1	BYTE	FCTL_annunAUTO_SLAT_FAIL	Boolean
NAVIGATION/DISPLAYS				
6451	1	BYTE	NAVDIS_VHFNavSelector	0: BOTH ON 1 1: NORMAL 2: BOTH ON 2
6452	1	BYTE	NAVDIS_IRSSelector	0: BOTH ON L 1: NORMAL 2: BOTH ON R
6453	1	BYTE	NAVDIS_FMCSelector	0: BOTH ON L 1: NORMAL 2: BOTH ON R
6454	1	BYTE	NAVDIS_SourceSelector	0: ALL ON 1 1: AUTO 2: ALL ON 2
6455	1	BYTE	NAVDIS_ControlPaneSelector	0: BOTH ON 1 1: NORMAL 2: BOTH ON 2
FUEL				
6458	4	FLT32	FUEL_FuelTempNeedle	
645C	1	BYTE	FUEL_CrossFeedSw	
645D	2	BYTE x 2	FUEL_PumpFwdSw[2	Boolean
645F	2	BYTE x 2	FUEL_PumpAftSw[2]	Boolean left aft / right aft
6461	2	BYTE x 2	FUEL_PumpCtrSw[2]	Boolean ctr left / ctr right
6463	2	BYTE x 2	FUEL_annunENG_VALVE_CLOSED[2]	Boolean
6465	2	BYTE x 2	FUEL _annunSPAR_VALVE_CLOSED[2]	Boolean
6467	2	BYTE x 2	FUEL_annunFILTER_BYPASS[2]	Boolean
6469	1	BYTE	FUEL_annunXFEED_VALVE_OPEN	Boolean
646A	2	BYTE x 2	FUEL_annunLOWPRESS_Fwd[2]	Boolean
646C	2	BYTE x 2	FUEL_annunLOWPRESS_Aft[2]	Boolean
646E	2	BYTE x 2	FUEL_annunLOWPRESS_Ctr[2]	Boolean

ELECTRICAL				
6470	1	BYTE	ELEC_annunBAT_DISCHARGE	Boolean
6471	1	BYTE	ELEC_annunTR_UNIT	Boolean
6472	1	BYTE	ELEC_annunELEC	Boolean
6473	1	BYTE	ELEC_DCMeterSelector	0: STBY PWR 1: BAT BUS ... 7: TEST
6474	1	BYTE	ELEC_ACMeterSelector	0: STBY PWR 1: GND PWR ... 6: TEST
6475	1	BYTE	ELEC_BatSelector	0: OFF 1: BAT 2: ON
6476	1	BYTE	ELEC_CabUtilSw	Boolean
6477	1	BYTE	ELEC_IFEPassSeatSw	Boolean
6478	2	BYTE x 2	ELEC_annunDRIVE[2]	Boolean
647A	1	BYTE	ELEC_annunSTANDBY_POWER_OFF	Boolean
647B	2	BYTE x 2	ELEC_IDGDisconnectSw[2]	Boolean
647D	1	BYTE	ELEC_StandbyPowerSelector	0: BAT 1: OFF 2: AUTO
647E	1	BYTE	ELEC_annunGRD_POWER_AVAILABLE	Boolean
647F	1	BYTE	ELEC_GrdPwrSw	Boolean
6480	1	BYTE	ELEC_BusTransSw_AUTO	Boolean
6481	2	BYTE x 2	ELEC_GenSw[2]	Boolean
6483	2	BYTE x 2	ELEC_APUGenSw[2]	Boolean
6485	2	BYTE x 2	ELEC_annunTRANSFER_BUS_OFF[2]	Boolean
6487	2	BYTE x 2	ELEC_annunSOURCE_OFF[2]	Boolean
6489	2	BYTE x 2	ELEC_annunGEN_BUS_OFF[2]	Boolean
648B	1	BYTE	ELEC_annunAPU_GEN_OFF_BUS	Boolean
APU				
648C	4	FLT32	APU_EGTNeedle	
6490	1	BYTE	APU_annunMAINT	Boolean
6491	1	BYTE	APU_annunLOW_OIL_PRESSURE	Boolean
6492	1	BYTE	APU_annunFAULT	Boolean
6493	1	BYTE	APU_annunOVERSPEED	Boolean
WIPERS				
6494	1	BYTE	OH_WiperLSelector	0: PARK 1: INT 2: LOW 3:HIGH
6495	1	BYTE	OH_WiperRSelector	0: PARK 1: INT 2: LOW 3:HIGH
CENTRE OVERHEAD CONTROLS & INDICATORS				
6496	1	BYTE	LTS_CircuitBreakerKnob	Position 0...150
6497	1	BYTE	LTS_OvereadPanelKnob	Position 0...150
6498	1	BYTE	AIR_EquipCoolingSupplyNORM	Boolean
6499	1	BYTE	AIR_EquipCoolingExhaustNORM	Boolean
649A	1	BYTE	AIR_annunEquipCoolingSupplyOFF	Boolean
649B	1	BYTE	AIR_annunEquipCoolingExhaustOFF	Boolean
649C	1	BYTE	LTS_annunEmerNOT_ARMED	Boolean
649D	1	BYTE	LTS_EmerExitSelector	0: OFF 1: ARMED 2: ON
649E	1	BYTE	COMM_NoSmokingSelector	0: OFF 1: AUTO 2: ON
649F	1	BYTE	COMM_FastenBeltsSelector	0: OFF 1: AUTO 2: ON
64A0	1	BYTE	COMM_annunCALL	Boolean
64A1	1	BYTE	COMM_annunPA_IN_USE	Boolean

