Offset Mapping for PMDG 737 (all variants)

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 737 for details.

Subject to the above condition, the facilities for reading the PMDG 737 data direct from FSUIPC7 offsets are included with kind permission of PMDG.

To enable the data communication output from the PMDG aircraft, you will first need to enable data broadcasts. To do this, open the **737_Options.ini** file which for Steam installations is located under the folder

```
[User]\AppData\Roaming\Microsoft Flight Simulator\Packages\pmdg-aircraft-737\work
```

and for MS Store installations under the folder

```
[User]\AppData\Local\Packages\Microsoft.FlightSimulator_8wekyb3d8bbwe\LocalState\packages\pmdg-aircraft-737\work
```

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 aircraft is running *and* SimConnect is supplying the "Client Data".

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 737 SDK. The numerical values of those controls can be used directly in button and key assignments in the FSUIPC7.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 255 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:*

737 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
\xA3: up arrow
\xA4: down 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
```

```c
// NG3 EFB Screen Dimensions
#define EFB_SCREEN_WIDTH                512
#define EFB_SCREEN_HEIGHT               645
#define EFB_SCREEN_BUFF_SIZE            (EFB_SCREEN_WIDTH * EFB_SCREEN_HEIGHT * 2)
```

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
642D	1	BYTE	IRS_aligned	at least one IRU is aligned
642E	7	BYTE x 7	IRS_DisplayLeft[7]	Left display string, zero terminated
6435	8	BYTE x 8	IRS_DisplayRight[8]	Right display string, zero terminated
643D	1	BYTE	IRS_DisplayShowsDots	True if the degrees and decimal dot symbols are shown on the IRS display
643E	1	BYTE	AFS_AutothrottleServosConnected	True if the autothrottle system is driving the thrust levers
643F	1	BYTE	AFS_ControlsPitch	The autoflight system is actively controlling pitch
6440	1	BYTE	AFS_ControlsRoll	The autoflight system is actively controlling roll
PSEU				
6441	1	BYTE	WARN_annunPSEU	Boolean
SERVICE INTERPHONE				
6442	1	BYTE	COMM_ServiceInterphoneSw	Boolean
LIGHTS				
6443	1	BYTE	LTS_DomeWhiteSw	0: DIM 1: OFF 2: BRIGHT
ENGINE				
6444	2	BYTE x 2	ENG_EECSwitch[2]	Boolean
6446	2	BYTE x 2	ENG_annunREVERSER[2]	Boolean
6448	2	BYTE x 2	ENG _annunENGINE_CONTROL[2]	Boolean
644A	2	BYTE x 2	ENG_annunALTN[2]	Boolean
