A Roll Down the Lane
Measuring Bowling Ball Dynamics from the Inside

A Master's Project and Paper in Engineering Science

by

Donald J. Hake II

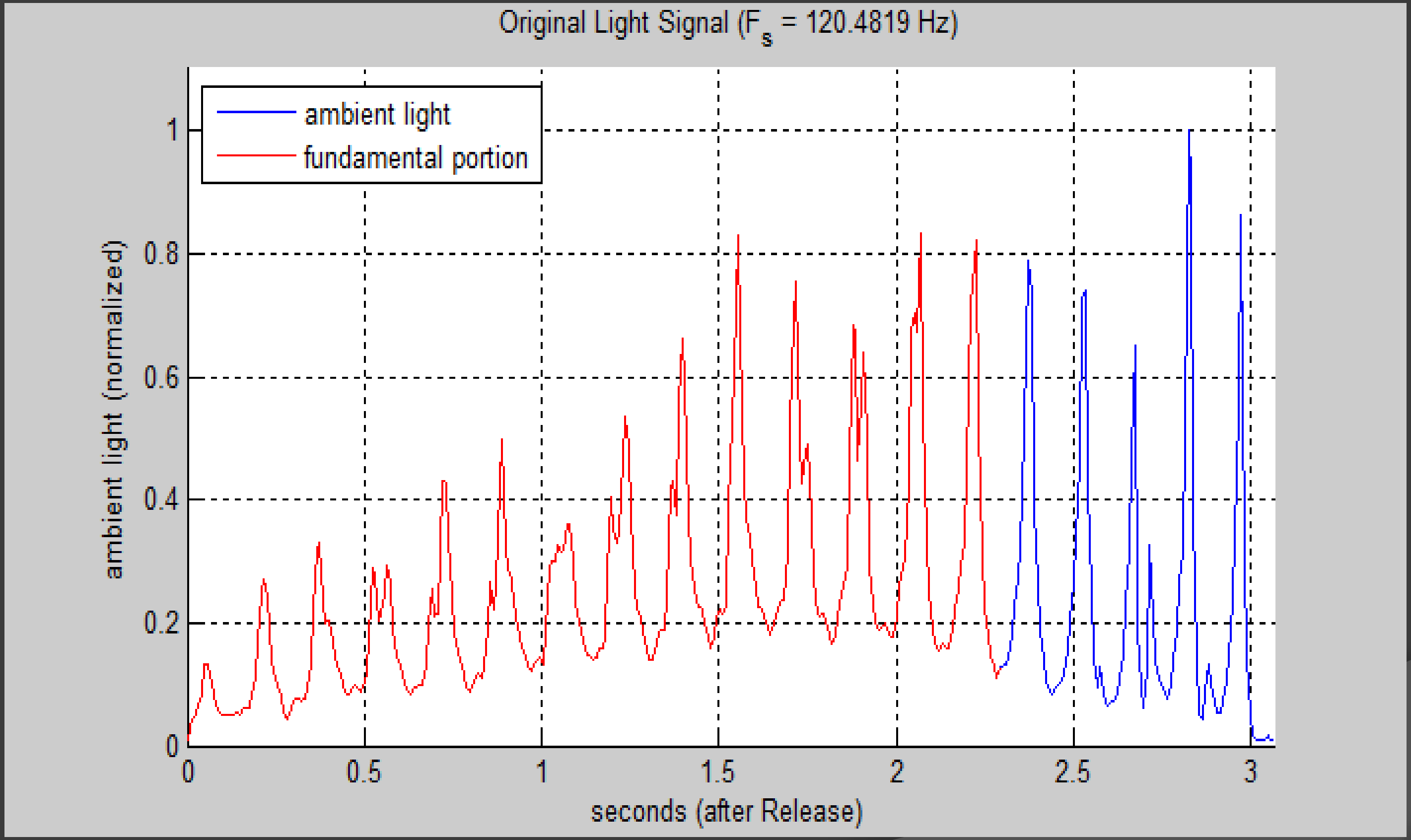
October 3, 2014

RevMetrix Performance Analysis System

- Project based upon previous research conducted and presented for my MSCS degree in 2002 w/Dr. Null
- *SmartDot* system: Sensor module placed in finger hole of bowling ball
- Recorded ambient light data as ball rolled down lane (rotating) under overhead lighting
- Used FIR filtering to isolate sinusoidal waveform related to rotation of ball
- Located revolutions from peaks and valleys
- Calculated angular (RPMs) and linear (MPH) velocities

RevMetrix Performance Analysis System

Typical *SmartDot* module ambient light waveform



RevMetrix Performance Analysis System

- *RevMetrix* system expands upon original *SmartDot* system to develop a practical accelerometer-based bowling performance analysis system that acquires the "internal" perspective of the dynamics of the ball
- Proposed using accelerometer in 2002 paper
- Composed and submitted project proposal to Dr. Wolpert in 2007 - 2008
- Development started in 2008
- First real world data collection in 2010
- Waveform filtering and analysis in MATLAB in 2010 – 2012 as part of EE 453, EE 551 and EE553 w/Dr. Morales

RevMetrix Performance Analysis System

- A dozen years after *SmartDot* project, an inexpensive, portable system that objectively quantifies a bowler's execution and performance still does not exist
- Existing systems (CATS, Brunswick's "Throbot", USBC's E.A.R.L. bowling robot) all rely on expensive, non-portable instrumentation affixed to the lane to observe the ball externally
- http://www.youtube.com/watch?v=s8yMFdPD68c
- http://www.youtube.com/watch?v=QEeLNxlKRrU

RevMetrix Performance Analysis System

- Consists of three components:
  - *SenseModule* (*SM*): in-situ sensor module
  - *ComModule* (*CM*): IR interface between *SenseModule* and *RevMetrixApp*
  - *RevMetrixApp* (*RMApp*): archival, analysis, and presentation software application running on smartphone, tablet, or PC (MATLAB and MS Excel were used for analysis purposes on this project)
- Ultimately, Bluetooth connection might be possible directly to *RevMetrixApp*

MOTIVATION

Motivation for such a system

- *RevMetrix* system fills several basic needs:
  - Records and quantifies release parameters:
    - Ball speed – initial linear velocity
    - Loft – distance ball travels before impacting lane
    - Lift – initial angular velocity
    - Tilt – angle between axis of rotation and lane surface
  - Correlates release parameters to subsequent ball reaction (how linear and angular velocities, and axis tilt change from release to impact)
  - Allows comparison across multiple frames, games, lane conditions, bowling balls, etc.

Motivation for such a system

- Success (higher scores) requires identifying and maintaining a consistent “line” to strike pocket
- Lane conditions (lane oil distribution) can vary significantly between bowling balls, frames, games, adjacent lanes, bowling establishments
- Bowlers compensate by:
  - Altering how they release ball, which increases/decreases amount of hook
  - Altering release location relative to foul line, allowing more or less room for ball to hook, and/or causes ball to encounter more or less lane oil on path to pins
  - Altering angle of release relative to foul line – angling ball out toward gutter, or in towards pins, for same reasons as above

Motivation for such a system

- Difficult to assess whether changing ball reaction is due to changing oil pattern or inconsistent delivery/release
- No convenient, widely available method to assess consistency of delivery/release
- Reviewing video is slow, tedious, imprecise, and does not facilitate comparison of multiple shots

RevMetrix Performance Analysis System

project Scope and Development

RevMetrix Performance Analysis System

Project Scope

- *SenseModule* and *ComModule* hardware design and development
- Embedded software design and development, evolving from manual raw data collection platform to fully autonomous operation
- Raw data analysis and filtering
- Automated algorithms for segmenting and componentizing waveforms and extracting metrics useful to bowler

RevMetrix Performance Analysis System

Project Scope

- *SenseModule* captures data:
  - Collects, stores, times stamps sensor data (ambient light, 3-axis acceleration readings)
  - Contains internal DB to store multiple ball records before upload is required
  - Uploads data to *RevMetrixApp* (through IR software UART to *ComModule)*
- *RevMetrixApp* is the “brains”
  - Archives data from *SenseModule*
  - Analyzes, calculates, and presents results
  - Analysis presented here used MS Excel and MATLAB for analysis and presentation purposes

RevMetrix Development

- Phase 1: Bench-Top *SenseModule* Breadboard
  - *SenseModule* and *ComModule* developed using F930DK development kit from Silicon Labs
  - Included Silicon Labs IDE, Keil's assembler and C51 compiler
  - Breadboard prototypes used to develop basic architecture and functionality, Ball Record DB, communications

RevMetrix Development

- Phase 2: Prototype Data Collection
  - 3-axis acceleration had not been collected from within ball
  - Real-world data had to be collected first before autonomous operation could be achieved
  - Schematic, lay out PCBs, order parts and PCBs, contract *SenseModule* prototype assembly
  - Earliest versions operated only in manual, single-shot mode
  - Lots of “basement” bowling to characterize release waveform, refine embedded software

RevMetrix Development

- Phase 3: *SenseModule* Autonomous Operation
  - Fully automated functionality evolved iteratively
  - Automated release detection was developed first
  - Automated shutdown detection next
  - False activation and false release detection last
  - Refinement of those functions is on-going
  - Additional data from broad range of bowling styles must be collected in order to develop truly robust automated operation

RevMetrix Development

- Phase 4: Raw Data Waveform Analysis
  - Initial visual analysis with Excel and raw data CSV files
  - Import raw data to MATLAB programs for display and manipulation
  - Characterize spectrum content of waveforms using FFTs
  - Experiment with wavelets due to discontinuous nature of waveform
  - Develop hybrid approach for segmenting waveform and isolating acceleration components

RevMetrix Development

- Phase 5: Bowling Metrics Extraction
  - Develop automated algorithms in MATLAB to isolate and filter acceleration components
  - Develop additional automated algorithms in MATLAB to analyze those component waveforms and extract metrics meaningful to bowler

Presentation Scope

- So – Development essentially consisted of 5 projects
- Any of which could fill a 90 minute presentation
- 200 slides is a bit too much
- So – I'll have to skim certain aspects to focus on others
- And leave room for questions

SenseModule Design

RevMetrix SenseModule

Basic Requirements

- Module must meet the following requirements
  - Unobtrusive – Bowler cannot detect physical presence of sensor module
  - “Transparent” – Operation cannot interfere with bowler’s normal routine
  - Small – Fits in an existing finger/thumb hole
  - Light Weight - Cannot appreciably affect static or dynamic balance of ball
  - Inexpensive – Substantially less than cost of a bowling ball
  - Low Power – Small battery, infrequent replacement
  - Convenient – Uploads required no more frequently than once per game

Hardware Design

RevMetrix SenseModule

Design Constraints

- Designed with commercialization in mind
- Consumer electronics design imposes challenging constraints:
  - Low-cost means simple design
  - Hardware costs money, use the μP to do the work in software
  - Consolidate as much functionality as possible into as few components as possible

RevMetrix SenseModule

Sensing Requirements

- Detect start-up condition (finger placed in insert)
- Detect translational motion (3-axis)
- Detect rotational motion (3-axis)
- Detect release (finger removed from insert)
- Detect impacts with lane and pins
- Detect shutdown condition
- Track accurate passage of time
- Detect and communicate with *ComModule*
- Discriminate between true activation events, communication events, spurious activations due to pinsetter, ball return, etc

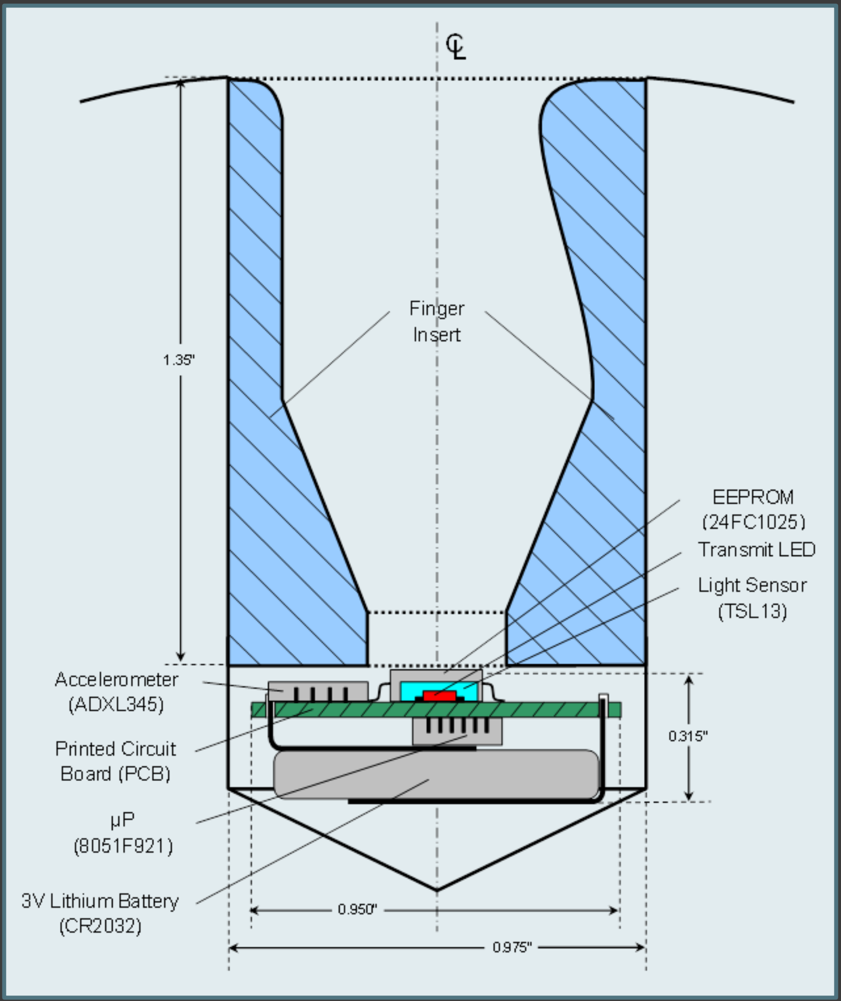
RevMetrix SenseModule

Design Constraints

- Located in finger hole, under finger insert
  - No additional hole required
  - Protects module from surface impacts
  - Module can detect presence of finger
  - User can install module and replace battery
- Size: diameter <= 0.950”, height <= 0.375”
- Weight: < 1/4 oz (≤ 7 gm)
- Battery: 1 year cycle (250-500 games)
- Cost: < $50 MSRP, < $15 MFG

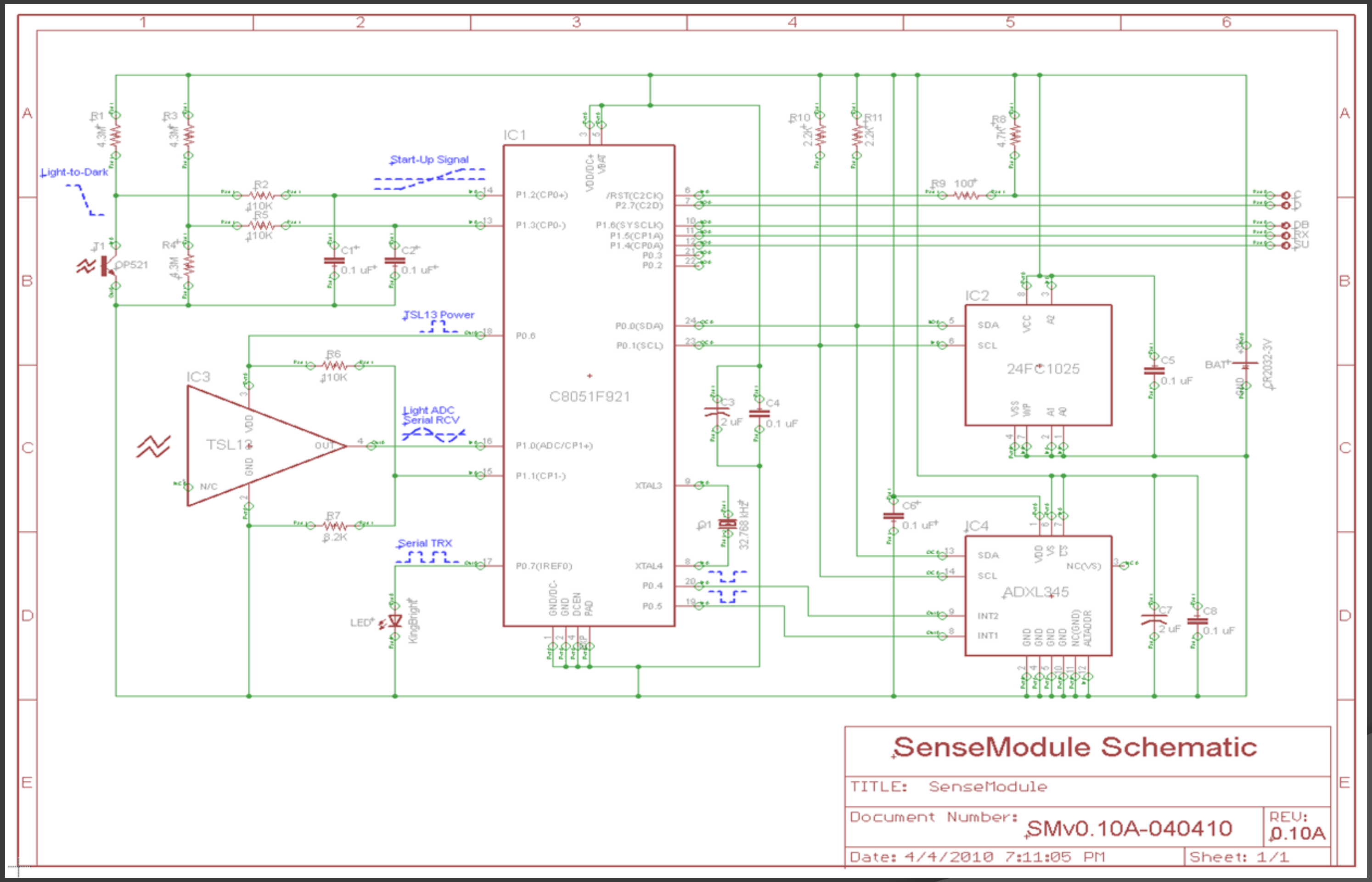
RevMetrix SenseModule

Installed Cut-Away View



- Size: 0.950” D x 0.315” H
- Weight: 3-7 grams (< 1/4 oz), depending on battery size
- µP: SiLabs 8051F921 system-on-a-chip
- Light Sensor: AMS TSL13T light-to-voltage converter
- Accelerometer: Analog Devices ADXL345 +/- 16g 3-axis, I²C interface
- Memory: Microchip 24FC1025 128 KB I²C EEPROM
- Battery: 3V lithium coin cell (CR2016 to CR2032)
- Cost: $13.50 – $15.00 (1,000s)

RevMetrix SenseModule Schematic



Start-Up Circuit

- Simple start-up circuit
  - Always powered, must be micro-power
  - Ambient light - transition from light to dark
  - Input to micro-power comparator (CP0) on μP
  - μP always powered, but in micro-power SleepMode
  - CP0 transition wakes μP from SleepMode
  - μP puts self back into SleepMode at conclusion of processing loop

RevMetrix SenseModule

Microprocessor

- Silicon Labs 8051F921 μP
  - System-on-a-chip - mixed analog and digital functions
  - Intel 8051 compatible
  - On-board 24.5 MHz system clock, configurable down to 3.05 MHz
  - Crossbar switch – assign on-board functions to port pins
  - In-system debugging capability
  - 32 kbytes in-system writeable program memory
  - 256 bytes of RAM, includes registers and stack space
  - 4096 bytes on-board XRAM – Light and ADXL circular buffers

RevMetrix SenseModule

Microprocessor

- Silicon Labs 8051F921 μP
  - Multiple timer/counters – light sample timer, I^2C timer
  - I^2C byte-wide bus – accelerometer and EEPROM
  - Micro-power comparators
    - CP0 – start-up circuit
    - CP1 – software IR iRTZ UART
  - Micro-power SleepMode, wake from CP0 interrupt
  - 12-bit 300 kHz ADC
    - Ambient light sampling
  - smaRTClock – built in micro-power RTC function
    - Accurate sample time-stamping
    - Tracks time and date of use
    - Just needs 32.768 kHz watch crystal

