AMD

Wireless LAN DSSS PC Card Reference Design

Application Note

Wireless LAN DSSS PC Card Reference Design

Application Note

This application note describes the connection of the PCnet™-Mobile Wireless LAN Media Access Controller (Am79C930) to the HARRIS Semiconductor Direct Sequence Spread Spectrum (DSSS) PRISM™ chipset.

INTRODUCTION

The PCnet-Mobile Wireless LAN Media Access Controller (Am79C930) functionality has been defined to allow for connection to various PHY types. This application note describes one possible connection of the Am79C930 device to an IEEE 802.11 Direct Sequence Spread Spectrum (DSSS) PHY. The 802.11-compliant DSSS PHY used in this application example is the HARRIS Semiconductor PRISM chipset, consisting of the HARRIS HFA3524, HFA3624, HFA3724, HSP3824, and HFA3925 devices. The complete solution is connected to a PC CARD bus connector and is intended to be built onto a TYPE II PC Card (formerly PCMCIA Type II card). A few additional supporting devices are required in order to complete the system, e.g., Am29F010 flash memory and 32K SRAM device.

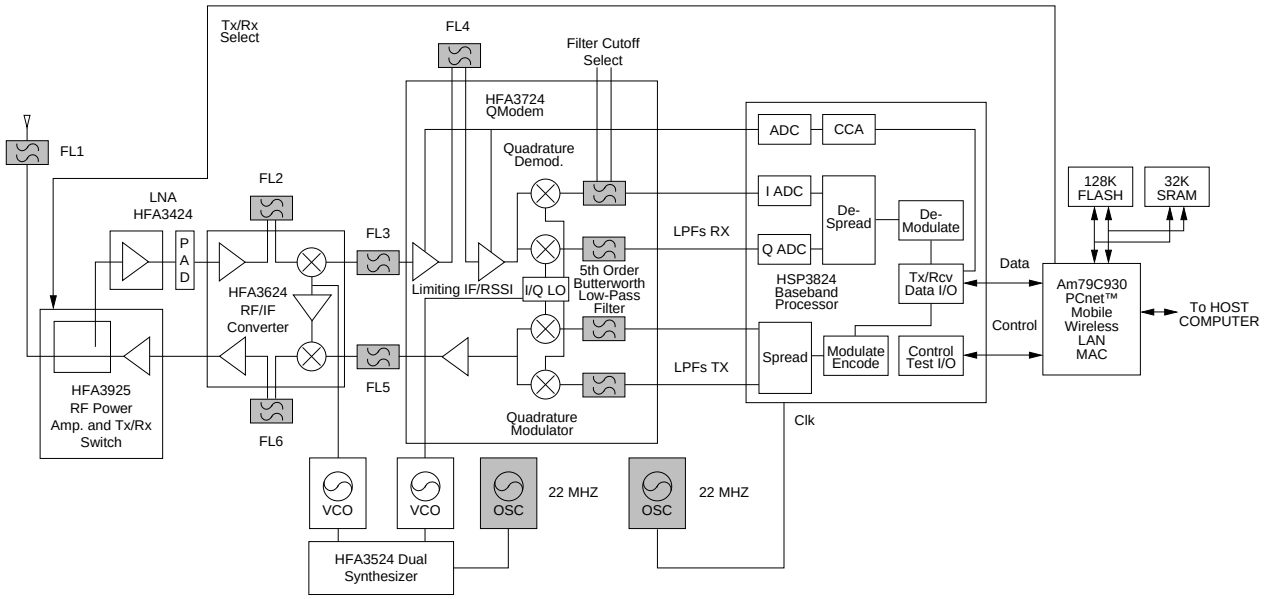
SYSTEM BLOCK DIAGRAM

A complete 802.11 PC CARD TYPE II PC Card includes the following functional blocks:

- Am79C930 MAC device
- Flash memory
- SRAM memory
- Oscillators
- PRISM chipset and antenna(s)
- PC CARD connector

These devices need to be interconnected as shown in the block diagram below.

BLOCK DIAGRAM



20575B-1

Publication# **20575** Rev: **B** Amendment/**0**
Issue Date: **April 1997**

TABLE OF CONTENTS

Parts List

The following is a list of the major components for this application:

Part Number	Description	Manufacturer	Quantity
Am79C930	WLAN MAC controller	AMD	1
HFA3624	2.4 GHz RF to IF Converter	HARRIS Semiconductor	1
HFA3724	400 MHz Quadrature IF Modulator/ Demodulator	HARRIS Semiconductor	1
HSP3824	Direct Sequence Spread Spectrum Baseband Processor	HARRIS Semiconductor	1
HFA3424	Low Noise Amplifier	HARRIS Semiconductor	1
HFA3925	2.4 GHz Power Amplifier	HARRIS Semiconductor	1
HFA3524	Dual frequency synthesizer	HARRIS Semiconductor	1
Am29F010-90EC or Am29F010-90FC or Am29F010-45EC or Am29F010-45FC E and F give normal or reverse pinout choice	128Kx8 5-V only flash memory	AMD	1
TC55257BFTL-10 or TC55257BTRL-10 Choice is normal or reverse pinout	32K SRAM - 100 ns	Toshiba	1
LFJ30-03B2442BA84	-3dB/50 ohm BPF 2450 MHz	muRata	2

Substitution List

The following list indicates acceptable device substitutions:

Recommended Part Number	Substitute Part Number	Manufacturer
Am79C930	None	AMD
HFA3624	None	HARRIS Semiconductor
HFA3724	None	HARRIS Semiconductor
HSP3824	None	HARRIS Semiconductor
HFA3925	None	HARRIS Semiconductor
Am29F010	None	AMD
TC55257BFTL-10 or TC55257BTRL-10 Choice is normal or reverse pinout		Toshiba
	LH52B256T-10LL or LH52B256TR-10LL	Sharp
	KM681000B-55T KM681000B-55R KM681000B-10T KM681000B-10R Note that these are 1M devices (128Kx8), because smaller devices are not availabe in TSOP from Samsung	Samsung

PIN CONNECTIONS

The pin connections among the Am79C930 device, the HARRIS PRISM™ chipset, Flash memory, SRAM and oscillator are given in Table 1.

Note: *The pin connections for a PC CARD design are different from the pin connections that would be used for an ISA Plug and Play design. This application note covers only the PC CARD case.*

Table 1. Am79C930 Pin Connections

AMD Am79C930 Pin Name	Pin No.	I/O	AMD Am 79C930 Pin Function	802.11 HARRIS DS PHY PC CARD System Connections	I/O	802.11 HARRIS DS PHY PC CARD System Function
USER2	1	I/O	API will determine signal timing	NC	I	
USER3	2	O	API will determine signal timing	Radio to Power Enable		Radio Power Enable
USER4	3	O	API will determine signal timing	HFA3524 to SYNTH.LE (Pin 13)	I	Synthesizer serial bus latch enable
VDDM	4, 16, 29	I	5-V supply (VCC)	5-V supply		
$\overline{XCE}$	5	O	Chip enable for extra peripheral on memory bus	NC		
MA[16:0]		O	Address bus for memory devices	MA[16:0] to FLASH.A[16:0]	I	Address bus for FLASH device
				MA[14:0] to SRAM.A[14:0]	I	Address bus for SRAM device
VSSM	7, 21, 32	I	GND	GND		
$\overline{MWE}$	11	O	Write enable for memory devices	FLASH.$\overline{WE}$	I	Write enable for FLASH device
				SRAM.$\overline{WE}$	I	Write enable for SRAM device
VCC	17, 89	I	3.5-V supply (VCC)	3.5-V supply		
MD[7:0]		I/O	Data bus for memory devices	FLASH.D[7:0]	I/O	Data bus for FLASH device
				SRAM.D[7:0]	I/O	Data bus for SRAM device
$\overline{MOE}$	38	O	Output enable for memory devices	FLASH.$\overline{OE}$	I	Output enable for FLASH device
				$\overline{SRAM.OE}$	I	Output enable for SRAM device
$\overline{SCE}$	39	O	Chip enable for SRAM device	$\overline{SRAM.CE}$	I	Chip enable for SRAM device
$\overline{FCE}$	40	O	Chip enable for FLASH device	$\overline{FLASH.CE}$	I	Chip enable for FLASH device
D[7:0]		I/O	Data bus for PC CARD	PC CARD.D[7:0]	I/O	Data bus for PC CARD
VSSP	44, 76	I	GND	GND		
$\overline{STSCHG}$	45	O	PC CARD $\overline{STSCHG}$ function	NC (PC CARD optional - not used in this implementation)		