644C	2	BYTE x 2	ENG_StartValve[2]	Boolean
OXYGEN				
644E	1	BYTE	OXY_Needle	Position 0...240
644F	1	BYTE	OXY_SwNormal	Boolean
6450	1	BYTE	OXY_annunPASS_OXY_ON	Boolean true: NORMAL

				false: ON
GEAR				
6451	1	BYTE	GEAR_annunOvhdLEFT	Boolean
6452	1	BYTE	GEAR_annunOvhdNOSE	Boolean
6453	1	BYTE	GEAR_annunOvhdRIGHT	Boolean
FLIGHT RECORDER				
6454	1	BYTE	FLTREC_SwNormal	Boolean true: NORMAL false: TEST
6455	1	BYTE	FLTREC_annunOFF	Boolean
6456	1	BYTE	CVR_annunTEST	Boolean
Forward overhead				
FLIGHT CONTROLS				
6457	2		FCTL_FltControl_Sw[2]	0: STBY/RUD 1: OFF 2: ON
6459	2	BYTE x 2	FCTL_Spoiler_Sw[2]	Boolean true: ON false: OFF
645B	1	BYTE	FCTL_YawDamper_Sw	Boolean
645C	1	BYTE	FCTL_AltnFlaps_Sw_ARM	Boolean true: ARM false: OFF
645D	1	BYTE	FCTL_AltnFlaps_Control_Sw	0: UP 1: OFF 2: DOWN
645E	2	BYTE x 2	FCTL_annunFC_LOW_PRESSURE[2]	Boolean
6460	1	BYTE	FCTL_annunYAW_DAMPER	Boolean
6461	1	BYTE	FCTL_annunLOW_QUANTITY	Boolean
6462	1	BYTE	FCTL_annunLOW_PRESSURE	Boolean
6463	1	BYTE	FCTL_annunLOW_STBY_RUD_ON;	Boolean
6464	1	BYTE	FCTL_annunFEEL_DIFF_PRESS	Boolean
6465	1	BYTE	FCTL_annunSPEED_TRIM_FAIL	Boolean
6466	1	BYTE	FCTL_annunMACH_TRIM_FAIL	Boolean
6467	1	BYTE	FCTL_annunAUTO_SLAT_FAIL	Boolean
NAVIGATION/DISPLAYS				
6468	1	BYTE	NAVDIS_VHFNavSelector	0: BOTH ON 1 1: NORMAL 2: BOTH ON 2
6469	1	BYTE	NAVDIS_IRSSelector	0: BOTH ON L 1: NORMAL 2: BOTH ON R
646A	1	BYTE	NAVDIS_FMCSelector	0: BOTH ON L 1: NORMAL 2: BOTH ON R
646B	1	BYTE	NAVDIS_SourceSelector	0: ALL ON 1 1: AUTO 2: ALL ON 2
646C	1	BYTE	NAVDIS_ControlPaneSelector	0: BOTH ON 1 1: NORMAL 2: BOTH ON 2
6470	4	Unsigned int	ADF_StandbyFrequency	Standby frequency multiplied by 10
FUEL				
6474	4	FLT32	FUEL_FuelTempNeedle	
6478	1	BYTE	FUEL_CrossFeedSw	
6479	2	BYTE x 2	FUEL_PumpFwdSw[2]	Boolean

647B	2	BYTE x 2	FUEL_PumpAftSw[2]	Boolean left aft / right aft
647D	2	BYTE x 2	FUEL_PumpCtrSw[2]	Boolean ctr left / ctr right
647F	2	BYTE x 2	FUEL_AuxFwd[2]	aux fwd A and aux fwd B
6481	2	BYTE x 2	FUEL_AuxAft[2]	aux aft A and aux aft B
6483	1	BYTE	FUEL_FWDBleed	Boolean
6484	1	BYTE	FUEL_AFTBleed	Boolean
6485	1	BYTE	FUEL_GNDXfr	Boolean
6486	2	BYTE x 2	FUEL_annunENG_VALVE_CLOSED[2]	Boolean
6488	2	BYTE x 2	FUEL_annunSPAR_VALVE_CLOSED[2]	Boolean
648A	2	BYTE x 2	FUEL_annunFILTER_BYPASS[2]	Boolean
648C	1	BYTE	FUEL_annunXFEED_VALVE_OPEN	0: CLOSED 1: OPEN (dim) 2: IN TRANSIT (highlighted)
648D	2	BYTE x 2	FUEL_annunLOWPRESS_Fwd[2]	Boolean
648F	2	BYTE x 2	FUEL_annunLOWPRESS_Aft[2]	Boolean