RevMetrix SenseModule

Ambient Light Sensor

- AMS TSL13 Light-to-Voltage Converter
  - Carryover from original *Smartdot* module
  - Ambient light sensing
  - Release detection
  - IR receiver
  - Overkill for this application
    - Included for comparison purposes w/Smartdot waveforms
    - Will be replaced with existing photodiode

RevMetrix SenseModule

Accelerometer

- Analog Devices ADXL345 Accelerometer
  - 3-axis acceleration
  - 3-axis tilt sensing (orientation to gravity)
  - ± 16 g range
  - 13-bit (4 mg) resolution
  - Always powered – micro-power standby mode
  - 200 μAmp operation current
  - Autonomous operation – self-clocked
  - Selectable sample frequency – 200 Hz for *SenseModule*
  - I^2C interface
  - 32 sample FIFO
  - 2 configurable interrupt pins (activity, inactivity, free fall)

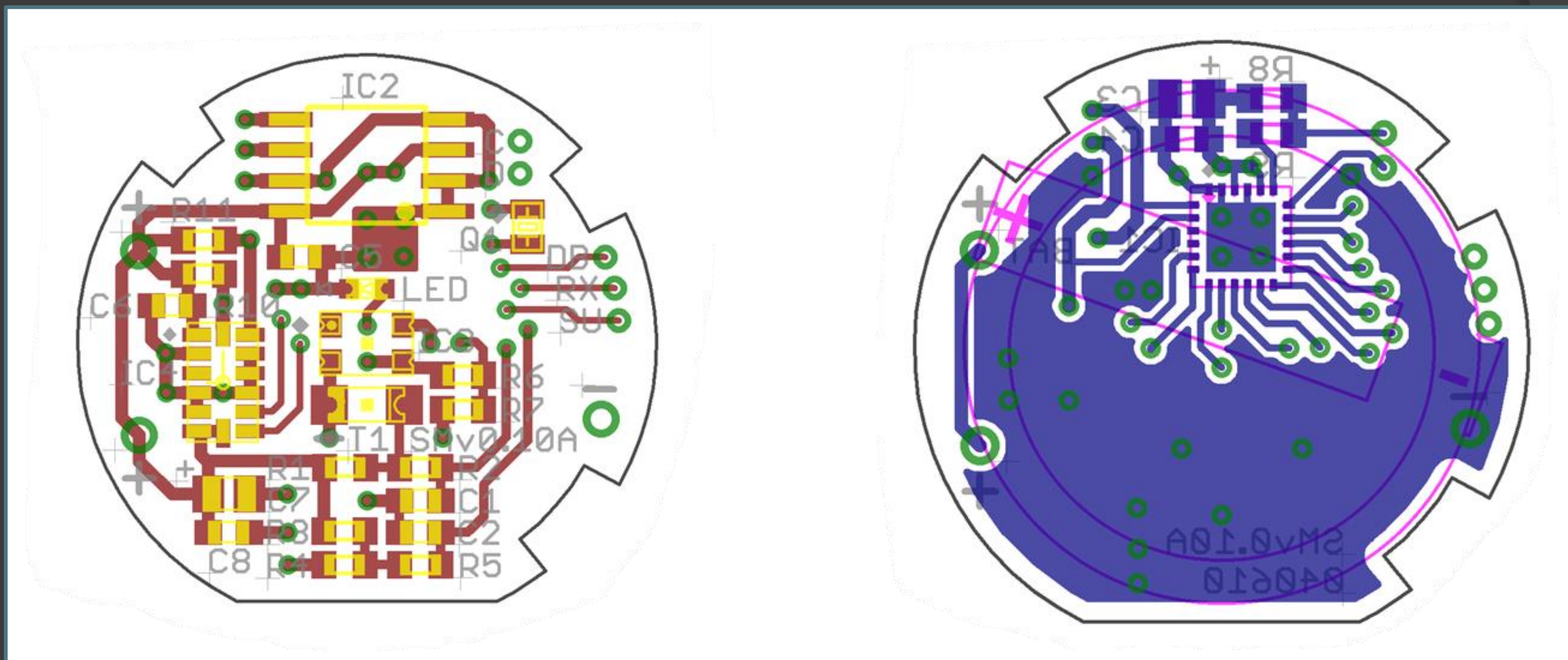
RevMetrix SenseModule

Non-Volatile Memory

- Microchip 24FC1025 Serial EEPROM
  - EEPROM provides extended non-volatile long-term storage for *SenseModule* configuration parameters and Ball Record database
  - Always-powered, nano-power standby mode
  - I^2C interface
  - 128 kbytes, configured as 1024 128-byte pages
  - Page read/write mode – up to 128 bytes at a time
  - 5 ms write cycle @ 3 mA per page
  - Self-timed write, μP doesn't wait for write cycle
  - 1,000,000 write cycles

RevMetrix SenseModule

PCB Layout

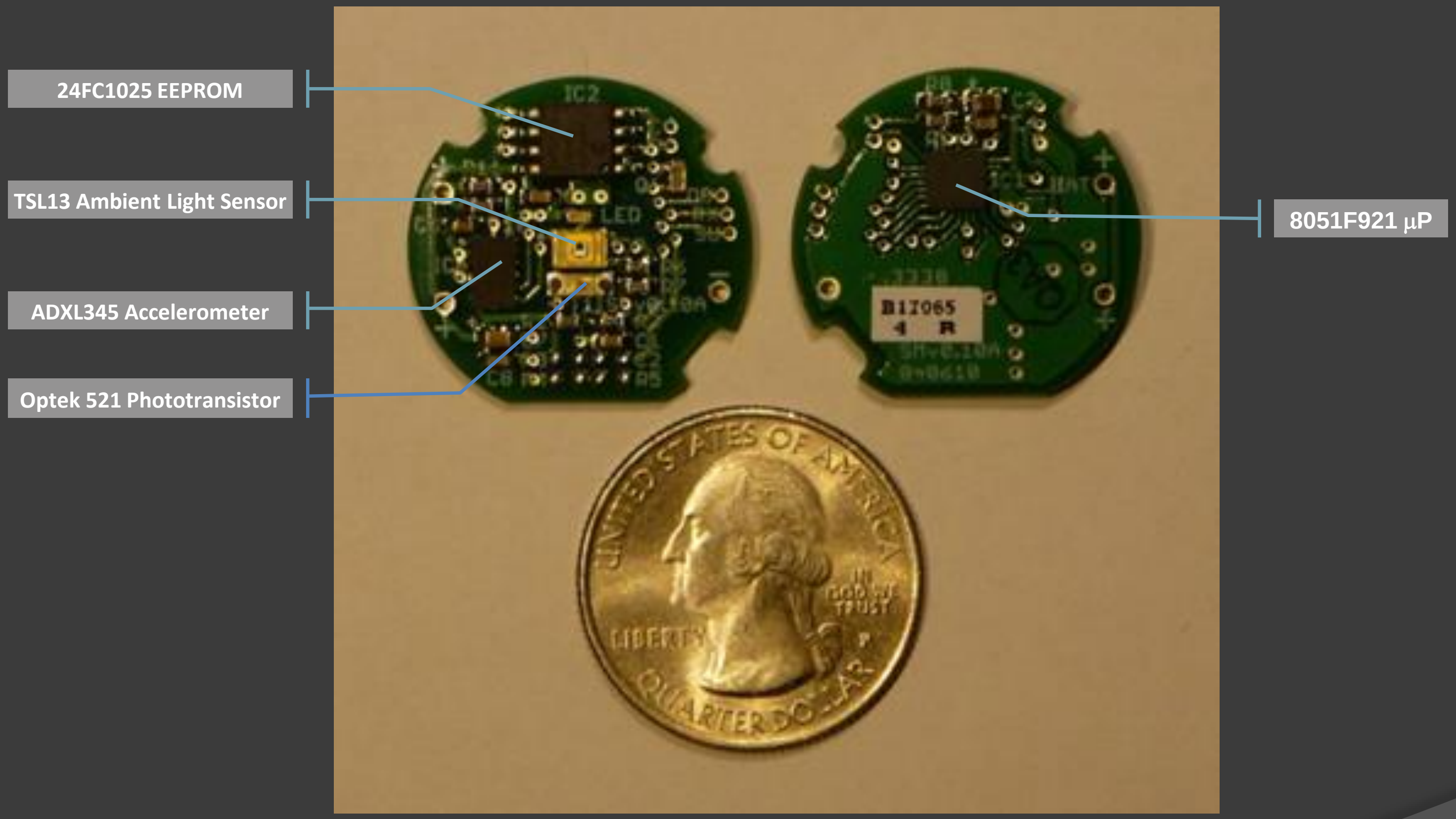


Top

Bottom

RevMetrix SenseModule

PCB Assembly



SenseModule

Hardware Performance

SenseModule Hardware Performance

Physical Constraints

- Transparent – Completely unobtrusive, fully automatic operation achieved
- Small and Light Weight:
  - 0.315” height, 0.185 oz (5.25 gm), as built (CR2032 battery)
  - 0.220” height, 0.088 oz (2.50 gm), w/0.031” PCB and CR2016

SenseModule	As Built	As Built w/CR2032 225 mAh Battery	0.080 mm PCB w/CR2032 225 mAh Battery	0.080 mm PCB w/CR2016 90 mAh Battery
Diameter	24.2 mm (0.951”)	-same -	-same -	- same -
Height	4.8 mm (0.190”)	8.0 mm (0.315”)	7.2 mm (0.285”)	5.6 mm (0.220”)
Weight	2.00 gm (0.071 oz)	5.25 gm (0.185 oz)	4.25 gm (0.150 oz)	2.50 gm (0.088 oz)

- Low Cost – component cost under $15 (1,000s)
  - Does not include plastic case to hold module and battery
  - Well under $1, excluding NRE for plastic injection molds
  - Will use 3D printed parts for prototypes

SenseModule Hardware Performance

Physical Constraints

- Low Power:
  - 2.5 µA @ 3V in SleepMode
  - 1.1 mA @ 3V in CommandMode, ApproachMode, SampleMode

SenseModule	Current (ave)	SleepMode	CommandMode (5-10 s)	ApproachMode (10-30 s)	SampleMode (<= 5 s)
Startup Circuit	1.3 µA	X			
CP0	0.5 µA	X			
smaRTClock (RTC)	0.6 µA	X			
8051F921 (µP)	600 µA		X	X	X
TSL13	50 µA		X	X	X
ADXL345 (sample)	180 µA			X	X
ADXL345 (read)	100 µA			X	X
EEPROM (write)	115 µA				X
EEPROM (read)	100 µA		X		
TRX LED (IREF0)	250 µA		X		
Average Current		2.5 µA	1.10 mA	1.03 mA	1.15 mA

- Battery Life:
  - CR2032 (225 mAh): 500 – 1000 games
  - CR2016 (90 mAh): 175 – 350 games (saves height and weight)

SenseModule Hardware Performance

Start-Up Circuit

- Responds to the following events:
  - Bowler's approach activation
  - ComModule activation
- And (unfortunately) these additional events:
  - Pinsetter elevating ball from pit to subway ramp
  - Ball entering subway at pinsetter
  - Ball exiting subway at ball return
  - Ball rolling on ball return
  - Ball placed in bag, locker, car trunk, closet, etc.
- Last 5 are "nuisance" activations, several of which also involve "impacts" and/or rotation
- Much effort placed on detecting false activations and/or release events – chew up battery life, Ball Record DB space
- Inexpensive micro-power proximity sensing circuit possible

RevMetrix SenseModule

Embedded Software

RevMetrix SenseModule
Embedded Software

- Cost savings of minimal hardware translates to greatly increased effort for embedded software
- *SenseModule* must deal with three basic scenarios:
  - Automatically wake-up, collect and record sensor data, go back to sleep
  - Automatically wake-up, detect ComModule, upload sensor data, go back to sleep
  - Detect and reject false activations due to pinsetter, subway, ball return, placing ball in bag, locker, trunk, closet

RevMetrix SenseModule

Embedded Software Use Cases

SenseModule Embedded Software Use Cases

Record Sensor Data

- Bowler picks up ball and places fingers in finger holes
- *SenseModule* wakes up due to transition from light to extended dark period
- *SenseModule* starts recording data to circular sample buffers
- Bowler delivers ball to lane (releases ball)
- *SenseModule* detects release, begins committing data to Ball Record DB (EEPROM)
- Ball hits pins and then falls into pit
- *SenseModule* finishes committing data to Ball Record DB (EEPROM)
- *SenseModule* returns to SleepMode

SenseModule Embedded Software Use Cases

upload Sensor Data

- Bowler picks up ball
- Places *ComModule* over finger hole
- *SenseModule* wakes up due to transition from light to extended dark period
- *SenseModule* attempts to contact *ComModule*
- *ComModule responds to SenseModule* and issues sequence of upload commands
- *SenseModule* uploads requested pages from EEPROM to *ComModule*
- *ComModule* indicates completion of command sequence
- *SenseModule* updates system parameters
- *SenseModule* returns to SleepMode.

SenseModule Embedded Software Use Cases

Reject False Wakeup Conditions

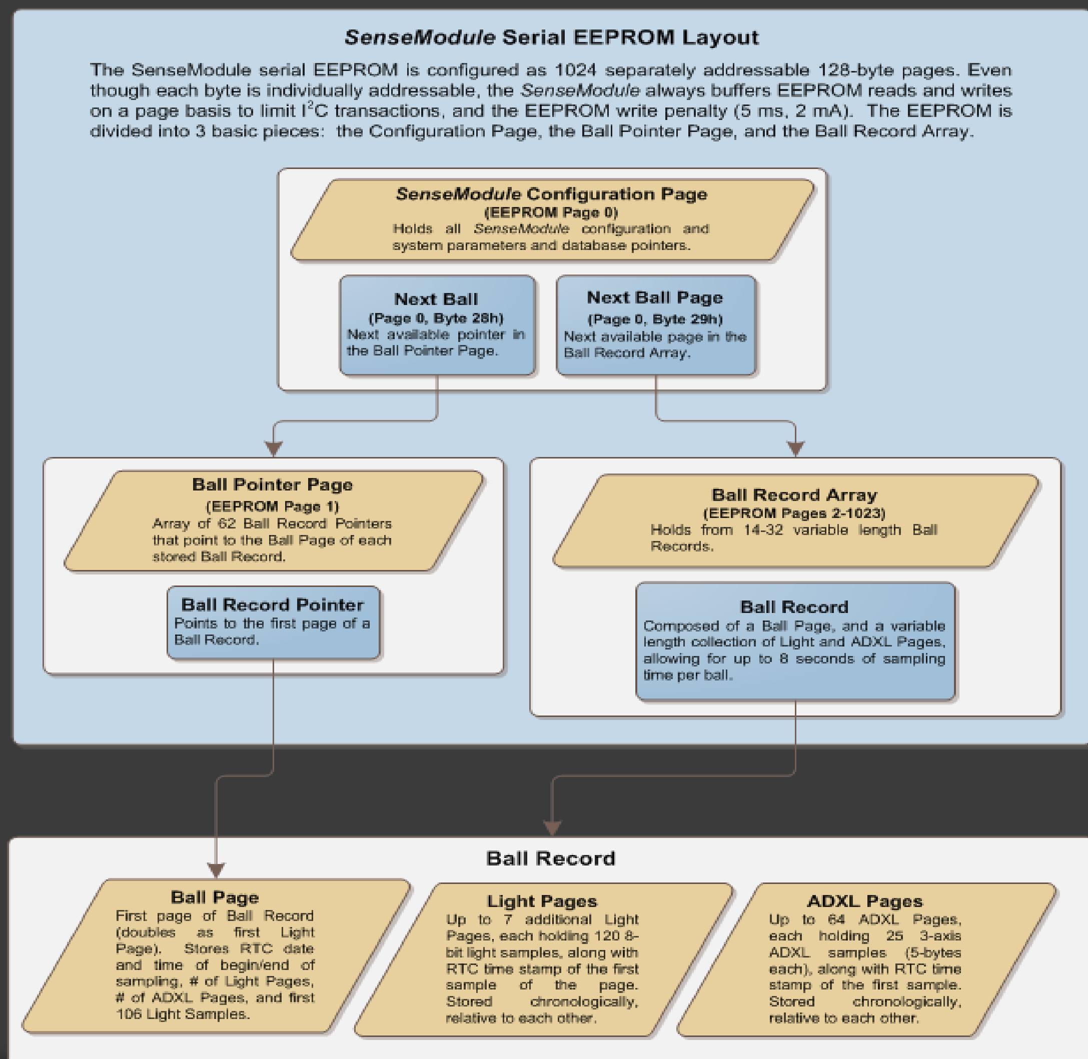
- Ball rolls down lane and falls into pit
- *SenseModule* finishes recording data to EEPROM and returns to SleepMode
- Pinsetter elevates ball to subway booster wheel (dark-to-light transition, impacts, rotation)
- Ball enters subway (light-to-dark transition, rotation)
- Ball rolls in darkness along subway (rotation)
- Ball encounters ball return booster wheel (impacts)
- Ball emerges onto ball return (dark-to-light transition, rotation)
- Multiple light-to-dark and dark-to-light transitions, combined with ball rotation and impacts
- To extend battery life, and conserve limited Ball Record DB entries, *SenseModule* must reject false wake-up conditions

SenseModule Embedded Software

SenseModule Ball Record Database

RevMetrix SenseModule

EEPROM Memory Map



EEPROM Memory Map

Configuration Page (page 0)

- Page Type: 0010 0000b (0x20h)
- Contains configuration, system, and operational parameters
- Unique module ID
- *SenseModule* password
- Bowling Ball name/description
- Embedded SW and DB versions
- Next available Ball Record Pointer
- Next available page in Ball Record database
- Various programmable thresholds, time out values, detection parameters
- CRC protected to detect corruption

EEPROM Memory Map

Ball Pointer Page (Page 1)

- Page Type: 0011 0000b (0x30h)
- Array of 62 Ball Record Pointers
- Ball Record Pointer array also forms circular buffer
- Ball Record Pointers do not point to fixed record locations, but rather to Ball Pages (first page of each Ball Record)
- Ball Records themselves know their own record number (Ball Count)
- CRC protected to detect corruption

Ball Pointer Page (EEPROM Page 1, Address: 0:0080, 128 bytes)			
0	1	2-125	126-127
Page Type	**unused**	**Ball Record Pointer Array (0 - 61)**	**Page CRC**
0x30		see below	16-bits
BYTE	BYTE	WORD[62]	WORD

EEPROM Memory Map

Ball Pointer

- Points to first page (Ball Page) of Ball Record in Ball Record database
- Contains status of Ball Record
  - In Use / Available – In Use, points to valid Ball Record
  - New / Old – New since last upload
  - Deleted – Overwritten before being uploaded

Ball Record Pointer (2 bytes)					
15	14	13	12	11	10 - 0
Ball Status Bits					**Ball Record Pointer**
In Use	**New**	**Deleted**	unused	unused	**EEPROM Page**
1: In Use 0: Available	1: New 0: Old	1: Deleted 0: Not Deleted			2 - 1023
WORD					

EEPROM Memory Map

Ball Record Database (Pages 2-1023)

- Stores variable length Ball Records
- Entire Ball Record database is configured as one large circular buffer
- Ball Records are stored in chronological order, oldest records are aged out (overwritten) by new records
- Each Ball Record starts with a Ball Page, and is followed by a variable number of Light and ADXL Pages
- Status of Ball Records (new, old, deleted) stored in Ball Pointers for each record.

