LPC5410x ROM Driver Deprecation

Overview

Due to complications associated with ROM based peripheral drivers, existing LPC5410x ROM peripheral drivers are being deprecated. This and subsequent releases of LPCOpen Software for LPC5410x will no longer provide support for ROM driver interfaces for the following peripherals:

- ADC
- DMA
- I2C (master, slave and monitor)
- SPI (master and slave)
- UART

These ROM drivers are being replaced by traditional LPCOpen software drivers. The functionality that was encapsulated in ROM drivers can be replicated using software drivers. This document augments the existing LPCOpen driver documentation and provides a consolidated list of supported API's organized by functionality as well as documenting critical points essential to successfully port code that used the ROM drivers to the updated LPCOpen Software v3.xx for the LPC5410x.

Existing ROM structure

The following considerations must be made to successfully replace the services provided by the ROM drivers.

- *ROM drivers did not configure peripheral clocks or enable power domains. Existing customer developed software that utilize ROM drivers will already have provisions for the required configuration and power settings.*
- *ROM handler functions (ROM_XXX_Handler) attempt to transfer data using registered call back functions and configuration data. This functionality must now be implemented using driver read / write routines in polled, IRQ or DMA transfer modes.*

Porting

Use the LPCOpen API's (listed below) to replace functionality provided by ROM drivers with consideration for the structural differences identified in the overview above. For convenience, the API's have been grouped into Configuration / Status, Send / Receive and Transfer Handler groups where applicable. Additional detail for each API is provided in corresponding LPCOpen header files (identified below).

Peripheral Interfaces

ADC LPCOpen Interface (chip_lpc5410x/inc/adc_5410x.h)

Config / Status

```
void Chip_ADC_SetIntBits(LPC_ADC_T *pADC, uint32_t intMask)
void Chip_ADC_ClearIntBits(LPC_ADC_T *pADC, uint32_t intMask)
void Chip_ADC_SetTHRSELBits(LPC_ADC_T *pADC, uint32_t mask)
void Chip_ADC_ClearTHRSELBits(LPC_ADC_T *pADC, uint32_t mask)
void Chip_ADC_Init(LPC_ADC_T *pADC, uint32_t flags);
```

void Chip_ADC_DeInit(LPC_ADC_T *pADC);
void Chip_ADC_SetDivider(LPC_ADC_T *pADC, uint8_t div)
void Chip_ADC_SetClockRate(LPC_ADC_T *pADC, uint32_t rate);
uint8_t Chip_ADC_GetDivider(LPC_ADC_T *pADC)
uint32_t Chip_ADC_Calibration(LPC_ADC_T *pADC);
void Chip_ADC_SetSequencerBits(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex, uint32_t bits)
void Chip_ADC_ClearSequencerBits(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex, uint32_t bits)
void Chip_ADC_SetupSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex, uint32_t options)
uint32_t Chip_ADC_GetSequencerCtrl(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
void Chip_ADC_EnableSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
void Chip_ADC_DisableSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
void Chip_ADC_StartBurstSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
void Chip_ADC_StopBurstSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
uint32_t Chip_ADC_GetGlobalDataReg(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)
void Chip_ADC_SetThrLowValue(LPC_ADC_T *pADC, uint8_t thrnum, uint16_t value)
void Chip_ADC_SetThrHighValue(LPC_ADC_T *pADC, uint8_t thrnum, uint16_t value)
void Chip_ADC_SelectTH0Channels(LPC_ADC_T *pADC, uint32_t channels)
void Chip_ADC_SelectTH1Channels(LPC_ADC_T *pADC, uint32_t channels)
void Chip_ADC_EnableInt(LPC_ADC_T *pADC, uint32_t intMask)
void Chip_ADC_DisableInt(LPC_ADC_T *pADC, uint32_t intMask)
void Chip_ADC_SetThresholdInt(LPC_ADC_T *pADC, uint8_t ch, ADC_INTEN_THCMP_T thInt)
uint32_t Chip_ADC_GetFlags(LPC_ADC_T *pADC)
void Chip_ADC_ClearFlags(LPC_ADC_T *pADC, uint32_t flags)
void Chip_ADC_StartSequencer(LPC_ADC_T *pADC, ADC_SEQ_IDX_T seqIndex)