6491	2	BYTE x 2	FUEL_annunLOWPRESS_Ctr[2]	Boolean
6494	4	FLT32	FUEL_QtyCenter	LBS
6498	4	FLT32	FUEL_QtyLeft	LBS
649C	4	FLT32	FUEL_QtyRight	LBS
ELECTRICAL				
64A0	1	BYTE	ELEC_annunBAT_DISCHARGE	Boolean
64A1	1	BYTE	ELEC_annunTR_UNIT	Boolean
64A2	1	BYTE	ELEC_annunELEC	Boolean
64A3	1	BYTE	ELEC_DCMeterSelector	0: STBY PWR 1: BAT BUS ... 7: TEST
64A4	1	BYTE	ELEC_ACMeterSelector	0: STBY PWR 1: GND PWR ... 6: TEST
64A5	1	BYTE	ELEC_BatSelector	0: OFF 1: BAT 2: ON
64A6	1	BYTE	ELEC_CabUtilSw	Boolean
64A7	1	BYTE	ELEC_IFEPassSeatSw	Boolean
64A8	2	BYTE x 2	ELEC_annunDRIVE[2]	Boolean
64AA	1	BYTE	ELEC_annunSTANDBY_POWER_OFF	Boolean
64AB	2	BYTE x 2	ELEC_IDGDisconnectSw[2]	Boolean
64AD	1	BYTE	ELEC_StandbyPowerSelector	0: BAT 1: OFF 2: AUTO
64AE	1	BYTE	ELEC_annunGRD_POWER_AVAILABLE	Boolean
64AF	1	BYTE	ELEC_GrdPwrSw	Boolean
64B0	1	BYTE	ELEC_BusTransSw_AUTO	Boolean
64B1	2	BYTE x 2	ELEC_GenSw[2]	Boolean
64B3	2	BYTE x 2	ELEC_APUGenSw[2]	Boolean
64B5	2	BYTE x 2	ELEC_annunTRANSFER_BUS_OFF[2]	Boolean
64B7	2	BYTE x 2	ELEC_annunSOURCE_OFF[2]	Boolean
64B9	2	BYTE x 2	ELEC_annunGEN_BUS_OFF[2]	Boolean
64BB	1	BYTE	ELEC_annunAPU_GEN_OFF_BUS	Boolean
64BC	13	char[13]	ELEC_MeterDisplayTop[13]	Top line of the display: 3 groups of 4 digits (or symbols) +

				terminating zero
64C9	13	char[13]	ELEC_MeterDisplayBottom[13]	Bottom line of the display
64D6	16	BYTE x 16	ELEC_BusPowered[16]	True if the corresponding bus is powered: DC HOT BATT 0 DC HOT BATT SWITCHED 1 DC BATT BUS 2 DC STANDBY BUS 3 DC BUS 1 4 DC BUS 2 5 DC GROUND SVC 6 AC TRANSFER 1 7 AC TRANSFER 2 8 AC GROUND SVC 1 9 AC GROUND SVC 210 AC MAIN 1 11 AC MAIN 2 12 AC GALLEY 1 13 AC GALLEY 2 14 AC STANDBY 15
APU				
64E8	4	FLT32	APU_EGTNeedle	
64EC	1	BYTE	APU_annunMAINT	Boolean
64ED	1	BYTE	APU_annunLOW_OIL_PRESSURE	Boolean
64EE	1	BYTE	APU_annunFAULT	Boolean
64EF	1	BYTE	APU_annunOVERSPEED	Boolean
WIPERS				
64F0	1	BYTE	OH_WiperLSelector	0: PARK 1: INT 2: LOW 3:HIGH
64F1	1	BYTE	OH_WiperRSelector	0: PARK 1: INT 2: LOW 3:HIGH
CENTRE OVERHEAD CONTROLS & INDICATORS				
64F2	1	BYTE	LTS_CircuitBreakerKnob	Position 0...150
64F3	1	BYTE	LTS_OvereadPanelKnob	Position 0...150
64F4	1	BYTE	AIR_EquipCoolingSupplyNORM	Boolean
64F5	1	BYTE	AIR_EquipCoolingExhaustNORM	Boolean
64F6	1	BYTE	AIR_annunEquipCoolingSupplyOFF	Boolean
64F7	1	BYTE	AIR_annunEquipCoolingExhaustOFF	Boolean
64F8	1	BYTE	LTS_annunEmerNOT_ARMED	Boolean
64F9	1	BYTE	LTS_EmerExitSelector	0: OFF 1: ARMED 2: ON
64FA	1	BYTE	COMM_NoSmokingSelector	0: OFF 1: AUTO 2: ON
64FB	1	BYTE	COMM_FastenBeltsSelector	0: OFF 1: AUTO 2: ON