EEPROM Memory Map
Ball Record

- Ball Records are variable length – max 72 sample pages (8 seconds)
  - Ball Page: Ball Record info, doubles as first Light Page
  - Light Pages: 1 to 7 additional Light Pages
  - ADXL Page: 1 to 64 ADXL Pages
  - Light Pages are stored chronologically
  - ADXL Pages are stored chronologically
  - Light and ADXL Pages are not stored in overall chronological order with respect to each other

Ball Record (9216 bytes max: 72 sample pages * 128 byes)	
0	1-71 (max)
Ball Page	**Sample Pages**
Doubles as first Light Page (106 samples, 883 ms)	Mix of 1–7 Light Pages, up to 7 seconds and 1-64 ADXL Pages, up to 8 seconds
1 EEPROM page (128 byes)	Up to 71 EEPROM pages (9088 BYTES max)

EEPROM Memory Map

Ball Page

- First page of Ball Record (doubles as first Light Page)
- Header indicates page type (11xxxxxxb) and Ball Pointer Index
- Bytes 0-21 store info pertinent to entire Ball Record
- Bytes 22-127 store first light samples

Ball Page (128 bytes)									
0	1-4	5-6	7	8-11	12-15	16-19	20	21	22-127
Ball Page Header	**Page Time Stamp**	**Ball Count**	**Sample Count**	**Ball Time Stamp**	**Start Time Stamp**	**End Time Stamp**	**Light Pages**	**ADXL Pages**	**Light Samples Array**
see below	RTC time @ start of page		# of samples stored in page	RTC date @ start of sampling	RTC time @ start of sampling	RTC time @ end of sampling	# of Light pages in Ball Record (1 - 8)	# of ADXL pages in Ball Record (1 - 64)	8-bit Samples 0 – 105 (833 ms)
BYTE	DWORD	WORD	BYTE	DWORD	DWORD	DWORD	BYTE	BYTE	BYTE[106]

Ball Page Header (byte 0 of Ball Page)							
7	6	5	4	3	2	1	0
Page Type Bits		**Ball Record #**					
1	1	0 - 61					
Ball Page Type = 11xxxxxxb		Ball index from Ball Pointer Page					
BYTE							

EEPROM Memory Map

Light Page

- Header indicates page type (10xx xxxxb) and Ball Pointer Index
- Stores RTC time stamp of 1st sample
- Light Pages store 120 light samples (1 second)
- Last Light Page might not be full

Light Page (128 bytes)				
0	1-4	5-6	7	8-127
Light Page Header	**Page Time Stamp**	**Ball Count**	**Sample Count**	**Light Samples Array**
see below	RTC time @ start of page		# of samples stored in page	8-bit Samples 0 – 119 (1 second)
BYTE	DWORD	WORD	BYTE	BYTE[120]

Light Page Header							
7	6	5	4	3	2	1	0
Page Type Bits		**Ball Record #**					
1	0	0 - 61					
Light Page Type = 10xxxxxxb		Ball index from Ball Pointer Page					
BYTE							

EEPROM Memory Map

ADXL Page

- Header indicates page type (01xx xxxxb) and Ball Pointer Index
- ADXL Pages store 25 3-axis accelerometer samples (125 ms)
- Stores low-order word of RTC time stamp of 1st sample

ADXL Page (128 bytes)		
0	1-2	3 - 127
ADXL Page Header	**Page Time Stamp**	**ADXL Samples Array**
see below	RTC time @ start of page (low-order WORD only)	Compressed 13-bit X,Y,Z-axis samples 0 – 24 (125 ms) see below
BYTE	WORD	ADXL Sample[25]

ADXL Page Header							
7	6	5	4	3	2	1	0
Page Type Bits		**Ball Record #**					
0	1	0 – 61					
ADXL Page Type – 01xxxxxxb		Ball index from Ball Pointer Page					
BYTE							

EEPROM Memory Map

ADXL Sample

- ADXL Sample stores compressed 3-axis accelerometer sample
- 3 16-bit readings (13 significant bits) compressed into 5 bytes
- MSB's of each axis are "easily" readable, LSB's are "mangled"

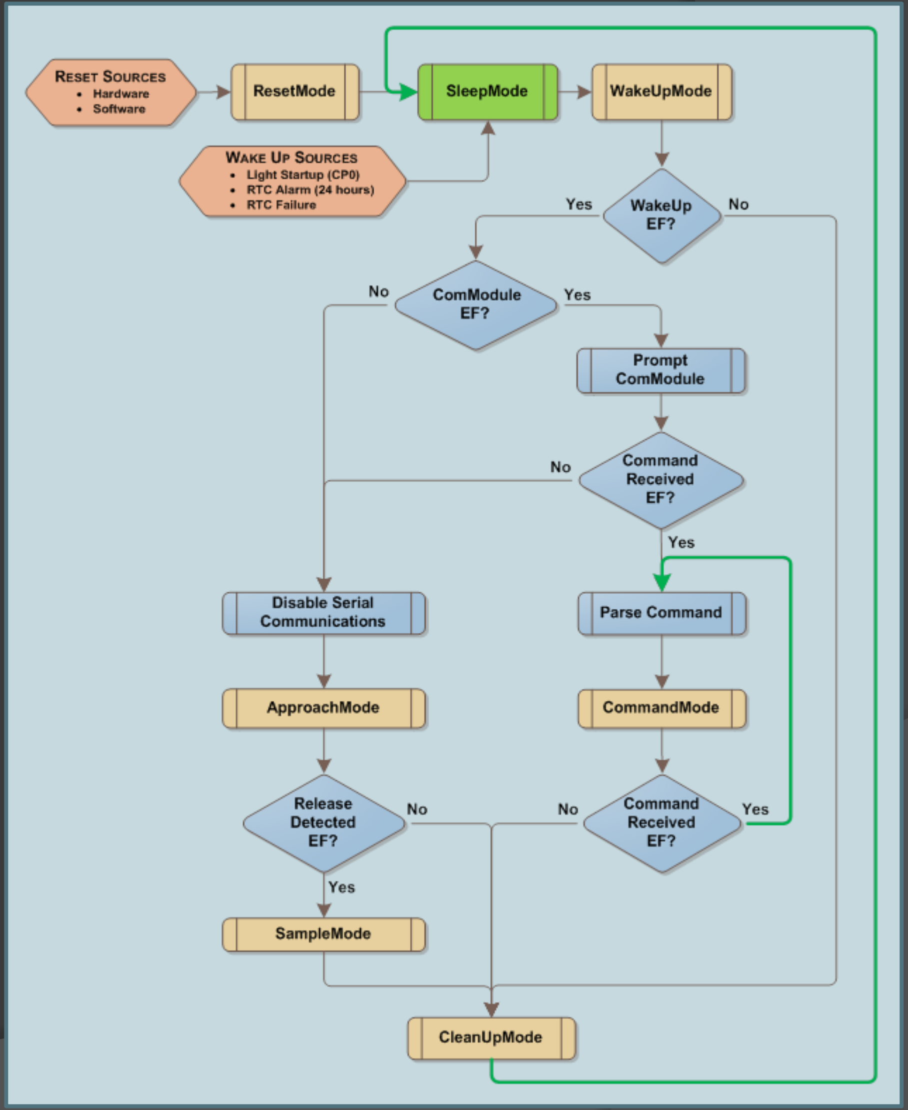
<table>
<tr><th colspan="19">ADXL Sample (5 bytes - compressed)</th></tr>
<tr><td colspan="8">0</td><td>1</td><td colspan="8">2</td><td>3</td><td>4</td></tr>
<tr><td colspan="8">X-axis (LSB), Z-axis (LSB)</td><td>X-axis (MSB)</td><td colspan="8">Y-axis (LSB), Z-axis (LSB)</td><td>Y-axis(MSB)</td><td>Z-axis (MSB)</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td rowspan="2">bits 15-8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td rowspan="2">bits 15-8</td><td rowspan="2">bits 15-8</td></tr>
<tr><td colspan="5">X-axis LSB
bits 7-3</td><td>unused</td><td colspan="2">Z-axis LSB
bits 7-6</td><td colspan="5">Y-axis LSB
bits 7-3</td><td colspan="3">Z-axis LSB
bits 5-3</td></tr>
<tr><td colspan="8">BYTE</td><td>BYTE</td><td colspan="8">BYTE</td><td>BYTE</td><td>BYTE</td></tr>
</table>

Embedded Software Architecture

SenseModule Embedded Software

MainLoop

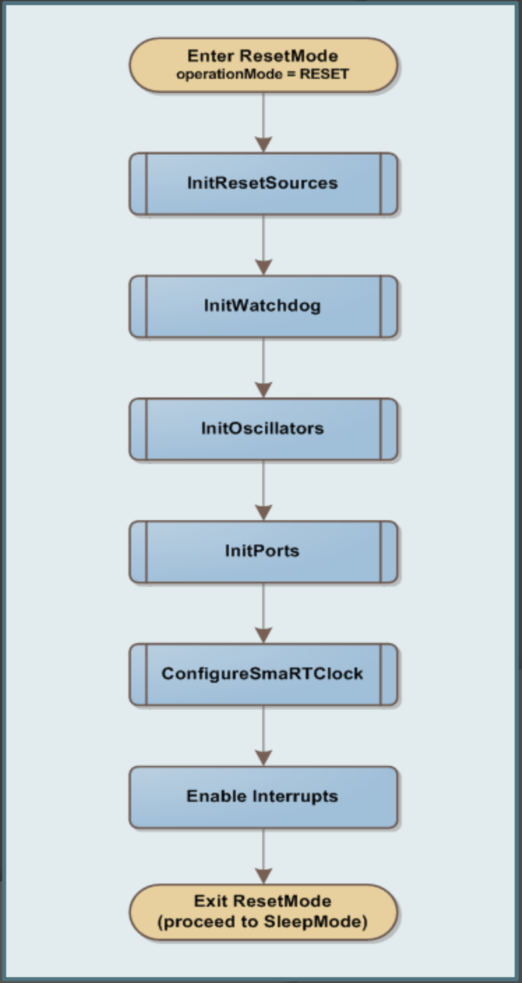
- Super loop architecture
- MainLoop progresses through series of processing modes based on interrupt-driven events
- Events trigger Event Flags (EFs), which determine path through MainLoop and processes
- Several processes are loops:
  - SleepMode
  - CommandMode
  - ApproachMode
  - SampleMode



SenseModule Embedded Software

ResetMode Process

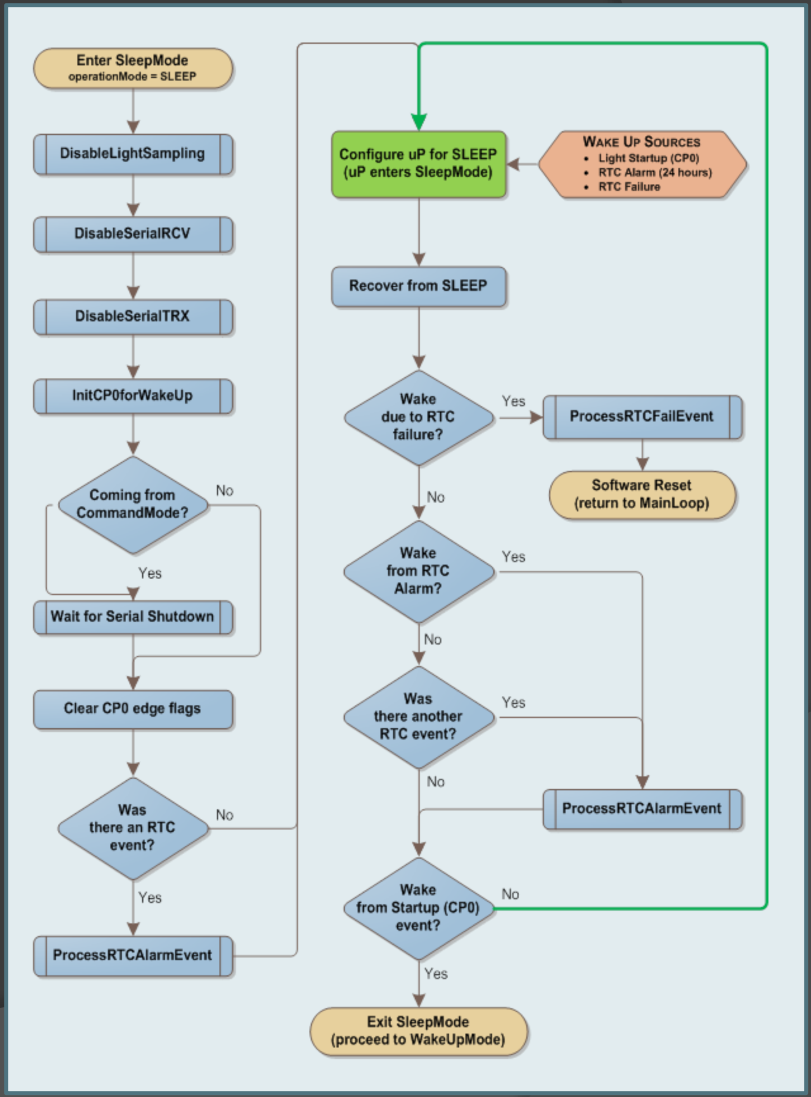
- Triggered by HW or SW events
- HW Events:
  - Loss-of-power (battery replacement)
  - Watchdog time-out
- SW Events (BIT functions):
  - CRC failure on EEPROM
  - RTC failure
  - Unresponsive external peripheral, e.g. ADXL345
- ResetMode reinitializes internal resources (oscillators, RTC, etc…)



SenseModule Embedded Software

SleepMode Process

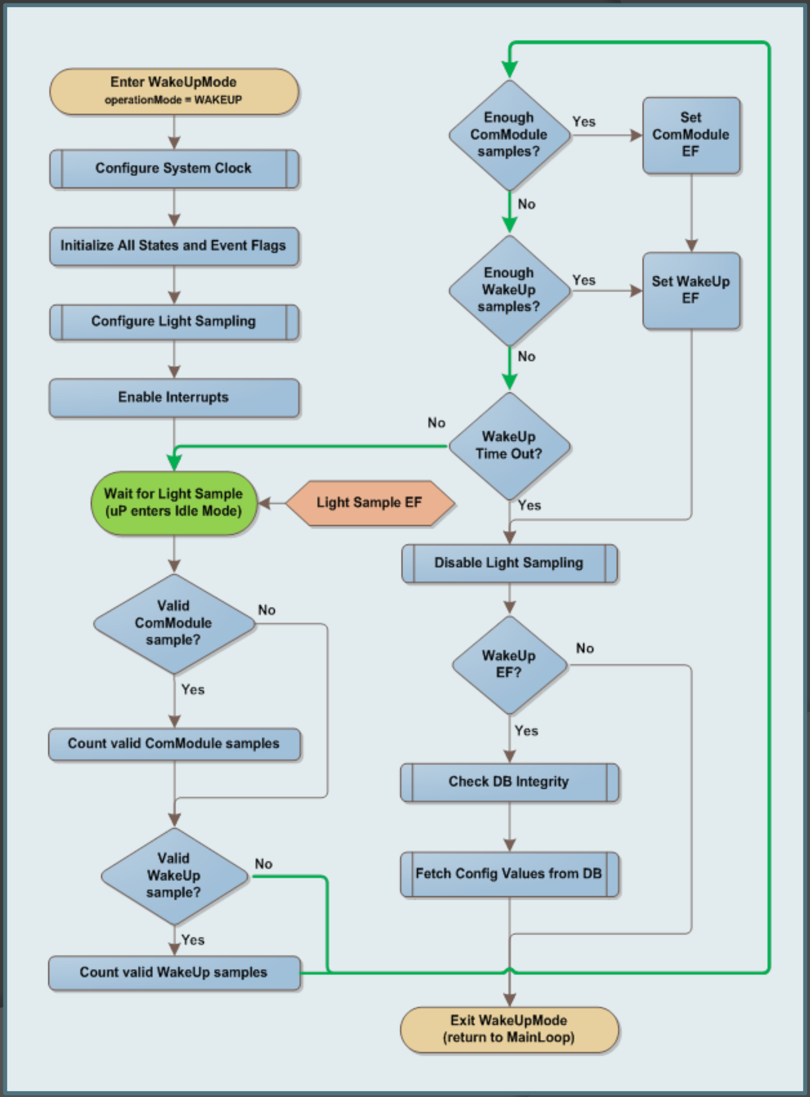
- *SenseModule* spends vast majority of time in SleepMode
- Returns to SleepMode after every iteration through MainLoop
- Performs orderly shutdown of µP HW functions, and external peripherals
- µP enters internal nano-power SleepMode
- While in SleepMode, only two internal functions remain active (both nano-power):
  - RTC – real time function
  - CP0 – startup comparator
- Wakeup events
  - Ambient light start-up at CP0
  - RTC – 24-hour alarm
  - RTC failure
- Returns to SleepMode for RTC events
- Returns to MainLoop for CP0 event



SenseModule Embedded Software

WakeUpMode Process

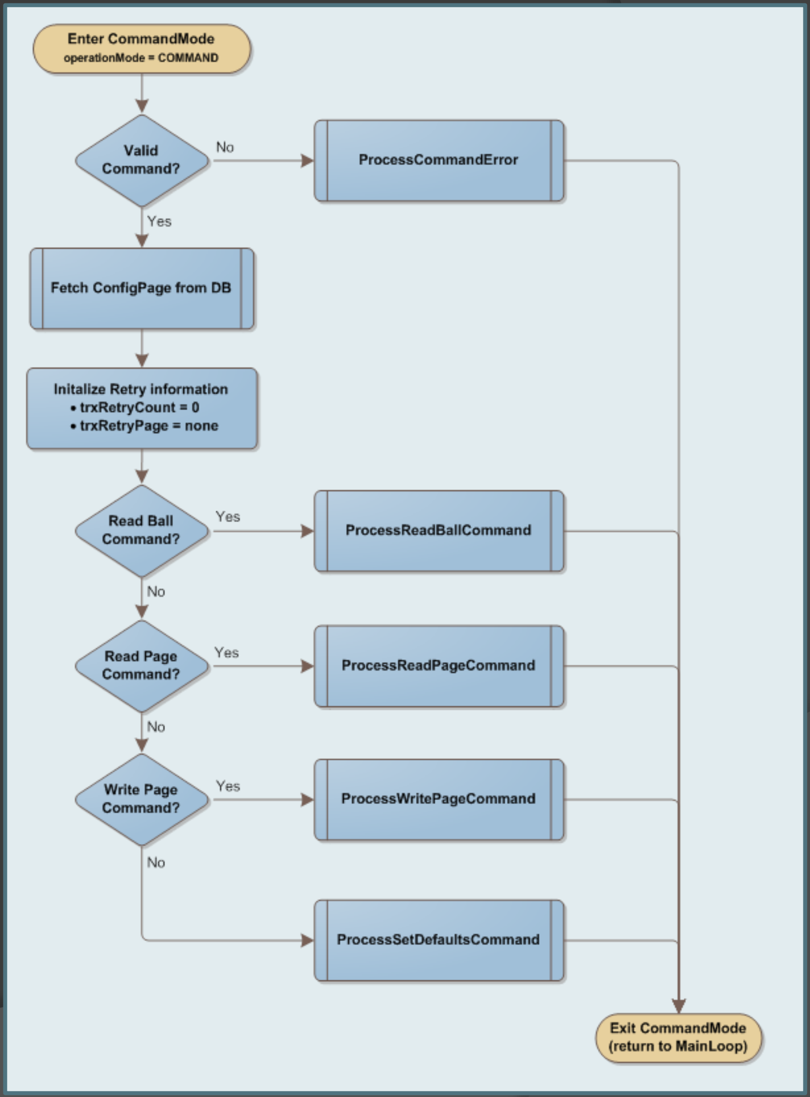
- *SenseModule* enters WakeUpMode upon detection of ambient light HW start-up event (CP0 Wakeup Event in SleepMode)
- Handles orderly start-up of µP and its peripherals
- Starts ambient light sampling
- Checks for valid start-up condition (must be sufficiently dark for a certain period after initial wake up
- Returns to MainLoop



