SAND2010-7562C

LPC / SPI Analysis Tool

Prototype System for Analysis of LPC / SPI Bus Devices

March 24, 2010

Tiffany A. S. Pierce
Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Overview

- Introduction
  - Motivation
  - Overview of LPC / SPI
- LPC/SPI Analysis System
  - Components
  - Passive Analysis
  - Active Analysis
- Examples
  - SPI analysis
  - LPC analysis

Sandia National Laboratories

Introduction: Motivation

- Want to support development, debugging, and analysis for a variety of LPC/SPI devices and drivers
- Understanding bus traffic can help identify where bugs and protocol errors occur

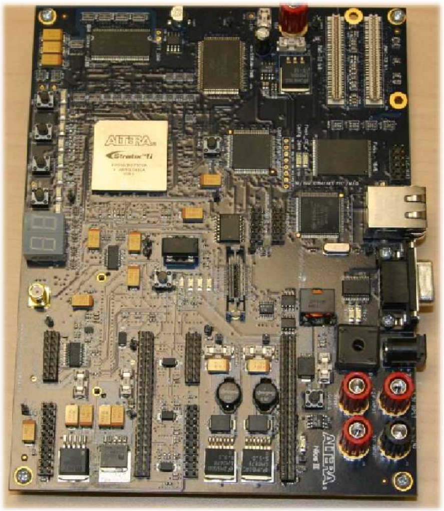
- Large volume of data on SPI or LPC buses
- Data needs to be parsed, filtered, interpreted
- Needs:
  - Observe devices interacting with a system
  - Isolate buggy devices to protect motherboard

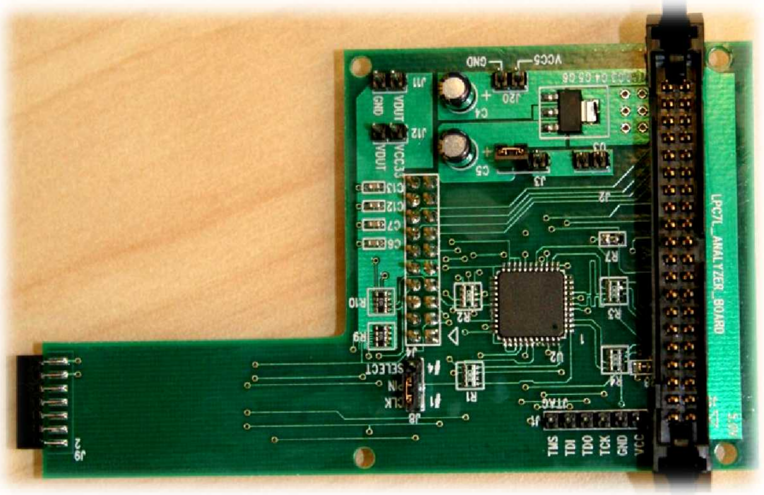
Introduction

- LPC / SPI buses
  - Shared bus; data visible for any device on bus
  - Relatively few pins used
- LPC devices
  - BIOS, serial and parallel ports, legacy keyboard, mouse, Trusted Platform Module (TPM)
- SPI devices
  - EEPROM, Flash memory, Ethernet, Real-time Clock
- Need for Custom Solution – existing solutions don’t do what we want.
- We developed a combined hardware and software system for LPC/SPI bus analysis