Table 1. Am79C930 Pin Connections

AMD Am79C930 Pin Name	Pin No.	I/O	AMD Am79C930 Pin Function	802.11 HARRIS DS PHY PC CARD System Connections	I/O	802.11 HARRIS DS PHY PC CARD System Function
A[14:0]		I	Address bus for PC CARD	PC CARD.A[14:0]	O	Address bus for PC CARD
$\overline{REG}$	48	I	PC CARD signal	PC CARD.$\overline{REG}$ (Pin 61)	O	PC CARD signal
$\overline{INPACK}$	50	O	PC CARD signal	PC CARD.$\overline{INPACK}$ (Pin 60)	O	PC CARD signal
$\overline{WAIT}$	52	O	PC CARD signal	PC CARD.$\overline{WAIT}$ (Pin 59)	O	PC CARD signal
VDDP	55	I	5-V supply (VCC)	5-V supply		
VSS	57	I	GND	GND		
RESET	58	I	PC CARD reset	PC CARD.RESET (Pin 58)	O	PC CARD reset
$\overline{IREQ}$	61	O	PC CARD signal	PC CARD.$\overline{IREQ}$ (Pin 16)	I	PC CARD signal
$\overline{WE}$	62	I	PC CARD signal	PC CARD.$\overline{WE}$ (Pin 15)	O	PC CARD signal
$\overline{IOWR}$	66	I	PC CARD signal	PC CARD.$\overline{IOWR}$ (Pin 45)	O	PC CARD signal
$\overline{IORD}$	67	I	PC CARD signal	PC CARD.$\overline{IORD}$ (Pin 44)	O	PC CARD signal
$\overline{OE}$	70	I	PC CARD signal	PC CARD.$\overline{OE}$ (Pin 9)	O	PC CARD signal
$\overline{CE1}$	72	I	PC CARD signal	PC CARD.$\overline{CE1}$ (Pin 7)	O	PC CARD signal
VSSP	44, 76	I	GND	GND	I	Ground
PC CARD	79	I	Mode selection pin	5 V supply	I	Power
CLK20	80	I	Digital Clock	40 MHz oscillator input	O	Clock source
$\overline{TEST}$	81	I	Test mode select pin	5 V supply		
PMX2	82	X*				
PMX1	83	I	Clock input for sleep function	32.768 kHz crystal with 20 Ω shunt and 10 pF loads		
TCK	84	I	JTAG test pin	VCC		3.5-V supply
TDO	85	O	JTAG test pin	NC		
TMS	86	I	JTAG test pin	NC		
TRST	87	I	JTAG test pin	PCCARD. RESET (Pin 58)		
TDI	88	I	JTAG test pin	VCC		3.5-V supply
USER0	90	O	API will determine signal timing	NC		
USER1	91	O	API will determine signal timing	NC		
USER7	92	I	API will determine signal timing	VCO startup circuit	1	VCO enable circuit
VSSU1	93	O	GND	GND		
RXC	94	I	RX clock output for test purposes	NC		
USER6/ EXTSDF	95	I	Used to start RX state machine	HSP3824.MD_RDY (Pin 34)	O	Asserted on last bit of Unique word (UW)
USER5/ EXTCHBSY	96	I	Accepts external channel busy indication	HSP3824.CCA (Pin 32)	O	Gives channel busy or idle indication
VDDU1	97	I	3.5-V supply (VCC)	3.5-V supply		

** The symbol X in the direction field (I/O column) denotes that this pin is the "output" side of a crystal oscillator amplifier.*

Table 1. Am79C930 Pin Connections

AMD Am79C930 Pin Name	Pin No.	I/O	AMD Am79C930 Pin Function	802.11 HARRIS DS PHY PC CARD System Connections	I/O	802.11 HARRIS DS PHY PC CARD System Function
$\overline{ACT}$	98	O	LED output, controlled by firmware	Yellow LED		Indicates currrent RX or TX activity
VSST	99	I	GND	GND		
$\overline{LNK}$	100	O	LED output, controlled by firmware	Green LED	I	Indicates a current association with a BSS
SDCLK	101	O	Note that this pin is shared by both serial busses (i.e., synthesizer chip serial bus and baseband chip serial bus)	HFA3524.SYNTH_CLK (Pin 11)	I	Synthesizer serial bus clock input
				HSP3824.SCLK	I	Control register serial bus clock input
SDDATA	102	I/O	Note that this pin is shared by both serial busses (i.e., synthesizer chip serial bus and baseband chip serial bus)	HFA3524.SYNTH_DATA (Pin 12)	I	Synthesizer serial bus data pin
				HSP3824.SDATA (Pin 25)	I/O	Control register serial bus clock input
$\overline{SDSEL3}$	103	O	API will determine signal timing	$\overline{HSP3824.RW}$ (Pin 8)	I	Control register serial bus read-write select
VDDT	104, 125	I	3.5-V supply	3.5-V supply		
$\overline{SDSEL2}$	105	O	API will determine signal timing	HSP3824.AS (Pin 23)	I	Selects between address and data registers in the HSP3824
$\overline{SDSEL1}$	107	O	API will determine signal timing	HSP3824.CS (Pin 9)	I	Selects the HSP3824 device for register access
SAR[6:0]		O	A/D converter output	NC		
TXC	115	I	TX clock - controls delivery of TXDATA	HSP3824.TXCLK (Pin 4)	O	TX clock - controls delivery of TXDATA
LFCLK	117	O	Buffered clock output	NC		
$\overline{LFPE}$	118	O	Enables LFCLK	HSP3824.RESET (Pin 28)	I	Reset input for HSP3824 device
HFCLK	119	O	Buffered clock output	NC		
$\overline{HFPE}$	120	O	Enables HFCLK	NC		
TXDATA	121	O	TX output data	HSP3824.TXD (Pin 3)	I	TX input data
$\overline{RXPE}$	122	O	API will create timing	HSP3824.RX_PE (Pin 33)	I	Baseband processor RX enable input.
RXDATA	123	I	RX input data	HSP3824.RXD (Pin 35)	O	RX output data
RXCIN	124	I	RX clock input for incoming RX frames	HSP3824.RXCLK (Pin 36)	O	RX clock decoded from channel activity

Table 1. Am79C930 Pin Connections

AMD Am79C930 Pin Name	Pin No.	I/O	AMD Am79C930 Pin Function	802.11 HARRIS DS PHY PC CARD System Connections	I/O	802.11 HARRIS DS PHY PC CARD System Function
$\overline{\text{TXCMD}}$	126	O	Signals beginning of Am79C930 transmission sequence	HFA3724.LPFRXPE (Pin 21) HFA3724.MODRXPE (Pin 43) HFA3724.LIM1PE (Pin 74) HFA3724.LIM2PE Pin 54) HFA3624.RXPE (Pin 28)	I	IF/RF modulator/ demodulator RX enable inputs, IF/RF converter RX enable input
$\overline{\text{FDET}}$	128	O	Indicates that UW has been located	NC		
$\overline{\text{TXPE}}$	129	O	Timing is generated by state machine per API programming	NC		
$\overline{\text{TXMOD}}$	131	O	Begins second stage of Am79C930 transmission power ramp sequence	HSP3824.TX_PE, HFA3925.PIN11 (Pin 11) HFA3925.PIN18 (Pin 18), HFA3925.PIN23 (Pin 23)	I	Standby mode for TX circuitry - also initiates TX state machine and freezes antenna selection for transmission, and enables power to transmitter
ANTSLT	132	O	Effectively a DC level set by an API module	HFA3724.LPFSEL0 (Pin 17)	I	Filter select control
PWRDWN	133	O		NC		
ADIN1	134	I		Ground		
ADIN2	135	I		Ground		
ADREF	137	I		Ground		
AVDD	138	I	5-V supply (VCC)	5-V supply		
VDD5	139	I	5-V supply (VCC)	5-V supply		
VDDU2	140	I	3.5-V supply (VCC)	3.5-V supply		
$\overline{\text{ANTSLT}}$	141	O	Effectively a DC level set by an API module	HFA3724.LPFSEL1 (Pin 16)	I	Filter select control
TXCMD	142	O	Timing is generated by state machine per API programming	HFA3624.TXPE (Pin 15) HFA3724.LPFTXPE (Pin 22) HFA3724.MODTXPE (Pin 41)	I	TX enables for RF/IF converter and modulator/ demodulator
$\overline{\text{TXDATA}}$	143	O	Inverted TX data output	NC		
LLOCKE	144	O	API will determine signal timing	NC		

Am79C930 Operational Modes

The Am79C930 device offers a very flexible MAC/PHY interface, just as the HARRIS PRISM chip chipset does. There are mode options for many of the functions of the Am79C930 device. The following sections describe the mode options that must be invoked in order for the Am79C930 device to interoperate with the HARRIS PRISM chipset.

The operational modes of the Am79C930 device are set by modifying control bits in the device's register sets, including bits contained in the MIR, TIR, and TCR register sets. MIR registers are only visible to the 80188 embedded core; TIR and TCR registers are accessible by both the 80188 core and the host system. However, since the MAC firmware must use TIR and TCR registers in order to perform the required MAC protocol operations, and since it is most convenient and straightforward to keep all TIR and TCR operations within a single piece of code (i.e., the MAC firmware), it is recommended that only the MAC firmware (*not* the driver software) modify TIR and TCR settings to accommodate the needs of a particular PHY implementation.

An API has been defined and is described in another section of this application note, which allows the user to create the appropriate calls required in order to allow the MAC firmware to set up the proper configuration for any particular Am79C930 application. Other API calls are needed in order to translate low-level radio instructions into the particular signalling that is required for the HARRIS PRISM chipset. A complete description of API functionality can be found in a later section.

