Altera FPGA 全速漂移 开发指南

LVDS 数据收发实例

欢迎加入 FPGA/CPLD 助学小组一同学习交流:

EDN:

http://group.ednchina.com/GROUP_GRO_14596_1375.HTM

ChinaAET: http://group.chinaaet.com/273

淘宝店链接: http://myfpga.taobao.com/ 技术咨询: orand_support@sina.com 特权 HSC 最新资料例程下载地址:

http://pan.baidu.com/s/1pLmZaFx

		版本信息
时间	版本	状态
2016-08-07	V1.00	创建。

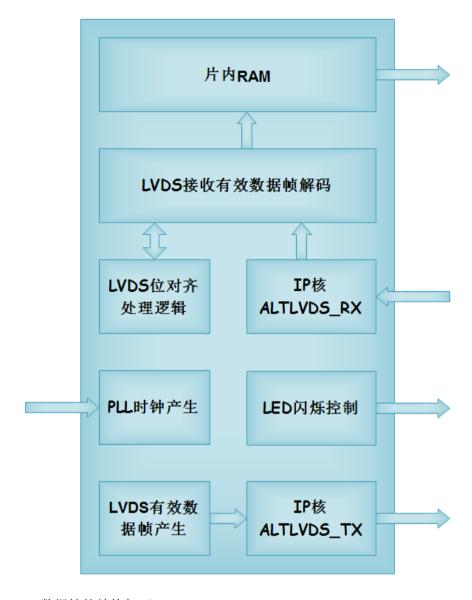


目录

Α	tera FPGA 全速漂移 开发指南	1
	LVDS 数据收发实例	
1	概述	3
2	IP 核 ALTLVDS_TX 创建与配置	6
4	bit align 处理	.13
5	· 装配说明	.16
	板级调试	

1 概述

本实例的 FPGA 功能大体如图所示,由 PLL 产生基准时钟,LED 闪烁用于指示工作运行状态; FPGA 内部产生固定的 1024 字节为单位的有效数据帧,通过 LVDS 发送出去;同时另一则,FPGA 也接收 LVDS 数据,进行位对齐处理,并且解析有效帧数据,同时将最新的有效 LVDS 接收数据帧缓存到片内 RAM 中;PC 上的 Quartus II 可以使用 In-System Memory Content Editor观察片内 RAM 的数据。在 HSC 板上,将 FPGA 的 LVDS 发送和接收数据、时钟通道互联在一起。这样,我们这个实例就可以通过同一颗 FPGA 芯片完成收发实验。



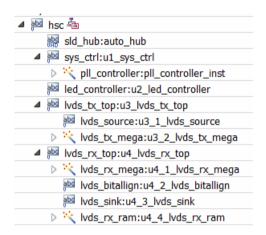
数据帧的结构如下:

N 个 Pattern 数据+4 个字节帧头+1024 字节有效数据+N 个 pattern 数据

如图所示,在 lvds_source.v 模块中定义了 pattern 数据为 0xa6,当 LVDS 数据线上不传输有效帧数据时,总是传输 pattern 数据。4 个字节帧头为固定的 0xcc+0x33+0xc3+0x3c,帧头用于判定接下来要传输有效数据。接下来 1024 个字节是真正有效的 LVDS 数据。

//pattern data, 当不传输有效数据时,不停的发送该数据,用于LVDS RX端bit同步对齐
parameter PATTERN TRAINING = 8'ha6;

本实例的代码层次结构如图所示。



- hsc.v 模式是顶层模块,用于各个子模块间的例化、互联,以及接口引脚的引出。
- sys ctrl.v 模块实现系统基本的复位与时钟(PLL 例化)生成。
- led_controller.v 模块实现 LED 闪烁计数逻辑,用于指示 FPGA 的运行状态。
- lvds_tx_top.v 模块涉及 LVDS 发送数据相关逻辑, 其下例化的 lvds_source.v 模块产生准备发送的 LVDS 数据源,以 1024 个字节为一个 帧单位作为有效数据,定时发送; lvds_tx_mega.v 模块是 IP 核 ALTLVDS_TX 的例化。
- Ivds_rx_top.v 模块涉及 LVDS 接收数据相关逻辑,其下例化的 Ivds_bitalign.v 模块对接收到的 LVDS 数据进行位对齐判断和处理; Ivds_rx_mega.v 模块是 IP 核 ALTLVDS_RX 的例化; Ivds_sink.v 模块对接收到的 LVDS 有效数据帧进行解析; Ivds_rx_ram.v 是 IP 核 RAM 的例化,用于缓存最新收到的一整个帧的 LVDS 数据。

