HAL API SPECIFICATION

Document Revision: 0.2

Issue Date: May 10, 2011

Revision History

Revision	Draft	Author	Date	Comment
0.1	Initial version	Minghua Fu	01/10/2011	Separated from G3 and PRIME PHY spec
0.2	PLCLite added	Jones Chen	05/09/2011	Added PLC Lite

Sign Off List

Group	Title	Authority	Date	Comment

Table of Content

Table of Figures

Table of Tables

1.0 Introduction

This document describes the Hardware Abstraction Layer (HAL) software interface. The HAL includes the controls of the peripherals of the MCU/DSP device and Analog Front End (AFE). It is used directly interfacing with the PHY Layer.

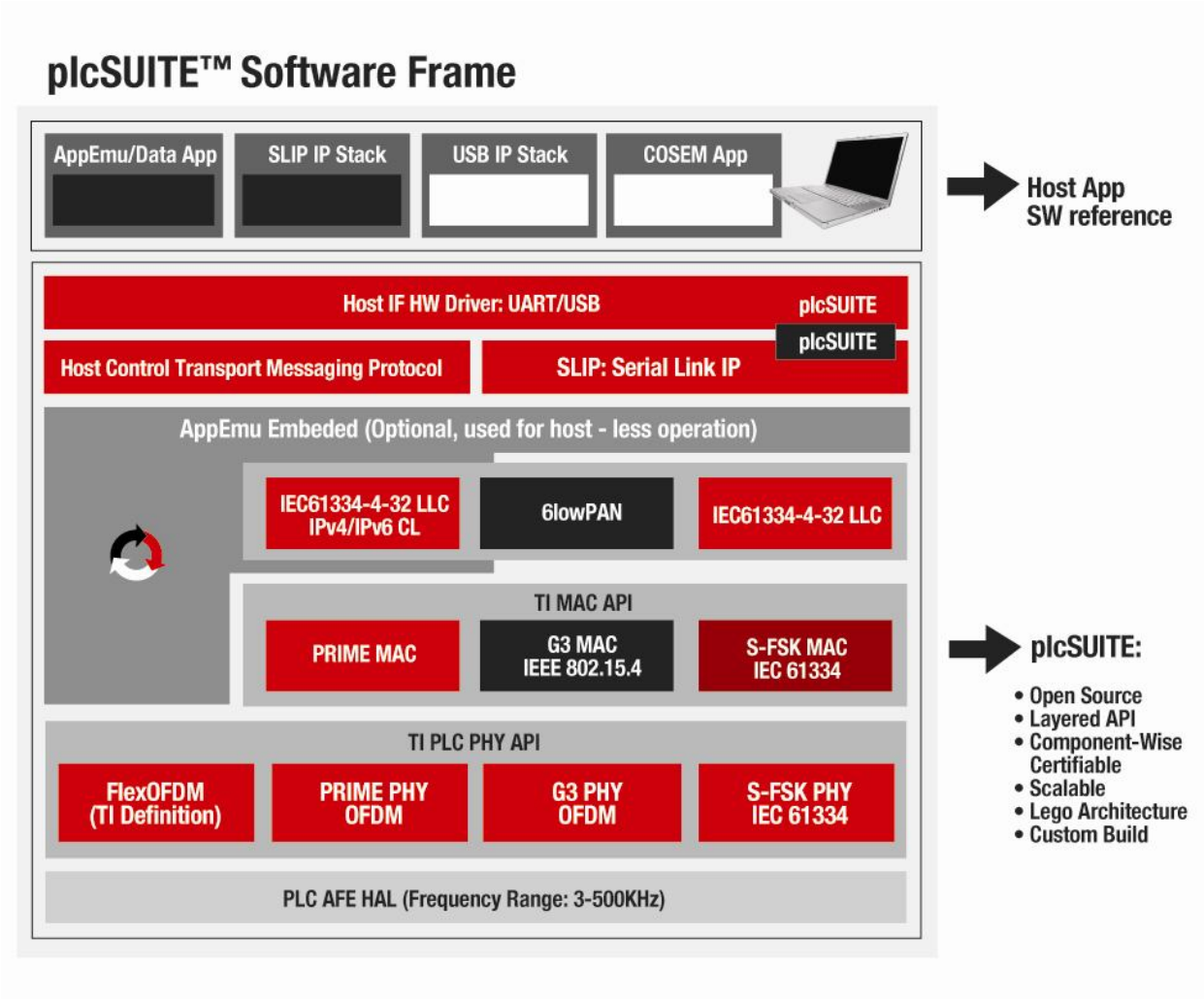

Figure 1 Overall plcSuite Software Architecture

2.0 HAL Layer Public APIs

Table 1 lists all the HAL layer library public API definitions.

API	Sync/Async	Descriptions
HAL_afeInit	Sync	Initialize HAL AFE
HAL_afeRxInit	Sync	Initialize HAL AFE RX chain.
HAL_afeTxInit	Sync	Initialize HAL AFE TX chain
HAL_afeSet	Sync	Set HAL AFE Parameters
HAL_afeGet	Sync	Get HAL AFE Parameters
HAL_afeDacCnv	Sync	HAL AFE DAC Conversion
HAL_afeRxDmaCh1IntFunc	Sync	RX DMA (Channel 1) Interrupt Processing Function
HAL_afeTxDmaCh2IntFunc	Sync	TX DMA (Channel 2) Interrupt Processing Function
HAL_ cpuTint0Func	Sync	CPU Timer0 Interrupt Processing Function
HAL_ afeEcap1IntFunc	Sync	ECAP1 Interrupt Processing Function
HAL_ afeEcap3IntFunc	Sync	ECAP3 Interrupt Processing Function
HAL_ afeEcap4IntFunc	Sync	ECAP4 Interrupt Processing Function
