#ifndef PTPD\_H\_﻿#define PTPD\_H\_﻿#include <stdio.h>﻿#include <string.h>﻿#include <stdbool.h>﻿// Include lwIP headers for networking types﻿#include "lwip/ip\_addr.h"﻿#include "lwip/udp.h"﻿#include "lwip/sys.h" // For sys\_mbox\_t﻿// Include Xilinx specific types﻿#include "xil\_printf.h"﻿#include "xil\_types.h"﻿// --- PTP Protocol Constants (from IEEE 1588-2008) ---﻿#define PTP\_EVENT\_PORT      319﻿#define PTP\_GENERAL\_PORT    320﻿// Multicast Addresses﻿#define PTP\_PRIMARY\_MULTICAST\_IP    IPADDR4\_INIT\_BYTES(224, 0, 1, 129)﻿#define PTP\_PEER\_MULTICAST\_IP       IPADDR4\_INIT\_BYTES(224, 0, 0, 107)﻿#define PTPD\_DEFAULT\_MAX\_FOREIGN\_RECORDS 5﻿#define ADJ\_FREQ\_MAX 500000 // Max frequency adjustment in ppb﻿// PTP Port States﻿typedef enum {﻿    PTP\_INITIALIZING, PTP\_FAULTY, PTP\_DISABLED, PTP\_LISTENING,﻿    PTP\_PRE\_MASTER, PTP\_MASTER, PTP\_PASSIVE, PTP\_UNCALIBRATED, PTP\_SLAVE﻿} ptp\_port\_state\_t;﻿// PTP Message Types﻿typedef enum {﻿    SYNC\_MSG, DELAY\_REQ\_MSG, PDELAY\_REQ\_MSG, PDELAY\_RESP\_MSG,﻿    FOLLOW\_UP\_MSG, DELAY\_RESP\_MSG, PDELAY\_RESP\_FOLLOW\_UP\_MSG,﻿    ANNOUNCE\_MSG, SIGNALING\_MSG, MANAGEMENT\_MSG,﻿} ptp\_message\_type\_t;﻿// --- Core PTP Data Structures ---﻿// Time representation (seconds and nanoseconds)﻿typedef struct {﻿    int64\_t seconds;﻿    int32\_t nanoseconds;﻿} TimeInternal;﻿// Uniquely identifies a PTP port﻿typedef struct {﻿    uint8\_t clockIdentity[8];﻿    uint16\_t portNumber;﻿} PortIdentity;﻿// Represents the quality of a clock﻿typedef struct {﻿    uint8\_t clock\_class;﻿    uint8\_t clock\_accuracy;﻿    uint16\_t offset\_scaled\_log\_variance;﻿} ClockQuality;﻿// Filter for PI controller﻿typedef struct {﻿    int32\_t n;﻿    int32\_t s;﻿    int32\_t y;﻿} Filter\_t;﻿// PTP Message Header﻿typedef struct {﻿    uint8\_t messageType;﻿    uint8\_t versionPTP;﻿    uint16\_t messageLength;﻿    uint8\_t domainNumber;﻿    uint16\_t flags;﻿    int64\_t correctionField;﻿    PortIdentity sourcePortIdentity;﻿    uint16\_t sequenceId;﻿    uint8\_t controlField;﻿    int8\_t logMessageInterval;﻿} PtpHeader;﻿// Announce Message Body﻿typedef struct {﻿    TimeInternal originTimestamp; // Not used, but part of the message﻿    int16\_t currentUtcOffset;﻿    uint8\_t grandmasterPriority1;﻿    ClockQuality grandmasterClockQuality;﻿    uint8\_t grandmasterPriority2;﻿    uint8\_t grandmasterIdentity[8];﻿    uint16\_t stepsRemoved;﻿    uint8\_t timeSource;﻿} AnnounceMessage;﻿// Stores information about potential masters﻿typedef struct {﻿    PortIdentity port\_identity;﻿    PtpHeader header; // Store the header of the last Announce﻿    AnnounceMessage announce\_message; // Store the body of the last Announce﻿} foreign\_master\_record\_t;﻿// Runtime configuration options﻿typedef struct {﻿    bool slave\_only;﻿    int8\_t sync\_interval;﻿    uint8\_t announce\_interval;﻿    ClockQuality clock\_quality;﻿    uint8\_t priority1;﻿    uint8\_t priority2;﻿} ptpd\_opts;﻿// --- Full PTP Data Set Definitions ---﻿// These structs hold the state of the clock as defined by the standard.