DMA controller API 函数说明

1. void FDmaPs_initDev(FDmaPs_T *pDmac, FDmaPs_Instance_T *pInstance, FDmaPs_Param_T *pParam)

描述	* This function is initialize the DMA device.
参数	* @param * pDmac is the pointer to the DMA controller device. * pInstance is the pointer to instance structure. * pParam is the pointer to parameter structure.
返回值	* @return * NA.

2. void FDmaPs_resetController(void);

描述	* Reset the DMA controller by SLCR registers.
参数	* @param * NA.
返回值	* @return * NA.

3. int FDmaPs_init(FDmaPs_T *pDmac);

描述	* This function is used to initialize the DMA controller. All * interrupts are cleared and disabled; DMA channels are disabled; and * the device instance structure is reset.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

4. void FDmaPs_enable(FDmaPs_T *pDmac);

描述	* This function will enable the DMA controller.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * NA.

5. int FDmaPs_disable(FDmaPs_T *pDmac);

描述	* This function will disable the dma controller.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return

	* returns 0 if the operation was successful. * otherwise returns an error code.

6. BOOL FDmaPs_isEnabled(FDmaPs_T *pDmac);

描述	* This function returns when the DMA controller is enabled.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * TRUE if the DMA controller is enabled. * FALSE if the DMA controller is not enabled.

7. int FDmaPs_enableChannel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function enables the specified DMA channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

8. int FDmaPs_disableChannel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function disables the specified DMA channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

9. BOOL FDmaPs_isChannelEnabled(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether the specified DMA channel is enabled. * Only ONE DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * TRUE if the channel is enabled. * FALSE if the channel is not enabled.

10. uint8_t FDmaPs_getChannelEnableReg(FDmaPs_T *pDmac);

描述	* This function returns the lower byte of the channel enable register(ChEnReg).
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * Contents of the lower byte of the ChEnReg.

11. int FDmaPs_enableChannelIrq(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function enables interrupts for the selected channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

12. int FDmaPs_disableChannelIrq(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function disables interrupts for the selected channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

13. BOOL FDmaPs_isChannelIrqEnabled(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether interrupts are enabled for the * specified DMA channel. Only ONE DMA channel can be specified for * the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * TRUE if channel interrupts are enabled. * FALSE if channel interrupts are not enabled.

14. enum FDmaPs_channelNumber FDmaPs_getFreeChannel(FDmaPs_T *pDmac);

描述	* This function returns a DMA channel number (enumerated) that is * disabled. The function starts at channel 0 and increments up * through the channels until a free channel is found.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * DMA channel number, as an enumerated type.

15. int FDmaPs_suspendChannel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function suspends transfers on the specified channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

16. int FDmaPs_resumeChannel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function resumes (remove suspend) on the specified channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

17. BOOL FDmaPs_isChannelSuspended(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether the specified channel is suspended. * Only ONE DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * TRUE if the channel is suspended. * FALSE if the channel is not suspended.

18. int FDmaPs_clearIrq(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_irq ch_irq);

描述	* This function clears the specified interrupt(s) on the specified channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument. * Multiple interrupt types can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

19. int FDmaPs_maskIrq(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_irq ch_irq);

描述	* This function masks the specified interrupt(s) on the specified channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument. * Multiple interrupt types can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

20. int FDmaPs_unmaskIrq(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_irq ch_irq);

描述	* This function unmasks the specified interrupt(s) on the specified channel(s). * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument. * Multiple interrupt types can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

21. BOOL FDmaPs_isIrqMasked(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_irq ch_irq);

描述	* This function returns whether the specified interrupt on the * specified channel is masked.

	* Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument. * Only 1 interrupt type can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * TRUE if the interrupt is masked. * FALSE if the interrupt is not masked.

22. BOOL FDmaPs_isRawIrqActive(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_irq ch_irq);

描述	* This function returns whether the specified raw interrupt on the * specified channel is active. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument. * Only 1 interrupt type can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * TRUE if the interrupt is masked. * FALSE if the interrupt is not masked.