HAL_ haltLPM	Sync	Low Power Mode (LPM-HALT) Interrupt Processing Function
HAL_ xint1Func	Sync	External Interrupt 1 (XINT1) Processing Function
HAL_ xint2Func	Sync	External Interrupt 2 (XINT2) Processing Function
HAL_getLibVersion	Sync	Get HAL library version

Table 1 HAL Layer Public APIs

3.0 HAL Layer AFE Device Driver APIs

3.1 HAL_afeInit

This API initializes the AFE device driver for TX and RX path. It initializes the PWM, Timers, ADC, DMAs (ADC and PWM), and SPI/McBSP (for PGA control). This API includes HAL_afe031Init (if AFE031 is used), HAL_afeRxInit, and HAL_afeTxInit.

Syntax `HAL_status_t HAL_afeInit(HAL_afe_prfParms_t *setParms_p)`

Parameter	Description

setParms_p	Pointer to profile parameters.
	typedef struct
	{
	  UINT16 rx_fs_kHz; // RX sampling frequency in kHz
	  UINT16 tx_fs_kHz; // TX sampling frequency in kHz
	  UINT16 tx_pwm_kHz; // PWM frequency in kHz
	}HAL_afe_prfParms_t;
	
	The valid TX/RX sampling frequencies are: 250, 400, 500 (kHz);
	The valid PWM frequencies are: 800, 1000, 2000 (kHz).

Return Values	HAL_status_t: HAL_STAT_SUCCESS HAL_STAT_FAILURE

3.2 HAL_afeRxInit

This API initializes the AFE device driver for RX path. It initializes the ADC, DMA (ADC to buffer), and SPI (for PGA control).

Syntax `HAL_status_t HAL_afeRxInit(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCESS HAL_STAT_FAILURE

3.3 HAL_afeTxInit

This API initializes the AFE device driver for TX path. It initializes PWM, does HRPWM scale factor optimization (SFO), and setup CPU timer 0.