SenseModule Embedded Software

CommandMode Process

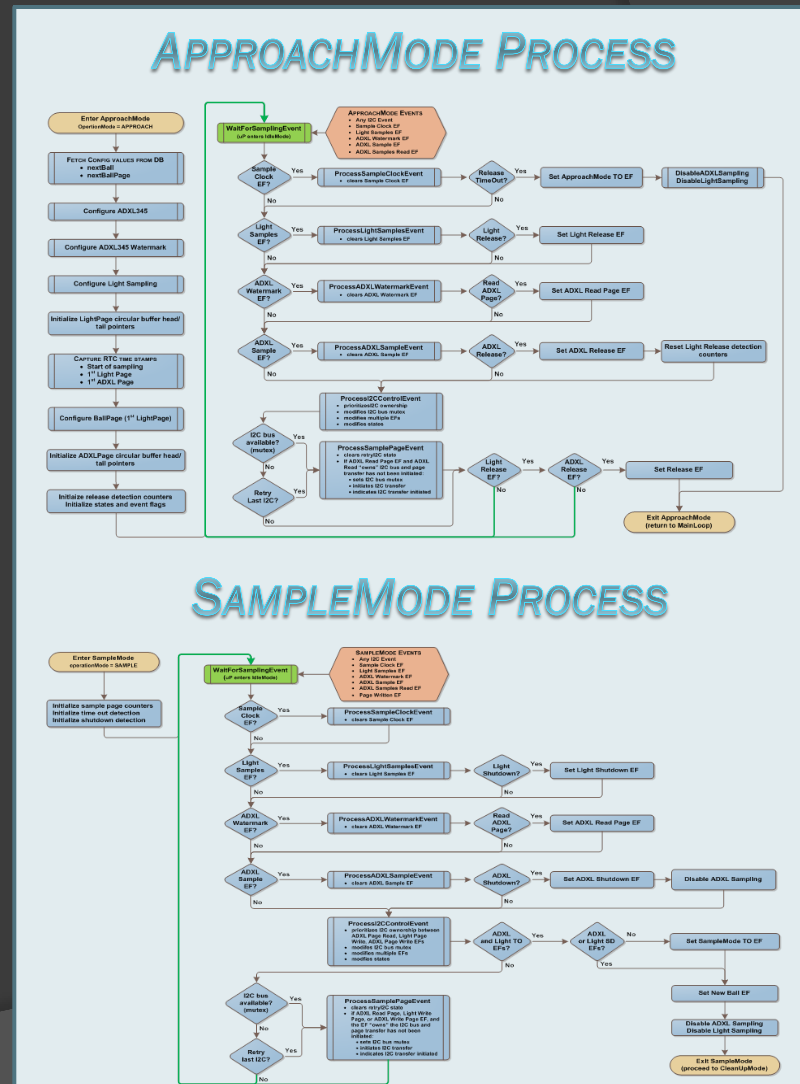
- MainLoop handles detecting presence of *ComModule*, receiving and parsing command strings
- CommandMode executes requested commands
- Four commands:
  - Read EEPROM Page
  - Read Ball Record
  - Write EEPROM Page
  - Set EEPROM Defaults
- MainLoop can receive sequences of commands



SenseModule Embedded Software

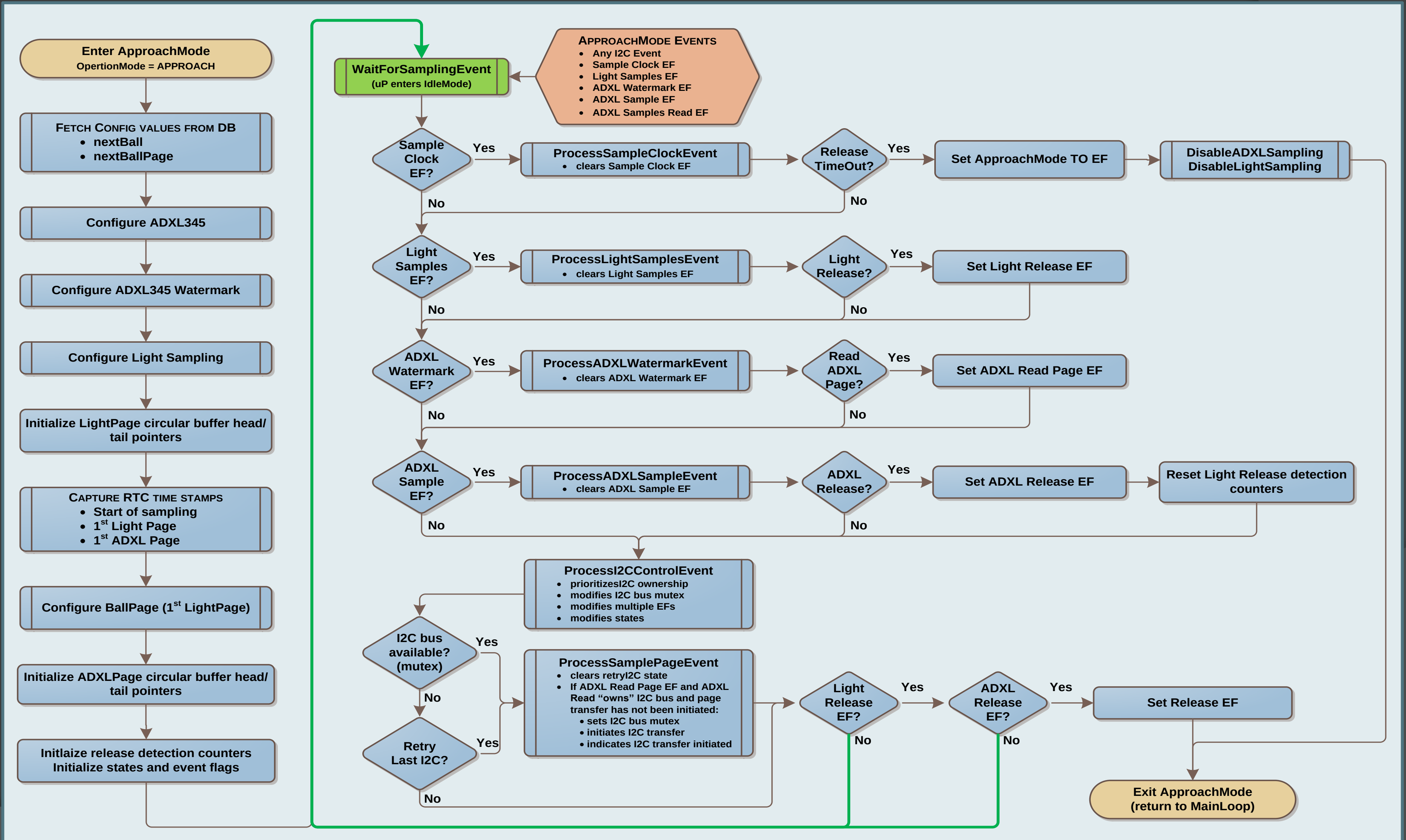
ApproachMode/SampleMode

- ApproachMode and SampleMode collaborate to detect, capture, and store ambient light (TSL13) and 3-axis acceleration (ADXL345) sensor readings
- ApproachMode stores sensor readings in Light and ADXL circular page buffers during bowler's approach and delivery, but does NOT commit readings to EEPROM
- SampleMode kicks in at release and commits previously stored circular buffer contents to EEPROM, while continuing to store new readings to those circular buffers



SenseModule Embedded Software

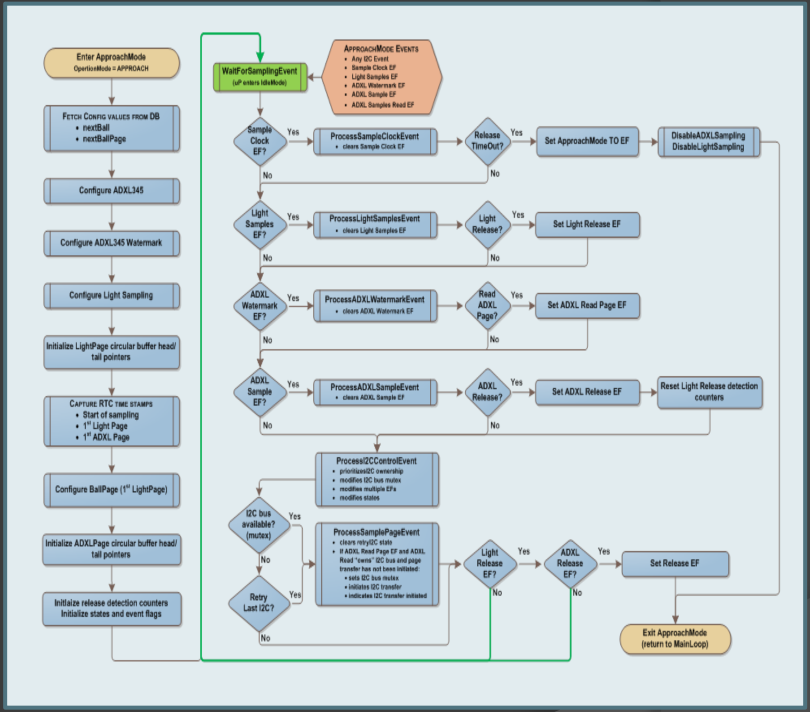
ApproachMode Process



SenseModule Embedded Software

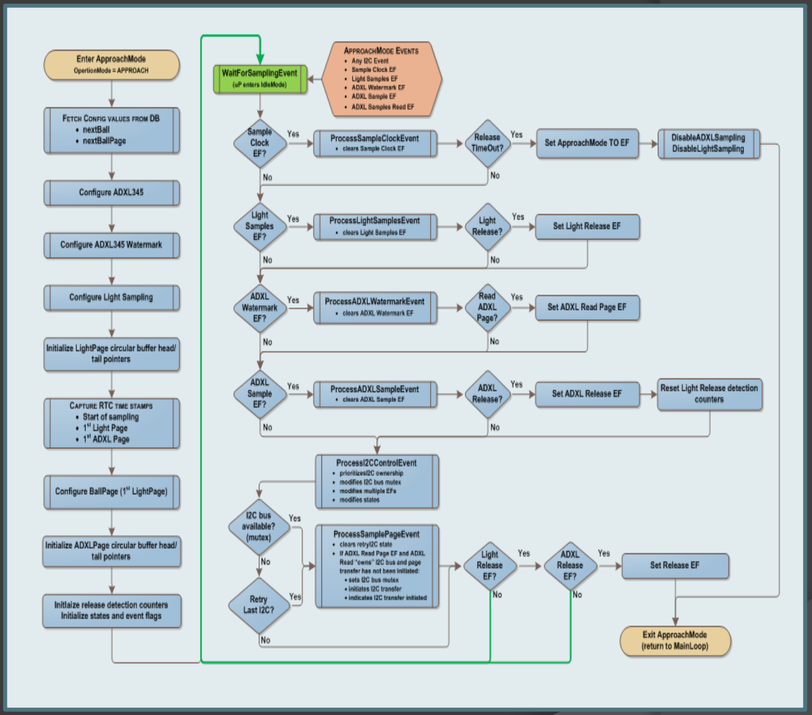
ApproachMode Process

- ApproachMode retrieves next available Ball Record pointer, and next available Ball Page location from Configuration Page
- Initializes Light and ADXL circular page buffers
- Enables ambient light (TSL13) and 3-axis acceleration (ADXL345) sensors
- Initializes release and shutdown variables
- Initiates waveform sampling (enables interrupts)
- µP enters internal low-power IdleMode and waits for interrupt events



APPROACHMODE PROCESS

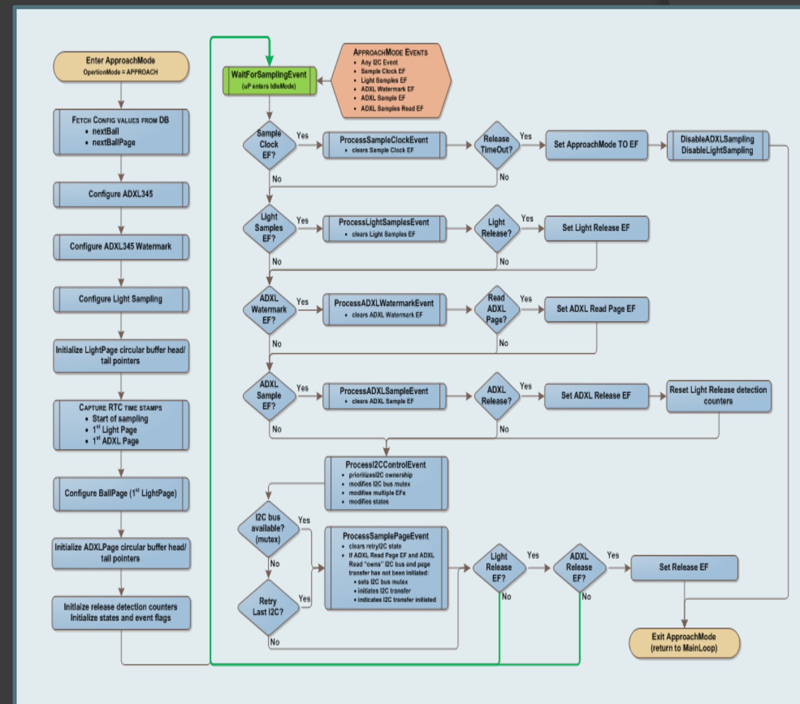
- Interrupts pull µP from IdleMode, ISRs retrieve sensor readings, and issue event flags EFs)
- ApproachMode captures snapshot of EventFlags register, clears EF register, and processes captured EFs in specific sequence
- Event flag processing can issue additional EFs for later processing
- Interrupts continue to occur during ApproachMode EF processing, EFs resulting from those interrupts will be processed in a second round, after current round completes
- µP returns to IdleMode when there are no new EFs



SenseModule Embedded Software

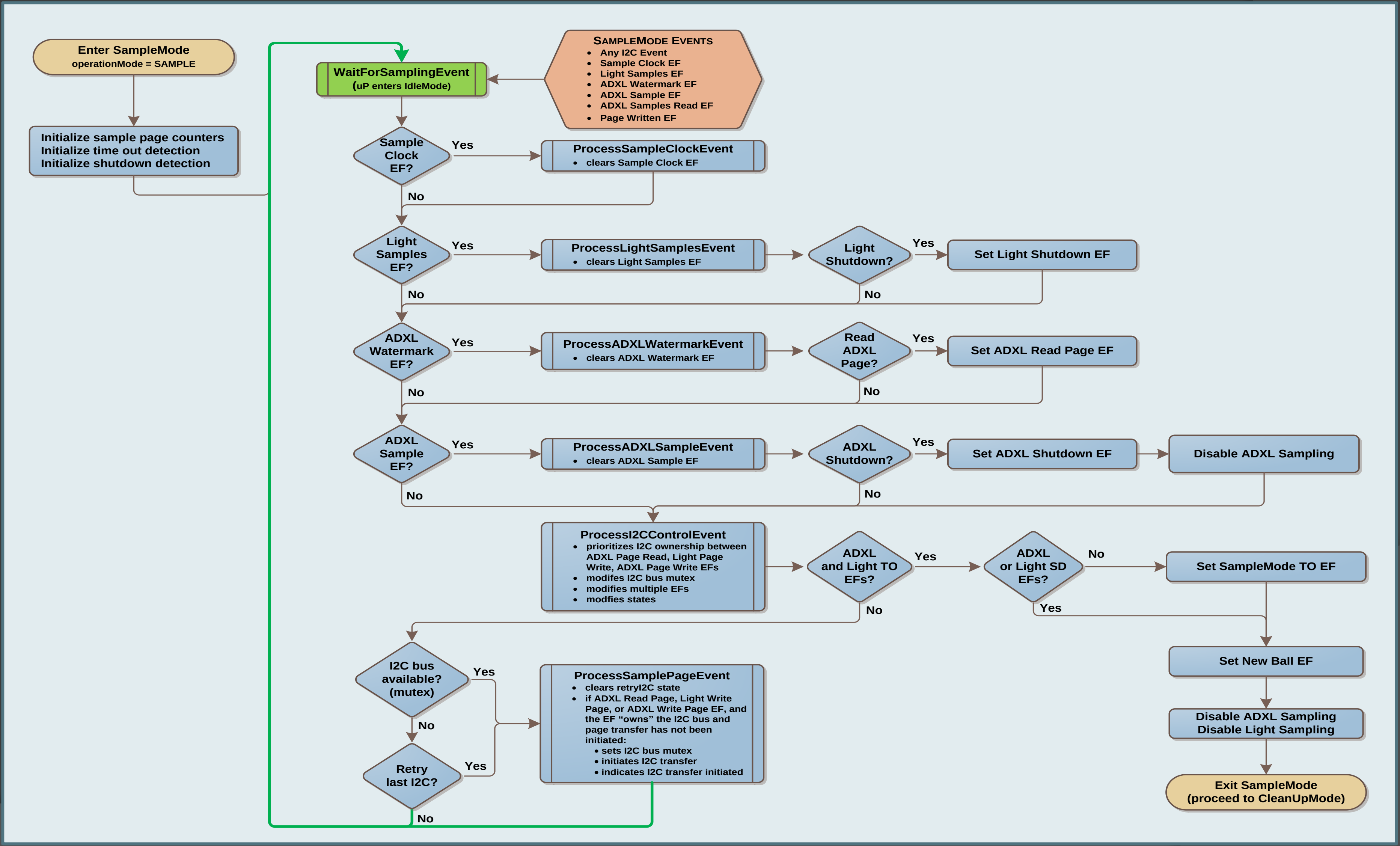
ApproachMode Process

- Light and ADXL samples are stored in circular page buffers in μP's XRAM during ApproachMode
- ApproachMode captures 3 seconds of sensor data immediately preceding release (pre-sampling)
- While sampling is going on, ApproachMode also checks for valid approach conditions, and for release
- If invalid approach condition is detected, sampling shuts down, return to MainLoop (back to SleepMode)
- If release not detected within 30 seconds, sampling shuts down, return to MainLoop (back to SleepMode)
- If release condition detected before time out, ReleaseEF is set, return to MainLoop (continues on to SampleMode)