1. BOOL FDmaPs_isIrqActive(FDmaPs_T *pDmac,
enum FDmaPs_channelNumber ch_num,
enum FDmaPs_irq ch_irq);

描述	* This function returns whether the specified interrupt on the * specified channel is active after masking. * All DMA channels OR only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. * Only 1 interrupt type can be specified for the FDmaPs_irq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_irq Enumerated interrupt type.
返回值	* @return * TRUE if the interrupt is active. * FALSE if the interrupt is not active.

23. int FDmaPs_setChannelConfig(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, FDmaPs_ChannelConfig_T *ch_config);

描述	* This function sets configuration parameters in the DMAC's

	* channel registers on the specified DMA channel. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch Configuration structure handle
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

24. int FDmaPs_getChannelConfig(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, FDmaPs_ChannelConfig_T *ch_config);

描述	* This function gets configuration parameters in the DMAC's * channel registers for the specified DMA channel. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch Configuration structure handle.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

25. int FDmaPs_setTransferType(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_transferType trans_type);

描述	* This function sets up the specified DMA channel(s) for the specified * transfer type. The FDmaPs_transferType enumerated type describes * all of the transfer types supported by the DMA controller. * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * trans_type Enumerated DMA transfer type.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

26. enum FDmaPs_transferType FDmaPs_getTransferType(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the DMA transfer type for the specified DMA channel. * The FDmaPs_transferType enumerated type describes all of the transfer * types supported by the DMA controller. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.

参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated DMA transfer type.

27. BOOL FDmaPs_isBlockTransDone(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether the block transfer of the selected * channel has completed. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * TRUE if the block transfer is done. * FALSE if the block transfer is not done.

28. BOOL FDmaPs_isFifoEmpty(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether the FIFO is empty on the specified channel. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * TRUE if channel FIFO is empty. * FALSE if channel FIFO is not empty.

29. void FDmaPs_setTestMode(FDmaPs_T *pDmac, enum FMSH_state state);

描述	* This function enables/disables test mode in the DMAC.
参数	* @param * pDmac is the pointer to the DMA controller device. * state Enumerated Enabled/Disabled state.
返回值	* @return * NA.

30. enum FMSH_state FDmaPs_getTestMode(FDmaPs_T *pDmac);

描述	* This function returns whether test mode is enabled or disabled * in the DMA controller.
参数	* @param * pDmac is the pointer to the DMA controller device.

返回值	* @return * Enumerated Enabled/Disabled state.

31. int FDmaPs_setSoftwareRequest(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_softwareReq sw_req, enum FMSH_state state);

描述	* This function is used to activate/de-activate the source and * destination software request registers. * Three registers exist for software requests on the source and destination. * These are: Request, Single Request, Last Request. * Use the FDmaPs_softwareReq enum to select which of the three * registers is accessed. * Use the FDmaPs_srcDstSelect enum to select either the source or * destination register. * Multiple DMA channels can be specified for the FDmaPs_channelNumber argument. * Both source and destination can be specified for * the FDmaPs_srcDstSelect argument. * Only 1 request register can be specified for the FDmaPs_softwareReq argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * sw_req Enumerated request register select. * state Enumerated enabled/disabled state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

32. enum FMSH_state FDmaPs_getSoftwareRequest(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_softwareReq sw_req);

描述	* This function is used to return the activate/in-activate status * of the source and destination software request registers. * Three registers exist for software requests on the source and destination. * These are: Request, Single Request, Last Request. * Use the FDmaPs_softwareReq enum to select which of the three * registers is accessed. * Use the FDmaPs_srcDstSelect enum to select either the source or * destination register. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument. * Only 1, source or destination, can be specified for the * FDmaPs_srcDstSelect argument. * Only 1 request register can be specified for the FDmaPs_softwareReq argument.
参数	* @param

	* pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * sw_req Enumerated request register select.
返回值	* @return * Enumerated enabled/disabled state.

33. int FDmaPs_setAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, u32 address);

描述	* This function sets the address on the specified source or/and * destination register of the specified channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination can * be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * address 32-bit address value.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

34. u32 FDmaPs_getAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the address on the specified source or * destination register of the specified channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * 32-bit contents of source/destination address register of the * specified DMA channel.

35. int FDmaPs_setBlockTransSize(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, uint16_t block_size);

描述	* This function sets the block size of a transfer on the specified channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.