System Components: Hardware

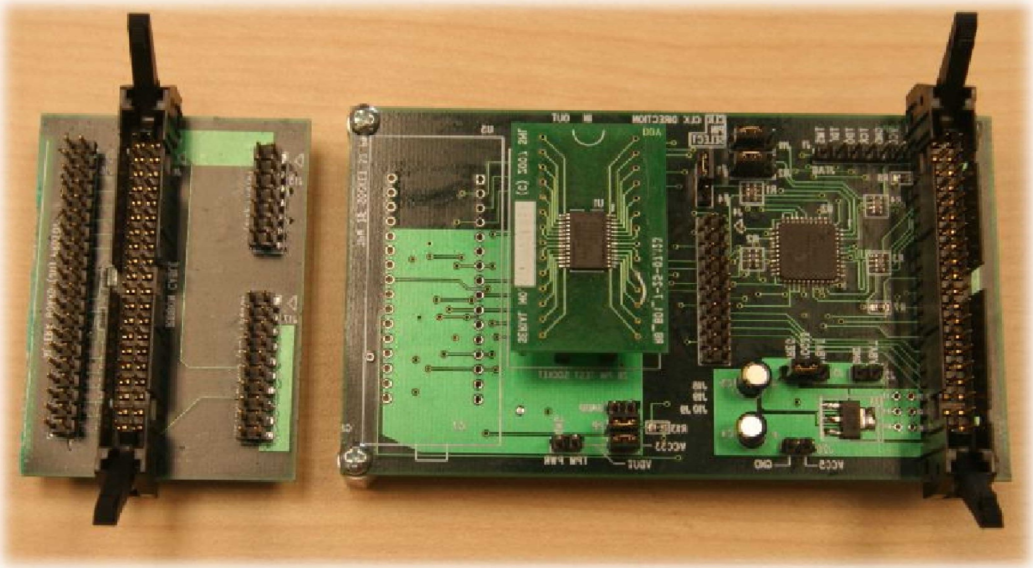
- FPGA prototyping board
  - Ethernet communications
  - Data Buffering
  - LPC/SPI basic protocol recognition
- Analyzer board
  - Plug-in to motherboard header
  - Passive analysis
- Analysis machine
  - Separate PC; not DUT
  - Runs analysis software
  - User Interface

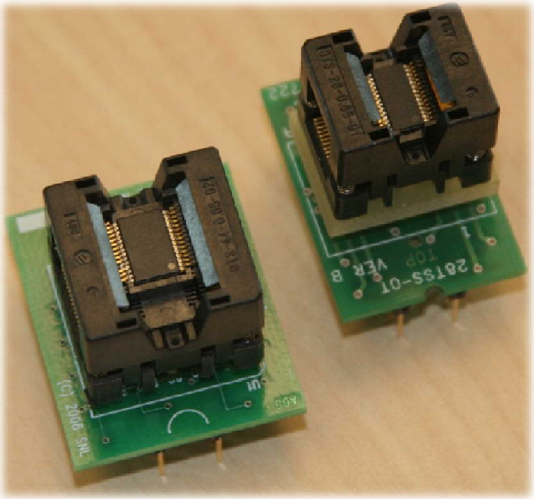




System Components: Hardware

- **Host board**
  - **Connects to FPGA board to send/receive data**
  - **Used to interact with loose components**
  - **Variety of pin setups for LPC / SPI devices**
  - **Some devices may require custom sockets**



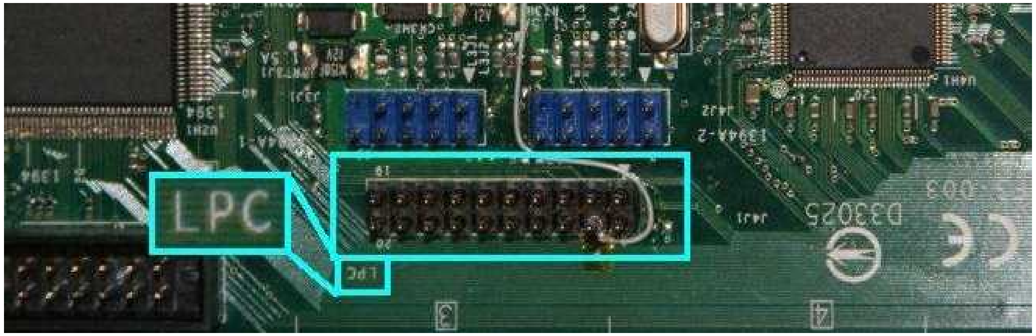


System Components: Software

- Runs on separate analysis machine
- Saves and reloads data for future work
- Passive analysis
  - Displays and Filters LPC & SPI data
  - Searchable events
  - Identifies protocol errors
- Active analysis
  - Send/receive events on host board
  - Parses and builds some device-specific commands
    - Prototype stage
    - Not implemented for most devices