64FC	1	BYTE	COMM_annunCALL	Boolean
64FD	1	BYTE	COMM_annunPA_IN_USE	Boolean
ANTI-ICE				
64FE	4	BYTE x 4	ICE_annunOVERHEAT[4]	Boolean
6502	4	BYTE x 4	ICE_annunON[4]	Boolean
6506	4	BYTE x 4	ICE_WindowHeatSw[4]	Boolean
650A	1	BYTE	ICE_annunCAPT_PITOT	Boolean
650B	1	BYTE	ICE_annunL_ELEV_PITOT	Boolean
650C	1	BYTE	ICE_annunL_ALPHA_VANE	Boolean
650D	1	BYTE	ICE_annunL_TEMP_PROBE	Boolean
650E	1	BYTE	ICE_annunFO_PITOT	Boolean
650F	1	BYTE	ICE_annunR_ELEV_PITOT	Boolean

6510	1	BYTE	ICE_annunR_ALPHA_VANE	Boolean
6511	1	BYTE	ICE_annunAUX_PITOT	Boolean
6512	2	BYTE x 2	ICE_TestProbeHeatSw[2]	Boolean
6514	2	BYTE x 2	ICE_annunVALVE_OPEN[2]	Boolean
6516	2	BYTE x 2	ICE_annunCOWL_ANTI_ICE[2]	Boolean
6518	2	BYTE x 2	ICE_annunCOWL_VALVE_OPEN[2]	Boolean
651A	1	BYTE	ICE_WingAntiIceSw	Boolean
651B	2	BYTE x 2	ICE_EngAntiIceSw[2]	Boolean
6520	4	INT	ICE_WindowHeatTestSw	0: OVHT 1: Neutral 2: PWR TEST
HYDRAULICS				
6524	2	BYTE x 2	HYD_annunLOW_PRESS_eng[2]	Boolean
6526	2	BYTE x 2	HYD_annunLOW_PRESS_elec[2]	Boolean
6528	2	BYTE x 2	HYD_annunOVERHEAT_elec[2]	Boolean
652A	2	BYTE x 2	HYD_PumpSw_eng[2]	Boolean
652C	2	BYTE x 2	HYD_PumpSw_elec[2]	Boolean
AIR SYSTEMS (part 1)				
652E	1	BYTE	AIR_TempSourceSelector	Positions 0..6
652F	1	BYTE	AIR_TrimAirSwitch	Boolean
6530	3	BYTE x 3	AIR_annunZoneTemp[3]	Boolean
6533	1	BYTE	AIR_annunDualBleed	Boolean
6534	1	BYTE	AIR_annunRamDoorL	Boolean
6535	1	BYTE	AIR_annunRamDoorR	Boolean
6536	2	BYTE x 2	AIR_RecircFanSwitch[2]	Boolean
6538	2	BYTE x 2	AIR_PackSwitch[2]	0=OFF 1=AUTO 2=HIGH
653A	2	BYTE x 2	AIR_BleedAirSwitch[2]	Boolean
653C	1	BYTE	AIR_APUBleedAirSwitch	Boolean
653D	1	BYTE	AIR_IsolationValveSwitch	Boolean
653E	2	BYTE x 2	AIR_annunPackTripOff[2]	Boolean
6540	2	BYTE x 2	AIR_annunWingBodyOverheat[2]	Boolean
6542	2	BYTE x 2	AIR_annunBleedTripOff[2]	Boolean
6544	1	BYTE	AIR_annunAUTO_FAIL	Boolean
6545	1	BYTE	AIR_annunOFFSCHED_DESCENT	Boolean
6546	1	BYTE	AIR_annunALTN	Boolean
6547	1	BYTE	AIR_annunMANUAL	Boolean
6548	8	FLT32 x 2	AIR_DuctPress[2]	PSI
6550	8	FLT32 x 2	AIR_DuctPressNeedle[2]	Value - PSI
6558	4	FLT32	AIR_CabinAltNeedle	Value - ft
655C	4	FLT32	AIR_CabinDPNeedle	Value - PSI
6560	4	FLT32	AIR_CabinVSNeedle	Value - ft/min
6564	4	FLT32	AIR_CabinValveNeedle	Value - 0 (closed) .. 1 (open)
6568	4	FLT32	AIR_TemperatureNeedle	Value - degrees C
656C	6	char[6]	AIR_DisplayFltAlt[6]	Pressurization system FLT ALT window, zero terminated, can be blank or show dashes or show test pattern