参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * block_size Size of the transfers block.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

36. uint16_t FDmaPs_getBlockTransSize(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the block size of a transfer on the specified channel. * Only ONE DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * 16-bit value of the transfer block size.

37. int FDmaPs_setMstSelect(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_masterNumber mst_num);

描述	* This function sets the specified source and/or destination master * select interface on the specified DMA channel(s). * Multiple DMA channels can be specified for the * the FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * mst_num Enumerated master interface number.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

38. enum FDmaPs_masterNumber FDmaPs_getMstSelect(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the specified source or destination master * select interface on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.

参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated master interface number.

39. int FDmaPs_setMemPeriphFlowCtl(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_transferFlow tt_fc);

描述	* This function sets the transfer device type and flow control * (TT_FC) for the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * tt_fc Enumerated transfer device type and flow control.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

40. enum FDmaPs_transferFlow FDmaPs_getMemPeriphFlowCtl(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether scatter mode is enabled or disabled * on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated Enabled/Disabled state.

41. int FDmaPs_setScatterEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FMSH_state state);

描述	* This function enables or disables the destination scatter mode * on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * state Enumerated Enable/Disable state.

返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

42. enum FMSH_state FDmaPs_getScatterEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether scatter mode is enabled or disabled * on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated Enabled/Disabled state.

43. int FDmaPs_setGatherEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FMSH_state state);

描述	* This function enables or disables the source gather mode on the * specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * state Enumerated Enable/Disable state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

44. enum FMSH_state FDmaPs_getGatherEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns whether gather mode is enabled or disabled * on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated Enabled/Disabled state.

45. int FDmaPs_setBurstTransLength(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_burstTransLength length);

描述	* This function sets the specified source and/or destination * burst size on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * length Enumerated burst size.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

46. enum FDmaPs_burstTransLength FDmaPs_getBurstTransLength(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the specified source or destination * burst size on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated burst size.

47. int FDmaPs_setAddressInc(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_addressIncrement addr_inc);

描述	* This function sets the address increment type on the specified * source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * addr_inc Enumerated increment type.
返回值	* @return

	* returns 0 if the operation was successful. * otherwise returns an error code.

48. enum FDmaPs_addressIncrement FDmaPs_getAddressInc(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the address increment type on the specified * source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated address increment type.

49. int FDmaPs_setTransWidth(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_transferWidth width);

描述	* This function sets the specified source and/or destination * transfer width on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * width Enumerated transfer width.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

50. enum FDmaPs_transferWidth FDmaPs_getTransWidth(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the specified source or destination * transfer width on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.

	* sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated transfer width.

51. int FDmaPs_setHsInterface(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_hsInterface hs_inter);

描述	* This function sets the handshaking interface on the specified * source or destination on the specified DMA channel. * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * hs_inter Enumerated handshaking interface number.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

52. enum FDmaPs_hsInterface FDmaPs_getHsInterface(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the handshaking interface on the specified * source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated handshaking interface number.

53. int FDmaPs_setStatUpdate(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FMSH_state state);

描述	* This function enables/disables the specified source and/or * destination status update feature on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device.

	* ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * state Enumerated Enable/Disable state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

54. enum FMSH_state FDmaPs_getStatUpdate(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns whether the status update feature is * enabled or disabled for the specified source or destination * on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated Enabled/Disabled state.

55. int FDmaPs_setProtCtl(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_protLevel prot_lvl);

描述	* This function sets the prot level for the AMBA bus * on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * prot_lvl Enumerated prot level.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

56. enum FDmaPs_protLevel FDmaPs_getProtCtl(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the prot level for the AMBA bus on the * specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param

	* pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated protection level.

57. int FDmaPs_setFifoMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_fifoMode fifo_mode);

描述	* This function sets the FIFO mode on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * fifo_mode Enumerated fifo mode for the DMA.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

58. enum FDmaPs_fifoMode FDmaPs_getFifoMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the FIFO mode on the specified DMA channel(s) * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated FIFO mode for the DMA.

59. int FDmaPs_setFlowCtlMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_flowCtlMode fc_mode);

描述	* This function sets the flow control mode on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * fc_mode Enumerated flow control mode.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

60. enum FDmaPs_flowCtlMode FDmaPs_getFlowCtlMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the flow control mode on the specified DMA channel. * Only 1 DMA channel can be specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated flow control mode.