﻿typedef struct {﻿    bool two\_step\_flag;﻿    uint8\_t clock\_identity[8];﻿    uint16\_t number\_ports;﻿    ClockQuality clock\_quality;﻿    uint8\_t priority1;﻿    uint8\_t priority2;﻿    uint8\_t domain\_number;﻿    bool slave\_only;﻿} DefaultDS\_t;﻿typedef struct {﻿    PortIdentity port\_identity;﻿    ptp\_port\_state\_t port\_state;﻿    int8\_t log\_min\_delay\_req\_interval;﻿    TimeInternal peer\_mean\_path\_delay;﻿    int8\_t log\_announce\_interval;﻿    uint8\_t announce\_receipt\_timeout;﻿    int8\_t log\_sync\_interval;﻿    uint8\_t delay\_mechanism;﻿    int8\_t log\_min\_pdelay\_req\_interval;﻿    uint8\_t versionNumber;﻿} PortDS\_t;﻿typedef struct {﻿    PortIdentity parent\_port\_identity;﻿    bool parent\_stats;﻿    int16\_t observed\_parent\_offset\_scaled\_log\_variance;﻿    int32\_t observed\_parent\_clock\_phase\_change\_rate;﻿    uint8\_t grandmaster\_identity[8];﻿    ClockQuality grandmaster\_clock\_quality;﻿    uint8\_t grandmaster\_priority1;﻿    uint8\_t grandmaster\_priority2;﻿} ParentDS\_t;﻿typedef struct {﻿    int16\_t current\_utc\_offset;﻿    bool current\_utc\_offset\_valid;﻿    bool leap59;﻿    bool leap61;﻿    bool time\_traceable;﻿    bool frequency\_traceable;﻿    bool ptp\_timescale;﻿    uint8\_t time\_source;﻿} TimePropertiesDS\_t;﻿// The main PTP clock data structure﻿typedef struct {﻿    // PTP data sets﻿    DefaultDS\_t default\_ds;﻿    PortDS\_t port\_ds;﻿    ParentDS\_t parent\_ds;﻿    TimePropertiesDS\_t time\_properties\_ds;﻿    // Foreign master records for BMC algorithm﻿    foreign\_master\_record\_t foreign[PTPD\_DEFAULT\_MAX\_FOREIGN\_RECORDS];﻿    // Software timers for PTP events﻿    int32\_t sync\_interval\_timer;﻿    int32\_t announce\_interval\_timer;﻿    int32\_t announce\_receipt\_timer;﻿    int32\_t delay\_req\_interval\_timer;﻿    // State machine and message handling data﻿    ptp\_port\_state\_t recommended\_state;﻿    bool waiting\_for\_followup;﻿    PtpHeader sync\_header;﻿    TimeInternal sync\_receive\_time;﻿    TimeInternal delay\_req\_send\_time;﻿    uint16\_t sent\_sync\_sequence\_id;﻿    uint16\_t sent\_delay\_req\_sequence\_id;﻿    // Servo and filter data﻿    TimeInternal offset\_from\_master;﻿    TimeInternal mean\_path\_delay;﻿    TimeInternal delay\_ms; // Master-to-slave delay component﻿    int32\_t observed\_drift;﻿    Filter\_t ofm\_filt; // Offset From Master filter﻿    Filter\_t owd\_filt; // One Way Delay filter﻿} ptp\_clock\_t;﻿// --- Function Prototypes (Public API) ---﻿// From ptpd.c (Core Protocol Engine)﻿int