6572	6	char[6]	AIR_DisplayLandAlt[6]	Pressurization system LAND ALT window, zero terminated, can

				be blank or show dashes or show test pattern
DOORS				
6578	1	BYTE	DOOR_annunFWD_ENTRY	Boolean
6579	1	BYTE	DOOR_annunFWD_SERVICE	Boolean
657A	1	BYTE	DOOR_annunAIRSTAIR	Boolean
657B	1	BYTE	DOOR_annunLEFT_FWD_OVERWING	Boolean
657C	1	BYTE	DOOR_annunRIGHT_FWD_OVERWING	Boolean
657D	1	BYTE	DOOR_annunFWD_CARGO	Boolean
657E	1	BYTE	DOOR_annunEQUIP	Boolean
657F	1	BYTE	DOOR_annunLEFT_AFT_OVERWING	Boolean
6580	1	BYTE	DOOR_annunRIGHT_AFT_OVERWING	Boolean
6581	1	BYTE	DOOR_annunAFT_CARGO	Boolean
6582	1	BYTE	DOOR_annunAFT_ENTRY	Boolean
6583	1	BYTE	DOOR_annunAFT_SERVICE	Boolean
AIR SYSTEMS (part 2)				
6584	4	DWORD	AIR_FltAltWindow	WARNING obsolete - use AIR_DisplayFltAlt instead
6588	4	DWORD	AIR_LandAltWindow	WARNING obsolete - use AIR_DisplayLandAlt instead
658C	4	DWORD	AIR_OutflowValveSwitch	0=CLOSE 1=NEUTRAL 2=OPEN
6590	4	DWORD	AIR_PressurizationModeSelector	0=AUTO 1=ALTN 2=MAN
BOTTOM OVERHEAD				
6594	2	BYTE x 2	LTS_LandingLtRetractableSw[2]	0: RETRACT 1: EXTEND 2: ON
6596	2	BYTE x 2	LTS_LandingLtFixedSw[2]	Boolean
6598	2	BYTE x 2	LTS_RunwayTurnoffSw[2]	Boolean
659A	1	BYTE	LTS_TaxiSw	Boolean
659B	1	BYTE	APU_Selector	0: OFF 1: ON 2: START
659C	2	BYTE x 2	ENG_StartSelector[2]	0: GRD 1: OFF 2: CONT 3: FLT
659E	1	BYTE	ENG_IgnitionSelector	0: IGN L 1: BOTH 2: IGN R
659F	1	BYTE	LTS_LogoSw	Boolean
65A0	1	BYTE	LTS_PositionSw	0: STEADY 1: OFF 2: STROBE & STEADY
65A1	1	BYTE	LTS_AntiCollisionSw	Boolean
65A2	1	BYTE	LTS_WingSw	Boolean
65A3	1	BYTE	LTS_WheelWellSw	Boolean

Glareshield

WARNINGS				
65A4	2	BYTE x 2	WARN_annunFIRE_WARN[2]	Boolean
65A6	2	BYTE x 2	WARN_annunMASTER_CAUTION[2]	Boolean
65A8	1	BYTE	WARN_annunFLT_CONT	Boolean

65A9	1	BYTE	WARN_annunIRS	Boolean
65AA	1	BYTE	WARN_annunFUEL	Boolean
65AB	1	BYTE	WARN_annunELEC	Boolean
65AC	1	BYTE	WARN_annunAPU	Boolean
65AD	1	BYTE	WARN_annunOVHT_DET	Boolean
65AE	1	BYTE	WARN_annunANTI_ICE	Boolean
65AF	1	BYTE	WARN_annunHYD	Boolean
65B0	1	BYTE	WARN_annunDOORS	Boolean
65B1	1	BYTE	WARN_annunENG	Boolean
65B2	1	BYTE	WARN_annunOVERHEAD	Boolean
65B3	1	BYTE	WARN_annunAIR_COND	Boolean
EFIS CONTROL PANELS				
65B4	2	BYTE x 2	EFIS_MinsSelBARO[2]	Boolean
65B6	2	BYTE x 2	EFIS_BaroSelHPA[2]	Boolean
65B8	2	BYTE x 2	EFIS_VORADFSel1[2]	0: VOR 1: OFF 2: ADF
65BA	2	BYTE x 2	EFIS_VORADFSel2[2]	0: VOR 1: OFF 2: ADF
65BC	2	BYTE x 2	EFIS_ModeSel[2]	0: APP 1: VOR 2: MAP 3: PLAN
65BE	2	BYTE x 2	EFIS_RangeSel[2]	0: 5 … 7: 640
MODE CONTROL PANEL				