Syntax `HAL_status_t HAL_afeTxInit(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCESS HAL_STAT_FAILURE

3.4 HAL_afeSet

AFE device driver main set function. It does the control for all AFE modules.

Syntax `HAL_status_t HAL_afeSet(HAL_afe_setCode_t setCode, void *setParms_p)`

Parameter	Description

setCode	Control set code.
	Typedef enum
	{
	AFE_TX_CFG = 0, // TX configure
	AFE_TX_START = 1, // TX start
	AFE_TX_STOP = 2, // TX stop
	AFE_TX_RECFG = 3, // Reconfigure TX
	AFE_TX_DMASIZE = 4, // Set TX DMA size
	AFE_TX_DMAADDR = 5, // Set TX DMA address
	AFE_TX_T0CFG = 6, // Configure CPU timer 0
	AFE_TX_T0START = 7, // Start CPU timer 0
	AFE_TX_T0STOP = 8, // Stop CPU timer 0
	
	AFE_RX_START = 9, // RX start
	AFE_RX_STOP = 10, // RX stop
	AFE_RX_RECFG = 11, // RX reconfigure
	AFE_RX_SETGAIN = 12, // Set AGC gain
	AFE_RX_UPDATEGAIN = 13, // Update AGC gain
	AFE_RX_CANCEL_GAIN = 14, // Cancel update gain
	
	AFE_SEL_ECAP = 15, // ECAP select
	AFE_SEL_XINT1 = 16, // External interrupt XINT1 select
	AFE_SEL_XINT2 = 17, // External interrupt XINT2 select
	
	NUM_AFE_SETPARMS = 18
	
	}HAL_afe_setCode_t;

setParms_p	Pointer to set parameters.
	RX Set Parameters:
	Typedef struct
	{
	Uint16 size[2];
	Uint16 *dstBufAddr[2];
	Uint16 dstBufIdx;
	}HAL_afe_rxSetParms_t;
	
	TX Set Parameters:
	typedef struct
	{
	int16 *txBuf_p; // Pointer to the TX buffer
	Uint16 txSize; // Size of TX buffer
	Uint32 t0PrdInUs; // CPU timre0 period (in us)
	Uint16 t0Flags; // CPU timer0 flag.
	// bit 0 : CPU timer0 interrupt enable (1) or disable (0) ;

	// bit 0 : CPU timer0 operation: one shot (1) or continuous (0)
	HAL_cbFunc_t cb_p; // Pointer to callback function when CPU timer0 timeout
	UINT32 cb_param; // Callback parameter
	}HAL_afe_txSetParms_t;
	
	Callback when CPU timer0 timeout:
	typedef void (*HAL_cbFunc_t)(UINT32);
	
	ECAP Selection:
	typedef enum
	{
	HAL_AFE_ECAP1 = 1, // Select ECAP1 channel (Capture interrupt API supported)
	HAL_AFE_ECAP2 = 2, // Select ECAP2 channel (Capture interrupt API **NOT** supported)
	HAL_AFE_ECAP3 = 3, // Select ECAP3 channel (Capture interrupt API **NOT** supported
	HAL_AFE_ECAP4 = 4, // Select ECAP4 channel (Capture interrupt API supported)
	HAL_AFE_ECAP5 = 5, // Select ECAP5 channel (Capture interrupt API supported)
	HAL_AFE_ECAP6 = 6, // Select ECAP6 channel (Capture interrupt API **NOT** supported
	}HAL_afe_ecapSel_t;
	
	XINT1/XINT2 Selection:
	*setParms_p: pointer to the variable (x) specifying which GPIO pin is selected as XINT1/XINT2 interrupt input.
	x = 0-31: select GPIOx as XINT1 or XINT2 interrupt input.

Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.5 HAL_afeGet

This API gets AFE parameters and status.

Syntax `HAL_status_t HAL_afeGet(HAL_afe_getCode_t setCode, void *getParms_p)`

Parameter	Description
getCode	Control get code. 0x0000: AFE_RX_DMATIME: RX DMA time 0x0001: AFE _ZCTIME: Zero Crossing (ZC) time 0x0002-0xFFFF: reserved

getParms_p	Pointer to get parameters .
	