ANTI-ICE				
64A2	4	BYTE x 4	ICE_annunOVERHEAT[4]	Boolean
64A6	4	BYTE x 4	ICE_annunON[4]	Boolean
64AA	4	BYTE x 4	ICE_WindowHeatSw[4]	Boolean
64AE	1	BYTE	ICE_annunCAPT_PITOT	Boolean
64AF	1	BYTE	ICE_annunL_ELEV_PITOT	Boolean
64B0	1	BYTE	ICE_annunL_ALPHA_VANE	Boolean
64B1	1	BYTE	ICE_annunL_TEMP_PROBE	Boolean
64B2	1	BYTE	ICE_annunFO_PITOT	Boolean
64B3	1	BYTE	ICE_annunR_ELEV_PITOT	Boolean
64B4	1	BYTE	ICE_annunR_ALPHA_VANE	Boolean
64B5	1	BYTE	ICE_annunAUX_PITOT	Boolean
64B6	2	BYTE x 2	ICE_TestProbeHeatSw[2]	Boolean
64B8	2	BYTE x 2	ICE_annunVALVE_OPEN[2]	Boolean
64BA	2	BYTE x 2	ICE_annunCOWL_ANTI_ICE[2]	Boolean
64BC	2	BYTE x 2	ICE_annunCOWL_VALVE_OPEN[2]	Boolean
64BE	1	BYTE	ICE_WingAntiIceSw	Boolean
64BF	2	BYTE x 2	ICE_EngAntiIceSw[2]	Boolean
HYDRAULICS				
64C1	2	BYTE x 2	HYD_annunLOW_PRESS_eng[2]	Boolean
64C3	2	BYTE x 2	HYD_annunLOW_PRESS_elec[2]	Boolean
64C5	2	BYTE x 2	HYD_annunOVERHEAT_elec[2]	Boolean
64C7	2	BYTE x 2	HYD_PumpSw_eng[2]	Boolean
64C9	2	BYTE x 2	HYD_PumpSw_elec[2]	Boolean
AIR SYSTEMS				
64CB	1	BYTE	AIR_TempSourceSelector	Positions 0..6
64CC	1	BYTE	AIR_TrimAirSwitch	Boolean
64CD	3	BYTE x 3	AIR_annunZoneTemp[3]	Boolean
64D0	1	BYTE	AIR_annunDualBleed	Boolean
64D1	1	BYTE	AIR_annunRamDoorL	Boolean
64D2	1	BYTE	AIR_annunRamDoorR	Boolean
64D3	2	BYTE x 2	AIR_RecircFanSwitch[2]	Boolean
64D5	2	BYTE x 2	AIR_PackSwitch[2]	0=OFF 1=AUTO 2=HIGH
64D7	2	BYTE x 2	AIR_BleedAirSwitch[2]	Boolean
64D9	1	BYTE	AIR_APUBleedAirSwitch	Boolean
64DA	1	BYTE	AIR_IsolationValveSwitch	Boolean
64DB	2	BYTE x 2	AIR_annunPackTripOff[2]	Boolean
64DD	2	BYTE x 2	AIR_annunWingBodyOverheat[2]	Boolean
64DF	2	BYTE x 2	AIR_annunBleedTripOff[2]	Boolean
64E4	4	DWORD	AIR_FltAltWindow	
64E8	4	DWORD	AIR_LandAltWindow	
64EC	4	DWORD	AIR_OutflowValveSwitch	0=CLOSE 1=NEUTRAL 2=OPEN
64F0	4	DWORD	AIR_PressurizationModeSelector	0=AUTO 1=ALTN 2=MAN
BOTTOM OVERHEAD				
64F4	2	BYTE x 2	LTS_LandingLtRetractableSw[2]	0: RETRACT 1: EXTEND 2: ON
64F6	2	BYTE x 2	LTS_LandingLtFixedSw[2]	Boolean
64F8	2	BYTE x 2	LTS_RunwayTurnoffSw[2]	Boolean
64FA	1	BYTE	LTS_TaxiSw	Boolean
64FB	1	BYTE	APU_Selector	0: OFF 1: ON 2: START