Send / Receive
uint32_t Chip_ADC_GetDataReg(LPC_ADC_T *pADC, uint8_t index)

DMA LPCOpen Interface (chip_lpc5410x/inc/dma_5410x.h)

Config / Status
void Chip_DMA_Init(LPC_DMA_T *pDMA)
void Chip_DMA_DeInit(LPC_DMA_T *pDMA)
void Chip_DMA_Enable(LPC_DMA_T *pDMA)
void Chip_DMA_Disable(LPC_DMA_T *pDMA)
uint32_t Chip_DMA_GetIntStatus(LPC_DMA_T *pDMA)
void Chip_DMA_SetSRAMBase(LPC_DMA_T *pDMA, uint32_t base)
uint32_t Chip_DMA_GetSRAMBase(LPC_DMA_T *pDMA)
void Chip_DMA_EnableChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_DisableChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
uint32_t Chip_DMA_GetEnabledChannels(LPC_DMA_T *pDMA)
uint32_t Chip_DMA_GetActiveChannels(LPC_DMA_T *pDMA)
uint32_t Chip_DMA_GetBusyChannels(LPC_DMA_T *pDMA)
uint32_t Chip_DMA_GetErrorIntChannels(LPC_DMA_T *pDMA)
void Chip_DMA_ClearErrorIntChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_EnableIntChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_DisableIntChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
uint32_t Chip_DMA_GetEnableIntChannels(LPC_DMA_T *pDMA)
uint32_t Chip_DMA_GetActiveIntAChannels(LPC_DMA_T *pDMA)
void Chip_DMA_ClearActiveIntAChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
uint32_t Chip_DMA_GetActiveIntBChannels(LPC_DMA_T *pDMA)
void Chip_DMA_ClearActiveIntBChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_SetValidChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_SetTrigChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_AbortChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_SetupChannelConfig(LPC_DMA_T *pDMA, DMA_CHID_T ch, uint32_t cfg)
uint32_t Chip_DMA_GetChannelStatus(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_SetupChannelTransfer(LPC_DMA_T *pDMA, DMA_CHID_T ch, uint32_t cfg)
void Chip_DMA_SetTranBits(LPC_DMA_T *pDMA, DMA_CHID_T ch, uint32_t mask)
void Chip_DMA_ClearTranBits(LPC_DMA_T *pDMA, DMA_CHID_T ch, uint32_t mask)
void Chip_DMA_SetupChannelTransferSize(LPC_DMA_T *pDMA, DMA_CHID_T ch, uint32_t trans)
void Chip_DMA_SetChannelValid(LPC_DMA_T *pDMA, DMA_CHID_T ch)
void Chip_DMA_SetChannelInValid(LPC_DMA_T *pDMA, DMA_CHID_T ch)
bool Chip_DMA_SetupTranChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch, DMA_CHDESC_T *desc);

Send / Receive
void Chip_DMA_SWTriggerChannel(LPC_DMA_T *pDMA, DMA_CHID_T ch)