2 IP 核 ALTLVDS_TX 创建与配置

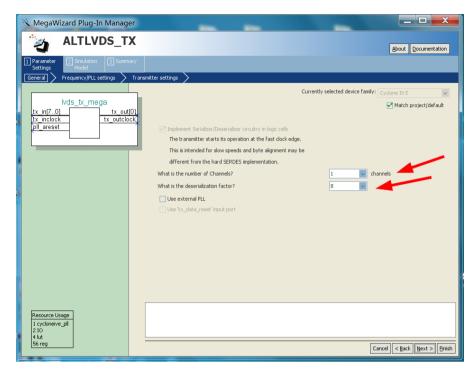
在新建的工程中,点击菜单"Tools→MegaWizard Plug-In Manager"。接着,选择"Creat a new custom megafunction variation",然后 Next。

接着选择我们所需要的IP核,设置如下。

- 在"Select a megafunction from the list below"下面选择 IP 核为"I/O → ALTLVDS_TX"。
- 在"What device family will you be using"后面的下拉栏中选择我们 所使用的器件系列为"Cyclone IV E"。
- 在 "What type of output file do you want to create?" 下面选择语言为 "Verilog"。
- 在"What name do you want for the output file?"下面输入工程所在的路径,并且在最后面加上一个名称,这个名称是我们现在正在例化的 ALTLVDS_TX 模块的名称,我们可以给他起名叫lvds_tx_mega.v,然后点击 Next 进入下一个页面。

如图所示,在第一个配置页面"Parameter Settings → General"中,设定如下。

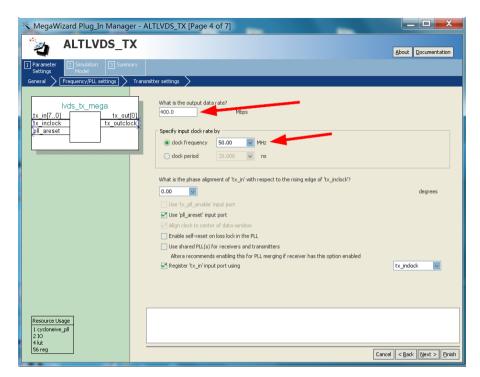
- 在 "What is the number of Channels?" 后面的下拉栏中选择 LVDS 的通道数为 "1" channels。
- 在 "What is the deserialization factor?" 后面的下拉栏中选择串化 因子为 "8"。



如图所示,在第二个配置页面"Parameter Settings → Frequency/PLL settings"中,设定如下。

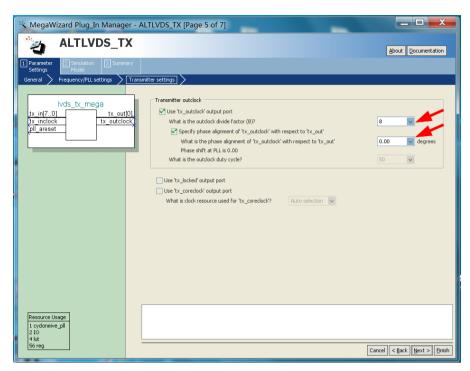
- 在 "What is the output data rate?" 下面输入单通道的 LVDS 数据速率为 "400.0" Mbps。
- 在 "Specify input clock rate by 下面的 "Clock frequency"后面输入 LVDS 时钟频率为 "50.00" MHz。
- 其他选项使用默认设置。

简单提下这里我们所设置的 LVDS 数据速率、LVDS 时钟频率以及上一个配置页面的串化因子之间的关系。LVDS 数据速率 = LVDS 时钟频率 * 串化因子,即 400Mbps = 8 * 50.00MHz。

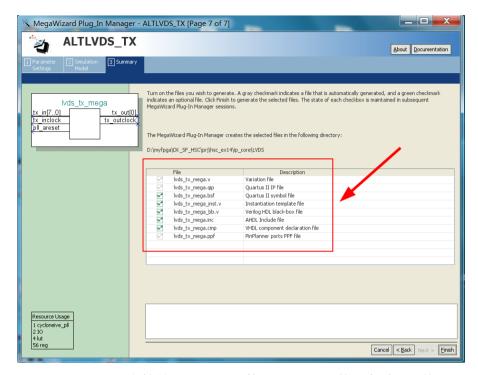


如图所示,在第三个配置页面"Parameter Settings → Transmitter settings"中,设定如下。

- 在"Transmitter outclock"下面选 "Use 'tx_outclock' output port", 设置"What is the outclock divide factor (B)?"为"8"。
- 设置 "Specify phase alignment ……" 选项的相位偏差为 "0.00" degrees。
- 其他选项使用默认设置。