64FC	2	BYTE x 2	ENG_StartSelector[2]	0: GRD 1: OFF 2: CONT 3: FLT
64FE	1	BYTE	ENG_IgnitionSelector	0: IGN L 1: BOTH 2: IGN R
64FF	1	BYTE	LTS_LogoSw	Boolean
6500	1	BYTE	LTS_PositionSw	0: STEADY 1: OFF 2: STROBE & STEADY
6501	1	BYTE	LTS_AntiCollisionSw	Boolean
6502	1	BYTE	LTS_WingSw	Boolean
6503	1	BYTE	LTS_WheelWellSw	Boolean

Glareshield

WARNINGS				
6504	2	BYTE x 2	WARN_annunFIRE_WARN[2]	Boolean
6506	2	BYTE x 2	WARN_annunMASTER_CAUTION[2]	Boolean
6508	1	BYTE	WARN_annunFLT_CONT	Boolean
6509	1	BYTE	WARN_annunIRS	Boolean
650A	1	BYTE	WARN_annunFUEL	Boolean
650B	1	BYTE	WARN_annunELEC	Boolean
650C	1	BYTE	WARN_annunAPU	Boolean
650D	1	BYTE	WARN_annunOVHT_DET	Boolean
650E	1	BYTE	WARN_annunANTI_ICE	Boolean
650F	1	BYTE	WARN_annunHYD	Boolean
6510	1	BYTE	WARN_annunDOORS	Boolean
6511	1	BYTE	WARN_annunENG	Boolean
6512	1	BYTE	WARN_annunOVERHEAD	Boolean
6513	1	BYTE	WARN_annunAIR_COND	Boolean
EFIS CONTROL PANELS				
6514	2	BYTE x 2	EFIS_MinsSelBARO[2]	Boolean
6516	2	BYTE x 2	EFIS_BaroSelHPA[2]	Boolean
6518	2	BYTE x 2	EFIS_VORADFSel1[2]	0: VOR 1: OFF 2: ADF
651A	2	BYTE x 2	EFIS_VORADFSel2[2]	0: VOR 1: OFF 2: ADF
651C	2	BYTE x 2	EFIS_ModeSel[2]	0: APP 1: VOR 2: MAP 3: PLAN
651E	2	BYTE x 2	EFIS_RangeSel[2]	0: 5 ... 7: 640
MODE CONTROL PANEL				
6520	4	WORD x 2	MCP_Course[2]	
6524	4	FLT32	MCP_IASMach	Mach if < 10.0
6528	1	BYTE	MCP_IASBlank	Boolean
6529	1	BYTE	MCP_IASOverspeedFlash	Boolean
652A	1	BYTE	MCP_IASUnderspeedFlash	Boolean
652C	2	WORD	MCP_Heading	
652E	2	WORD	MCP_Altitude	
6530	2	Signed short	MCP_VertSpeed	
6532	1	BYTE	MCP_VertSpeedBlank	Boolean
6533	2	BYTE x 2	MCP_FDSw[2]	Boolean
6535	1	BYTE	MCP_ATArmSw	Boolean