ptp\_startup(ptp\_clock\_t \*clock, ptpd\_opts \*opts, foreign\_master\_record\_t \*records);﻿void ptpd\_periodic\_handler(void);﻿// From protocol.c﻿void to\_state(ptp\_clock\_t \*clock, ptp\_port\_state\_t state);﻿void protocol\_tick(ptp\_clock\_t \*clock);﻿void handle\_announce(const PtpHeader \*header, const AnnounceMessage \*announce);﻿void handle\_sync(const PtpHeader \*header, const TimeInternal \*originTimestamp);﻿void handle\_follow\_up(const PtpHeader \*header, const TimeInternal \*preciseOriginTimestamp);﻿void handle\_delay\_req(const PtpHeader \*header, const TimeInternal \*rx\_ts);﻿void handle\_delay\_resp(const PtpHeader \*header, const TimeInternal \*receiveTimestamp, const PortIdentity \*requestingPortIdentity);﻿// From net.c (Network Layer)﻿bool ptpd\_net\_init(ptp\_clock\_t \*clock);﻿void ptpd\_net\_shutdown(ptp\_clock\_t \*clock);﻿int net\_send\_event(const void \*data, int len);﻿int net\_send\_general(const void \*data, int len);﻿// From sys\_arch\_ptp.c (Hardware Abstraction Layer)﻿void ptpd\_hw\_timer\_init(void);﻿void getTime(TimeInternal \*time);﻿void setTime(const TimeInternal \*time);﻿bool adjTime(int32\_t adj\_ns);﻿// From bmc.c (Best Master Clock Algorithm)﻿void init\_data(ptp\_clock\_t \*clock, ptpd\_opts \*opts);﻿void bmc\_add\_foreign\_master(ptp\_clock\_t \*clock, const PtpHeader \*header, const AnnounceMessage \*announce);﻿uint8\_t bmc(ptp\_clock\_t \*clock);﻿void update\_local\_as\_master(ptp\_clock\_t \*clock);﻿// From servo.c (Clock Servo)﻿void servo\_init\_clock(ptp\_clock\_t \*clock);﻿void servo\_update\_offset(ptp\_clock\_t \*clock, const TimeInternal \*sync\_event\_ingress\_timestamp, const TimeInternal \*precise\_origin\_timestamp);﻿void servo\_update\_delay(ptp\_clock\_t \*clock, const TimeInternal \*delay\_event\_egress\_timestamp, const TimeInternal \*recv\_timestamp);﻿void servo\_update\_clock(ptp\_clock\_t \*clock);﻿// From timer.c (Software Timers)﻿void init\_timer\_lists(ptp\_clock\_t \*clock);﻿void timer\_start(int32\_t \*timer\_id, uint32\_t interval\_ms);﻿void timer\_stop(int32\_t \*timer\_id);﻿bool timer\_expired(int32\_t \*timer\_id);﻿void timer\_tick(ptp\_clock\_t \*clock);﻿// From msg.c (Message Packing/Unpacking)﻿void handle\_msg(void \*data, int len);﻿void msg\_pack\_announce(uint8\_t \*buf, ptp\_clock\_t \*clock);﻿void msg\_pack\_sync(uint8\_t \*buf, ptp\_clock\_t \*clock, const TimeInternal \*originTimestamp);﻿void msg\_pack\_follow\_up(uint8\_t \*buf, ptp\_clock\_t \*clock, const TimeInternal \*preciseOriginTimestamp);﻿void msg\_pack\_delay\_req(uint8\_t \*buf, ptp\_clock\_t \*clock, const TimeInternal \*originTimestamp);﻿void msg\_pack\_delay\_resp(uint8\_t \*buf, ptp\_clock\_t \*clock, const PtpHeader \*req\_header, const TimeInternal \*receiveTimestamp);﻿#endif /\* PTPD\_H\_ \*/