I2C LPCOpen Interface (chip_lpc5410x/inc/i2c_comon_5410x.h, i2cm_5410x.h, i2cs_5410x.h)

```
Config / Status
void Chip_I2C_Init(LPC_I2C_T *pI2C)
void Chip_I2C_DeInit(LPC_I2C_T *pI2C)
void Chip_I2C_SetRegMask(volatile uint32_t *pReg32, uint32_t mask, uint32_t bits)
void Chip_I2C_ClearRegMask(volatile uint32_t *pReg32, uint32_t mask, uint32_t bits)
void Chip_I2C_SetClockDiv(LPC_I2C_T *pI2C, uint32_t clkdiv)
uint32_t Chip_I2C_GetClockDiv(LPC_I2C_T *pI2C)
void Chip_I2C_EnableInt(LPC_I2C_T *pI2C, uint32_t intEn)
void Chip_I2C_DisableInt(LPC_I2C_T *pI2C, uint32_t intClr)
void Chip_I2C_ClearInt(LPC_I2C_T *pI2C, uint32_t intClr)
uint32_t Chip_I2C_GetPendingInt(LPC_I2C_T *pI2C)

    Master:
        Config / Status
        void Chip_I2CM_SetDutyCycle(LPC_I2C_T *pI2C, uint16_t sclH, uint16_t sclL);
        void Chip_I2CM_SetBusSpeed(LPC_I2C_T *pI2C, uint32_t busSpeed);
        void Chip_I2CM_Enable(LPC_I2C_T *pI2C)
        void Chip_I2CM_Disable(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CM_GetStatus(LPC_I2C_T *pI2C)
        void Chip_I2CM_ClearStatus(LPC_I2C_T *pI2C, uint32_t clrStatus)
        bool Chip_I2CM_IsMasterPending(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CM_GetMasterState(LPC_I2C_T *pI2C)

        Send / Receive
        void Chip_I2CM_SendStart(LPC_I2C_T *pI2C)
        void Chip_I2CM_SendStop(LPC_I2C_T *pI2C)
        void Chip_I2CM_MasterContinue(LPC_I2C_T *pI2C)
        void Chip_I2CM_WriteByte(LPC_I2C_T *pI2C, uint8_t data)
        uint8_t Chip_I2CM_ReadByte(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CM_XferHandler(LPC_I2C_T *pI2C, I2CM_XFER_T *xfer);
        void Chip_I2CM_Xfer(LPC_I2C_T *pI2C, I2CM_XFER_T *xfer);
        uint32_t Chip_I2CM_XferBlocking(LPC_I2C_T *pI2C, I2CM_XFER_T *xfer);

    Slave:
        Config / Status
        void Chip_I2CS_Enable(LPC_I2C_T *pI2C)
        void Chip_I2CS_Disable(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CS_GetStatus(LPC_I2C_T *pI2C)
        void Chip_I2CS_ClearStatus(LPC_I2C_T *pI2C, uint32_t clrStatus)
        bool Chip_I2CS_IsSlavePending(LPC_I2C_T *pI2C)
        bool Chip_I2CS_IsSlaveSelected(LPC_I2C_T *pI2C)
        bool Chip_I2CS_IsSlaveDeSelected(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CS_GetSlaveState(LPC_I2C_T *pI2C)
        uint32_t Chip_I2CS_GetSlaveMatchIndex(LPC_I2C_T *pI2C)
        void Chip_I2CS_SlaveEnableDMA(LPC_I2C_T *pI2C)
        void Chip_I2CS_SlaveDisableDMA(LPC_I2C_T *pI2C)
        void Chip_I2CS_SetSlaveAddr(LPC_I2C_T *pI2C, uint8_t slvNum, uint8_t slvAddr)
        uint8_t Chip_I2CS_GetSlaveAddr(LPC_I2C_T *pI2C, uint8_t slvNum)
        void Chip_I2CS_EnableSlaveAddr(LPC_I2C_T *pI2C, uint8_t slvNum)
        void Chip_I2CS_DisableSlaveAddr(LPC_I2C_T *pI2C, uint8_t slvNum)
        void Chip_I2CS_SetSlaveQual0(LPC_I2C_T *pI2C, bool extend, uint8_t slvAddr)

        Send / Receive
        void Chip_I2CS_SlaveContinue(LPC_I2C_T *pI2C)
        void Chip_I2CS_SlaveNACK(LPC_I2C_T *pI2C)
        void Chip_I2CS_WriteByte(LPC_I2C_T *pI2C, uint8_t data)
        uint8_t Chip_I2CS_ReadByte(LPC_I2C_T *pI2C)

        Transfer Handler / Callbacks
        uint32_t Chip_I2CS_XferHandler(LPC_I2C_T *pI2C, const I2CS_XFER_T *xfers);
        typedef void (*I2CSlaveXferStart)(uint8_t addr);
        typedef uint8_t (*I2CSlaveXferSend)(uint8_t *data);
        typedef uint8_t (*I2CSlaveXferRecv)(uint8_t data);
        typedef void (*I2CSlaveXferDone)(void);
```

SPI LPCOpen Interface (chip_lpc5410x/inc/spi_common_5410x.h, spim_5410x.h, spis_5410x.h)