6536	1	BYTE	MCP_BankLimitSel	0: 10 ... 4: 30
6537	1	BYTE	MCP_DisengageBar	Boolean
6538	2	BYTE x 2	MCP_annunFD[2]	Boolean
653A	1	BYTE	MCP_annunATArm	Boolean
653B	1	BYTE	MCP_annunN1	Boolean
653C	1	BYTE	MCP_annunSPEED	Boolean
653D	1	BYTE	MCP_annunVNAV	Boolean
653E	1	BYTE	MCP_annunLVL_CHG	Boolean
653F	1	BYTE	MCP_annunHDG_SEL	Boolean
6540	1	BYTE	MCP_annunLNAV	Boolean
6541	1	BYTE	MCP_annunVOR_LOC	Boolean
6542	1	BYTE	MCP_annunAPP	Boolean
6543	1	BYTE	MCP_annunALT_HOLD	Boolean
6544	1	BYTE	MCP_annunVS	Boolean
6545	1	BYTE	MCP_annunCMD_A	Boolean
6546	1	BYTE	MCP_annunCWS_A	Boolean
6547	1	BYTE	MCP_annunCMD_B	Boolean
6548	1	BYTE	MCP_annunCWS_B	Boolean

Forward Panel

6549	1	BYTE	MAIN_NoseWheelSteeringSwNORM	Boolean, false: ALT
654A	2	BYTE x 2	MAIN_annunBELOW_GS[2]	Boolean
654C	2	BYTE x 2	MAIN_MainPanelDUSel[2];	0: OUTBD PFD ... 4 MFD for Capt Reverse sequence for FO
654E	2	BYTE x 2	MAIN_LowerDUSel[2];	0: ENG PRI .. 2 ND for Capt Reverse sequence for FO
6550	2	BYTE x 2	MAIN_annunAP[2]	Boolean
6552	2	BYTE x 2	MAIN_annunAT[2]	Boolean
6554	2	BYTE x 2	MAIN_annunFMC[2]	Boolean
6556	2	BYTE x 2	MAIN_DisengageTestSelector[2]	0: 1 1: OFF 2: 2
6558	1	BYTE	MAIN_annunSPEEDBRAKE_ARMED	Boolean
6559	1	BYTE	MAIN_annunSPEEDBRAKE_DO_NOT_ARM	Boolean
655A	1	BYTE	MAIN_annunSPEEDBRAKE_EXTENDED	Boolean
655B	1	BYTE	MAIN_annunSTAB_OUT_OF_TRIM	Boolean
655C	1	BYTE	MAIN_LightsSelector	0: TEST 1: BRT 2: DIM
655D	1	BYTE	MAIN_RMISelector1_VOR	Boolean
655E	1	BYTE	MAIN_RMISelector2_VOR	Boolean
655F	1	BYTE	MAIN_N1SetSelector	0: 2 1: 1 2: AUTO 3: BOTH
6560	1	BYTE	MAIN_SpdRefSelector	0: SET 1: AUTO 2: V1 3: VR 4: WT 5: VREF 6: Bug
6561	1	BYTE	MAIN_FuelFlowSelector	0: RESET 1: RATE 2: USED
6562	1	BYTE	MAIN_AutobrakeSelector	0: RTO 1: OFF ... 5: MAX
6563	1	BYTE	MAIN_annunANTI_SKID_INOP	Boolean
6564	1	BYTE	MAIN_annunAUTO_BRAKE_DISARM	Boolean
6565	1	BYTE	MAIN_annunLE_FLAPS_TRANSIT	Boolean