The remainder of this section describes the bit locations that modify the operational modes of the Am79C930 device and describes the mode affected by each bit. The material contained within this section is intended to serve as a guide in generating the proper code for each of the API calls that must be written in order to allow the Am79C930 device and its MAC firmware to communicate with the HARRIS PRISM chipset. In particular, the API calls that will be affected by the descriptions in this section are those calls that initialize the configuration of the Am79C930 device and those that initialize the configuration of the HARRIS PRISM chipset.

A summarized version of the register bit locations and the required register settings for this application are given in Table 2.

Table 2. HSP3824 Register Setting Summary

Register Number	Register Description	Type	Address	Required Setting
CR0	Modem configuration register A	R/W	00h	1Ch
CR1	Modem configuration register B	R/W	04h	02h
CR2	Modem configuration register C	R/W	08h	27h
CR3	Modem configuration register D	R/W	0Ch	00h
CR4	Internal test register A	R/W	10h	00h
CR5	Internal test register B	R/W	14h	40h
CR6	Internal test register C	R	18h	NA
CR7	Modem status register A	R	1Ch	NA
CR8	Modem status register B	R	20h	NA
CR9	I/O definition register	R/W	24h	00h
CR10	RSSI status register	R	28h	NA
CR11	A/D Calibration positive	R/W	2Ch	01h
CR12	A/D calibration negative	R/W	30h	FDh
CR13	TX spread code(high)	R/W	34h	05h
CR14	TX spread code (low)	R/W	38h	B8h
CR15	Scramble seed	R/W	3Ch	70h
CR16	Scramble tap	R/W	40h	48h
CR17	CCA timer threshold	R/W	44h	2Ch
CR18	CCA cycle threshold	R/W	48h	03h
CR19	RSSI threshold	R/W	4Ch	3Fh
CR20	RX de-spread code (high)	R/W	50h	05h
CR21	RX de-spread code (low)	R/W	54h	B8h
CR22	RX bit sync signal quality for acquisition threshold (high)	R/W	58h	02h
CR23	RX bit sync signal quality for acquisition threshold (low)	R/W	5Ch	10h
CR24	RX bit sync signal quality for acquisition (high)	R	60h	NA
CR25	RX bit sync signal quality for acquisition (low)	R	64h	NA
CR26	RX bit sync signal quality for data threshold (high)	R/W	68h	0Fh
CR27	RX bit sync signal quality for data threshold (low)	R/W	6Ch	FFh
CR28	RX bit sync signal quality for data (high)	R	70h	NA
CR29	RX bit sync signal quality for data (low)	R	74h	NA
CR30	RX frequency error signal quality for acquisition threshold (high)	R/W	78h	00
CR31	RX frequency error signal quality for acquisition threshold (low)	R/W	7Ch	90h
CR32	RX frequency error signal quality for acquisition (high)	R	80h	NA
CR33	RX frequency error signal quality for acquisition (low)	R	84h	NA
CR34	RX frequency error signal quality for data threshold (high)	R/W	88h	09h
CR35	RX frequency error signal quality for data threshold (low)	R/W	8Ch	80h
CR36	RX frequency error signal quality for data (high)	R	90h	NA
CR37	RX frequency error signal quality for data (low)	R	94h	NA
CR38	RX bit sync signal quality for 802.11	R	98h	NA
CR39	Reserved	R/W	9Ch	NA
CR40	Reserved	R/W	A0h	NA
CR41	Unique word search timeout length	R/W	A4h	90h
CR42	DBPSK modulation type field value	R/W	A8h	0Ah
CR43	DQPSK modulation type field value	R/W	ACh	14h
CR44	RX service field of 802.11	R	B0h	NA
CR45	RX MPDU length field (high)	R	B4h	NA
CR46	RX MPDU length field (low)	R	B8h	NA
CR47	RX PLCP CRC16 field (high)	R	BCh	NA
CR48	RX PLCP CRC16 field (low)	R	C0h	NA

Table 2. HSP3824 Register Setting Summary

Register Number	Register Description	Type	Address	Required Setting
CR49	Unique Word (high)	R/W	C4h	F3h
CR50	Unique Word (low)	R/W	C8h	A0h
CR51	TX service field of 802.11	R/W	CCh	00h
CR52	TX MPDU length field (high)	R/W	D0h	NA
CR53	TX MPDU length field (low)	R/W	D4h	NA
CR54	TX PLCP CRC16 field (high)	R	D8h	NA
CR55	TX PLCP CRC16 field (low)	R	DCh	NA
CR56	TX preamble length	R/W	E0h	80h

Note: *The basic operational mode (PC CARD system interface bus mode) is selected with a pin-strapping option through the PCMCIA pin. For this Am79C930/ HARRIS PRISM PC CARD application, the PCMCIA pin should be connected to VCC.*

MIR, TIR, and TCR bit locations not mentioned in this section are controlled by the MAC firmware that is shipped with the Am79C930 device. These bit locations do not need to be altered by API code, hence, they are excluded from this section.

Am79C930 Register Settings

MIR8[1:0] FLASHWAIT[1:0]

The FLASHWAIT bits of MIR8[1:0] should be set to a value that is appropriate for the FLASH being used in the design. With a 40-MHz signal at the CLKIN input, the FLASHWAIT bits are interpreted as follows in Table 3.

Note: *Each wait state on the asynchronous Am79C930 memory interface bus is two CLKIN periods in length (i.e., at CLKIN = 40 MHz, then two CLKIN periods = 2*25 ns = 50 ns.)*

Table 3. FLASHWAIT Bits

FLASHWAIT[1:0] Programmed Value	Number Of Wait States Created On Am79c930 Memory Interface Bus	Maximum Guaranteed tmAA Access Time Allowed For Flash Device
11 (RESET default)	3	205 ns
10	2	155 ns
01	1	105 ns
00	0	55 ns

MIR9[5:4] SRAMWAIT[1:0]

The SRAMWAIT bits of MIR9[5:4] should be set to a value that is appropriate for the SRAM being used in the design. With a 40-MHz signal at the CLKIN input, the SRAMWAIT bits are interpreted as follows in Table 4.

Table 4. SRAMWAIT Bits

SRAMWAIT[1:0] Programmed Value	Number Of Wait States Created On Am79c930 Memory Interface Bus	Maximum Guaranteed Access tmAA Time Allowed For SRAM Device
11 (RESET default)	3	205 ns
10	2	155 ns
01	1	105 ns
00	0	55 ns

Note: *Each wait state on the asynchronous Am79C930 memory interface bus is two CLKIN periods in length (i.e. at CLKIN = 40 MHz, then two CLKIN periods = 2 x 25 ns = 50 ns.)*

Additional memory parameter restrictions are as follows with CLKIN = 40 MHz.

Table 5. Memory Parameter Restrictions with CLKIN = 40 MHz

Memory Speed	WAIT States
t_{ACC} 55 ns MAX, t_{CE} 55 ns MAX, t_{OE} 30 ns MAX	0
t_{ACC} 105 ns MAX, t_{CE} 105 ns MAX, t_{OE} 80 ns MAX	1
t_{ACC} 155 ns MAX, t_{CE} 155 ns MAX, t_{OE} 130 ns MAX	2

TIR2[2] SDC

TIR2[7:0] should be set to 40h.

The SDC bit of TIR2 determines the polarity of the clock pulse that will appear on the SDCLK output pin of the Am79C930 device. A setting of 0 yields a positive going pulse when the TIR3 register is used for serial communications. A setting of 1 yields a negative going pulse when the TIR3 register is used for serial communications. For RADIO API calls that access the HSP3824, it is required that the SDC bit of TIR2 be set to 1, in order to provide the necessary setup and hold time of the serial data with respect to the serial clock from the Am79C930 device. For RADIO API calls that access other devices within the PRISM subsystem (such as the HFA3524 dual synthesizer), it is required that the SDC bit of TIR2 be set to 0 in order to provide a positive clock for the serial data transfer. The switching of the SDC bit of TIR2 from 1 to 0, and vice versa, should occur within the respective RADIO API calls.

TIR3[7:0] Fast Serial Port

The Fast Serial Port is used for serial communications with the HSP3824 baseband processor and the HFA3524 frequency synthesizer devices. The serial data transferred through this means is used to program the initial operating state of the HSP3824 device and is used to program the operating frequencies of the synthesizers in order to tune the radio to the proper channel. Channel switching is required for periodic scanning operations in order to locate access points for potential connections, for changing associations from one access point to another and for creating *ad hoc* network connections. Channel switching is generally automatically performed as part of the MAC management function and, therefore, is generally transparent to the user. However, the specific signalling sequence required to

program these devices must be coded into an API in order to allow the MAC code to be independent of PHY implementation. Therefore, system integrators will write API code that accesses TIR3 for the purpose of executing a channel reprogramming operation. This API routine will be called at the appropriate time and with the appropriate arguments by the MAC code. The HSP3824 and frequency synthesizer programming operations are performed through the TIR3 fast serial port.

There are two other modes for operating the serial communications link with the HSP3824 and frequency synthesizer devices. Both of these modes are controlled with the bits in the TIR2 register and are slower than the mode that uses the fast serial port (TIR3 register). Therefore, the TIR3 method of serial transfer has been chosen. This minimizes code execution time for API modules that include serial communications between the Am79C930 and the PHY-layer devices.