61. int FDmaPs_setMaxAmbaBurstLength(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, uint16_t burst_length);

描述	* This function sets the maximum amba burst length on the specified * DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * burst_length AMBA burst length value.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

62. uint16_t FDmaPs_getMaxAmbaBurstLength(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the maximum amba burst length * on the specified DMA channel. * Only ONE DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * AMBA burst length value.

63. int FDmaPs_setHsPolarity(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_polarityLevel pol_level);

描述	* This function sets the handshaking interface polarity on the * specified source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination

	* can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * pol_level Enumerated polarity level.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

64. enum FDmaPs_polarityLevel FDmaPs_getHsPolarity(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the handshaking interface polarity on the * specified source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated polarity level.

65. int FDmaPs_setLockLevel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_lockBusCh bus_ch, enum FDmaPs_lockLevel lock_l);

描述	* This function sets the lock level for the specified bus and/or * channel on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * bus_ch Enumerated channel or bus lock select. * lock_l Enumerated level for the lock feature.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

66. enum FDmaPs_lockLevel FDmaPs_getLockLevel(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_lockBusCh bus_ch);

描述	* This function returns the lock level for the specified bus or * channel on the specified DMA channel. Only 1 DMA channel can be

	* specified for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * bus_ch Enumerated channel or bus lock select.
返回值	* @return * Enumerated level for the lock feature.

67. int FDmaPs_setLockEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_lockBusCh bus_ch, enum FMSH_state state);

描述	* This function enables/disables the lock feature on the specified * bus and/or channel on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * bus_ch Enumerated channel or bus lock select. * state Enumerated Enable/Disable state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

68. enum FMSH_state FDmaPs_getLockEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_lockBusCh bus_ch);

描述	* This function returns the enabled or disabled the lock status * on the specified bus or channel on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * bus_ch Enumerated channel or bus lock select.
返回值	* @return * Enumerated Enabled/Disabled state.

69. int FDmaPs_setHandshakingMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FDmaPs_swHwHsSelect hs_hwsw_sel);

描述	* This function sets the handshaking mode from hardware to software * on the specified source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.

参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * hs_hwsw_sel Enumerated software/hardware handshaking select.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

70. enum FDmaPs_swHwHsSelect FDmaPs_getHandshakingMode(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the handshaking mode hardware or software * on the specified source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated software/hardware handshaking select.

71. int FDmaPs_setChannelPriority(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_channelPriority ch_priority);

描述	* This function sets the priority level on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_priority Enumerated priority level.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

72. enum FDmaPs_channelPriority FDmaPs_getChannelPriority(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the priority level on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device.

	* ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated channel priority level.

73. int FDmaPs_setListMstSelect(FDmaPs_T *pDmac,enum FDmaPs_channelNumber ch_num, enum FDmaPs_masterNumber mst_num);

描述	* This function sets the list master select interface on the specified * DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * mst_num Enumerated master interface number.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

74. enum FDmaPs_masterNumber FDmaPs_getListMstSelect(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the list master select interface on the * specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Enumerated master interface number.

75. int FDmaPs_setListPointerAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, u32 address);

描述	* This function sets the address for the first linked list item * in the system memory for the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * address Linked list item address.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

76. u32 FDmaPs_getListPointerAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the address for the first linked list item * in the system memory for the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Linked list item address.

77. int FDmaPs_setLlpEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FMSH_state state);

描述	* This function enables or disables the block chaining on the * specified source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * state Enumerated enable/disable state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

78. enum FMSH_state FDmaPs_getLlpEnable(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns whether block chaining is enabled or disabled * on the specified source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated Enabled/Disabled state.

79. int FDmaPs_setReload(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, enum FMSH_state state);

描述	* This function enables or disables the reload feature on the * specified source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the ch_num argument. Both * source and destination can be specified for the sd_sel argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * state Enumerated enable/disable state.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

80. enum FMSH_state FDmaPs_getReload(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns whether the reload feature is enabled or * disabled on the specified source or destination on the specified * DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * Enumerated Enabled/Disabled state.

81. int FDmaPs_setStatus(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, u32 value);

描述	* This function sets the status registers on the specified source * and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * value 32-bit status value.
返回值	* @return

	* returns 0 if the operation was successful. * otherwise returns an error code.