	AFE_RX_DMATIME: pointer to 32-bit DMA time (in 10's us). Needs 32-bit aligned.
	
	AFE_ZCTIME:
	typedef struct
	{
	UINT32 zcaTime; // ZC time from ECAP1 or ECAP3
	UINT32 zcbTime; // ZC time from ECAP4
	}HAL_afe_zcTime_t;

Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.6 HAL_afeDacCnv

This API converts the digital data into DAC format that in ready for the SPI DAC (12-bit) or the HRPWM (32-bit). It also scales the TX output level and clips the TX signal to control the PAPR of the OFDM signal. This API is applicable to G3 only; for PRIME the DAC conversion is combined in a PHY upsampling FIR.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_afeDacCnv(int16 *in_p, Uint16 len, int16 lev, Uint16 clip, int16 *out_p))`

Parameters	Description
in_p	Pointer to input data array
len	Length of the data array
lev	TX output level (Q14)
clip	TX output clip level (Q15)
out_p	Output DAC data array (ready for SPI DAC or HRPWM)
Return Values	HAL_status_t: HAL_STAT_SUCCESS HAL_STAT_FAILURE

3.7 HAL_afeRxDmaCh1IntFunc

This API processes interrupt for RX DMA channel 1. This function is to be called in the DMA channel 1 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_afeRxDmaCh1IntFunc(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.8 HAL_afeTxDmaCh2IntFunc

This API processes interrupt for TX DMA channel 2. This function is to be called in the DMA channel 2 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_afeTxDmaCh2IntFunc(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.9 HAL_cpuTint0Func

This API processes interrupt for CPU timer 0. This function is to be called in the CPU timer 0 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_cpuTint0Func (void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.10 HAL_afeEcap1IntFunc

This API processes interrupt for enhanced capture (ECAP) channel 1. It records the zero-crossing time from the PHY timer for phase 1. This function is to be called in the ECAP1 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_afeEcap1IntFunc (void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.11 HAL_afeEcap3IntFunc

This API processes interrupt for enhanced capture (ECAP) channel 3. It records the zero-crossing time from the PHY timer for phase 1. This function is to be called in the ECAP3 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_afeEcap3IntFunc (void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.12 HAL_afeEcap4IntFunc

This API processes interrupt for enhanced capture (ECAP) channel 4. It records the zero-crossing time from the PHY timer for phase 2. This function is to be called in the ECAP4 interrupt service routine.
This API does not apply to the PLC Lite.

yntax `HAL_status_t HAL_afeEcap4IntFunc (void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.13HAL_haltLPM

This function puts the device in low power HALT mode.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_haltLPM(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.14HAL_xint1Func

This API processes interrupt for external interrupt 1. It will put the device in HALT mode. This function is to be called in the XINT1 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_xint1HaltFunc(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.15HAL_xint2Func

This API processes interrupt for external interrupt 2. This function is to be called in the XINT2 interrupt service routine.
This API does not apply to the PLC Lite.

Syntax `HAL_status_t HAL_xint2FlagFunc(void)`

Parameter	Description
Return Values	HAL_status_t: HAL_STAT_SUCCES HAL_STAT_FAILURE

3.16 HAL_getLibVersion

This API gets HAL library version string.
This API does not apply to the PLC Lite.

Syntax `const char *HAL_getLibVersion(void)`

Parameter	Description
Return Value	Pointer to the HAL library version string. const char HAL_LIB_VERSION[16]

3.17 PLC Lite ISR configurations

For PLC Lite ADC and Timer0 ISR installations, please refer to “TI_FlexLite_UG_v1.0” and PHY examples.

4.0 References

[1] “FSM SDS”
[2] “PLC G3 Physical Layer Specification”
[3] “PRIME – Specification”