Note: *The proper use of the TIR3 register is found in the Am79C930 data sheet. The description in the data sheet should be used when writing API modules that require serial communications with the HARRIS devices.*

The HARRIS PRISM chip requires the following timing to appear on the serial port lines during read and write operations (see Figures 1 and 2, respectively).

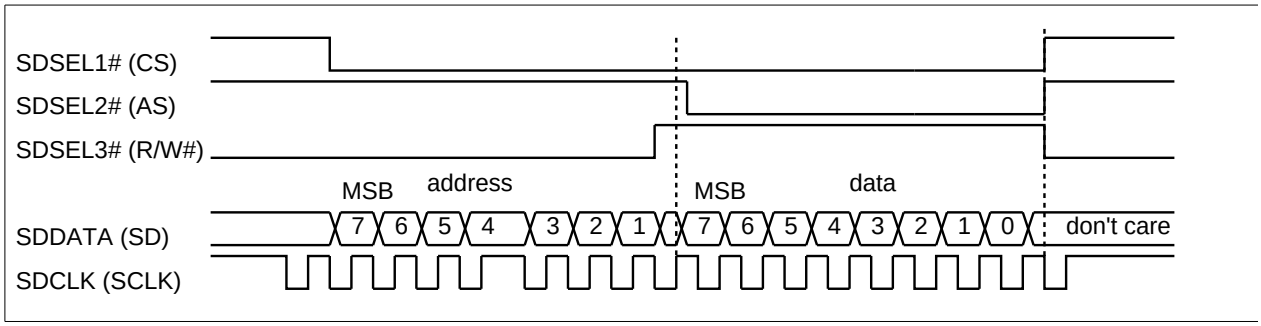


Figure 1. HSP3824 Serial Port Timing Diagram (Read Operation)

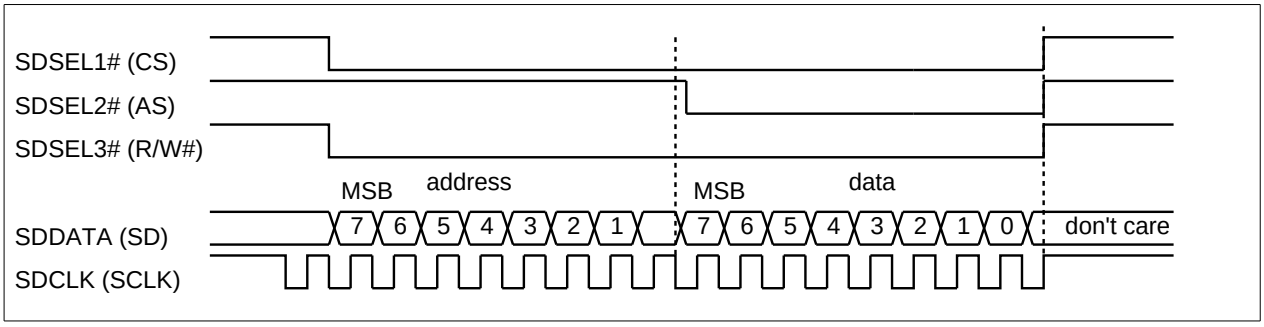


Figure 2. HSP3824 Serial Port Timing Diagram (Write Operation)

The HFA3524 synthesizer chip requires the following timing to appear on the serial port lines:

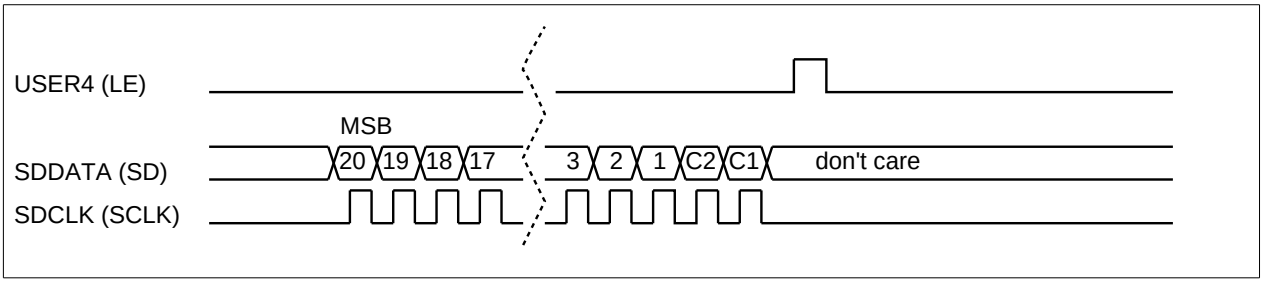


Figure 3. HFA3524 Serial Port Timing Diagram

TIR26[3] ANTSEN

The ANTSEN bit of TIR26 should be set to a 1 at all times.

The ANTSEN bit is used to determine the nature of the signalling that will appear on the ANTSLT and $\overline{\text{ANTSLT}}$ pins of the Am79C930 device. When ANTSEN is reset to a 0, then the timing of the signals named above will be determined by the ANT_SEL state machine in the TAI portion of the Am79C930 device. The setting of ANTSEN = ZERO should be used whenever the antenna selection will be performed by the Am79C930 device. A setting of ANTSEN = ONE is used when the ANTSLT and $\overline{\text{ANTSLT}}$ pins are to be used for any other purpose in the system. A setting of ANTSEN = ONE disables the Am79C930 internal antenna selection logic.

Note: In the application described herein, the HARRIS PRISM chipset will perform the antenna selection function and, hence, a setting of ANTSEN = ONE should be used.

When ANTSEN is set to a 1, then the timing of the signals appearing on the ANTSLT and $\overline{\text{ANTSLT}}$ pins will be determined by firmware manipulation of the ANTSLTD bit (TIR26[4]) and the ANTSLTLD bit (TCR7[1]). For the application described in this application note, antenna selection will be performed by the HSP3824 and, hence, the Am79C930 device ANTSLT and $\overline{\text{ANTSLT}}$ pins will be available for other use. In the HARRIS PRISM PC CARD application, ANTSLT and $\overline{\text{ANTSLT}}$ pins will be used to control the filter selection in the HFA3724 device. The Am79C930 firmware will create the appropriate signalling through register bit manipulation that will occur as part of the RADIO API call “initialize()”.

TCR7[7:0] User Pins Control

TCR7[7:0] should be set to 02h.

TCR7 controls several functions.

TCR7[7] controls the CTS function of the USER1 pin. The CTS function is not used for the HARRIS PRISM PC CARD application, and, therefore, TCR7[7] must be reset to a 0 in order to disable the function.

TCR7[6] controls the function of the USER6 pin when the Am79C930 device is operating in ISA PnP mode. PC CARD mode is used for the HARRIS PRISM PC CARD application and, therefore, TCR7[6] must be reset to a 0.

TCR7[5] controls the function of the USER5 pin when the Am79C930 device is operating in ISA PnP mode. PC CARD mode is used for the HARRIS PRISM PC CARD application and, therefore, TCR7[5] must be reset to a 0.

TCR7[4:3] controls the active edge for generation of interrupts from the USER1/INT188 input to the 80188 core. The USER1 INT188 function is not used for the HARRIS PRISM PC CARD application and, therefore, TCR7[4:3] must be reset to ZERO-ZERO binary.

TCR7[2] affects the output value at the TXCMD pin. The TXCMD function is not used for the HARRIS PRISM PC CARD application and, therefore, TCR7[2] must be reset to a 0.

TCR7[1] affects the output value at the $\overline{\text{ANTSLT}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{ANTSLT}}$ output value will be controlled by the RADIO API call “initialize().” The initial value for TCR7[1] must be a 1, since this signal will control the characteristics of a tunable filter on the HARRIS PRISM PC CARD.

TCR7[0] affects the output value at the $\overline{\text{TXDATA}}$ pin. The $\overline{\text{TXDATA}}$ function is not used for the HARRIS PRISM PC CARD application and, therefore, TCR7[0] must be reset to a 0.

TCR8[7:0] Start Delimiter LSB

TCR8[7:0] can be set to any value for IEEE 802.11 protocol operation, since the start delimiter for the 802.11 DS PHY is only two bytes in length.

TCR8[7:0] is used to match the third arriving byte of the Start of Frame Delimiter (Unique Word) in incoming and outgoing frames. This information is used to determine

when the MAC CRC state machine should begin calculation.

TCR9[7:0] Start Delimiter CSB

TCR9[7:0] should be set to A0 for IEEE 802.11 protocol operation.

TCR9[7:0] is used to match the second arriving byte of the Start of Frame Delimiter (Unique Word) in incoming and outgoing frames. This information is used to determine when the MAC CRC state machine should begin calculation.

TCR10[7:0] Start Delimiter MSB

TCR10[7:0] should be set to F3 for IEEE 802.11 protocol operation.

TCR10[7:0] is used to match the first arriving byte of the Start of Frame Delimiter (Unique Word) in incoming and outgoing frames. This information is used to determine when the MAC CRC state machine should begin calculation.

TCR13[7:0] Pin Configuration A

TCR13[7:0] should be set to FFh.