82. u32 FDmaPs_getStatus(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the status registers on the specified * source or destination on the specified DMA channel. * Only ONE DMA channel can be specified for the * FDmaPs_channelNumber argument. Only ONE, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * 32-bit status value.

83. int FDmaPs_setStatusAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel, u32 address);

描述	* This function sets the status address registers on the specified * source and/or destination on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument. Both source and destination * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select. * address 32-bit address from where status is fetched.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

84. u32 FDmaPs_getStatusAddress(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the status address register on the * specified source or destination on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.

	* sd_sel Enumerated source/destination select.
返回值	* @return * 32-bit address from where status is fetched.

85. int FDmaPs_setGatherParam(FDmaPs_T *pDmac,enum FDmaPs_channelNumber ch_num, enum FDmaPs_scatterGatherParam cnt_int, u32 value);

描述	* This function sets the specified gather interval or count * on the specified DMA channel(s). * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * cnt_int Enumerated count/interval select. * value Count or interval value.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

86. u32 FDmaPs_getGatherParam(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_scatterGatherParam cnt_int);

描述	* This function returns the specified gather interval or count * on the specified DMA channel. Only 1 DMA channel can be specified * for the FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * cnt_int Enumerated count/interval select.
返回值	* @return * Count or interval value.

87. int FDmaPs_setScatterParam(FDmaPs_T *pDmac,enum FDmaPs_channelNumber ch_num, enum FDmaPs_scatterGatherParam cnt_int, u32 value);

描述	* This function sets the specified scatter interval or count on the * specified DMA channel. * Multiple DMA channels can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * cnt_int Enumerated count/interval select. * value Count or interval value.
返回值	* @return

	* returns 0 if the operation was successful. * otherwise returns an error code.

88. u32 FDmaPs_getScatterParam(FDmaPs_T *pDmac,enum FDmaPs_channelNumber ch_num, enum FDmaPs_scatterGatherParam cnt_int);

描述	* This function returns the specified scatter interval or count * on the specified DMA channel. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * ch_int Enumerated count/interval select.
返回值	* @return * Count or interval value.

89. unsigned FDmaPs_getChannelIndex(enum FDmaPs_channelNumber ch_num);

描述	* This function returns the channel index from the specified channel * enumerated type. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * ch_num Enumerated DMA channel number.
返回值	* @return * The DMA channel index.

90. uint8_t FDmaPs_getNumChannels(FDmaPs_T *pDmac);

描述	* This function returns the number of channels that the DMA controller * is configured to have. This function returns the value on the * DMAH_NUM_CHANNELS hardware parameter for the specified DMA * controller device.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * Number of channels that the DMA controller was * configured to have.

91. int FDmaPs_getChannelFifoDepth(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the FIFO depth of the specified DMA channel * that the DMA controller is configured to have. This function * returns the value on the DMAH_CHx_FIFO_DEPTH hardware * parameter for the specified DMA controller device.

参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Depth of the FIFO for the specified DMA channel.

92. void FDmaPs_addLliItem(FMSH_listHead *lhead, FDmaPs_LliItem_T *lli_item, FDmaPs_ChannelConfig_T *config);

描述	* This function creates a Linked List Item or appends a current linked * list with a new item. The FDmaPs_ChannelConfig_T structure handle * contains the values for the FDmaPs_lli_item structure members.
参数	* @param * lhead Handle to a dw_list_head structure. * lli_item Handle to a FDmaPs_lli_item structure. * config Handle to a FDmaPs_ChannelConfig_T structure.
返回值	* @return * NA.

93. int FDmaPs_userIrqHandler(FDmaPs_T *pDmac);

描述	* This function identifies the current highest priority active * interrupt, if any, and forwards it to a user-specified listener * function for processing. This allows a user absolute control over * how each DMAC interrupt is processed.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * FMSH_SUCCESS an interrupt was processed. * FMSH_FAILURE no interrupt was processed.

94. int FDmaPs_irqHandler(FDmaPs_T *pDmac);

描述	* This function handles and processes any DMA controller interrupts. * It works in conjuntion with the Interrupt API and user listener * functions to manage interrupt-driven DMA transfers. Before using * this function, the user must set up a listener function using * FDmaPs_setListener() for the relevant channel(s). When fully using * the Interrupt API, this function should be called whenever a * dmac interrupt occurs.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * FMSH_SUCCESS an interrupt was processed. * FMSH_FAILURE no interrupt was processed.