65C0	4	WORD x 2	MCP_Course[2]	
65C4	4	FLT32	MCP_IASMach	Mach if < 10.0
65C8	1	BYTE	MCP_IASBlank	Boolean
65C9	1	BYTE	MCP_IASOverspeedFlash	Boolean
65CA	1	BYTE	MCP_IASUnderspeedFlash	Boolean
65CC	2	WORD	MCP_Heading	
65CE	2	WORD	MCP_Altitude	
65D0	2	Signed short	MCP_VertSpeed	
65D2	1	BYTE	MCP_VertSpeedBlank	Boolean
65D3	2	BYTE x 2	MCP_FDSw[2]	Boolean
65D5	1	BYTE	MCP_ATArmSw	Boolean
65D6	1	BYTE	MCP_BankLimitSel	0: 10 … 4: 30
65D7	1	BYTE	MCP_DisengageBar	Boolean
65D8	2	BYTE x 2	MCP_annunFD[2]	Boolean
65DA	1	BYTE	MCP_annunATArm	Boolean
65DB	1	BYTE	MCP_annunN1	Boolean
65DC	1	BYTE	MCP_annunSPEED	Boolean
65DD	1	BYTE	MCP_annunVNAV	Boolean
65DE	1	BYTE	MCP_annunLVL_CHG	Boolean
65DF	1	BYTE	MCP_annunHDG_SEL	Boolean
65E0	1	BYTE	MCP_annunLNAV	Boolean
65E1	1	BYTE	MCP_annunVOR_LOC	Boolean
65E2	1	BYTE	MCP_annunAPP	Boolean
65E3	1	BYTE	MCP_annunALT_HOLD	Boolean
65E4	1	BYTE	MCP_annunVS	Boolean
65E5	1	BYTE	MCP_annunCMD_A	Boolean
65E6	1	BYTE	MCP_annunCWS_A	Boolean
65E7	1	BYTE	MCP_annunCMD_B	Boolean
65E8	1	BYTE	MCP_annunCWS_B	Boolean
65E9	1	BYTE	MCP_indication_powered	Boolean: true when the MCP is powered

				and the MCP windows are indicating

Forward Panel

65EA	1	BYTE	MAIN_NoseWheelSteeringSwNORM	Boolean, false: ALT
65EB	2	BYTE x 2	MAIN_annunBELOW_GS[2]	Boolean
65ED	2	BYTE x 2	MAIN_MainPanelDUSel[2];	0: OUTBD PFD … 4 MFD for Capt Reverse sequence for FO
65EF	2	BYTE x 2	MAIN_LowerDUSel[2];	0: ENG PRI .. 2 ND for Capt Reverse sequence for FO
65F1	2	BYTE x 2	MAIN_annunAP[2]	Boolean
65F3	2	BYTE	MAIN_annunAP_Amber[2]	Boolean
65F5	2	BYTE x 2	MAIN_annunAT[2]	Boolean
65F7	2	BYTE	MAIN_annunAT_Amber[2]	Boolean
65F9	2	BYTE x 2	MAIN_annunFMC[2]	Boolean
65FB	2	BYTE x 2	MAIN_DisengageTestSelector[2]	0: 1 1: OFF 2: 2
65FD	1	BYTE	MAIN_annunSPEEDBRAKE_ARMED	Boolean
65FE	1	BYTE	MAIN_annunSPEEDBRAKE_DO_NOT_AR M	Boolean
65FF	1	BYTE	MAIN_annunSPEEDBRAKE_EXTENDED	Boolean
6600	1	BYTE	MAIN_annunSTAB_OUT_OF_TRIM	Boolean
6601	1	BYTE	MAIN_LightsSelector	0: TEST 1: BRT 2: DIM
6602	1	BYTE	MAIN_RMISelector1_VOR	Boolean
6603	1	BYTE	MAIN_RMISelector2_VOR	Boolean
6604	1	BYTE	MAIN_N1SetSelector	0: 2 1: 1 2: AUTO 3: BOTH
6605	1	BYTE	MAIN_SpdRefSelector	0: SET 1: AUTO 2: V1 3: VR 4: WT 5: VREF 6: Bug
6606	1	BYTE	MAIN_FuelFlowSelector	0: RESET 1: RATE 2: USED
6607	1	BYTE	MAIN_AutobrakeSelector	0: RTO 1: OFF … 5: MAX
6608	1	BYTE	MAIN_annunANTI_SKID_INOP	Boolean
6609	1	BYTE	MAIN_annunAUTO_BRAKE_DISARM	Boolean
660A	1	BYTE	MAIN_annunLE_FLAPS_TRANSIT	Boolean
660B	1	BYTE	MAIN_annunLE_FLAPS_EXT	Boolean