Config / Status
```
void Chip_SPI_SetCFGRegBits(LPC_SPI_T *pSPI, uint32_t bits)
void Chip_SPI_ClearCFGRegBits(LPC_SPI_T *pSPI, uint32_t bits)
void Chip_SPI_Enable(LPC_SPI_T *pSPI)
void Chip_SPI_Disable(LPC_SPI_T *pSPI)
void Chip_SPI_Init(LPC_SPI_T *pSPI)
void Chip_SPI_DeInit(LPC_SPI_T *pSPI)
void Chip_SPI_EnableMasterMode(LPC_SPI_T *pSPI)
void Chip_SPI_EnableSlaveMode(LPC_SPI_T *pSPI)
void Chip_SPI_EnableLSBFirst(LPC_SPI_T *pSPI)
void Chip_SPI_EnableMSBFirst(LPC_SPI_T *pSPI)
void Chip_SPI_SetSPIMode(LPC_SPI_T *pSPI, SPI_CLOCK_MODE_T mode)
void Chip_SPI_SetCSPolHigh(LPC_SPI_T *pSPI, uint8_t csNum)
void Chip_SPI_SetCSPolLow(LPC_SPI_T *pSPI, uint8_t csNum)
void Chip_SPI_ConfigureSPI(LPC_SPI_T *pSPI, SPI_CFGSETUP_T *pCFG);
uint32_t Chip_SPI_GetStatus(LPC_SPI_T *pSPI)
void Chip_SPI_ClearStatus(LPC_SPI_T *pSPI, uint32_t Flag)
void Chip_SPI_EnableInts(LPC_SPI_T *pSPI, uint32_t intMask)
void Chip_SPI_DisableInts(LPC_SPI_T *pSPI, uint32_t intMask)
uint32_t Chip_SPI_GetEnabledInts(LPC_SPI_T *pSPI)
uint32_t Chip_SPI_GetPendingInts(LPC_SPI_T *pSPI)
void Chip_SPI_FlushFifos(LPC_SPI_T *pSPI)
void Chip_SPI_SetTXCTRLRegBits(LPC_SPI_T *pSPI, uint32_t bits)
void Chip_SPI_ClearTXCTRLRegBits(LPC_SPI_T *pSPI, uint32_t bits)
void Chip_SPI_SetTXCtl(LPC_SPI_T *pSPI, uint32_t ctrlBits)
void Chip_SPI_ClearTXCtl(LPC_SPI_T *pSPI, uint32_t ctrlBits)
void Chip_SPI_SetXferSize(LPC_SPI_T *pSPI, uint32_t ctrlBits)
```

Send / Receive
```
void Chip_SPI_WriteTXData(LPC_SPI_T *pSPI, uint16_t data)
uint32_t Chip_SPI_ReadRXData(LPC_SPI_T *pSPI)
uint32_t Chip_SPI_ReadRawRXFifo(LPC_SPI_T *pSPI)
```

Master:

Config / Status
```
uint32_t Chip_SPIM_GetClockRate(LPC_SPI_T *pSPI);
uint32_t Chip_SPIM_SetClockRate(LPC_SPI_T *pSPI, uint32_t rate);
void Chip_SPIM_DelayConfig(LPC_SPI_T *pSPI, SPIM_DELAY_CONFIG_T *pConfig)
void Chip_SPIM_ForceEndOfTransfer(LPC_SPI_T *pSPI)
void Chip_SPIM_AssertSSEL(LPC_SPI_T *pSPI, uint8_t sselNum)
void Chip_SPIM_DeAssertSSEL(LPC_SPI_T *pSPI, uint8_t sselNum)
void Chip_SPIM_EnableLoopBack(LPC_SPI_T *pSPI)
void Chip_SPIM_DisableLoopBack(LPC_SPI_T *pSPI)
```

Send / Receive
```
void Chip_SPIM_Xfer(LPC_SPI_T *pSPI, SPIM_XFER_T *xfer);
void Chip_SPIM_XferBlocking(LPC_SPI_T *pSPI, SPIM_XFER_T *xfer);
```