Passive Analysis

- Set up and connect the hardware components
- Connect analysis board to LPC or SPI header
- Good for driver development and debugging
  - Identify protocol errors
  - Compare expected data with actual data
  - Analyze speed of hardware resources

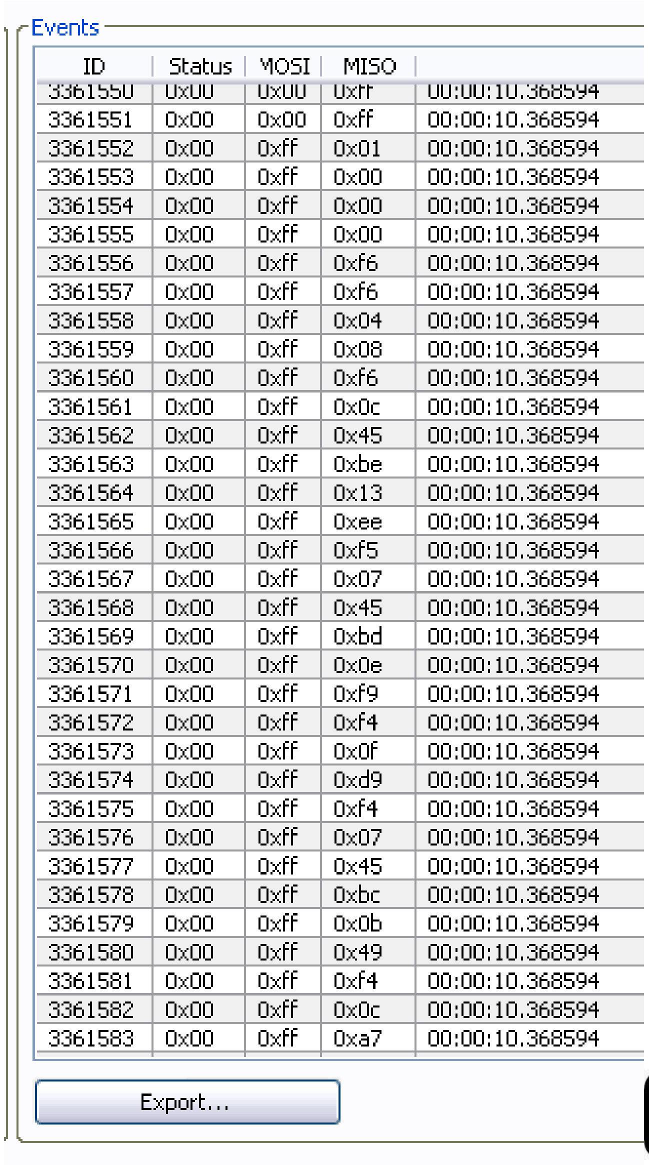


Active Analysis

- Set up and connect the system with host board
- Plug in device-in question
- Generate commands and data with software
- View, analyze and interpret output
- Use for debugging and acceptance testing
  - Check behavior of DUT against specifications
  - Verify performance under normal operating conditions
  - Protect system from malfunctioning device

Example: SPI device traffic

- **Which devices are these events to/from?**
  - **NIC, BIOS, memory**
- **Slow Read or Fast Read?**
- **User Interface will**
  - **Interpret SPI protocol**
  - **Establish filters to show relevant data**
  - **Ex: ignore BIOS events**