660C	8	FLT32 x 2	MAIN_TEFlapsNeedle[2]	
6614	3	BYTE x 3	MAIN_annunGEAR_transit[3]	Boolean
6617	3	BYTE x 3	MAIN_annunGEAR_locked[3]	Boolean
661A	1	BYTE	MAIN_GearLever	0: UP 1: OFF 2: DOWN
661C	4	FLT32	MAIN_BrakePressNeedle	
6C00	1	BYTE	MAIN_annunCABIN_ALTITUDE	Boolean
6C01	1	BYTE	MAIN_annunTAKEOFF_CONFIG	Boolean
6C02	1	BYTE	HGS_annun_AIII	Boolean
6C03	1	BYTE	HGS_annun_NO_AIII	Boolean
6C04	1	BYTE	HGS_annun_FLARE	Boolean
6C05	1	BYTE	HGS_annun_RO	Boolean
6C06	1	BYTE	HGS_annun_RO_CTN	Boolean
6C07	1	BYTE	HGS_annun_RO_ARM	Boolean
6C08	1	BYTE	HGS_annun_TO	Boolean

6C09	1	BYTE	HGS_annun_TO_CTN	Boolean
6C0A	1	BYTE	HGS_annun_APCH	Boolean
6C0B	1	BYTE	HGS_annun_TO_WARN	Boolean
6C0C	1	BYTE	HGS_annun_Bar	Boolean
6C0D	1	BYTE	HGS_annun_FAIL	Boolean

Lower Forward Panel

6C0E	2	BYTE x 2	LTS_MainPanelKnob[2]	Position 0...150
6C10	1	BYTE	LTS_BackgroundKnob	Position 0...150
6C11	1	BYTE	LTS_AFDSFloodKnob	Position 0...150
6C12	2	BYTE x 2	LTS_OutbdDUBrtKnob[2];	Position 0...127
6C14	2	BYTE x 2	LTS_InbdDUBrtKnob[2]	Position 0...127
6C16	2	BYTE x 2	LTS_InbdDUMapBrtKnob[2]	Position 0...127
6C18	1	BYTE	LTS_UpperDUBrtKnob	Position 0...127
6C19	1	BYTE	LTS_LowerDUBrtKnob	Position 0...127
6C1A	1	BYTE	LTS_LowerDUMapBrtKnob	Position 0...127
6C1B	1	BYTE	GPWS_annunINOP	Boolean
6C1C	1	BYTE	GPWS_FlapInhibitSw_NORM	Boolean
6C1D	1	BYTE	GPWS_GearInhibitSw_NORM	Boolean
6C1E	1	BYTE	GPWS_TerrInhibitSw_NORM	Boolean

Control Stand

6C1F	2	BYTE x 2	CDU_annunEXEC[2]	Boolean
6C21	2	BYTE x 2	CDU_annunCALL[2]	Boolean
6C23	2	BYTE x 2	CDU_annunFAIL[2]	Boolean
6C25	2	BYTE x 2	CDU_annunMSG[2]	Boolean
6C27	2	BYTE x 2	CDU_annunOFST[2]	Boolean
6C29	2	BYTE x 2	CDU_BrtKnob[2]	Position 0...127
6C2B	1	BYTE	COMM_Attend_PressCount	incremented with each button press
6C2C	1	BYTE	COMM_GrdCall_PressCount	incremented with each button press
6C2D	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
6C30	12	dword x 3	COMM_ReceiverSwitches[3]	Bit flags for selector receivers (see ACP_SEL_RECV_VHF1 etc):[0]=Capt, [1]=FO, [2]=Overhead
6C3C	1	BYTE	TRIM_StabTrimMainElecSw_NORMAL	Boolean
6C3D	1	BYTE	TRIM_StabTrimAutoPilotSw_NORMAL	Boolean
6C3E	1	BYTE	PED_annunParkingBrake	Boolean
6C3F	2	BYTE x 2	FIRE_OvhtDetSw[2]	0: A 1: NORMAL 2: B
6C41	2	BYTE x 2	FIRE_annunENG_OVERHEAT[2]	Boolean

6C43	1	BYTE	FIRE_DetTestSw	0: FAULT/INOP 1: neutral 2: OVHT/FIRE
6C44	3	BYTE x 3	FIRE_HandlePos[3]	0: In 1: Blocked 2: Out 3: Turned Left 4: Turned right
6C47	3	BYTE x 3	FIRE_HandleIlluminated[3]	Boolean
6C4A	1	BYTE	FIRE_annunWHEEL_WELL	Boolean