6566	1	BYTE	MAIN_annunLE_FLAPS_EXT	Boolean
6568	8	FLT32 x 2	MAIN_TEFlapsNeedle[2]	
6570	3	BYTE	MAIN_annunGEAR_transit[3]	Boolean
6573	3	BYTE	MAIN_annunGEAR_locked[3]	Boolean
6576	1	BYTE	MAIN_GearLever	0: UP 1: OFF 2: DOWN
6578	4	FLT32	MAIN_BrakePressNeedle	
657C	1	BYTE	HGS_annun_AIII	Boolean
657D	1	BYTE	HGS_annun_NO_AIII	Boolean
657E	1	BYTE	HGS_annun_FLARE	Boolean
657F	1	BYTE	HGS_annun_RO	Boolean
6580	1	BYTE	HGS_annun_RO_CTN	Boolean
6581	1	BYTE	HGS_annun_RO_ARM	Boolean
6582	1	BYTE	HGS_annun_TO	Boolean
6583	1	BYTE	HGS_annun_TO_CTN	Boolean
6584	1	BYTE	HGS_annun_APCH	Boolean
6585	1	BYTE	HGS_annun_TO_WARN	Boolean
6586	1	BYTE	HGS_annun_Bar	Boolean
6587	1	BYTE	HGS_annun_FAIL	Boolean

Lower Forward Panel

6588	2	BYTE x 2	LTS_MainPanelKnob[2]	Position 0...150
658A	1	BYTE	LTS_BackgroundKnob	Position 0...150
658B	1	BYTE	LTS_AFDSFloodKnob	Position 0...150
658C	2	BYTE x 2	LTS_OutbdDUBrtKnob[2];	Position 0...127
658E	2	BYTE x 2	LTS_InbdDUBrtKnob[2]	Position 0...127
6590	2	BYTE x 2	LTS_InbdDUMapBrtKnob[2]	Position 0...127
6592	1	BYTE	LTS_UpperDUBrtKnob	Position 0...127
6593	1	BYTE	LTS_LowerDUBrtKnob	Position 0...127
6594	1	BYTE	LTS_LowerDUMapBrtKnob	Position 0...127
6595	1	BYTE	GPWS_annunINOP	Boolean
6596	1	BYTE	GPWS_FlapInhibitSw_NORM	Boolean
6597	1	BYTE	GPWS_GearInhibitSw_NORM	Boolean
6598	1	BYTE	GPWS_TerrInhibitSw_NORM	Boolean

Control Stand

6599	2	BYTE x 2	CDU_annunEXEC[2]	Boolean
659B	2	BYTE x 2	CDU_annunCALL[2]	Boolean
659D	2	BYTE x 2	CDU_annunFAIL[2]	Boolean
659F	2	BYTE x 2	CDU_annunMSG[2]	Boolean
65A1	2	BYTE x 2	CDU_annunOFST[2]	Boolean
65A3	2	BYTE x 2	CDU_BrtKnob[2]	Position 0...127
65A5	1	BYTE	TRIM_StabTrimMainElecSw_NORMAL	Boolean
65A6	1	BYTE	TRIM_StabTrimAutoPilotSw_NORMAL	Boolean
65A7	1	BYTE	PED_annunParkingBrake	Boolean
65A8	2	BYTE x 2	FIRE_OvhtDetSw[2]	0: A 1: NORMAL 2: B
65AA	2	BYTE x 2	FIRE_annunENG_OVERHEAT[2]	Boolean
65AC	1	BYTE	FIRE_DetTestSw	0: FAULT/INOP 1: neutral 2: OVHT/FIRE