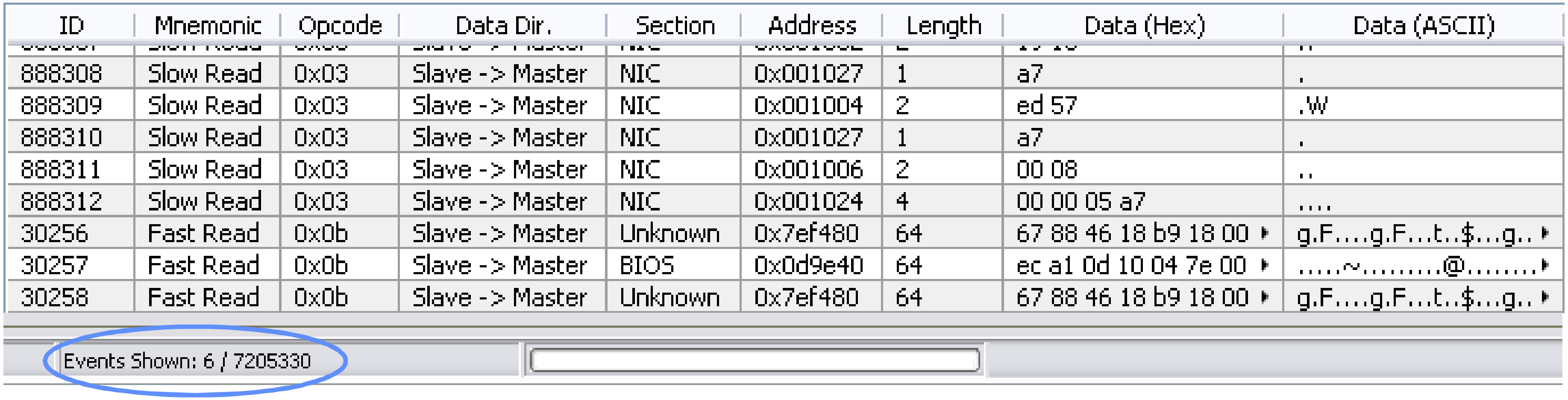


ID	Status	MOSI	MISO	
3361550	0x00	0x00	0xff	00:00:10.368594
3361551	0x00	0x00	0xff	00:00:10.368594
3361552	0x00	0xff	0x01	00:00:10.368594
3361553	0x00	0xff	0x00	00:00:10.368594
3361554	0x00	0xff	0x00	00:00:10.368594
3361555	0x00	0xff	0x00	00:00:10.368594
3361556	0x00	0xff	0xf6	00:00:10.368594
3361557	0x00	0xff	0xf6	00:00:10.368594
3361558	0x00	0xff	0x04	00:00:10.368594
3361559	0x00	0xff	0x08	00:00:10.368594
3361560	0x00	0xff	0xf6	00:00:10.368594
3361561	0x00	0xff	0x0c	00:00:10.368594
3361562	0x00	0xff	0x45	00:00:10.368594
3361563	0x00	0xff	0xbe	00:00:10.368594
3361564	0x00	0xff	0x13	00:00:10.368594
3361565	0x00	0xff	0xee	00:00:10.368594
3361566	0x00	0xff	0xf5	00:00:10.368594
3361567	0x00	0xff	0x07	00:00:10.368594
3361568	0x00	0xff	0x45	00:00:10.368594
3361569	0x00	0xff	0xbd	00:00:10.368594
3361570	0x00	0xff	0x0e	00:00:10.368594
3361571	0x00	0xff	0xf9	00:00:10.368594
3361572	0x00	0xff	0xf4	00:00:10.368594
3361573	0x00	0xff	0x0f	00:00:10.368594
3361574	0x00	0xff	0xd9	00:00:10.368594
3361575	0x00	0xff	0xf4	00:00:10.368594
3361576	0x00	0xff	0x07	00:00:10.368594
3361577	0x00	0xff	0x45	00:00:10.368594
3361578	0x00	0xff	0xbc	00:00:10.368594
3361579	0x00	0xff	0x0b	00:00:10.368594
3361580	0x00	0xff	0x49	00:00:10.368594
3361581	0x00	0xff	0xf4	00:00:10.368594
3361582	0x00	0xff	0x0c	00:00:10.368594
3361583	0x00	0xff	0xa7	00:00:10.368594