The TCR13[7:0] bits are used to determine if the I/O structure at the pin location specified for each bit location is to be enabled to drive output values. For the HARRIS PRISM PC CARD application, each of the following pins will either be used as an output from the Am79C930 device, or as an NC in the design. Therefore, the I/O structure associated with the pin needs to be enabled to drive as an output: $\overline{\text{LNK}}$, $\overline{\text{LFPE}}$, $\overline{\text{HFPE}}$, SDCLK, $\overline{\text{SDSEL3}}$, $\overline{\text{SDSEL2}}$, $\overline{\text{SDSEL1}}$, and $\overline{\text{RXPE}}$.

TCR14[7:0] Pin Configuration B

TCR14[7:0] should be set to D8h.

The TCR14[7:0] bits are used to determine if the I/O structure at the pin location specified for each bit location is to be enabled to drive output values. For the HARRIS PRISM PC CARD application, USER7, LLOCKE, USER4, and USER3 will all be outputs from the Am79C930 device, or will be NC in the design, hence, TCR14[7:6] and TCR14[4;3] are set to 1. Note that TCR14[5] is a reserved location and, therefore, must be reset to a 0.

TCR15[7:0] ANTSLTLEN

TCR15[7:0] should be set to 82h.

The TCR15[7:1] bits are used to determine if the I/O structure at the pin location specified for each bit location is to be enabled to drive output values. For the HARRIS PRISM PC CARD application, $\overline{\text{ANTSLT}}$ and $\overline{\text{ACT}}$ are outputs or NC in the design, while $\overline{\text{TXDATA}}$, TXCMD, RXC, USER6, and USER5 are inputs.

The TCR15[0] bit is used to determine the functionality of the $\overline{\text{STSCHG}}$ pin. For the HARRIS PRISM PC CARD application, this pin is not to be used for $\overline{\text{STSTCHG}}$ functionality; hence, the TCR15[0] bit should be set to 0.

TCR27[7:0] TIP LED Scramble

TCR27[7:0] should be set to 83h.

TCR27 controls several functions.

TCR27[7] controls the RUNERR function. The RUNERR function is not used for the HARRIS PRISM PC CARD application. Therefore, TCR27[7] must be set to a 1 in order to disable the function.

TCR27[6] is reserved and must be reset to 0.

TCR27[5] controls the use of the input signal at the USER5 input pin for the EXTCS function. For the HARRIS PRISM PC CARD application, the USER5 input signal is not used for the EXTCS function. Therefore, TCR27[5] must be reset to 0.

TCR27[4] controls the drive type on the $\overline{\text{LNK}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{LNK}}$ pin drive should be open drain. Therefore, TCR27[4] must be reset to a 0.

TCR27[3] controls the drive type on the $\overline{\text{ACT}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{ACT}}$ pin drive should be open drain.Therefore, TCR27[3] must be reset to a 0.

TCR27[2] controls the polarity of the $\overline{\text{FDET}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{FDET}}$ pin is not used, Therefore, TCR27[2] must be reset to a 0.

TCR27[1] controls the polarity of the $\overline{\text{TXPE}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{TXPE}}$ pin is not used. Therefore, TCR27[1] must be reset to a 1.

TCR27[0] controls the polarity of the $\overline{\text{TXMOD}}$ pin. For the HARRIS PRISM PC CARD application, the $\overline{\text{TXMOD}}$ pin polarity should be high assert. Therefore, TCR27[0] must be reset to a 1.

TCR28[7:0] CCA Configuration

TCR28[7:0] should be set to 20h.

TCR28[7:0] controls several functions.

TCR28[7] controls the RXC pin function. The RXC function of this pin is not used for the HARRIS PRISM PC CARD application. Therefore, TCR28[7] must be reset to a 0 in order to disable the function.

TCR28[6] controls the EXTSDF pin function. The Am79C930 device will perform the Start Delimiter Detection (Unique Word Detection) for incoming receive frames and, therefore, TCR28[6] must be set to a 0 in order to disable the external input function.

TCR28[5] controls the EXTCHBSY pin function. The HARRIS HSP3824 chip will perform the Clear Channel

Assessment function in this application and, therefore, TCR28[5] must be set to a 1 in order to enable the function. (Note that ENXCHBSY (TCR28[5]) must also be set to a 1 in order for the EXTSDF input to be fully functional.)

TCR28[4] controls the Am79C930 device's use of its internal antenna diversity decision to control the transition of the receive state machine from RCVR_ENABLED to SFD_SEARCH. For the HARRIS PRISM PC CARD application, the Antenna Diversity function will be performed by the HARRIS HSP3824 device. Therefore, TCR28[4] must be set to a 0 in order to allow the RCVR_ENABLED to SFD_SEARCH state transition to occur unconditionally.

TCR28[3] controls the Am79C930 device's use of its internal CCA logic decision to control its internal antenna diversity decision. Since neither of these functions is used in the HARRIS PRISM PC CARD application, TCR28[3] must be set to a 0.

TCR28[2] controls the Am79C930 device's use of its internal baud detect logic output to control its internal antenna diversity decision. Since the Am79C930 device's antenna diversity logic is not used in the HARRIS PRISM PC CARD application, TCR28[2] must be set to a 0.

TCR28[1] controls the Am79C930 device's use of its internal baud detect logic output to control its internal CCA decision. Since the Am79C930 device's CCA logic is not used in the HARRIS PRISM PC CARD application, TCR28[1] must be set to a 0.

TCR28[0] controls the Am79C930 device's use of its internal RSSI threshold comparison logic output to control its internal CCA and antenna diversity decisions. Since neither of these functions is used in the HARRIS PRISM PC CARD application, TCR28[0] must be set to a 0.

TCR30[7] Pin Function and Data Rate

TCR30[7:0] should be set to 89h.

TCR30[7:0] controls several functions.

TCR30[7] controls the $\overline{\text{ANTSLT}}$ pin function. The $\overline{\text{ANTSLT}}$ pin is used as an output of the Am79C930 device for the HARRIS PRISM PC CARD application. Therefore, TCR30[7] must be set to a 1 in order to invoke the proper pin function.

TCR30[6] controls the $\overline{\text{TXDATA}}$ pin function. The $\overline{\text{TXDATA}}$ pin is not used for the HARRIS PRISM PC CARD application. Therefore, TCR30[6] must be reset to a 0 in order to invoke the proper pin function.

TCR30[5] controls the TXCMD pin function. The TXCMD pin is not used for the HARRIS PRISM PC CARD application and, therefore, TCR30[5] must be reset to a 0 in order to invoke the proper pin function.

TCR30[4] controls the USER7 pin function. The USER7 pin is not used for the HARRIS PRISM PC CARD application and, therefore, TCR30[4] must be reset to a 0 in order to invoke the proper pin function.

TCR30[3] controls the direction of the TXC pin. The TXC pin is an input to the Am79C930 device within the HARRIS PRISM PC CARD application. Therefore, TCR30[3] must be set to a 1 in order to invoke the proper pin function.

TCR30[2:0] control the data rate of transmission and reception of serial network data. The nominal data rate will be 1 Mbps. The HARRIS PRISM PC CARD system is capable of switching automatically to a rate of 2 Mbps as indicated by the device driver software. The rate switch to 2 Mbps will be performed by setting a dynamic rate switch bit at the time of transmission for each frame, if the device driver has indicted that the frame should be transmitted at the higher rate. The MAC control firmware will set the dynamic rate switch bit as necessary. The setting of the Data Rate bits in this register should conform to the nominal rate of 1 Mbps, because the rate switching mechanism in the Am79C930 device allows for an automatic internal modification of the Data Rate bits such that in the middle of a frame to be transmitted a 2 Mbps, the internal Data Rate bits will automatically be modified at the correct time to allow the MAC portion of the frame to be transmitted a 2 Mbps. Therefore, the proper data rate setting for the HARRIS PRISM PC CARD application is 1 Mbps. With the CLKIN input frequency set to 40 MHz, TCR30[2:0] must be set to ZERO ZERO ONE binary in order to invoke the proper 1 Mbps nominal data rate.

NC Pins Configured As Output

As a general note, unused pins (NC) may have an input/output structure inside of the Am79C930 device. Therefore, if the unused pin is left as an NC on the PCB and the output structure of the Am79C930 device is not driving a value, then the pin will float and the input structure could enter a state which sinks a relatively undesirable amount of DC current, thereby, reducing the effectiveness of the power-down mode. In order to guarantee the lowest possible power-down current consumption, *all Am79C930 pins that are NC in the design need to be configured for output mode of operation*. Following the register setting recommendations described in this application note will achieve this objective.

Am79C930 Register Settings Summary

Table 6 indicates the required configuration register settings for the Am79C930 device, when combined in a design with the HARRIS PRISM chip on a PC CARD card.