SenseModule Embedded Software

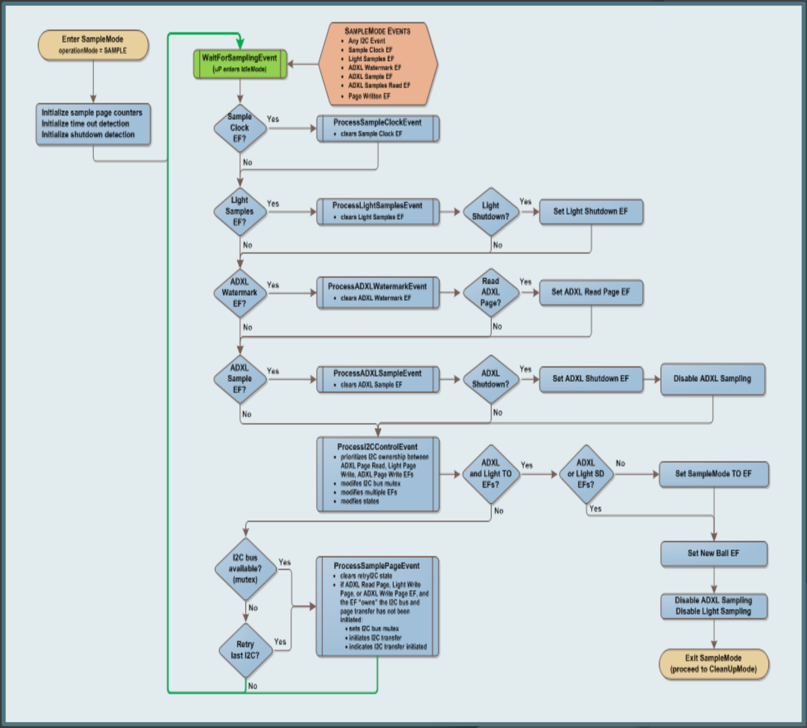
SampleMode Process



SenseModule Embedded Software

SampleMode Process

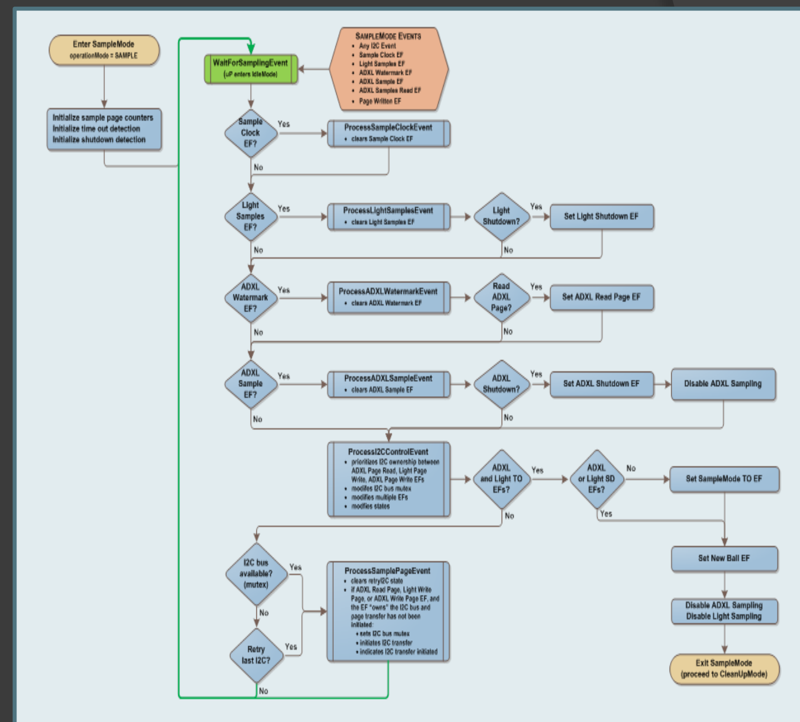
- Similar operation to ApproachMode – Waits in IdleMode, captures EFs, processes EFs, stores samples in circular page buffers
- SampleMode commits page buffer contents to new Ball Record in EEPROM
- New pages stored at buffer “head” pointer, pages written to EEPROM from buffer “tail” pointer
- Producer/consumer problem:
  - Buffers are full upon entering SampleMode
  - To avoid buffer overrun, must “consume” pages (write to EEPROM) faster than new pages are “produced” (stored in page buffers)
- Starvation problem:
  - 8 ADXL pages produced for each Light page
  - Separate Light and ADXL buffers
  - Write pages to EEPROM from both buffers
  - ADXL has higher priority, but must allow some alternation between buffers



SenseModule Embedded Software

SampleMode Process

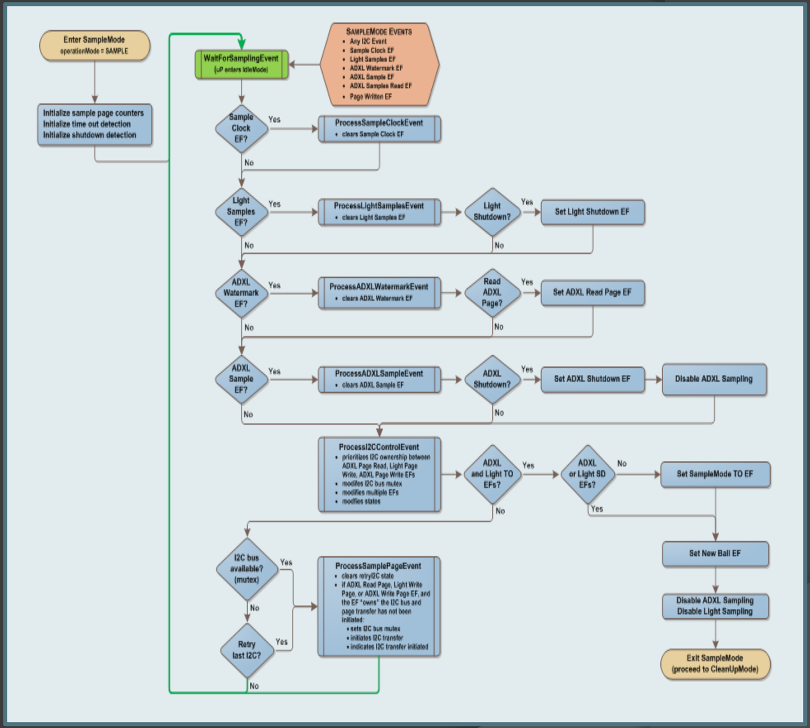
- Additional EFs and increased complexity due to writing pages to EEPROM
- I^2C bus is shared resource:
  - ADXL reads from ADXL345
  - Light Page writes to EEPROM
  - ADXL Page writes to EEPROM
- I^2C bus contention is an issue, can lead to deadlock if not managed correctly
- 25 ADXL samples retrieved every 125 ms from ADXL345, takes 7 ms via I^2C bus
- Transfers to EEPROM also take 7 ms
- Must use mutex on I^2C resource
- EEPROM writes take additional 5 ms, EEPROM is unresponsive during writes, must periodically poll and retry
- ProcessI2CControlEvent routine handles all bus contention – all bus traffic flows through this routine
- Assigns “ownership” based on priority, handles retries, tracks progress of transfer



SenseModule Embedded Software

SampleMode Process

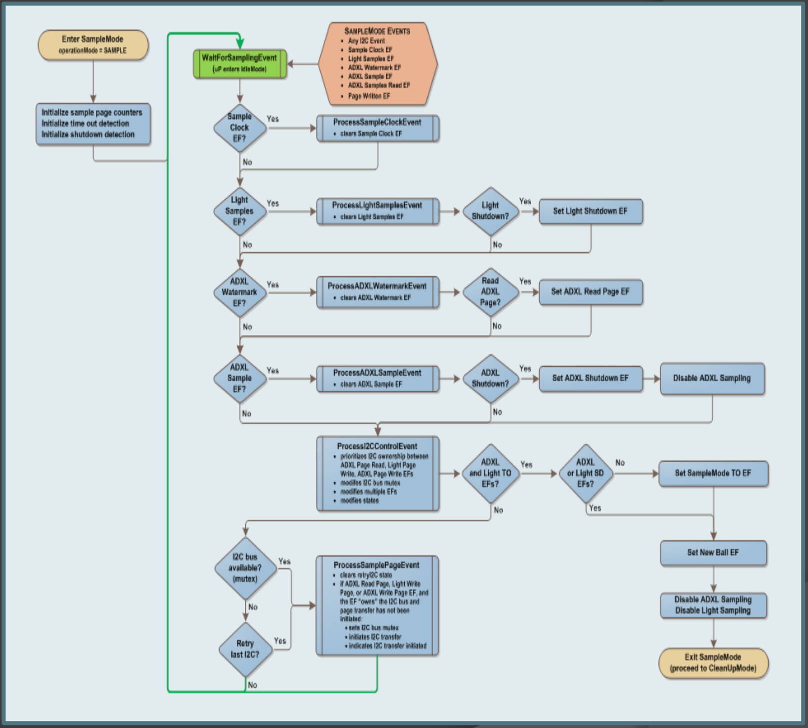
- ADXL345 captures samples in internal FIFO buffer – holds up to 32 3-axis samples
- New ADXL sample added to FIFO every 5 ms (200 Hz sample rate)
- FIFO interrupts µP at 25 samples
- 7 ms to transfer 25 entries (1 ADXL Page) from FIFO
- Another producer/consumer problem – how to keep from losing a sample during I²C transfer?
  - ADXL345 continues to post new samples to tail of FIFO while I²C transfer pulls from head of FIFO
  - 7 unused FIFO entries create 35 ms window for retrieving 25 ADXL samples before FIFO overrun occurs



SenseModule Embedded Software

SampleMode Process

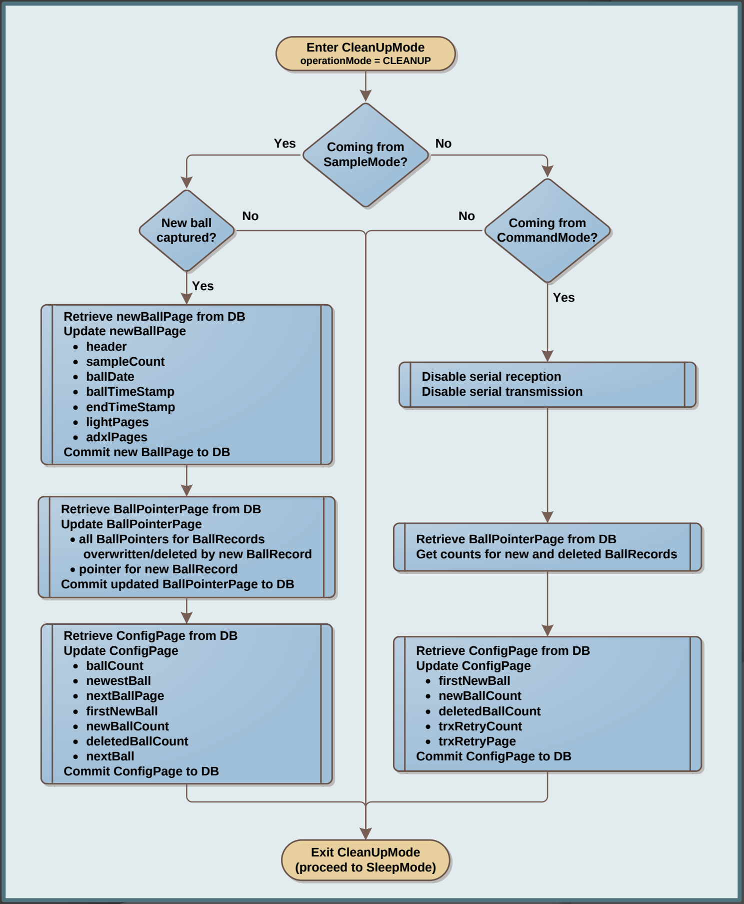
- Valid release detection:
  - Does waveform match that of rolling ball?
  - If not, sampling shuts down
- Shutdown detection:
  - Has ball stopped rolling (fallen in pit at end of lane?)
  - If so, sampling shuts down
- Sampling time out:
  - Shuts down sampling if shutdown condition not detected within 8 seconds from start of ApproachMode
- Return to MainLoop (CleanUpMode)



SenseModule Embedded Software

CleanUpMode Process

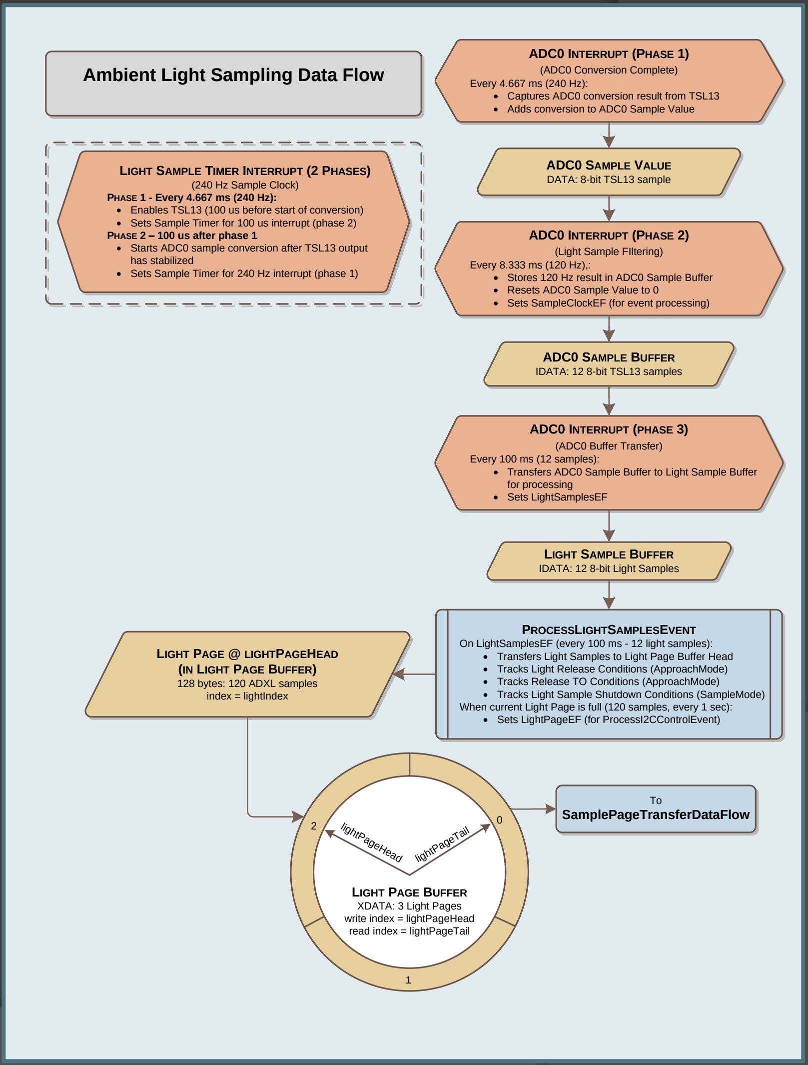
- CleanUpMode handles remaining processing coming out of CommandMode, ApproachMode, SampleMode
- CommandMode Clean Up:
  - Disable serial communications
  - Update firstNewBall, newBallCount, deletedBallCount in Config Page
- ApproachMode Clean Up:
  - Updates false activation count, run time in Config Page
- SampleMode Clean Up:
  - Valid new Ball Record:
    - Updates sampleCount, ballDate, ballTimeStamp, endTimeStamp, lightPages, adxlPages in new Ball Page
    - Updates Ball Record Pointer in Ball Pointer Page
    - Updates ballCount, newestBall, nextBallPage, firstNewBall, newBallCount, deletedBallCount, nextBall in Config Page
  - False release:
    - Updates falseReleaseCount, runTime, deletedBallCount in Config Page
    - New EEPROM data overwritten next time
- Returns to MainLoop (SleepMode)



SenseModule Embedded Software

Ambient Light Sampling Data Flow

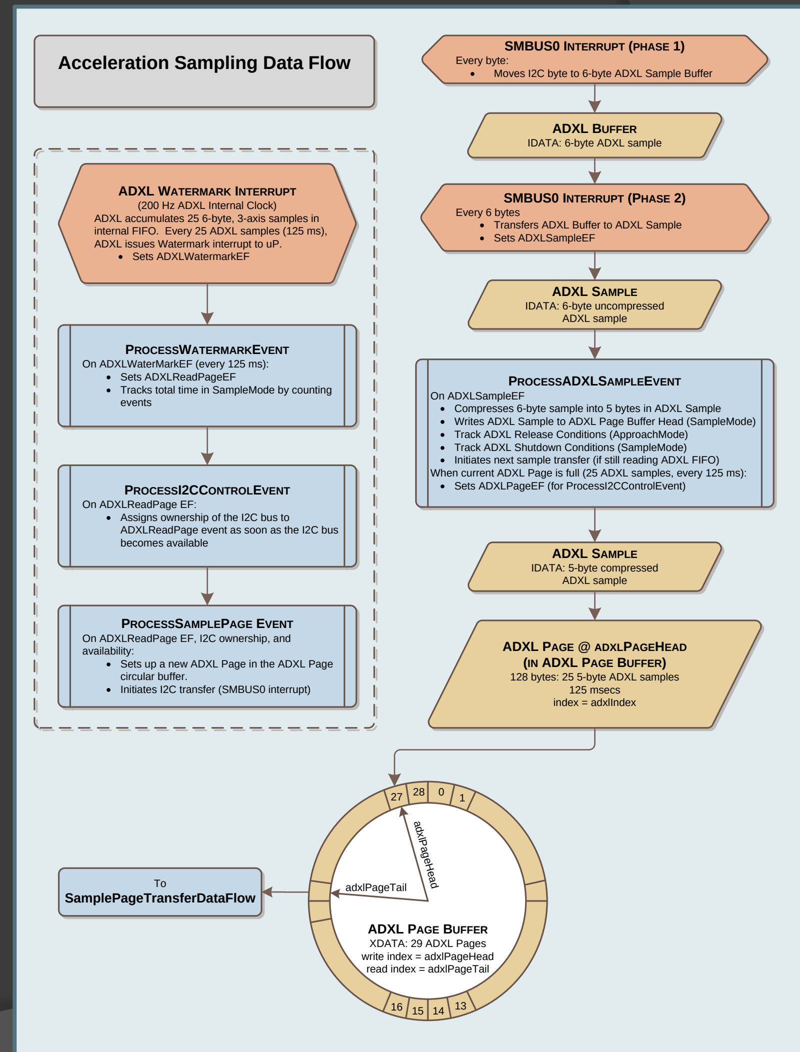
- µP's ADC0 samples TSL13 output @ 240 Hz, averages 2 samples to cancel 120 Hz fluorescent light ripple
- µP timer generates 240 Hz sample clock
  - Phase 1: 100 µS to enable TSL3
  - Phase 2: Starts ADC0 conversion, sets SampleClockEF
- ADC0 conversion complete ISR
  - Phase 1: Grabs 1st 240 Hz sample
  - Phase 2: Averages 1st and 2nd samples, places result in 12-sample (100 ms) ISR buffer
  - Phase 3: ISR buffer full, transfers contents for EF processing, resets ISR buffer, issues LightSampleEF
- ProcessLightSamplesEvent
  - ApproachMode: Detects Light release
  - SampleMode: Detects Light shutdown
  - Transfers light sample buffer to current page in Light circular buffer
  - When page fills, advances buffer to next page, issues LightPageEF
- ProcessI2CControlEvent
  - Processes LightPageEF



SenseModule Embedded Software

Acceleration Sampling Data Flow

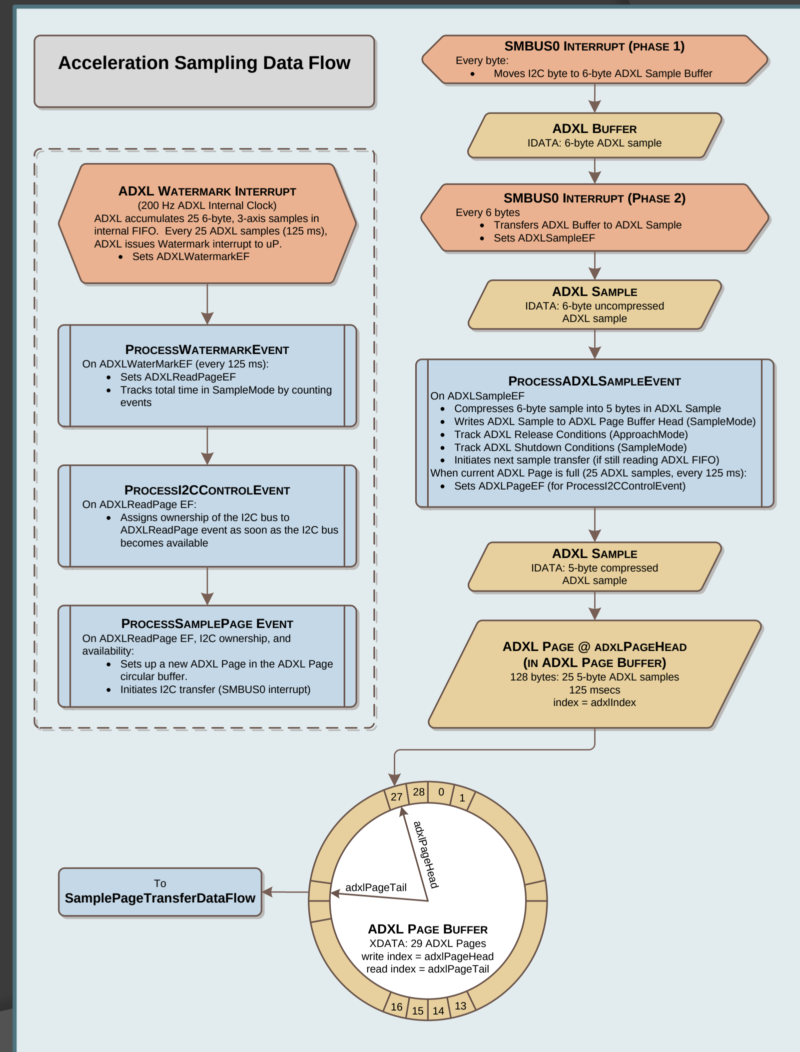
- ADXL345 samples 3-axis acceleration autonomously @ 200 Hz
- Issues Watermark interrupt to uP @ 25 samples in FIFO
- Watermark ISR:
  - Saves last RTC time for current FIFO contents (ADXL Page)
  - Captures new RTC time (for next ADXL page)
  - Issues ADXLWatermarkEF
- ProcessWatermarkEvent
  - Sets ADXLReadPageEF
  - SampleMode: Tracks ADXL shutdown detection
- ProcessI2CControlEvent:
  - Starts I^2C transfer when I^2C bus becomes available