95. int FDmaPs_startTransfer(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, int num_blocks, FMSH_callback cb_func);

描述	* This function is used to start an interrupt-driven transfer on a * DMA channel. Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. * * The function enables DMA channel interrupts and stores * information needed by the IRQ Handler to control the transfer. * The DMA channel is also enabled to begin the DMA transfer. The * following channel interrupts are enabled and unmasked by this function: * * IntTfr - transfer complete interrupt * IntBlock - block transfer complete interrupt * IntErr - error response on the AMBA AHB bus * * If software handshaking is used on the source and the source * device is a peripheral, the following interrupt is unmasked. If * the transfer set up does not match that described and the user * wants to use this interrupt, the user should unmask the * interrupt using the FDmaPs_unmaskIrq() function prior to calling * this function. * * IntSrcTran - source burst/single tranfer completed * * If software handshaking is used on the destination and the * destination device is a peripheral, the following interrupt * is unmasked. If the transfer setup does not match that described * and the user wants to use this interrupt, the user should * unmask the interrupt using the FDmaPs_unmaskIrq() function prior * to calling this function. * * IntDstTran - destination burst/single tranfer completed * * All channel interrupts are masked and disabled on completion of * the DMA transfer. * * If the number of blocks that make up the DMA transfer is not known, * the user should enter 0 for the num_blocks argument. The user's * listener function is called by the FDmaPs_irqHandler() function * each time a block interrupt occurs. The user can use the * FDmaPs_getBlockCount() API function to fetch the number of blocks * completed by the DMA Controller from within the listener function. * When the total number of blocks is known, the user should call the * FDmaPs_nextBlockIsLast() function also from within the Listener function.

	* The listener function has two arguments, the DMAC device handle * and the interrupt type (FDmaPs_irq). * * At the end of the DMA transfer, the FDmaPs_irqHandler() calls * the user's callback function if the user has specified one. The * callback function has two arguments: the DMAC device handle and * the number of blocks transferred by the DMA Controller.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * num_blocks Number of blocks in the DMA transfer. * cb_func User callback function (can be NULL) - called by ISR.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

96. void FDmaPs_sourceReady(FDmaPs_T *pDmac, unsigned ch_index, BOOL single, BOOL last);

描述	* This function is part of the interrupt-driven interface for the * DMA controller driver. This function writes to the source * software request registers on the DMA controller. * * This function is ONLY useful when the source device is a * peripheral (non-memory) AND that source device is interfacing to * the DMA controller via software handshaking. Under all other * transfer conditions, this function should NOT be used. * * This function should ideally be called inside an ISR for the * source peripheral device to indicate that it is ready for a DMA transfer. * * If the source peripheral is not the flow control device, the * single and last arguments are ignored and should be left at 'FALSE'.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_index DMA channel index (0 to DMAC_MAX_CHANNELS). * * The following arguments are only useful when the source * peripheral is the flow control device: * * single when 'TRUE' requests a single transfer. * when 'FALSE' requests a burst transfer. * last when 'TRUE' the next single/burst transfer is the * last in the current block. * when 'FALSE' the next single/burst transfer is NOT

	* the last in the current block.
返回值	* @return * NA.

97. void FDmaPs_destinationReady(FDmaPs_T *pDmac, unsigned ch_index, BOOL single, BOOL last);

描述	* This function is part of the interrupt-driven interface for the * DMA controller driver. This function writes to the destination * software request registers on the DMA controller. * * This function is ONLY useful when the destination device is a * peripheral (non-memory) AND that destination device is * interfacing to the DMA controller via software handshaking. * Under all other transfer conditions, this function should NOT be used. * * This function should ideally be called inside an ISR for the * destination peripheral device to indicate that it is ready for a * DMA transfer. * * If the destination peripheral is not the flow control device, the * single and last arguments are ignored and should be left at 'FALSE'.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_index DMA channel index (0 to DMAC_MAX_CHANNELS). * * The following arguments are only useful when the destination * peripheral is the flow control device: * * single when 'TRUE' requests a single transfer. * when 'FALSE' requests a burst transfer. * last when 'TRUE' the next single/burst transfer is the * last in the current block. * when 'FALSE' the next single/burst transfer is NOT * the last in the current block.
返回值	* @return * NA.