Sandia National Laboratories

Example: SPI device traffic

- **SPI protocol parsing provides better interpretation**



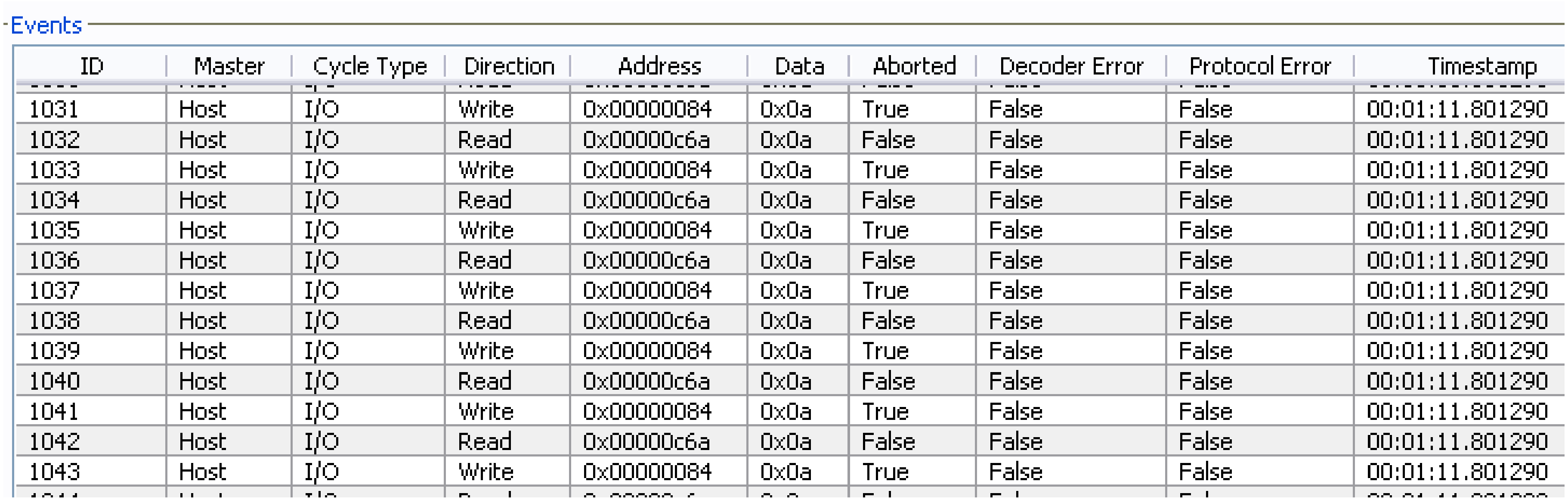
ID	Mnemonic	Opcode	Data Dir.	Section	Address	Length	Data (Hex)	Data (ASCII)
888308	Slow Read	0x03	Slave -> Master	NIC	0x001027	1	a7	.
888309	Slow Read	0x03	Slave -> Master	NIC	0x001004	2	ed 57	.W
888310	Slow Read	0x03	Slave -> Master	NIC	0x001027	1	a7	.
888311	Slow Read	0x03	Slave -> Master	NIC	0x001006	2	00 08	..
888312	Slow Read	0x03	Slave -> Master	NIC	0x001024	4	00 00 05 a7	....
30256	Fast Read	0x0b	Slave -> Master	Unknown	0x7ef480	64	67 88 46 18 b9 18 00 ▸	g.F....g.F...t..$...g.. ▸
30257	Fast Read	0x0b	Slave -> Master	BIOS	0x0d9e40	64	ec a1 0d 10 04 7e 00 ▸	.....~.........@........ ▸
30258	Fast Read	0x0b	Slave -> Master	Unknown	0x7ef480	64	67 88 46 18 b9 18 00 ▸	g.F....g.F...t..$...g.. ▸

Events Shown: 6 / 7205330

Over 7.2 million SPI events captured

Example: LPC Passive Analysis

- During computer boot, large volume of data
- Parse, filter, analyze protocols
- Verify assumptions about resource usage
- This data still needs interpretation