SenseModule Embedded Software

Acceleration Sampling Data Flow

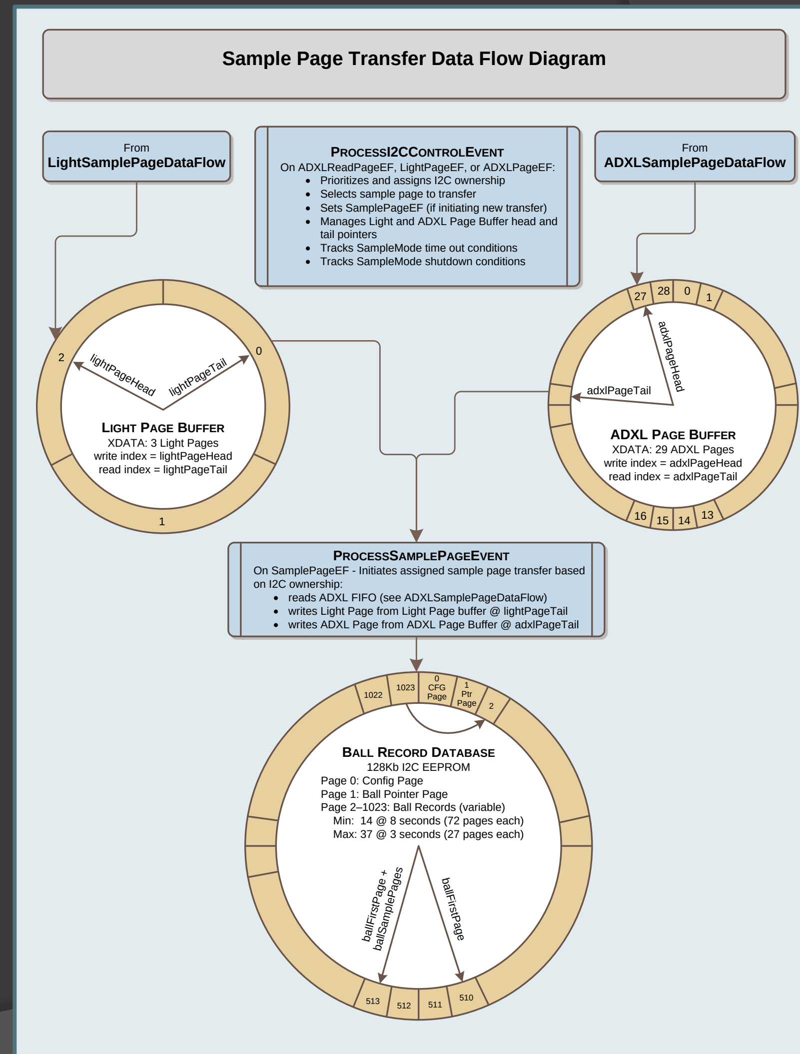
- SMBus0 ISR (I^2C transfer)
  - Phase 1: Reads sample bytes into ISR buffer – each ADXL sample has 6 bytes
  - Phase 2: Transfers 6-byte sample to ADXLSampleBuffer, sets ADXLSampleEF
- ProcessADXLSampleEvent
  - Compresses 6-byte sample into 5-byte sample
  - Places compressed sample in “head” page of ADXL circular buffer
  - ApproachMode: Detects ADXL release
  - SampleMode: Detects ADXL shutdown
    - When “head” buffer page fills, advances buffer pointer
    - Issues ADXLPageEF
- ProcessI2CControlEvent:
  - Handles ADXLPageEF



SenseModule Embedded Software

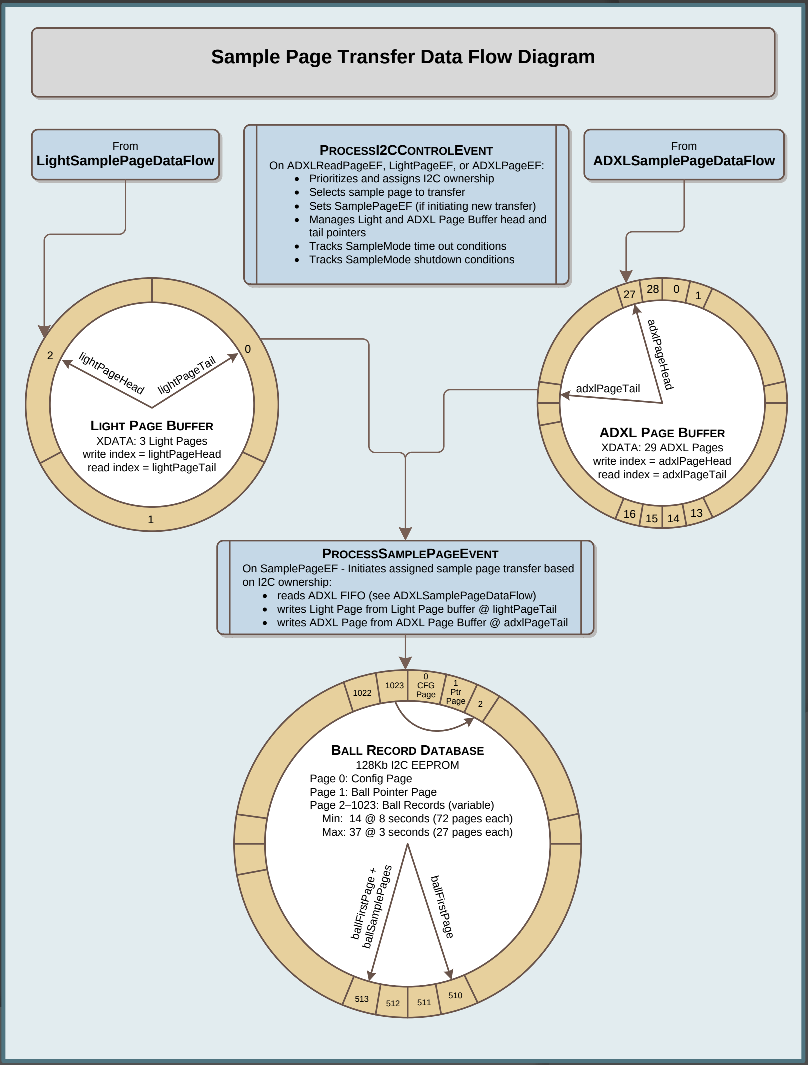
Sample Page Transfer and Storage

- ApproachMode has no bus contention, only I^2C reads from ADXL345
- SampleMode has ADXL reads, ADXL Page writes, Light Page writes
- Light and ADXL circular page buffers are always full coming from ApproachMode
- The start of SampleMode is “busy”
  - Reading ADXL pages from ADXL345
  - Storing new Light and ADXL pages at “heads” of circular buffers
  - Transferring old pages from “tails” of circular buffers to EEPROM



Sample Page Transfer and Storage

- ProcessI2CControlEvent called on every pass through SampleMode
  - Prioritizes competing I²C bus requests
  - Assigns and manages "ownership" of I²C bus
  - ADXLReadPageEF
  - ADXLWritePageEF
  - LightWritePageEF
  - Sets SamplePageEF if initiating new I²C transfer
  - SampleMode:
    - Detects sampling shutdown
    - Detects sampling time out
- ProcessSamplePageEvent
  - Initiates sample page transfer based on I²C ownership assigned by ProcessI2CControlEvent
  - Transfers ADXL FIFO to ADXL circular buffer "head"
  - Writes Light Page from Light circular buffer "tail"
  - Writes ADXL Page from ADXL circular buffer "tail"
  - Issues "retries" if EEPROM was busy last time (EEPROM non-responsive during 5 ms write)



SenseModule Performance

Raw Data

Waveforms

SenseModule Performance

Raw Data Waveforms

- After about 3 years of part-time research, design, and development, the first functional *SenseModule* prototype emerged from my basement and made its way down a real bowling lane…
- I threw 20 first balls with it – about what I thought the database would hold, at the time
- The potential IP implications were still unknown, so I couldn’t upload and view the data at that lanes – I had to wait until I got back home to look at the data…
- So, after 8 years of anticipation, and 3 years of development, what did I see when I uploaded the data to my PC…?

SenseModule Performance

Raw Data Waveforms

NOTHING…

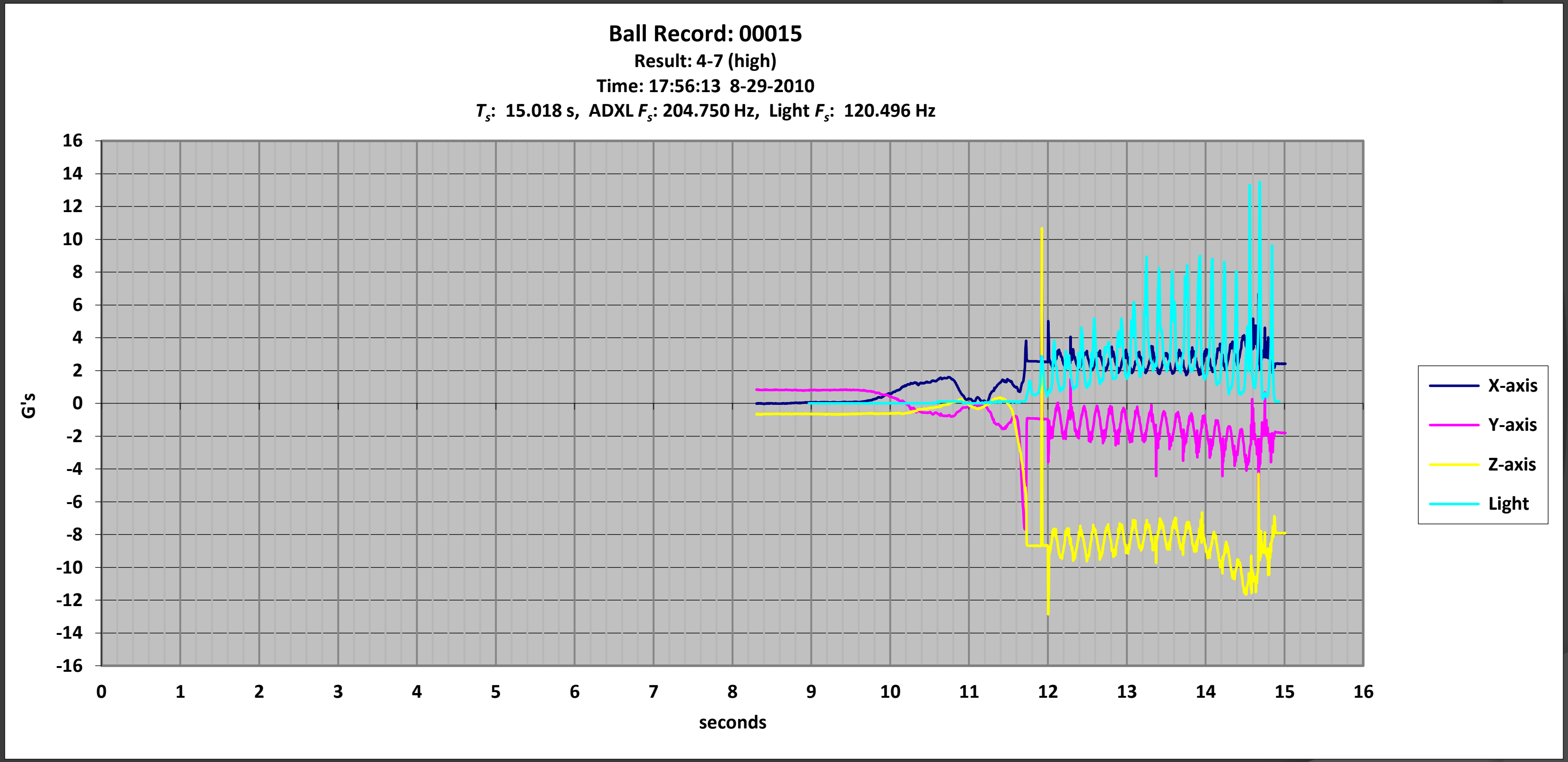
☹

Turns out my first attempt at automatic release detection was a little too restrictive for real-world use…

SenseModule Performance

Raw Data Waveforms

- After some further tweaking, and another trip to the lanes, here's what I saw…

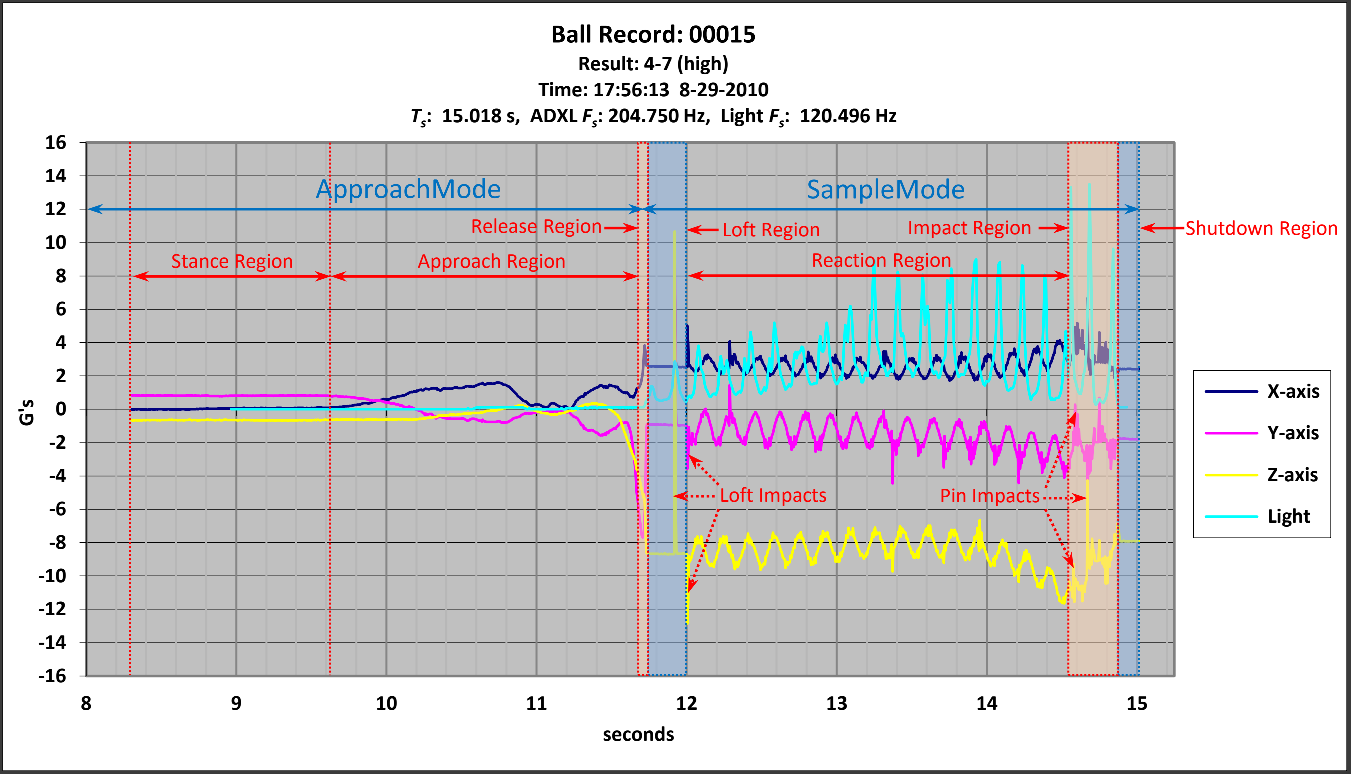


- Typical raw data waveform - time starts at beginning of ApproachMode
- First 8.25 seconds of data were overwritten in circular page buffers during ApproachMode while waiting for release

SenseModule Performance

Typical Raw Data Waveform

- Waveform evolves through several regions from ApproachMode to SampleMode

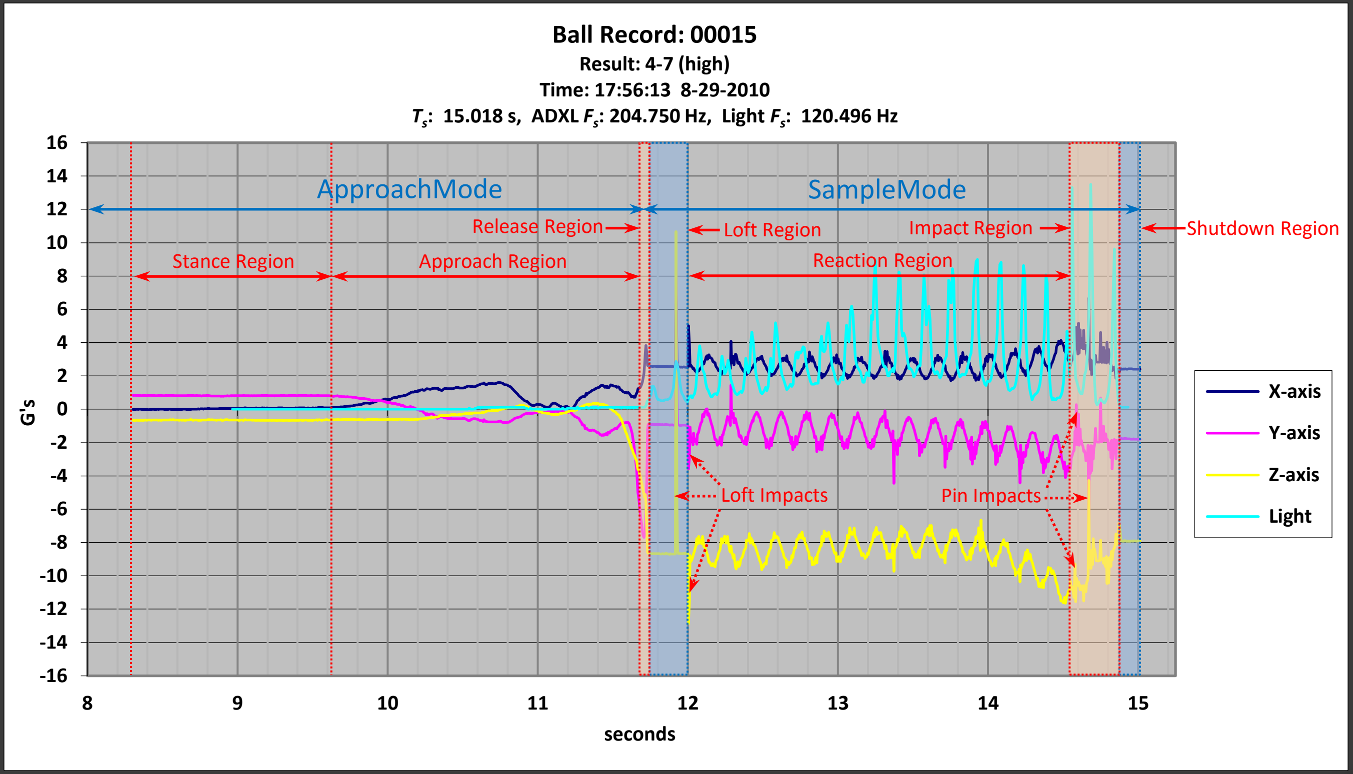


- Stance: Bowler is relatively still, preparing to start approach
- Approach: Bowler starts approach, response to arm swing is evident
- Release: Bowler applies lift and turn to ball, notice sudden sharp increase in acceleration, notice increase in light level, as bowler removed fingers from ball
- ApproachMode ends, SampleMode begins