配置完成后,最后在 Summary 页面,如图所示,勾选上*_inst.v 文件,这是一个 ALTLVDS 例化的模板文件,一会我们可以在工程目录下找到这个文件,然后打开它,将它的代码复制到工程中,修改对应接口即可完成这个 IP 核的集成。



ug_altlvds.pdf 文档是 Altera 对 IP 核 ALTLVDS_TX 的配套说明,关于以上配置的详细说明,大家可以参考该文档。

3 IP 核 ALTLVDS_RX 创建与配置

在新建的工程中,点击菜单"Tools→MegaWizard Plug-In Manager"。接着,选择"Creat a new custom megafunction variation",然后 Next。

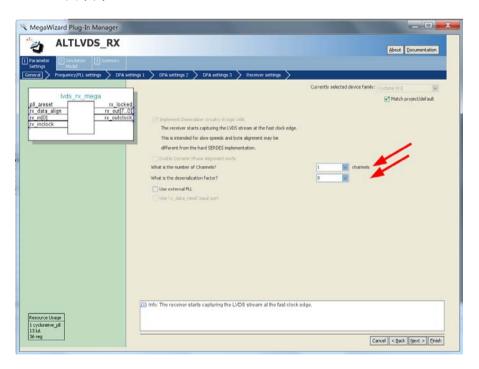
接着选择我们所需要的IP核,设置如下。

- 在"Select a megafunction from the list below"下面选择 IP 核为"I/O → ALTLVDS_RX"。
- 在"What device family will you be using"后面的下拉栏中选择我们所使用的器件系列为"Cyclone IV E"。

- 在"What type of output file do you want to create?"下面选择语言为"Verilog"。
- 在"What name do you want for the output file?"下面输入工程所在的路径,并且在最后面加上一个名称,这个名称是我们现在正在例化的 ALTLVDS_RX 模块的名称,我们可以给他起名叫lvds rx mega.v,然后点击 Next 进入下一个页面。

如图所示,在第一个配置页面"Parameter Settings → General"中,设定如下。

- 在 "What is the number of Channels?" 后面的下拉栏中选择 LVDS 的通道数为 "1" channels。
- 在 "What is the deserialization factor?" 后面的下拉栏中选择串化 因子为 "8"。

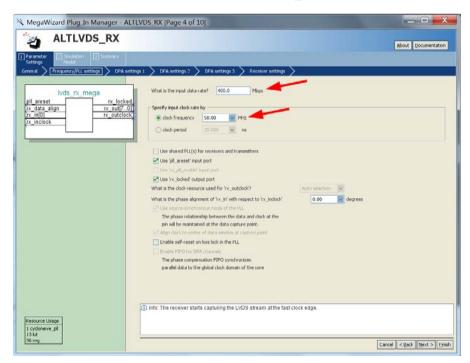


如图所示,在第二个配置页面"Parameter Settings → Frequency/PLL

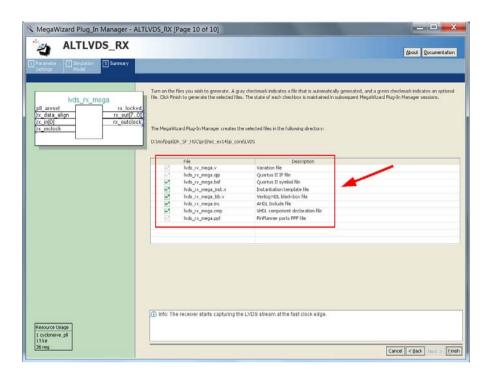
settings"中,设定如下。

- 在 "What is the output data rate?" 下面输入单通道的 LVDS 数据速率为 "400.0" Mbps。
- 在 "Specify input clock rate by 下面的 "Clock frequency"后面输入 LVDS 时钟频率为 "50.00" MHz。
- 其他选项使用默认设置。

这里我们所设置的 LVDS 数据速率、LVDS 时钟频率以及上一个配置页面的串化因子之间的关系。LVDS 数据速率 = LVDS 时钟频率 * 串化因子,即 400Mbps = 8 * 50.00MHz。这和例化 ALTLVDS_TX 的 IP 核是对应的。