Table 6. Am79C930 Register Settings

Am79C930 Register	Required Configuration Setting
MIR8	08h - 0 wait state FLASH 09h - 1 wait state FLASH 0Ah - 2 wait state FLASH 0Bh - 3 wait state FLASH
MIR9	82h - 0 wait state SRAM 92h - 1 wait state SRAM write to this register after all other registers have been configured because the MIR9[1]=1 setting will prevent writes to some of the configuration registers
TIR2	40h
TIR11	00h
TIR26	08h
TCR7	02h
TCR8	don't care
TCR9	A0h
TCR10	F3h
TCR13	FFh
TCR14	D8h
TCR15	82h
TCR27	83h
TCR28	20h
TCR30	89h

Host PC/Adapter Card Interaction

Normal system operation (i.e., transmission and reception of frames) is controlled through the use of a software device driver. The device driver will interface with the Am79C930 device through command, status, and data structures that exist in the SRAM component of the system and through registers that are part of the Am79C930 device. The Am79C930 device, in turn, will perform all interface functions that are required in order to operate the remaining system components (i.e., the HARRIS PRISM chipset).

TX Flow

The Am79C930 device will perform the following functions of the TX operation:

- Monitor CCA input for TX defer procedure; execute backoff if necessary
- Execute TX power ramp-up sequence for radio when defer=FALSE and a TX frame is queued
- Generate preamble (uses DMA to transfer to TX FIFO)
- Generate Unique Word (uses DMA to transfer to TX FIFO)
- Generate PLCP header (uses DMA to transfer to TX FIFO)
- Generate PLCP CRC16 value (uses DMA to transfer to TX FIFO)
- Dynamic rate switch from 1 Mbps to 2 Mbps (if necessary)
- Generate MAC header (uses DMA transfer to TX FIFO)
- Generate MAC data (uses DMA to transfer to TX FIFO)
- Generate MAC CRC32 (generated by TX hardware state machine)
- Execute TX power ramp-down sequence for radio (by TX hardware state machine control)

The transmit operation is initiated by a request from the software driver to the Am79C930-based adapter card. The driver will first place the data to be transmitted into a predefined TX buffer area of the SRAM. The firmware will poll the TX descriptors at a periodic interval. At the next poll of the TX descriptors, the firmware will discover the new TX frame and will initiate the MAC TX Sequence as follows.

TX SEQUENCE for the Am79C930 Device Generates and Strips PHY Fields

Note: *Only the interaction between the Am79C930 device and the HARRIS PRISM subsystem is described.*

The mode of operation selected for the Am79C930/HARRIS PRISM chip application is one in which the Am79C930 device will generate preamble, Unique Word, and PLCP header for transmission, for an RX frame, the Am79C930 device will remove these fields. As such, the first operation for the Am79C930 device in response to the discovery of a TX frame waiting in the TX descriptor is to prepare the necessary PPDU(s) from the MSDU. This step and those that follow in sequence are described below.

1. The Am79C930 firmware has prepared ahead and already has enabled the DMA0 engine to move the preamble plus Unique Word (UW) from a template in SRAM into the TX FIFO. Effectively, these fields have been preloaded into the TX FIFO, even before the device driver has written a descriptor for a transmit frame.
2. The Am79C930 firmware determines whether the MSDU should be fragmented. If no fragmentation is needed, then the firmware creates a single "internal descriptor" which contains the appropriately formed PLCP header, including a calculated CRC16, and an appropriately formed MAC header (which was created by the device driver and passed to the firmware as part of the MSDU "data." However, the firm-

ware must modify certain bit fields of the MAC header which the device driver is unable to fill in at the time of creation). The internal descriptor contains a pointer which points to the SRAM location that contains the MSDU data that has been passed from the device driver. If fragmentation is required, then multiple internal descriptors are created, each containing a PLCP header and a MAC header for one fragment, and each pointing to a portion of the MSDU data that has been passed from the device driver. The "internal descriptor" structures are not shared with the device driver. The device driver is unaware that such descriptors have been created. The purpose of the internal descriptors is to allow for fragmentation, if needed, and to create a location to store the appropriately formed PLCP headers and modified MAC headers.

Note: *The Am79C930 device firmware will compute the CRC16 for the PHY PLCP, i.e., this operation is not performed in hardware. The firmware CRC16 operation consists of a few lookups into a small lookup table. Therefore, the firmware implementation is quite simple and fast.*

3. The Am79C930 firmware monitors CHBSY (CHBSY is supplied by the PHY through the USER5/EXTCHBSY input). The CHBSY signal is available to Am79C930 firmware as a direct read of a register bit. Changes to the CCA signal value are signaled to firmware through interrupts to the embedded 80188 core of the Am79C930 device, or they are seen as the CHBSY bit is polled, depending upon which firmware procedure is examining the CHBSY status.
4. When the Am79C930 device firmware sees CHBSY=0 (medium IDLE) for the required DIFS time plus the selected backoff time, then the firmware initiates the TX operation by asserting the TXS bit of TIR8.
5. TXCMD - TX_PEB is asserted under state machine control in response to the assertion of the TXS bit of TIR8.
6. $\overline{\text{TXPE}}$ - TX_PEA is asserted under state machine control after a programmable number of bit times. (The programmable delay time was set during Am79C930 configuration, as part of an API call.)
7. TXDATA - TXD provides data from the TX FIFO after a programmable time following the assertion of $\overline{\text{TXPE}}$ - TX_PEA. (The programmable delay time was set during Am79C930 configuration, as part of an API call.)
8. The SFD DETECT logic inside of the Am79C930 device is programmed to recognize the Unique Word for outgoing transmissions. When the Unique Word is detected, the Am79C930 device will wait the programmed PFL time (TCR3[3:0]) and then begin CRC32 calculation. At the same time that the CRC32 logic is started, the device will switch clock rates to allow for a dynamic rate change of the frame, *if needed*. The dynamic rate change will occur if the dynamic rate bit (TIR8[3] has been set. If the bit has been set, then the preamble, Unique Word and PLCP header fields will leave the Am79C930 device at a 1 Mbps rate, and the MAC header and the remainder of the frame will leave the Am79C930 device at a 2 Mbps rate.
9. At the end of the data portion of the TX, the TXDATA pin of the Am79C930 device returns to its default state.
10. After a programmable delay, the $\overline{\text{TXPE}}$ (TX_PEA) pin will become deasserted.
11. After another programmable delay, the TXCMD - TX_PEB pin is deasserted, ending the transmission.

Interface Timing

The following timing diagrams indicate the logical signalling that will be generated between the HARRIS DS PHY chipset and the Am79C930 device for various operations.

Initialization to Receive (RX)

See Figure 4 for the initialization-to-receive timing diagram.

Receive to Transmit to Receive (RX to TX to RX)

See Figure 5 for the receive-to-transmit-to-receive timing diagram.

Receive (RX) to Sleep

For the SLEEP mode of the HSP3824 and the other devices of the HARRIS HFA3x24 to be placed into the sleep mode, the following signals should be placed into the states shown in Table 7.

Table 7. SLEEP Mode Signal States

Pin Name: AMD_NAME (HARRIS_NAME)	Pin State During Sleep Mode
TXCMD (TX_PEB)	0
$\overline{\text{TXPE}}$ (TX_PEA)	0
$\overline{\text{RXPE}}$ (RX_PE)	0

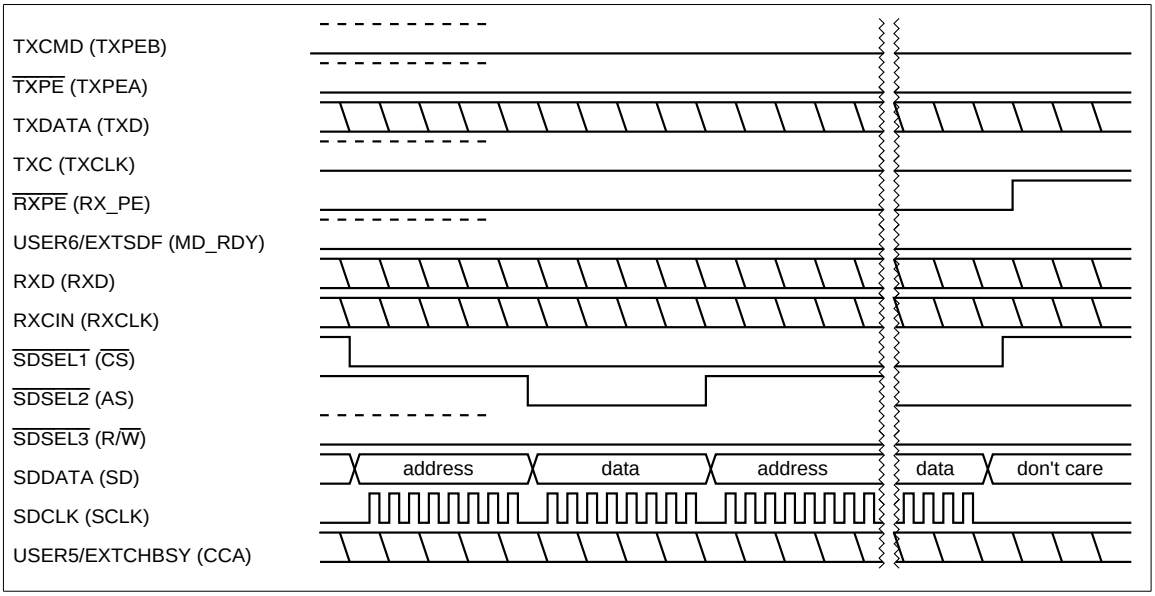
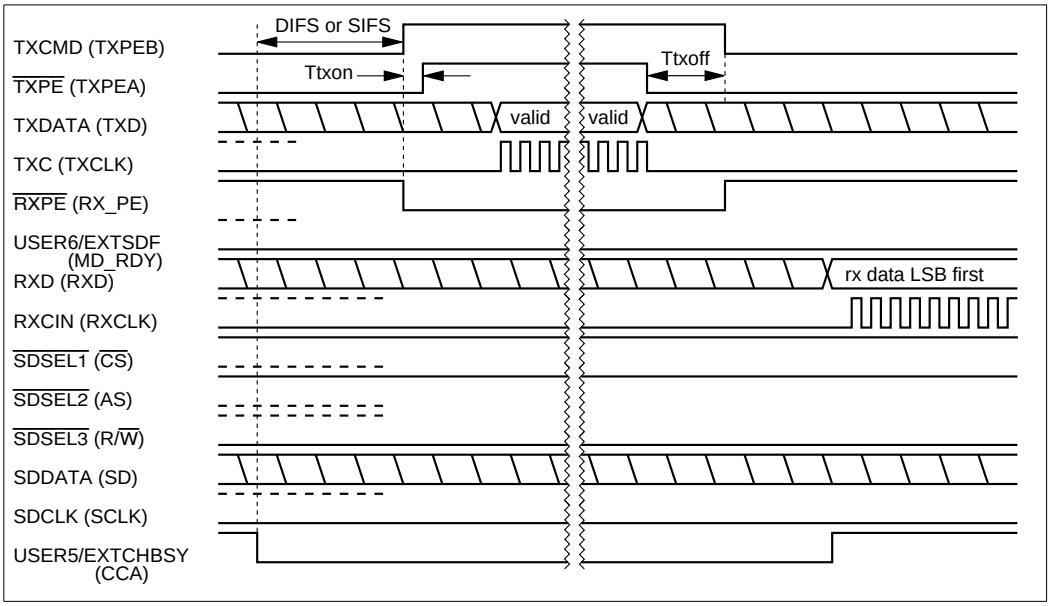