6C4B	1	BYTE	FIRE_annunFAULT	Boolean
6C4C	1	BYTE	FIRE_annunAPU_DET_INOP	Boolean
6C4D	1	BYTE	FIRE_annunAPU_BOTTLE_DISCHARGE	Boolean
6C4E	2	BYTE x 2	FIRE_annunBOTTLE_DISCHARGE[2]	Boolean
6C50	1	BYTE	FIRE_ExtinguisherTestSw	0: 1 1: neutral 2: 2
6C51	3	BYTE x 3	FIRE_annunExtinguisherTest[3]	Left, Right, APU
6C54	2	BYTE x 2	CARGO_annunExtTest[2]	Fwd, Aft
6C56	2	BYTE x 2	CARGO_DetSelect[2]	0: A 1: NORM 2: B
6C58	2	BYTE x 2	CARGO_ArmedSw[2]	Boolean
6C5A	1	BYTE	CARGO_annunFWD	Boolean
6C5B	1	BYTE	CARGO_annunAFT	Boolean
6C5C	1	BYTE	CARGO_annunDETECTOR_FAULT	Boolean
6C5D	1	BYTE	CARGO_annunDISCH	Boolean
6C5E	1	BYTE	HGS_annunRWY	Boolean
6C5F	1	BYTE	HGS_annunGS	Boolean
6C60	1	BYTE	HGS_annunFAULT	Boolean
6C61	1	BYTE	HGS_annunCLR	Boolean
6C62	1	BYTE	XPDR_XpndrSelector_2;	false: 1 true: 2
6C63	1	BYTE	XPDR_AltSourceSel_2	false: 1 true: 2
6C64	1	BYTE	XPDR_ModeSel	0: STBY 1: ALT RPTG OFF ... 4: TA/RA
6C65	1	BYTE	XPDR_annunFAIL	Boolean
6C66	1	BYTE	LTS_PedFloodKnob	Position 0...150
6C67	1	BYTE	LTS_PedPanelKnob	Position 0...150
6C68	1	BYTE	TRIM_StabTrimSw_NORMAL	Boolean
6C69	1	BYTE	PED_annunLOCK_FAIL	Boolean
6C6A	1	BYTE	PED_annunAUTO_UNLK	Boolean
6C6B	1	BYTE	PED_FltDkDoorSel	0: UNLKD 1 AUTO pushed in 2: AUTO 3: DENY

FMS

6C6C	1	BYTE	FMC_TakeoffFlaps	degrees, 0 if not set
6C6D	1	BYTE	FMC_V1	knots, 0 if not set
6C6E	1	BYTE	FMC_VR	knots, 0 if not set
6C6F	1	BYTE	FMC_V2	knots, 0 if not set
6C70	1	BYTE	FMC_LandingFlaps	degrees, 0 if not set
6C71	1	BYTE	FMC_LandingVREF	knots, 0 if not set
6C72	2	WORD	FMC_CruiseAlt	ft, 0 if not set
6C74	2	WORD	FMC_LandingAltitude	ft; -32767 if not available
6C76	2	WORD	FMC_TransitionAlt	ft
6C78	2	WORD	FMC_TransitionLevel	ft

6C7A	1	BYTE	FMC_PerfInputComplete	Boolean
6C7C	4	FLT32	FMC_DistanceToTOD	nm; 0.0 if passed, negative if n/a
6C80	4	FLT32	FMC_DistanceToDest	nm, negative if n/a
6C84	9	STR [9]	FMC_flightNumber[9]	
Generic and misc				
6C8E	2	WORD	AircraftModel	1: -600 2: -700 3: -700 BW 4: -700 SSW 5: -800 6: -800 BW 7: -800 SSW 8: -900 9: -900 BW 10: -900 SSW 11: -900ER BW 12: -900ER SSW 13: -700 14: -700 BDSF SSW 15: -800 BDSF BW 16: -800 BDSF SSW 17: -800 BCF BW 18 -800 BCF SSW 19: -700 BBJ BW 20: -700 BBJ SSW 21: -800 BBJ BW 22: -800 BBJ SSW
6C90	1	BYTE	WeightInKg	false: LBS true: KG
6C91	1	BYTE	GPWS_V1CallEnabled	GPWS V1 callout option enabled
6C92	1	BYTE	GroundConnAvailable	can connect/disconnect ground air/electrics
6C93			**Last byte of first reserved area for PMDG 737**	

John L. Dowson, January 2023, by permission of PMDG