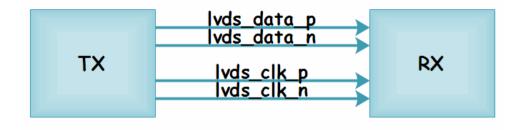
其他页面基本不需要配置,使用默认即可。最后在 Summary 页面,如图所示,勾选上*_inst.v 文件,这是一个 ALTLVDS 例化的模板文件,一会我们可以在工程目录下找到这个文件,然后打开它,将它的代码复制到工程中,修改对应接口即可完成这个 IP 核的集成。



ug_altlvds.pdf 文档是 Altera 对 IP 核 ALTLVDS_RX 的配套说明,关于以上配置的详细说明,大家可以参考该文档。

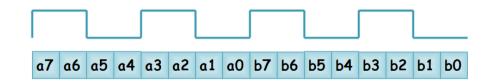
4 bit align 处理

一般情况下,LVDS 传输只有一个固定的时钟差分对和多个数据差分对。每个时钟对应的采集多个数据位的数据。例如,如图所示,只有1个时钟和1个数据的LVDS 传输,1个时钟周期可以传输1bit、2bit、3bit······多个数据位,我们通常称这个时钟和数据的关系为串化因子或解串因子。



我们所使用这个 LVDS 收发实例的串化因子为 8,即一个时钟信号对应 采集的数据是 8bit。

接着,那为什么需要做 bit align 处理呢。如图所示,这里假设有一个LVDS 时钟,对应每个时钟周期传输 4bit 数据,那么实际在接收端,我们需要将这单通道的数据转换为 8bit 的数据字节。这就存在一个问题,当 LVDS 的 TX 端和 RX 端工作时,他们可以不是同时处于工作状态,可能 TX 端先于 RX 端工作,那么 RX 端开始接收数据时,所抓到的数据就不一定是图示最开始一个时钟周期对应的 a7(bit7)了,那么如果它采集到的数据时 a3 或者其他任何一个数据位,直接就把这个数据位作为一个字节的 bit7,可想而知,并不是我们所期望的,组合出来的所有数据都存在"位移",这些数据也无法正确的解析为 TX 端的有效数据信息。

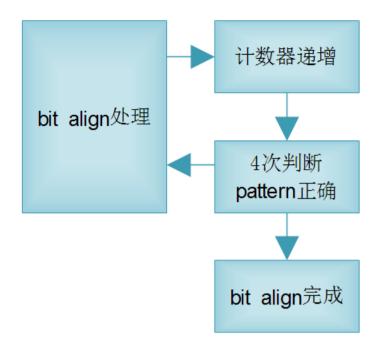


为了解决这个问题,LVDS 的 RX 解串器中提供了一个功能接口,即 bit align(或者称之为 bit slip)。每个时钟周期,LVDS 解串器都会判断这个 bit align 接口信号的状态,若为高电平,则解析出来的数据会相应的移动一位,那么在我们自己的逻辑中,必须设计一个状态机,判断当前数据的各个位是否就绪,即判断所接收到的数据是否为 RX 端和 TX 端事先预定好的数据,

若不是,则拉高一个时钟周期的 bit align 信号,再判断,直到数据接收正确。通常情况下,为了完成这个所谓的 bit align 功能,LVDS 的 TX 端和 RX 端都会约定好,在不传输有效数据时,数据通道上所传输的数据都是固定的数据,并且 TX 端和 RX 端都清楚这个固定数据值。

再回到我们的这里实例上来,每个 LVDS 时钟采样 8bit 数据,正好是 1个字节。LVDS 的 TX 端和 RX 端约定的数据传输协议是:每秒产生一个1028*8bit 的一帧数据,其中头 4*8bit 为固定的帧头 0xcc+0x33+0xc3+0x3c,随后 1024*8bit 为实际有效数据,在其他不传输有效数据时,固定传输Pattern 数据,pattern 数据为固定的数据 8'ha6,在 lvds_source.v 代码中,有如下定义。

如图所示,实例工程中,我们设定一个计数器,在没有完成 bit align 时会循环递增。当计数器值为 16,17,18 和 19 时,分别判断当前字节数据 是否为 pattern 值,若任何一次不是 pattern 值,则 bit align 处理一次;若 判定 4 次数据都为 pttern 值,则 bit align 完成,输出标志信号,同时计数 器清零,不再计数,即不再重新判定 pattern 是否正确。