Figure 4. Initialization to RX



Ttxon = 100 ns min, 150 ns max; Ttxoff = 1.8 µsec min, 2.0 µsec max

Figure 5. RX to TX to RX

RX Flow

The Am79C930 device will perform the following functions of the RX operation:

- Enable the RX portion of the HSP3824 chipset
- Monitor USER6/EXTSDF input for RX Unique Word detection event as signalled by the HSP3824
- Parse the PLCP header
- Check the PLCP CRC16
- Parse MAC header
- Pass MAC data to the LLC layer
- Check the MAC CRC32
- Disable the RX portion of the HSP3824 chipset

The receive operation is enabled by the host driver software. Individual receive events are not generally predictable. Therefore, the receive operation is initiated by the Am79C930 firmware whenever the transmit operation is not in progress, the current power savings state dictates remaining awake, and the host driver software has enabled the receive operation. An indication from the HARRIS PRISM chip will begin the sequence that constitutes the reception of a frame.

RX SEQUENCE for the Am79C930 Device Generates and Strips PHY Fields

Note: *Only the interaction between the Am79C930 device and the HARRIS PRISM subsystem is described.*

The steps of the Am79C930 RX operation in the HARRIS PRISM PC CARD application are listed below:

1. The Am79C930 firmware prepares the DMA1 engine for DMA transfers from the RX FIFO to the next available RX buffer space. The DMA1 engine is enabled.
2. The Am79C930 firmware sets the RXS bit of TIR16 to enable the RX state machine whenever there is no transmit activity (and the current power savings state allows it). RX data and RX clock from the decoder in the HARRIS HSP3824 arrives at the RXD and RXCIN inputs. The RX DATA will be descrambled by the HARRIS HSP3824 device. RX DATA is not stored by the Am79C930 device until the Unique Word has been identified by the SFD detection logic of the Am79C930 device.
3. Once the Unique Word field has been detected, the Am79C930 device's RX state machine will move from the Unique-Word-search state to the DATA-accept state. The Am79C930 device will now begin accepting data at the RXD input and placing it into the RX FIFO.
4. The first bit that is placed into the RX FIFO is the first bit of the PLCP header.
5. As RX data continues to arrive, the RX FIFO signals a request for DMA to the embedded 80188. The DMA machine moves bytes from the RX FIFO to the RX buffer in the SRAM.
6. If the incoming RX frame is a 2 Mbps frame, then the bit rate will change from 1 Mbps to 2 Mbps at the PHY/MAC boundary. At this point, the RX clock from the HARRIS HSP3824 will switch from 1 MHz to 2 MHz. The Am79C930 device is able to accept any RX clock rate at any time, up to a limit of 8 MHz.
7. As RX data continues to arrive, the Am79C930 device firmware examines the data that has been placed into the SRAM in order to determine the nature of the RX frame. At this time, PLCP header information is parsed for correct field values and a valid CRC16. If the CRC16 value is corrupted, then the firmware will terminate the reception of the frame at this point by sending a deassertion pulse on the RX_PE signal.

   Note that the HSP3824 contains an OVERRIDE bit (CR2[5]) to allow the message to continue despite the presence of a bad PLCP CRC16 indication. This allows for the Am79C930 firmware to have complete control of the subsequent action in the case of a bad CRC16 value.
8. If CRC16 indication is OK, then RX continues. As the MAC header is received, the Am79C930 firmware continues to parse incoming fields, determines future actions (such as responding with a transmission) and prepares for them by copying appropriate fields of the MAC header into a TX buffer.
9. When PLCP length (obtained from the PLCP header) number of bytes occurs, the Am79C930 firmware will deassert the RX_PE signal to end the frame reception to reset the RX portion of the HSP3824.
10. End of reception.

API REQUIREMENTS

This section gives examples of the API modules that are required to be written in order to support the HARRIS PRISM chipset. These modules must then be compiled and finally linked with the MAC protocol object code. The complete list of API modules is found in a separate document, the *Am79C930 Device Driver Interface Description.*

API Procedures

Enable_TX()

This routine keys the transmitter on. It is called after a frame has been completely formatted and TX_DMA is programmed. Upon return from this routine, the Am79C930 device will have started the data transmission. Any register and transceiver-dependent signalling that is required to turn the transmitter on will be executed in this routing, including assertion of the TXS bit of TIR8.

The transceiver will remain in the transmit state until the transmission is terminated by either the Am79C930 hardware and/or the call tot he Disable_TX routine.

uint8 Enable_TX if Good(uint16 good length, uint8 dma length)

This API call is used by firmware when it is receiving a frame to which it knows that a response will be sent if the CRC32 on the frame is good. In order to minimize the SIFS turnaround of the firmware, this API call is made a few bytes before the end of the frame being received. This routine will wait until DMA zero transfer count has reached DMA length, at which time it will check if the CRC is good by comparing the Am79C930 CRC good length register with the good length value passed in the API call. If they match, then this API routing will start the transmitter and return.

Disable_TX()

Most transceiver transmitters would typically be keyed off by the Am79C930 hardware which provides automatic termination of the transmission through state machine generated signals at the Am79C930 pins. After all bytes of the frame and the CRC have been sent, the Am79C930 firmware will call this routine. The Disable_TX routine will key off the transmitter if automatic termination cannot be supported and enable any RX functions that are required.

reset_CCA()

This routine is called by the firmware whenever it wishes CCA to clear and start again. The actual steps taken by this routine will depend on the PHY hardware used, but as a minimum it must clear any CCA busy indication.

Enable_RX()

This routine is used to enable the receiver and is called after each transmission. Enable_RX() must also perform the equivalent of the reset_CCA function.

Sleep (uint8 Sleep_Lvl)

This routine is used tot put the radio into its low power consumption Sleep state. Once in the Sleep state, the Wake routine must be called to resume operation. Different degrees of sleep are allowed by passing a parameter indicating which level of sleep is currently desired.

Wake()

The Wake routine will cause the radio to exit the Sleep state and power up its circuitry. When the routine returns the radio will be in the Receive state.

Initialize (uint8 Domain)

This routine is called during the firmware's initialization procedure. The Am79C930 registers will be programmed by this routine bringing he Am79C930 to a known default state. Following Am79C930 initialization, the transceiver should be reset and then specific registers programmed, bringing the transceiver to the required initial default state. This includes a default channel and power level for the transmitter. The transceiver may then be programmed for a specific channel and power setting with the following commands.

The Domain of operation is passed as an argument in the Initialize() routine in order to allow any special processing or function that needs to be performed for a particular domain to be correctly executed. As an example, in the MKK domain, it is a requirement that all radiators in the 2.4 GHz band send an identification frame upon power up. In order to allow this function to be carried out, additional information must be available in order to form the frame properly, and this information can be obtained at a fixed location in the code space, immediately following the normal MIB structures. The structure that follows contains one byte identifying the domain, one byte giving a length, and subsequent bytes that contain any necessary data (MKK Callsign ID, for example).

Preset_Channel (uint8 Channel)

This routine is called to move the programming information for the *NEXT* channel from the table in the flash into a next channel information variable in the SRAM, and also, to copy this same value into the next channel register inside of the radio synthesizer, if such a register exists.