98. void FDmaPs_setSingleRegion(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function is part of the interrupt-driven interface for the * DMA controller driver. The function is used to instruct the * driver that subsequent transfers in a block are to be completed * using single transfers. * Only 1 DMA channel can be specified for the

	* FDmaPs_channelNumber argument. * * This function is ONLY useful when either the source or * destination device is a peripheral (non-memory) AND that source or * destination device is interfacing to the DMA controller via * software handshaking. Under all other transfer conditions this * function should NOT be used. * * This function is only be needed if the source or destination * transfer can enter a single transaction region. * * If the source or destination enters a single transaction region, * the user has the choice of completing the block transaction using * a single transfer (in which case this function should be called) * or completing the block transaction using a burst transfer, and * allowing the DMA controller to early terminate. * Care should be taken here to set the threshold levels in the * peripheral device to match the requested transfer (single/burst).
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * sd_sel Enumerated source/destination select.
返回值	* @return * NA.

99. void FDmaPs_nextBlockIsLast(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function is part of the interrupt-driven interface in * the DMAC driver. This function is only needed for the * special case where the number of blocks that make up the DMAC * transfer is not known when the transfer is initiated. If this * is the case, the user can monitor the block count in the listener * function and call this function when the last block of the DMAC * transfer is known. Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * NA.

100. void FDmaPs_setListener(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, FMSH_callback userFunction);

描述	* This function is used to set up a listener function for the

	* interrupt handler of the DMAC driver. The listener function is * responsible for handling all interrupts that are not handled * by the Driver Kit interrupt handler. A listener function need to be * setup for each channel that is being used. * * Only 1 DMA channel can be specified for the ch_num argument. There * is no need to clear any interrupts in the listener as this is * handled automatically by the Driver Kit interrupt handlers. Note * that when using the FDmaPs_irqHandler() interrupt handler, the * Dmac_irq_tfr interrupt is never passed to the listener function. * Instead, an optional user-provided callback function is called.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number. * userFunction function pointer to user listener function.
返回值	* @return * NA.

101. int FDmaPs_getBlockCount(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function returns the number of blocks that a DMA * channel has completed transferring. This function should only be * used for interrupt driven transfers. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * Number of blocks completed by the DMA controller.

102. int FDmaPs_getBlockByteCount(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num, enum FDmaPs_srcDstSelect sd_sel);

描述	* This function returns the number of bytes within a DMA block * that have been transferred by the DMA controller on the specified * source or destination. This function should only be used for * interrupt driven transfers, where the SRCTRAN or DSTTRAN * interrupts are enabled and unmasked. * Only 1 DMA channel can be specified for the * FDmaPs_channelNumber argument. Only 1, source or destination, * can be specified for the FDmaPs_srcDstSelect argument.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.

	* sd_sel Enumerated source/destination select.
返回值	* @return * Number of bytes completed on source or destination.

103. void FDmaPs_resetInstance(FDmaPs_T *pDmac);

描述	* This functions resets the FDmaPs_Instance_T structure.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * NA.

104. int FDmaPs_autoCompParams(FDmaPs_T *pDmac);

描述	* This function attempts to automatically discover the hardware * component parameters. * This is usually controlled by the ADD_ENCODED_PARAMS coreConsultant * parameter.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

105. int FDmaPs_checkChannelBusy(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function checks if the specified DMA channel is Busy (enabled) * or not. Also checks if the specified channel is in range.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

106. int FDmaPs_checkChannelRange(FDmaPs_T *pDmac, enum FDmaPs_channelNumber ch_num);

描述	* This function checks if the specified DMA channel is in range.
参数	* @param * pDmac is the pointer to the DMA controller device. * ch_num Enumerated DMA channel number.
返回值	* @return * returns 0 if the operation was successful. * otherwise returns an error code.

107. void FDmaPs_setChannelPriorityOrder(FDmaPs_T *pDmac);

描述	* This function places each channel number into an ordered array * based on the priority level setting for each channel.
参数	* @param * pDmac is the pointer to the DMA controller device.
返回值	* @return * NA.