Transfer Handler / Callbacks
```
typedef void (*SPIMasterXferCSAssert)(struct SPIM_XFER *pMasterXfer);
typedef void (*SPIMasterXferSend)(struct SPIM_XFER *pMasterXfer);
typedef void (*SPIMasterXferRecv)(struct SPIM_XFER *pMasterXfer);
typedef void (*SPIMMasterXferCSDeAssert)(struct SPIM_XFER *pMasterXfer);
typedef void (*SPIMMasterXferDone)(struct SPIM_XFER *pMasterXfer);
void Chip_SPIM_XferHandler(LPC_SPI_T *pSPI, SPIM_XFER_T *xfer);
```

Slave:

Send / Receive
```
void Chip_SPIS_PreBuffSlave(LPC_SPI_T *pSPI, SPIS_XFER_T *xfer)
uint32_t Chip_SPIS_XferBlocking(LPC_SPI_T *pSPI, SPIS_XFER_T *xfer);
```

Transfer Handler / Callbacks
```
typedef void (*SPISlaveXferCSAssert)(struct SPIS_XFER *pSlaveXfer);
typedef void (*SPISlaveXferSend)(struct SPIS_XFER *pSlaveXfer);
typedef void (*SPISlaveXferRecv)(struct SPIS_XFER *pSlaveXfer);
```

```
    typedef void (*SPISlaveXferCSDeAssert)(struct SPIS_XFER *pSlaveXfer);
    uint32_t Chip_SPIS_XferHandler(LPC_SPI_T *pSPI, SPIS_XFER_T *xfer);
```

UART LPCOpen Interface (chip_lpc5410x/inc/uart_5410x.h)

Config / Status

```
uint32_t Chip_UART_CalcBaud(LPC_USART_T *pUART, UART_BAUD_T *pBaud);
void Chip_UART_Enable(LPC_USART_T *pUART)
void Chip_UART_Disable(LPC_USART_T *pUART)
void Chip_UART_TXEnable(LPC_USART_T *pUART)
void Chip_UART_TXDisable(LPC_USART_T *pUART)
uint32_t Chip_UART_AutoBaud(LPC_USART_T *pUART)
void Chip_UART_Div(LPC_USART_T *pUART, uint32_t div, uint32_t ovr)
void Chip_UART_IntEnable(LPC_USART_T *pUART, uint32_t intMask)
void Chip_UART_IntDisable(LPC_USART_T *pUART, uint32_t intMask)
uint32_t Chip_UART_GetIntsEnabled(LPC_USART_T *pUART)
uint32_t Chip_UART_GetIntStatus(LPC_USART_T *pUART)
void Chip_UART_ConfigData(LPC_USART_T *pUART, uint32_t config)
uint32_t Chip_UART_GetStatus(LPC_USART_T *pUART)
void Chip_UART_ClearStatus(LPC_USART_T *pUART, uint32_t stsMask)
void Chip_UART_Init(LPC_USART_T *pUART);
void Chip_UART_DeInit(LPC_USART_T *pUART);
void Chip_UART_SetBaud(LPC_USART_T *pUART, uint32_t baudrate);
```

Send / Receive

```
void Chip_UART_SendByte(LPC_USART_T *pUART, uint8_t data)
uint32_t Chip_UART_ReadByte(LPC_USART_T *pUART)
int Chip_UART_Send(LPC_USART_T *pUART, const void *data, int numBytes);
int Chip_UART_Read(LPC_USART_T *pUART, void *data, int numBytes);
int Chip_UART_SendBlocking(LPC_USART_T *pUART, const void *data, int numBytes);
int Chip_UART_ReadBlocking(LPC_USART_T *pUART, void *data, int numBytes);
void Chip_UART_RXIntHandlerRB(LPC_USART_T *pUART, RINGBUFF_T *pRB);
void Chip_UART_TXIntHandlerRB(LPC_USART_T *pUART, RINGBUFF_T *pRB);
uint32_t Chip_UART_SendRB(LPC_USART_T *pUART, RINGBUFF_T *pRB, const void *data, int count);
int Chip_UART_ReadRB(LPC_USART_T *pUART, RINGBUFF_T *pRB, void *data, int bytes);
void Chip_UART_IRQRBHandler(LPC_USART_T *pUART, RINGBUFF_T *pRXRB, RINGBUFF_T *pTXRB);
```