65AD	3	BYTE x 3	FIRE_HandlePos[3]	0: In 1: Blocked 2: Out 3: Turned Left 4: Turned right
65B0	3	BYTE x 3	FIRE_HandleIlluminated[3]	Boolean
65B3	1	BYTE	FIRE_annunWHEEL_WELL	Boolean
65B4	1	BYTE	FIRE_annunFAULT	Boolean
65B5	1	BYTE	FIRE_annunAPU_DET_INOP	Boolean
65B6	1	BYTE	FIRE_annunAPU_BOTTLE_DISCHARGE	Boolean
65B7	2	BYTE x 2	FIRE_annunBOTTLE_DISCHARGE[2]	Boolean
65B9	1	BYTE	FIRE_ExtinguisherTestSw	0: 1 1: neutral 2: 2
65BA	3	BYTE x 3	FIRE_annunExtinguisherTest[3]	Left, Right, APU
65BD	2	BYTE x 2	CARGO_annunExtTest[2]	Fwd, Aft
65BF	2	BYTE x 2	CARGO_DetSelect[2]	0: A 1: NORM 2: B
65C1	2	BYTE x 2	CARGO_ArmedSw[2]	Boolean
65C3	1	BYTE	CARGO_annunFWD	Boolean
65C4	1	BYTE	CARGO_annunAFT	Boolean
65C5	1	BYTE	CARGO_annunDETECTOR_FAULT	Boolean
65C6	1	BYTE	CARGO_annunDISCH	Boolean
65C7	1	BYTE	HGS_annunRWY	Boolean
65C8	1	BYTE	HGS_annunGS	Boolean
65C9	1	BYTE	HGS_annunFAULT	Boolean
65CA	1	BYTE	HGS_annunCLR	Boolean
65CB	1	BYTE	XPDR_XpndrSelector_2;	false: 1 true: 2
65CC	1	BYTE	XPDR_AltSourceSel_2	false: 1 true: 2
65CD	1	BYTE	XPDR_ModeSel	0: STBY 1: ALT RPTG OFF ... 4: TA/RA
65CE	1	BYTE	XPDR_annunFAIL	Boolean
65CF	1	BYTE	LTS_PedFloodKnob	Position 0...150
65D0	1	BYTE	LTS_PedPanelKnob	Position 0...150
65D1	1	BYTE	TRIM_StabTrimSw_NORMAL	Boolean
65D2	1	BYTE	PED_annunLOCK_FAIL	Boolean
65D3	1	BYTE	PED_annunAUTO_UNLK	Boolean
65D4	1	BYTE	PED_FltDkDoorSel	0: UNLKD 1 AUTO pushed in 2: AUTO 3: DENY

Additional variables: used by FS2Crew

65D5	2	BYTE x 2	ENG_StartValve[2]	true: valve open
65D8	8	FLT32 x 2	AIR_DuctPress[2]	PSI
65E0	1	BYTE	COMM_Attend_PressCount	Iincremented with each button press
65E1	1	BYTE	COMM_GrdCall_PressCount	Iincremented with each button press
65E2	3	BYTE x 3	COMM_SelectedMic[3]	Array: 0=capt, 1=F/O, 2=observer. Values: : 0=VHF1 1=VHF2 2=VHF3 3=HF1 4=HF2 5=FLT 6=SVC 7=PA
65E8	4	FLT32	FUEL_QtyCenter	LBS
65EC	4	FLT32	FUEL_QtyLeft	LBS