在 LVDS 的接收端,我们在上电初始需要做一次上述的 bit align 处理, 完成后则可以正常接收数据。

5 装配说明

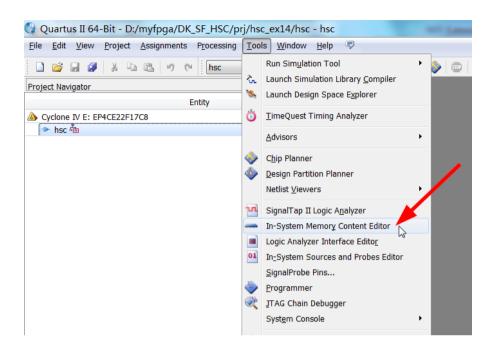
如图所示,首先需要将 P2 连接器用跳线帽短路 P2-7 和 P2-9、P2-8 和 P2-10、P2-15 和 P2-17、P2-16 和 P2-18。



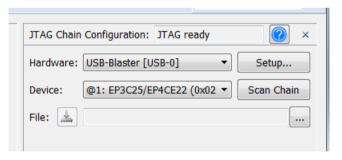
6 板级调试

给 HSC 开发板上电。

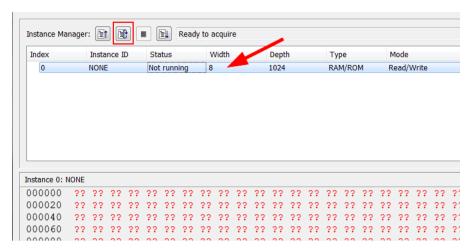
将工程 "…\prj\hsc_ex14\output_files"下的 hsc. sof 下载到 HSC 开发板中,接着如图所示,点击 Quartus II 的菜单 "Tools → In-System Memory Content Editor"。



在 In-System Memory Content Editor 的界面右侧,如图所示,选择 Hardware 为 USB-Blaster (点击 Setup…),接着点击 Scan Chain 确认器 件连接好。