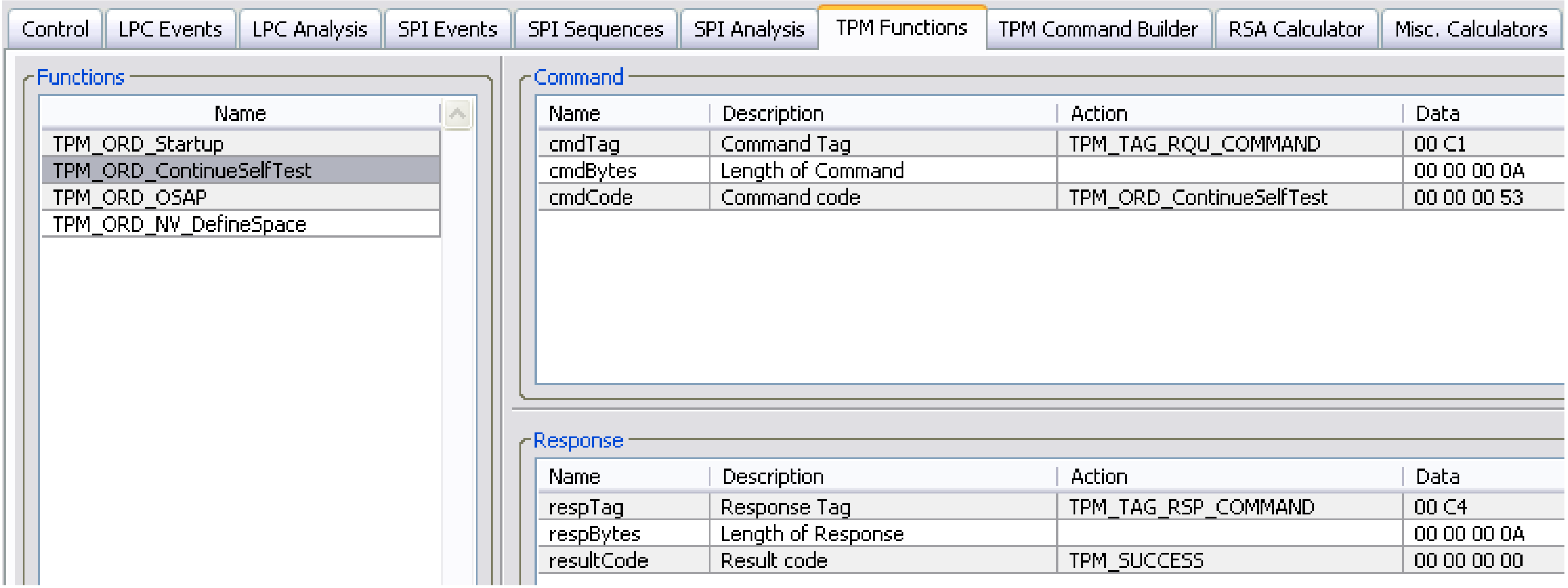


Events

ID	Master	Cycle Type	Direction	Address	Data	Aborted	Decoder Error	Protocol Error	Timestamp
1031	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1032	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1033	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1034	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1035	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1036	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1037	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1038	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1039	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1040	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1041	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290
1042	Host	I/O	Read	0x00000c6a	0x0a	False	False	False	00:01:11.801290
1043	Host	I/O	Write	0x00000084	0x0a	True	False	False	00:01:11.801290

Example: LPC Active Analysis

- Active Analysis of an LPC device
- Generate commands to send to ‘loose’ device
- Proof of Concept: communication with Trusted Platform Module (TPM)



Functions

Name
TPM_ORD_Startup
TPM_ORD_ContinueSelfTest
TPM_ORD_OSAP
TPM_ORD_NV_DefineSpace

Command

Name	Description	Action	Data
cmdTag	Command Tag	TPM_TAG_RQU_COMMAND	00 C1
cmdBytes	Length of Command		00 00 00 0A
cmdCode	Command code	TPM_ORD_ContinueSelfTest	00 00 00 53