65F0	4	FLT32	FUEL_QtyRight	LBS
65F4	1	BYTE	IRS_aligned	Boolean: At least one IRU is aligned
65F5	1	BYTE	AircraftMode	1: -600 2: -700 3: -700WL 4: -800 5: -800WL 6: -900 7: -900ER
65F6	1	BYTE	WeightInKg	Boolean: false: LBS true: KG
65F7	1	BYTE	GPWS_V1CallEnabled	GPWS V1 callout option enabled
65F8	1	BYTE	GroundConnAvailable	can connect/disconnect ground air/electrics
65F9	1	BYTE	FMC_TakeoffFlaps	degrees, 0 if not set
65FA	1	BYTE	FMC_V1	knots, 0 if not set
65FB	1	BYTE	FMC_VR	knots, 0 if not set
65FC	1	BYTE	FMC_V2	knots, 0 if not set
65FD	1	BYTE	FMC_LandingFlaps	degrees, 0 if not set
65FE	1	BYTE	FMC_LandingVREF	knots, 0 if not set
6600	2	WORD	FMC_CruiseAlt	ft, 0 if not set
6602	2	WORD	FMC_LandingAltitude	ft; -32767 if not available
6604	2	WORD	FMC_TransitionAlt	ft
6606	2	WORD	FMC_TransitionLevel	ft
6608	1	BYTE	FMC_PerfInputComplete	Boolean
660C	4	FLT32	FMC_DistanceToTOD	nm; 0.0 if passed, negative if n/a
6610	4	FLT32	FMC_DistanceToDest	nm, negative if n/a
6614	9	STR [9]	FMC_flightNumber[9]	
661F			Last byte of first reserved area for PMDG 737NGX	

Additional variables: added by update SP1D March 2015

6C00	12	DWORD x 3	COMM_ReceiverSwitches[3]	Bit flags for selector receivers (see ACP_SEL_RECV_VHF1 etc)
6C0C	2	BYTE x 2	MAIN_annunAP_Amber[2]	Boolean
6C0E	2	BYTE X 2	MAIN_annunAT_Amber[2]	Boolean
6C10	4	DWORD	ICE_WindowHeatTestSw	0: OVHT 1: Neutral 2: PWR TEST
6C14	1	BYTE	DOOR_annunFWD_ENTRY	Boolean
6C15	1	BYTE	DOOR_annunFWD_SERVICE	Boolean
6C16	1	BYTE	DOOR_annunAIRSTAIR	Boolean
6C17	1	BYTE	DOOR_annunLEFT_FWD_OVERWING	Boolean
6C18	1	BYTE	DOOR_annunRIGHT_FWD_OVERWING	Boolean
6C19	1	BYTE	DOOR_annunFWD_CARGO	Boolean
6C1A	1	BYTE	DOOR_annunEQUIP	Boolean
6C1B	1	BYTE	DOOR_annunLEFT_AFT_OVERWING	Boolean
6C1C	1	BYTE	DOOR_annunRIGHT_AFT_OVERWING	Boolean
6C1D	1	BYTE	DOOR_annunAFT_CARGO	Boolean

6C1E	1	BYTE	DOOR_annunAFT_ENTRY	Boolean
6C1F	1	BYTE	DOOR_annunAFT_SERVICE	Boolean
6C20	1	BYTE	AIR_annunAUTO_FAIL	Boolean
6C21	1	BYTE	AIR_annunOFFSCHED_DESCENT	Boolean
6C22	1	BYTE	AIR_annunALTN	Boolean
6C23	1	BYTE	AIR_annunMANUAL	Boolean
6C24	4	FLT32	AIR_CabinAltNeedle	ft
6C28	4	FLT32	AIR_CabinDPNeedle	PSI
6C2C	4	FLT32	AIR_CabinVSNeedle	Ft/min
6C30	4	FLT32	AIR_CabinValveNeedle	0 closed … 1 open
6C34	4	FLT32	AIR_TemperatureNeedle	Degrees C
6C38	8	FLT32 x 2	AIR_DuctPressNeedle[2]	
6C40	13	STR[13]	ELEC_MeterDisplayTop[13]	Top line: 3 groups of 4 digits (or symbols) + terminating zero
6C4D	13	STR[13]	ELEC_MeterDisplayBottom[13]	
6C5A	7	STR[7]	IRS_DisplayLeft[7]	
6C61	8	STR[8]	IRS_DisplayRight[8]	
6C69	1	BYTE	IRS_DisplayShowsDots	True if the degrees and decimal dot symbols are shown on the IRS display
6C6A			Last byte of second reserved area for PMDG 737NGX	

Peter L. Dowson, April 2012, by permission of PMDG, Revised 2nd May 2016