Note: *Some radio hardware does not contain a "next channel" register. For such radios, the Preset_channel() operation will only be able to copy the next channel information programming information from the flash tot he SRAM.*

A typical application of Preset_channel() is that following a Change_channel() call during an FH hop, a Preset_channel() API call should be executed. This allows the Change_channel() call to be executed at the

dwell boundary, thereby, removing the synthesizer channel information download from the Am79C930 device to the radio from the critical 240 µsec time allotted for channel hops.

Preset_channel() will be a null routine for the DS PHY since this routine is not called by the DS section of the code.

Change_Channel ()

This routine is called to move the preset channel from the preset register in the radio hardware's synthesizer into the synthesizer's working register, also inside of the radio hardware, thereby causing the synthesizer to begin the relock operation which will result in the retuning of the radio to a new channel. This routine also copies the next_channel_information variable into the current_channel_information variable. This copy operation is required because scanning operations (and other possible operations) will cause non-hop channel changes. And, in order to return to the correct channel at the end of a scan operation, the current_channel_information must be available to the scan procedure.

Typically, this routine should cause the assertion of a SYNTH_LOAD signal that executes a transfer from a holding register to a working register in the synthesizer. The intent of having separate Preset_channel() and Change_Channel() calls is to allow the fastest possible channel change operation. The Change_channel() call should NOT perform the transfer of the synthesizer programming information from the MAC to the PHY, unless the PHY does NOT support a preload of the next channel programming information.

In the case of a radio that does not support the maintenance of next channel information, the Change_Channel() call will perform the move of the next channel information from the next_channel_information variable (in SRAM) to the synthesizer hardware in the radio. This will be followed by the assertion of whatever signalling is required to enable the synthesizer to re-sync according to the new channel programming information.

Change_Channel() will be a null routine for the DS PHY since this routine is not called by the DS section of the code.

Force_Channel (uint8 Channel)

This routine is called to change the frequency of the transceiver to the channel specified in the passed parameter. The current_channel_information variable in SRAM is modified.

Force_channel() will typically be used for scanning operations and for channel changes that are the result of AP association changes. For the scanning application, the main program flow is one where Force_channel() is called in order to force a channel change for the scan. Additional channel changes may be performed during the scan. When the scan operation has been completed, the MAC Management code of the Am79C930 firmware will perform a re-sync operation with the saved state information, thereby solving the problem of a scan covering multiple hop intervals and also allowing the current_channel_information to be modified (since this information is stored at the MAC management level).

The range of the Channel parameter will depend on the PHY type being used. Geographic restrictions will also force the use to subsets of the channel range.

FH = 2 - 95

DS = 1 - 12

IR = n/a

Force_Channel() will be a null routine for the IR PHY since it has only one defined channel.

Set_Power (uint8 Power_Lvl)

This routine is called to change the output power the transceiver uses when transmitting to the level specified in the passed parameter.

The range of the Power_Lvl parameter will depend on the PHY type being used.

FH = 1 - 4

DS = 1 - n

IR = n/a

Set_Power() will be a null routine for the IR PHY since it has only one defined output power.

Some radios determine transmit power level by signalling at the pin level. Therefore, the power level as determined by the PHY MGMT procedure and communicated with the Set_Power() call needs to be stored in a variable (e.g., TX_Power) such that each time Enable_TX() is called, the Enable_TX() routine can reference the stored TX_Power level variable and create the appropriate signalling for the radio power ramp function.

For a radio that maintains a local copy of the power level setting (for example, inside of a radio control register), the Set_Power() API will, in addition to modifying the FW copy of the global power level variable, execute a write to the radio's register.

The API will attempt to get the best fit for the power level that is communicated without exceeding the requested power level.

uint8 Get_PHY_Type()

This routine is used to get the type of PHY attached to the API. The type is hard coded into the API function. The two supported types are FREQUENCY_HOPPING and DIRECT_SEQUENCE.

uint8 Get_Tx_Preamble_Len(uint8 rate)

This function returns the actual length of the transmit Preamble that will be DMAed by the firmware when it is transmitting a frame. The length of the preamble/SFD is different for each PHY, and in certain implementations, different for each bit rate.

uint8 Get_Rx_PLCP_Header_Len()

This function returns the number of PLCP header bytes that will be in received frames. This function is needed by modules like PHY_Rx for its buffer management.

uint8 Get_Rx_Rate(uint8* PLCP_ptr)

This function is used to obtain the rate from the PLCP header pointed to by PLCP_ptr. The encoding of the rate information is different for each PHY. The return values are '0' for 1 Mbps and '1' for 2 Mbps.

uint8 Get_Tx_PLCP_header_len(uint8 rate)

This function returns the length of the PLCP header in bytes that must be transmitted for the particular PHY and data rate. The rate parameter indicates the rate at which the MAC portion of the frame will be transmitted and is '0' for 1 Mbps and '1' for 2 Mbps.

uint8 Get_RSSI()

This function returns an RSSI value between 0 and 255, sampled when this function is called. PHYs that do not have an RSSI indicator shall return a value of zero. This function is called by PHY_Rx when it receives the SFD of a frame.

BOOLEAN Is_PLCP_Header_Good(uint8* PLCP_ptr)

This function is called by the firmware when it wants to verify that the PLCP Header pointed to by PLCP_ptr has a good CRC16. This function returns zero if CRC was bad and non-zero if CRC was good.

uint16 Get_Length(uint8* PLCP_ptr)

This function extracts and returns the length field from the PLCP Header pointed to by PLPC_ptr.

Build_PLCP_Header(uint8* PLCP_start, uint16 length, uint8 rate)

This function is called by the firmware to build the PHY specific PLCP Header in the buffer pointed to by PLCP_start. The MPDU length and rate information.

uint8* Get_Tx_Preamble(uint8 rate)

This function returns a pointer to the buffer containing the Preamble that is to be transmitted for each frame. Since preambles may be different for different rates the rate parameter is passed to the routine. '0' implies 1 Mbps and '1' implies 2 Mbps. The Preamble buffer contains the 1-0-1-0 pattern as well as the SFD sequence and any other special control bytes needed by the PHY. This buffer must not be changed since the firmware does not copy it to its own array but uses it in place.

Set_PHY_Rate(uint8 rate)

This function is called to allow the programming of any PHY specific registers that may be required in order to change the rate of the PHY. The rate parameter is '0' for 1 Mbps and '1' for 2 Mbps.

The function is called by the firmware when it pre-loads the TX DMA FIFO with the rate specific preamble for the next expected transmission.

User_Function()

This API function is called by the firmware once every time around its main loop. Any function that the integrator may need can be placed here. A byte in the Control Block is reserved as a means for the driver to communicate with this function. Possible uses for this function are, updating any LED's, monitoring any PHY status lines, etc.

Unsigned16 rel_time_to_μsec_est()

This API function is called by the firmware in order to get a quick conversion from the Am79C930 real time clock to a value in microseconds. This operation is called at time-critical points in the firmware and, therefore, the API routine must have a short execution time. An indexed jump into a small table with interpolations thereafter would be one possible implementation that could meet the execution-time criteria. The sample API routine that is provided is sufficient for all implementations that use a 32.768kHz crystal. For any system implementation in which any other value of crystal is used, this routine needs to be modified to accommodate the difference in crystal frequency.

Unsigned32 rel_time_to_μsec()

This API function is called by the firmware in order to get a much more accurate conversion from the Am79C930 real time clock to a value in microseconds than the rel_time_to_usec_est() function can provide. This operation is called at points in the firmware when execution time is not critical and, therefore, the API routine should include a much more sophisticated algorithm giving a much more exact conversion. The sample API routine that is provided is sufficient for all implementations that use a 32.768 kHz crystal. For any system implementation in which any other value of crystal is used, this routine needs to be modified to accommodate the difference in crystal frequency.

Unsigned32 μsec_to_rel_time()

This routine converts a 32-bit number (max allowed value 8,000,000) in units of μsecs to 32768 Hz clock ticks. The conversion is accurate to one tick. Partial ticks are truncated since the error is not expected to accumulate. This routine accomplishes the divide by 30.51 μsec by first multiplying by 512 and dividing by 15625. (equivalent to 32768/1000000). *This operation is critical in time!* The sample API routine that is provided is sufficient for

all implementations that use a 32.768 kHz crystal. For any system implementation in which any other value of crystal is used, this routine needs to be modified to accommodate the difference in crystal frequency.

pgm_clkgt20()

This routine sets the CLKGT20 bit of the MIR9 register, since the 80188 microcontroller inside of the Am79C930 device must be running at 20 MHz (Am79C930 CLKIN pin = 40 MHz) in order to support the MIPS requirement of the IEEE 802.11 MAC protocol firmware. This routine may set the CLKGT20 bit of MIR9 to zero only if non-IEEE 802.11 MAC protocol firmware is operating and the Am79C930 CLKIN input pin is running at a frequency of 20 MHz or less.

set_wait_states()

This routine sets the number of wait states to be introduced for SRAM and FLASH memory accesses by altering the contents of the MIR8 and MIR9 registers of the Am79C930 device. With the Am79C930 CLKIN input running at 40 MHz, memory devices that require a setting of greater than one wait state will cause improper function. Memory devices that allow wait state settings of less than one are allowed for either SRAM or FLASH.