SenseModule Performance

Typical Raw Data Waveform

- Waveform evolves through several regions from ApproachMode to SampleMode

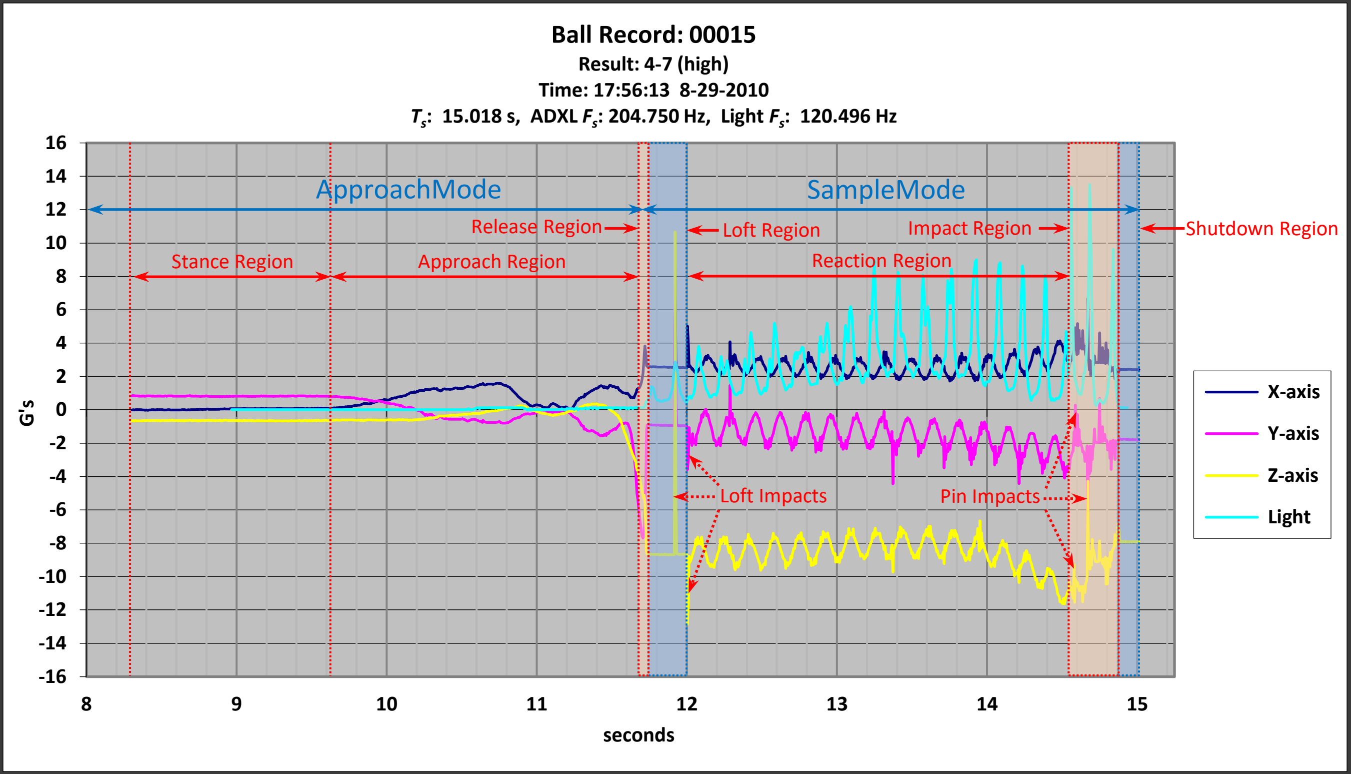


- Loft: Bowler has released ball, ball is in free fall, flat acceleration due to centripetal force generated by rotation
- Loft Impacts: Ball contacts lane, generating impact, second impact is from bounce
- Reaction: Ball rolling on lane, tilt sensing aspect superimposed on acceleration, acceleration increases as ball “revs” up, light waveform also indicates rotation

SenseModule Performance

Typical Raw Data Waveform

- Waveform evolves through several regions from ApproachMode to SampleMode

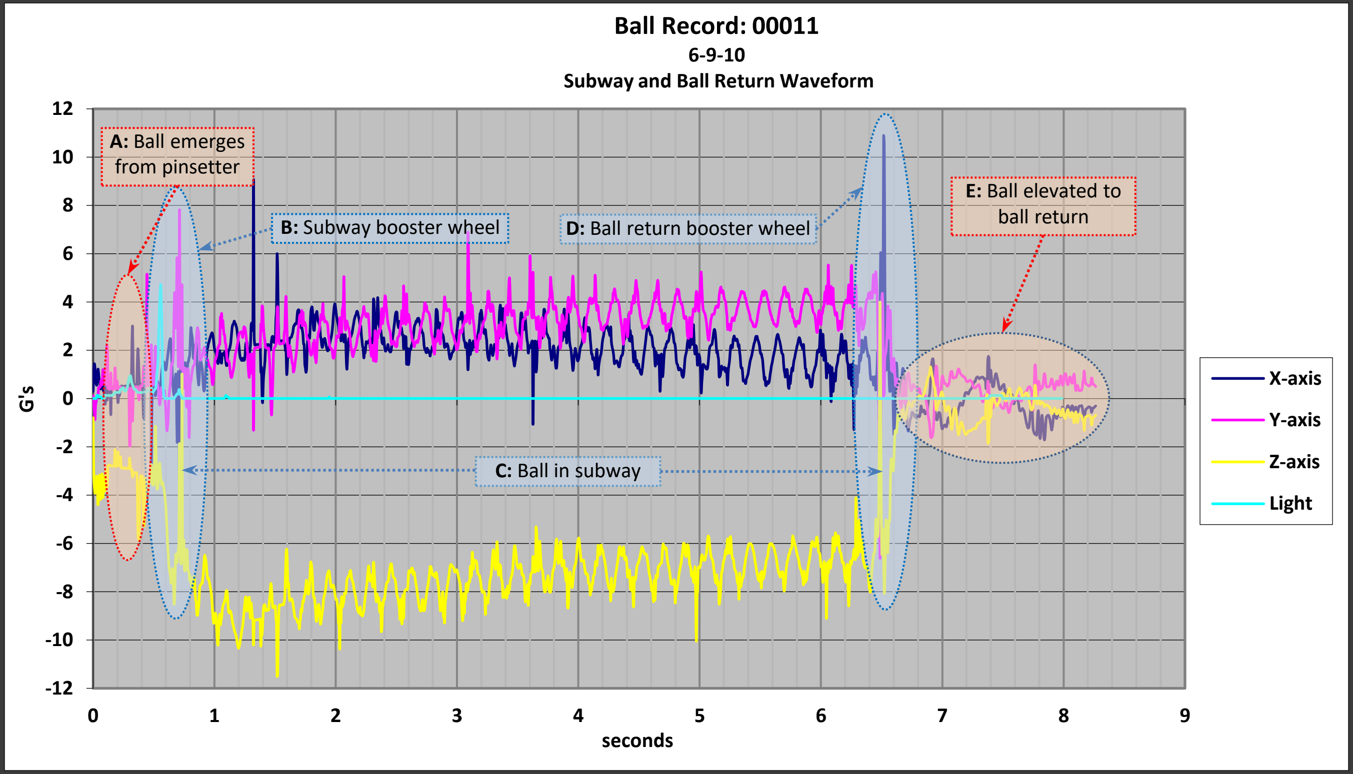


- Impact: Ball impacts pins – multiple spikes, plus increased high frequency noise level, ambient light spikes as ball passes under pin light
- Shutdown: Ball falls off end of lane into pit, free fall again evident, *SenseModule* shuts down

SenseModule Performance

False Activation Waveform

- SenseModule also "sees" waveforms that result from false activations

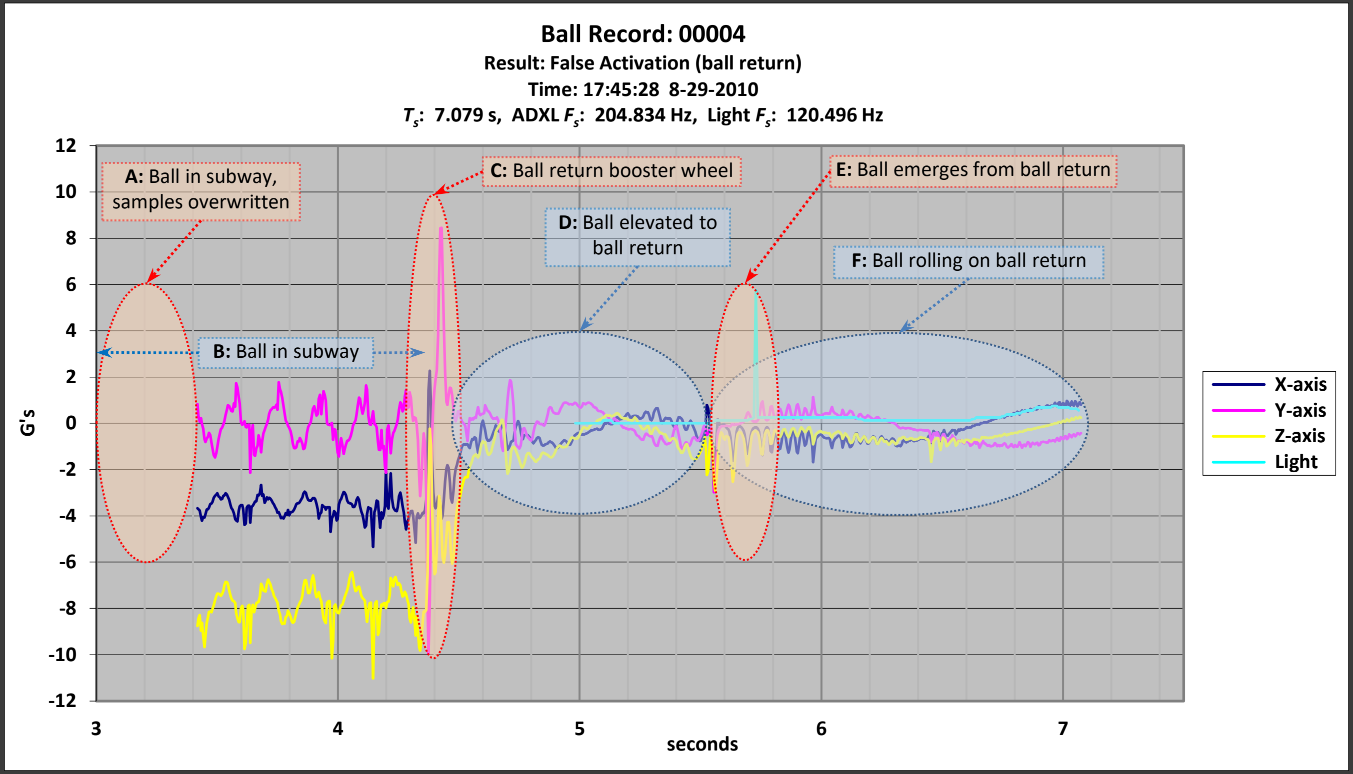


- SenseModule started up in pinsetter (dark)
- Light release condition detected when ball emerged onto subway ramp, combined with "impacts" from encountering booster wheel
- Rotation occurs as ball rolls along subway toward ball return
- "Impacts" at ball return booster wheel, sampling timed out as ball rolled on ball return

SenseModule Performance

False Activation Waveform

- SenseModule also "sees" waveforms that result from false activations



  - SenseModule started up when entering subway at pinsetter
  - Light release condition detected when ball emerged from subway to ball return

SenseModule Performance

Autonomous Operation

SenseModule Performance

Autonomous operation

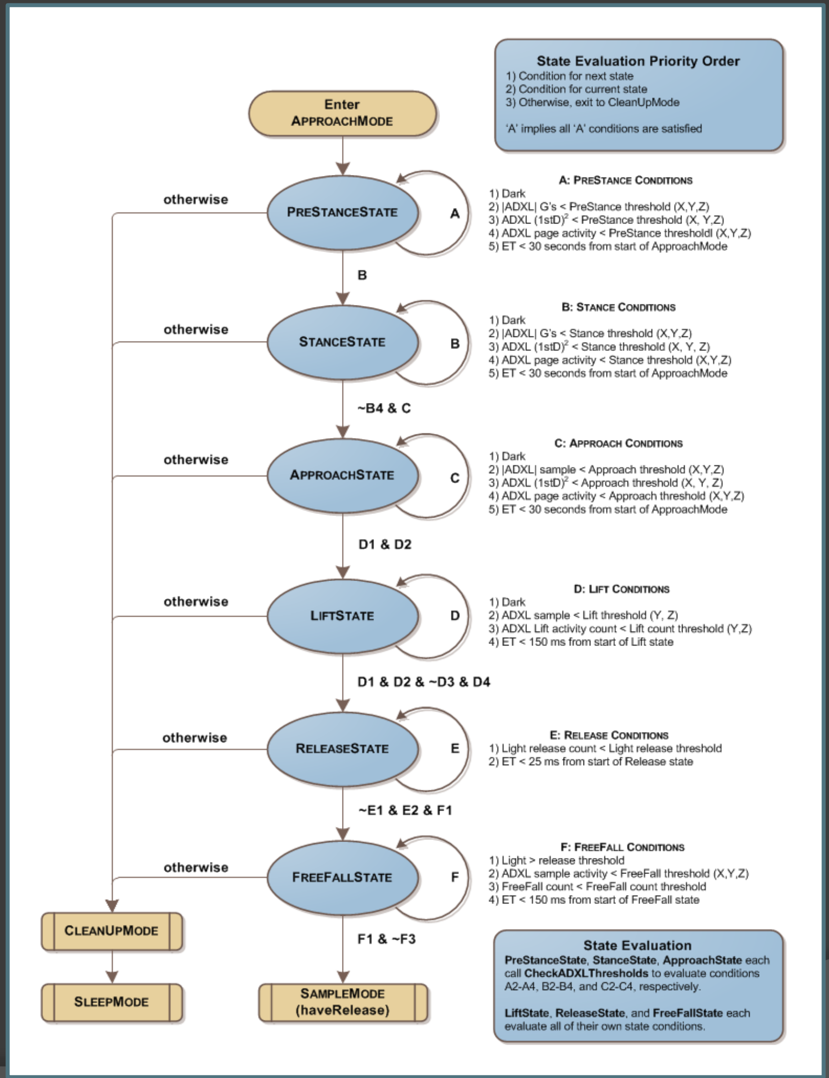
- *SenseModule* should record data for every valid activation and release
- *SenseModule* start-up circuit cannot reject subway and ball return activations
- Autonomous operation requires reliable detection and discrimination routines
- To conserve Ball Record DB space and battery life:
  - *SenseModule* should detect invalid ApproachMode waveforms, and shutdown
  - *SenseModule* should then detect invalid release conditions, and shutdown
  - *SenseModule* should then detect invalid loft/reaction conditions, and shutdown
- Discrimination is not easy, since subway and ball return activation waveforms have similar morphology as typical waveform

Autonomous Operation

- Complexity of task increases with processor and memory constraints:
  - 8-bit μP running at 3.05 MHz
  - 256 bytes of RAM, including stack (24 bytes)
  - 4 kbytes of XRAM used for circular page buffers
  - 32 kbytes of code space
  - Must detect in real-time
- True challenge of working in 8-bit embedded environment – working within those constraints
- Must identify minimum amount of information (data) necessary to make reliable decisions quickly
- Requires very efficient algorithms for detection – no heavy-duty DSP going on in the *SenseModule*

SenseModule Performance

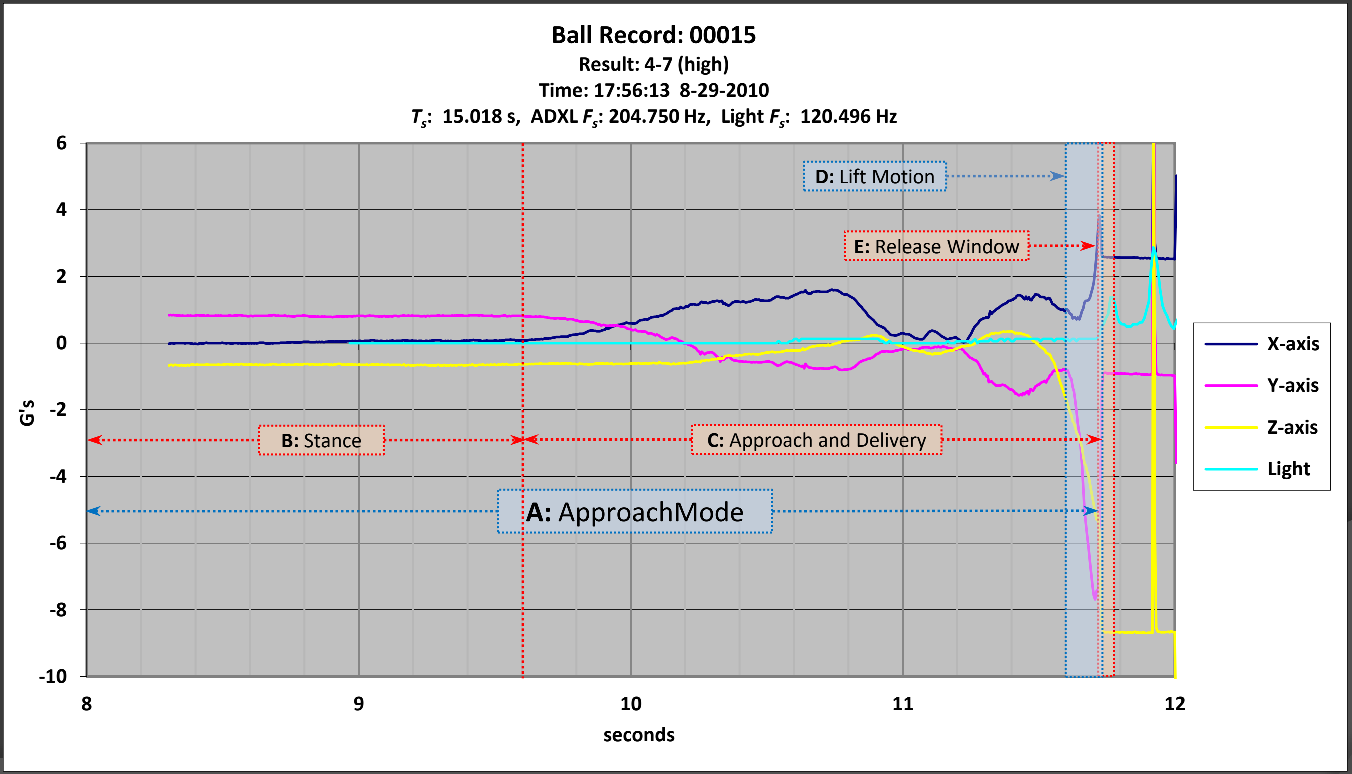
Activation and Release Detection State Machine



SenseModule Performance

False Activation Detection

- Typical ApproachMode waveform, leading up to release
  - Absence of light
  - Initial flat acceleration – tilt sensing only
  - Followed by low frequency content
  - Low acceleration amplitude – under ±2 g



SenseModule Performance

False Activation Detection

- ApproachMode times out in 30 seconds if release not detected
- Must quickly detect invalid waveform to limit battery consumption
- Limited breadth of waveforms with which to develop false activation routine – single user during development
- Subway and ball return scenarios present false activation detection challenges
- Not as simple as low light, low amplitude and low frequency acceleration levels