Response

Name	Description	Action	Data
respTag	Response Tag	TPM_TAG_RSP_COMMAND	00 C4
respBytes	Length of Response		00 00 00 0A
resultCode	Result code	TPM_SUCCESS	00 00 00 00

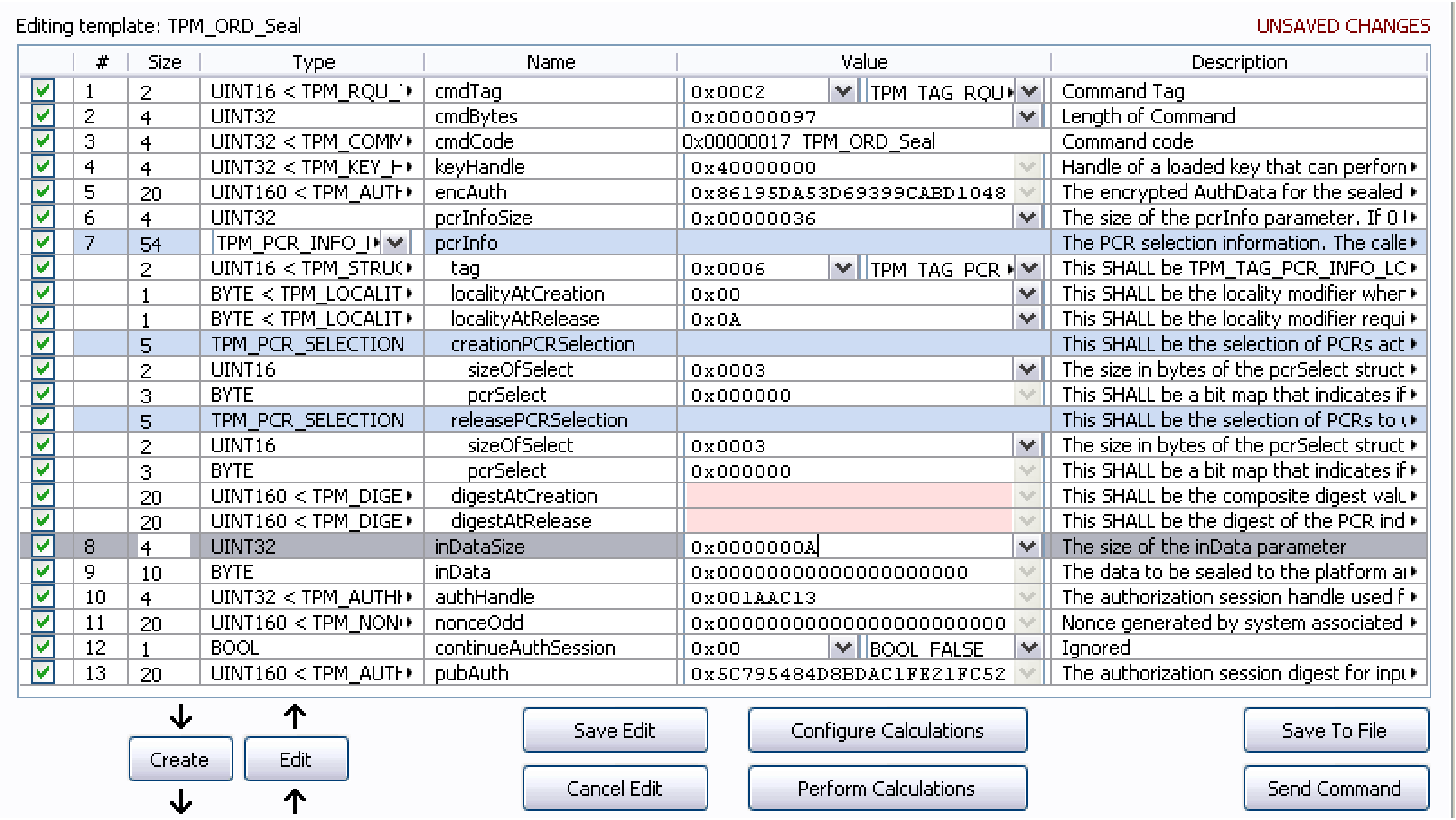
Example: LPC device traffic

- **Trusted Platform Modules**
  - **Monitor how the TPM is being used by software**
  - **Exercise devices to verify specification compliance (e.g. use of deprecated commands)**
  - **Observe how drivers use the device**
  - **Help develop custom security drivers.**