如图所示,选中当前的 RAM/ROM, 然后点击 Instance Manager 后面第二个 run 按钮实时读取 FPGA 中的 RAM 数据。



随机读取到的一组数据如图所示。图示中奇数列的数据始终保持不变, 而偶数列的数据将会持续递增。

Instance 0: N	ONE																															
000000	8E	00	8E	01	8E	02	8E	03	8E	04	8E	05	8E	06	8E	07	8E	08	8E	09	8E	0A	8E	0B	8E	0C	8E	0D	8E	0E	8E	01
000020	8E	10	8E	11	8E	12	8E	13	8E	14	8E	15	8E	16	8E	17	8E	18	8E	19	8E	1A	8E	1B	8E	1C	8E	1D	8E	1E	8E	11
000040	8E	20	8E	21	8E	22	8E	23	8E	24	8E	25	8E	26	8E	27	8E	28	8E	29	8E	2A	8E	2B	8E	2C	8E	2D	8E	2E	8E	21
000060	8E	30	8E	31	8E	32	8E	33	8E	34	8E	35	8E	36	8E	37	8E	38	8E	39	8E	3A	8E	3B	8E	3C	8E	3D	8E	3E	8E	31
000080	8E	40	8E	41	8E	42	8E	43	8E	44	8E	45	8E	46	8E	47	8E	48	8E	49	8E	4A	8E	4B	8E	4C	8E	4D	8E	4E	8E	4
0000a0	8E	50	8E	51	8E	52	8E	53	8E	54	8E	55	8E	56	8E	57	8E	58	8E	59	8E	5A	8E	5B	8E	5C	8E	5D	8E	5E	8E	5
0000c0	8E	60	8E	61	8E	62	8E	63	8E	64	8E	65	8E	66	8E	67	8E	68	8E	69	8E	6A	8E	6B	8E	6C	8E	6D	8E	6E	8E	6
0000e0	8E	70	8E	71	8E	72	8E	73	8E	74	8E	75	8E	76	8E	77	8E	78	8E	79	8E	7A	8E	7B	8E	7C	8E	7D	8E	7E	8E	7
000100	8E	80	8E	81	8E	82	8E	83	8E	84	8E	85	8E	86	8E	87	8E	88	8E	89	8E	8A	8E	8B	8E	8C	8E	8D	8E	8E	8E	8
000120	8E	90	8E	91	8E	92	8E	93	8E	94	8E	95	8E	96	8E	97	8E	98	8E	99	8E	9A	8E	9B	8E	9C	8E	9D	8E	9E	8E	9
000140	8E	A0	8E	A1	8E	A2	8E	A3	8E	Α4	8E	A5	8E	Α6	8E	A7	8E	A8	8E	Α9	8E	AA	8E	AB	8E	AC	8E	AD	8E	ÆΕ	8E	A
000160	8E	B0	8E	B1	8E	B2	8E	B3	8E	B4	8E	B5	8E	В6	8E	В7	8E	B8	8E	В9	8E	BA	8E	BB	8E	BC	8E	BD	8E	BE	8E	В
000180	8E	C0	8E	C1	8E	C2	8E	C3	8E	C4	8E	C5	8E	C6	8E	C7	8E	C8	8E	C9	8E	CA	8E	CB	8E	CC	8E	CD	8E	CE	8E	C
0001a0	8E	D0	8E	D1	8E	D2	8E	D3	8E	D4	8E	D5	8E	D6	8E	D7	8E	D8	8E	D9	8E	DA	8E	DB	8E	DC	8E	DD	8E	DE	8E	I
0001c0	8E	ΕŌ	8E	E1	8E	E2	8E	E3	8E	E4	8E	E5	8E	E6	8E	E7	8E	E8	8E	E9	8E	EA	8E	EB	8E	EC	8E	ED	8E	EE	8E	E
0001e0	8E	F0	8E	F1	8E	F2	8E	F3	8E	F4	8E	F5	8E	F6	8E	F7	8E	F8	8E	F9	8E	FA	8E	FB	8E	FC	8E	FD	8E	FΕ	8E	E
000200	8E	00	8E	01	8E	02	8E	03	8E	04						07	8E		8E	09	8E	0A	8E	0B	8E	0C	8E	0D	8E	0E	8E	0
000220	8E	10	8E	11	8E	12	8E	13	8E	14	8E			16	8E	17	8E			19	8E	1A		1B				1D	8E	1E	8E	1
000240	8E		8E	21	8E	22	8E	23			8E			26	8E	27	8E	28			8E			2B			8E	2D	8E	2E	8E	
000260	8E	30	8E	31	8E	32	8E	33	8E	34	8E	35	8E	36	8E	37	8E	38	8E	39	8E	ЗA	8E	3B	8E	3C	8E	3D	8E	3E	8E	1
000280	8E	40	8E	41	8E		8E		8E		8E			46			8E				8E	4A		4B				4D	8E	4E	8E	4
0002a0	8E	50	8E	51	8E	52	8E	53	8E	54	8E	55	8E	56	8E	57	8E	58	8E	59	8E	5A		5B	8E	5C	8E	5D	8E	5E	8E	
0002c0	8E	60	8E	61	8E	62	8E	63	8E	64	8E	65	8E	66	8E	67	8E	68	8E	69	8E	6A	8E	6B	8E	6C	8E	6D	8E	6E	8E	
0002e0	8E		8E	71	8E	72	8E	73	8E	74	8E	75	8E	76	8E	77	8E	78	8E	79	8E	7A	8E	7B	8E	7C	8E	7D	8E	7E	8E	
000300	8E		8E	81	8E	82	8E	83		84	8E	85		86	8E	87	8E	88			8E				8E	8C	8E	8D	8E	8E	8E	
000320	8E	90	8E	91	8E	92	8E	93	8E	94		95	8E	96	8E	97	8E	98	8E	99	8E	9A	8E	9B	8E			9D	8E	9E	8E	9
000340	8E	A0	8E	A1	8E	A2	8E	A3	8E	A4	8E	A5	8E	А6	8E	A7	8E	A8	8E	Α9	8E	AA	8E	AB	8E	AC	8E	AD	8E	ΑE	8E	7
000360			8E			B2		B3			8E			_		_				B9		BA			8E			BD			8E	_
000380	8E	C0	8E	C1		C2													8E	C9	8E	CA	8E	CB	8E	CC	8E	CD	8E	CE	8E	(
0003a0			8E			D2																DA						DD		DE	8E	I
0003c0			8E			E2					8E						8E				8E			EB				ED		BB	8E	E
0003e0	8E	FO	8E	F1	8E	F2	SE.	F3	8E	F4	8E	FS	8E	F6	SE.	F7	SE.	F8	8E	F9	SE.	FA	SE.	FB	8E	FC	8E	FD	SE.	FE	8E	F