- **User interface also provides hash calculator to verify understanding of commands, specifications**

Sandia National Laboratories

Building a TPM Seal Command



Editing template: TPM_ORD_Seal

UNSAVED CHANGES

	#	Size	Type	Name	Value	Description
☑	1	2	UINT16 < TPM_RQU_▸	cmdTag	0x00C2 TPM TAG RQU▸	Command Tag
☑	2	4	UINT32	cmdBytes	0x00000097	Length of Command
☑	3	4	UINT32 < TPM_COMM▸	cmdCode	0x00000017 TPM_ORD_Seal	Command code
☑	4	4	UINT32 < TPM_KEY_H▸	keyHandle	0x40000000	Handle of a loaded key that can perforn▸
☑	5	20	UINT160 < TPM_AUTH▸	encAuth	0x86195DA53D69399CABD1048	The encrypted AuthData for the sealed ▸
☑	6	4	UINT32	pcrInfoSize	0x00000036	The size of the pcrInfo parameter. If 0 ▸
☑	7	54	TPM_PCR_INFO_I▸	pcrInfo		The PCR selection information. The calle▸
☑		2	UINT16 < TPM_STRUC▸	tag	0x0006 TPM TAG PCR ▸	This SHALL be TPM_TAG_PCR_INFO_LC▸
☑		1	BYTE < TPM_LOCALIT▸	localityAtCreation	0x00	This SHALL be the locality modifier wher▸
☑		1	BYTE < TPM_LOCALIT▸	localityAtRelease	0x0A	This SHALL be the locality modifier requi▸
☑		5	TPM_PCR_SELECTION	creationPCRSelection		This SHALL be the selection of PCRs act▸
☑		2	UINT16	sizeOfSelect	0x0003	The size in bytes of the pcrSelect struct▸
☑		3	BYTE	pcrSelect	0x000000	This SHALL be a bit map that indicates if▸
☑		5	TPM_PCR_SELECTION	releasePCRSelection		This SHALL be the selection of PCRs to ▸
☑		2	UINT16	sizeOfSelect	0x0003	The size in bytes of the pcrSelect struct▸
☑		3	BYTE	pcrSelect	0x000000	This SHALL be a bit map that indicates if▸
☑		20	UINT160 < TPM_DIGE▸	digestAtCreation		This SHALL be the composite digest valu▸
☑		20	UINT160 < TPM_DIGE▸	digestAtRelease		This SHALL be the digest of the PCR ind▸
☑	8	4	UINT32	inDataSize	0x0000000A	The size of the inData parameter
☑	9	10	BYTE	inData	0x00000000000000000000	The data to be sealed to the platform ar▸
☑	10	4	UINT32 < TPM_AUTHH▸	authHandle	0x001AAC13	The authorization session handle used f▸
☑	11	20	UINT160 < TPM_NON▸	nonceOdd	0x00000000000000000000000	Nonce generated by system associated ▸
☑	12	1	BOOL	continueAuthSession	0x00 BOOL FALSE	Ignored
☑	13	20	UINT160 < TPM_AUTH▸	pubAuth	0x5C795484D8BDAC1FE21FC52	The authorization session digest for inpu▸

Conclusions

- Prototype LPC / SPI Bus Analyzer
  - Can be extended to parse/exercise more devices
  - Flash memory, legacy BIOS, keyboard & mouse, USB, Ethernet, etc
- Passive Analysis
  - Debugging
  - Identifying available functionality
  - Analysis of use of system resources
- Active Analysis
  - Acceptance testing and verification
  - Isolation of a malfunctioning part
  - Carefully targeted debugging

Questions / Discussion