SenseModule Performance

False Activation Detection

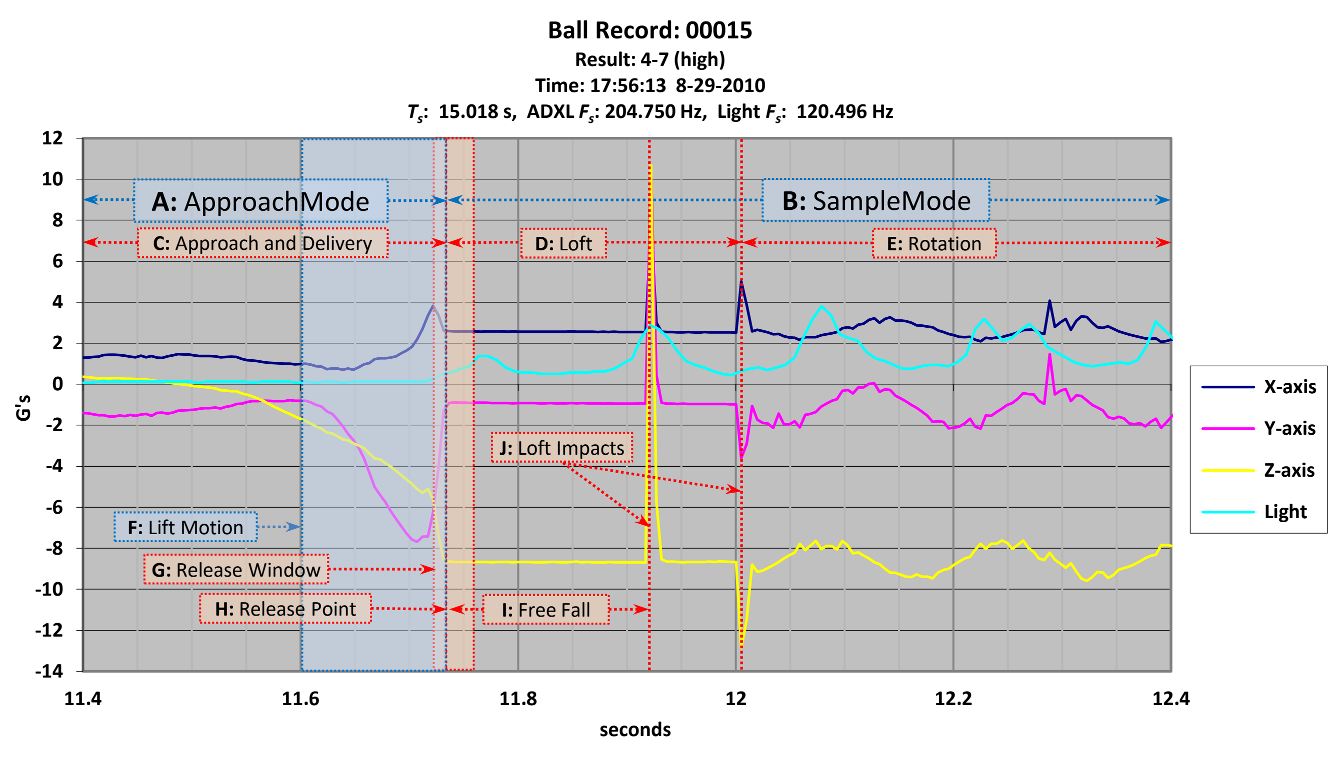
- False activation detection algorithm considers:
  - Ambient light level: Magnitude, frequency, rate of change
  - 3-axis acceleration: magnitude, frequency, rate of change
  - Relative phase of changes in light and acceleration
- SenseModule must err on side of caution – detect every valid activation, at the expense of missing some false activations
- Results of false activation routine have been mixed
  - Rejected 2.3% of valid activations
  - Missed $^1/_3$ of false activations

SenseModule (219 frames)	Valid Activations	False Activations
Captured	214	56
Rejected	5	123
Total Events	219	178
Detection Efficiency (%)	97.7%	68.5%

SenseModule Performance

Valid Release Detection

- Typical release and loft region waveform
  - Rapidly increasing acceleration levels, especially on *Y*-axis *Z*-axis
  - Amplitudes exceeding ± 2 *g* on multiple axes
  - Absence of light followed by sudden increase in light
  - Sudden drop-off to flat acceleration – *SenseModule* in free fall during loft



SenseModule Performance

Valid Release Detection

- False activation detection and release detection run concurrently
- If false activation detected first – shuts down sampling, returns to SleepMode
- Otherwise, release detection switches *SenseModule* from ApproachMode to SampleMode
- SampleMode starts committing Light and ADXL Pages to EEPROM
- Missed false activation/release overwrites valid Ball Record(s)
- Subway and Ball Return scenarios also include false release content
- Release detection NOT as simple as:
  - Rapidly increasing acceleration
  - Followed by light transition
  - Followed by flat acceleration

SenseModule Performance

Valid Release Detection

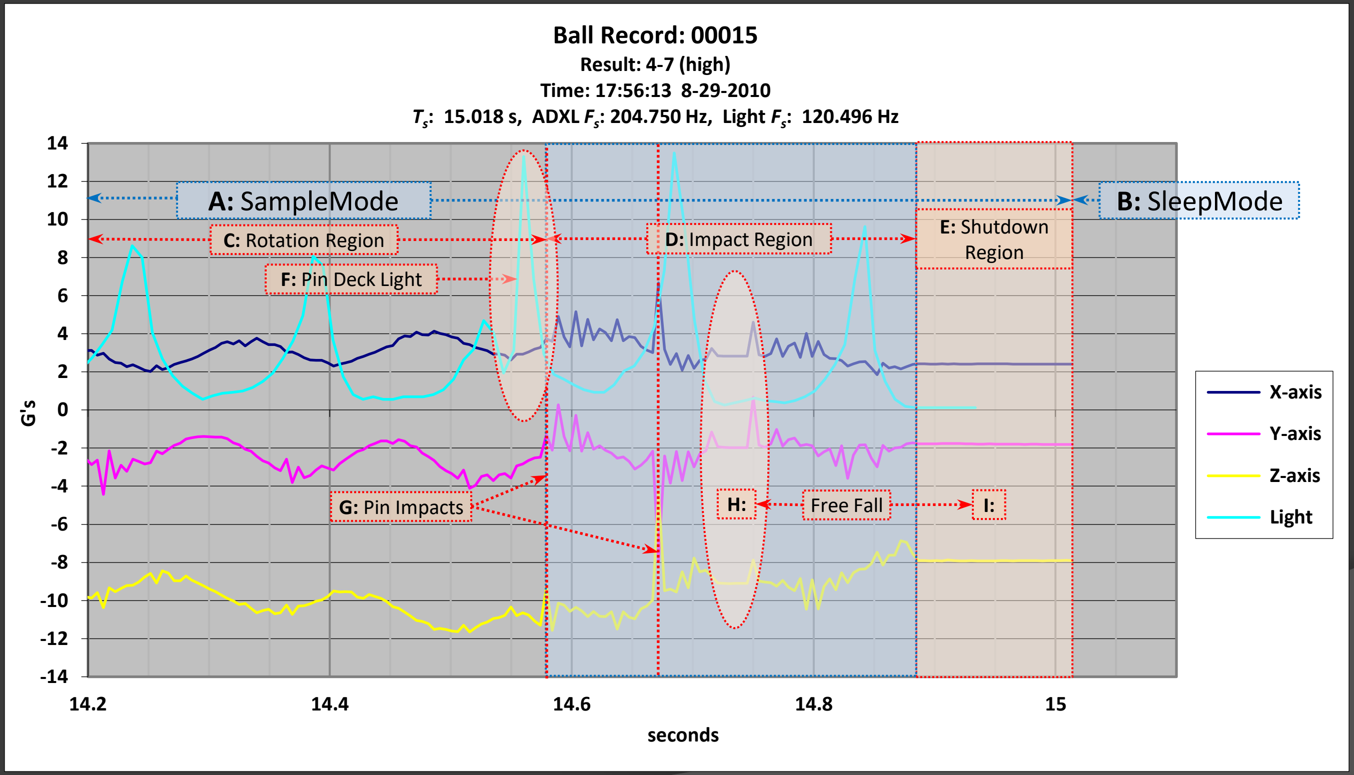
- Release detection algorithm considers:
  - Ambient light level: Rate of change
  - 3-axis acceleration: magnitude, rate of change
  - Timing constraints between certain light and acceleration events
- Similar story - *SenseModule* must detect every valid release, at the expense of missing some false releases
- Results of release detection have been mixed
  - Rejected 2.3% of valid releases
  - Missed $^{1}/_{4}$ of false releases
- Combined efficiency of false activation detection and release detection is 93%
- Still room for improvement – must capture 100% valid waveforms

SenseModule (219 frames)	Valid Releases	Pinsetter “False” Releases	Subway/Ball Return “False” Releases
Captured	214	2	13
Rejected	5	41	
Total Events	219	56	
Detection Efficiency (%)	97.7%	73.2%	

SenseModule Performance

Shutdown Detection

- Typical Shutdown region waveform:
  - Light spike from passing under pin deck light
  - Multiple impacts with pins and increased high frequency noise content
  - Free fall as ball falls into pit (flat acceleration)



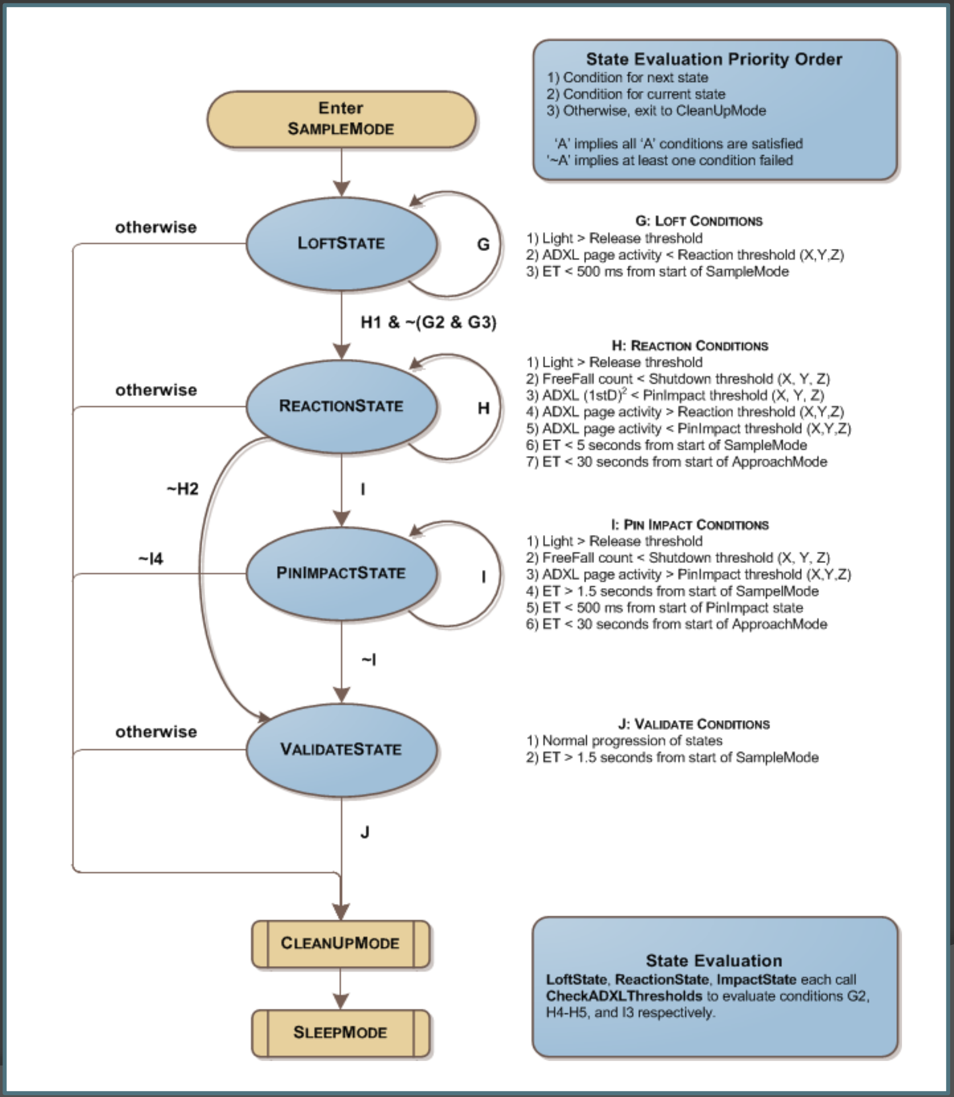
SenseModule Performance

Shutdown Detection

- Variable length Ball Records conserve Ball Record DB space, *SenseModule* run time
- Detects cessation of activity, shuts down sampling before time out
- Could double as last line of "defense" against recording false activations:
  - Constantly monitor Reaction region sensor response
  - Shut down sampling, does not advance pointers, if invalid conditions detected
  - Only a portion of oldest Ball Record gets overwritten
  - Next Valid Ball Record overwrites invalid data

SenseModule Performance

Shutdown Detection State Machine



SenseModule Performance

Shutdown Detection

- Simple shutdown algorithm:
  - Low light threshold

    OR
  - 50 ms free fall
- Invalid reaction detection algorithm:
  - Acceleration frequency content
  - Relative 3-axis acceleration amplitudes
  - Light level
  - Minimum time between release and pin impacts
- Results:
  - 100% effective shutting down valid waveform sampling before expiration of sampling time out
  - Invalid reaction detection not yet implemented

RevMetrix Application

Waveform Analysis

Waveform Analysis

- Now that the *SenseModule* has collected this data, what can we do with it?
- Must extract and present useful metrics in a form easy to visualize and comprehend
- So what metrics interest the bowler?

Waveform Analysis

- Everything the bowler can control happens at release – consistent release is at the heart of consistent execution
  - Release linear velocity (ball speed)
  - Release angular velocity (RPMs)
  - Axis turn (angle with respect to direction of travel)
  - Axis tilt (angle with respect to lane surface)
  - Loft distance

Waveform Analysis

- Bowler cannot control what happens after release – how ball reacts to lane conditions:
  - Breakpoint (lane location where ball starts to hook toward pins)
  - Revolutions from release to impact ("revs")
  - Impact linear velocity
  - Impact angular velocity
  - Impact axis turn
  - Impact axis tilt

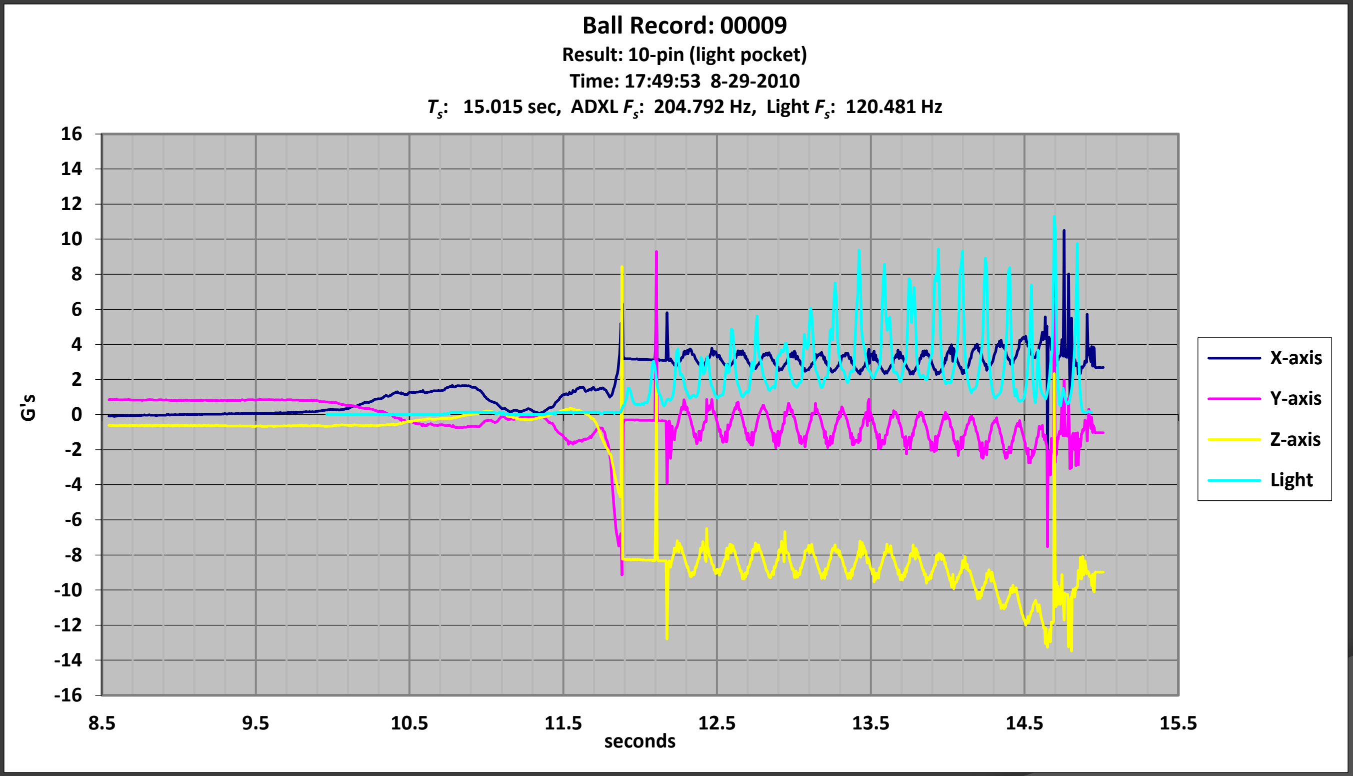
Waveform Analysis

- So what can the *SenseModule* tell us?
  - Apart from the automatic detection routines, *SenseModule* makes no decisions and draws no conclusions from raw data it collects
  - *SenseModule* identifies characteristics directly from raw data in real-time indicative of approach, release, ball rolling down lane, impact with pins, falling into pit
  - *SenseModule* knows nothing about release velocity, angular velocity, RPMs, loft distance, etc
  - Raw data must be uploaded to *RevMetrixApp*, which then must extract those quantities of interest from *SenseModule* raw data waveforms, and present them to bowler

How do we do that?

WAVEFORM ANALYSIS

- How do we get from this (raw data):



Waveform Analysis

- To this (metrics)…?

Release linear velocity:	15.00 mph
Impact linear velocity:	14.57 mph
Average linear velocity:	14.93 mph
Release angular velocity:	357.9 rpms
Impact angular velocity:	423.7 rpms
Total Revolutions:	15.3
Loft Distance:	54 in (4.51 ft)
Reaction Distance:	76 in (6.34 ft)
Breakpoint Distance:	37.7 ft

Waveform Analysis

- A modicum of math…
- A pinch of Physics…
- A dash of DSP…
- A smattering of Wavelet Theory…
- And a fair amount of algorithm development to tie all of that together…
- Rather a lot, really… ☺

Waveform Analysis

And It was time consuming…

And complicated…

And frustrating…

And "fascinating"…

If you're into that kind of thing…

Did I mention that I'm a gEEk…☺

Waveform Analysis

In more formal terms, it ultimately involved all of the following:

- Fast Fourier Transforms (FFTs)
- Symmetric Finite Impulse Response (FIR) filters
- Wavelet decomposition and reconstruction
  - 1st-level Haar
  - 3rd and 5th-level biorthogonal 6.8
- Statistical analysis:
  - Mean
  - Variance
  - Standard deviation
- Numerical methods
  - Interpolation
  - Extrapolation
  - Curve fitting
  - Derivatives

Waveform Analysis

- *SenseModule* waveform data is uploaded through ComModule to PC
- Stored as CSV files on PC
- MS Excel used for initial data visualization, developing *SenseModule* algorithms
- MATLAB used to isolate and filter acceleration components and develop bowling metric extraction routines

Waveform Analysis

- Waveforms contain multiple regions with sudden transitions and varying frequency content
- FFTs and FIR techniques are suited to repetitive signal content
- Wavelet techniques work better on disjoint, non-repetitive signals
- 1st-level *Haar* wavelet details are used to identify distinct temporal regions of 3-axis acceleration waveforms
- Waveform is then segmented into components with common spectral composition
- FIR and wavelet-based filtering techniques tuned to segment morphology and frequency content extract meaningful metrics

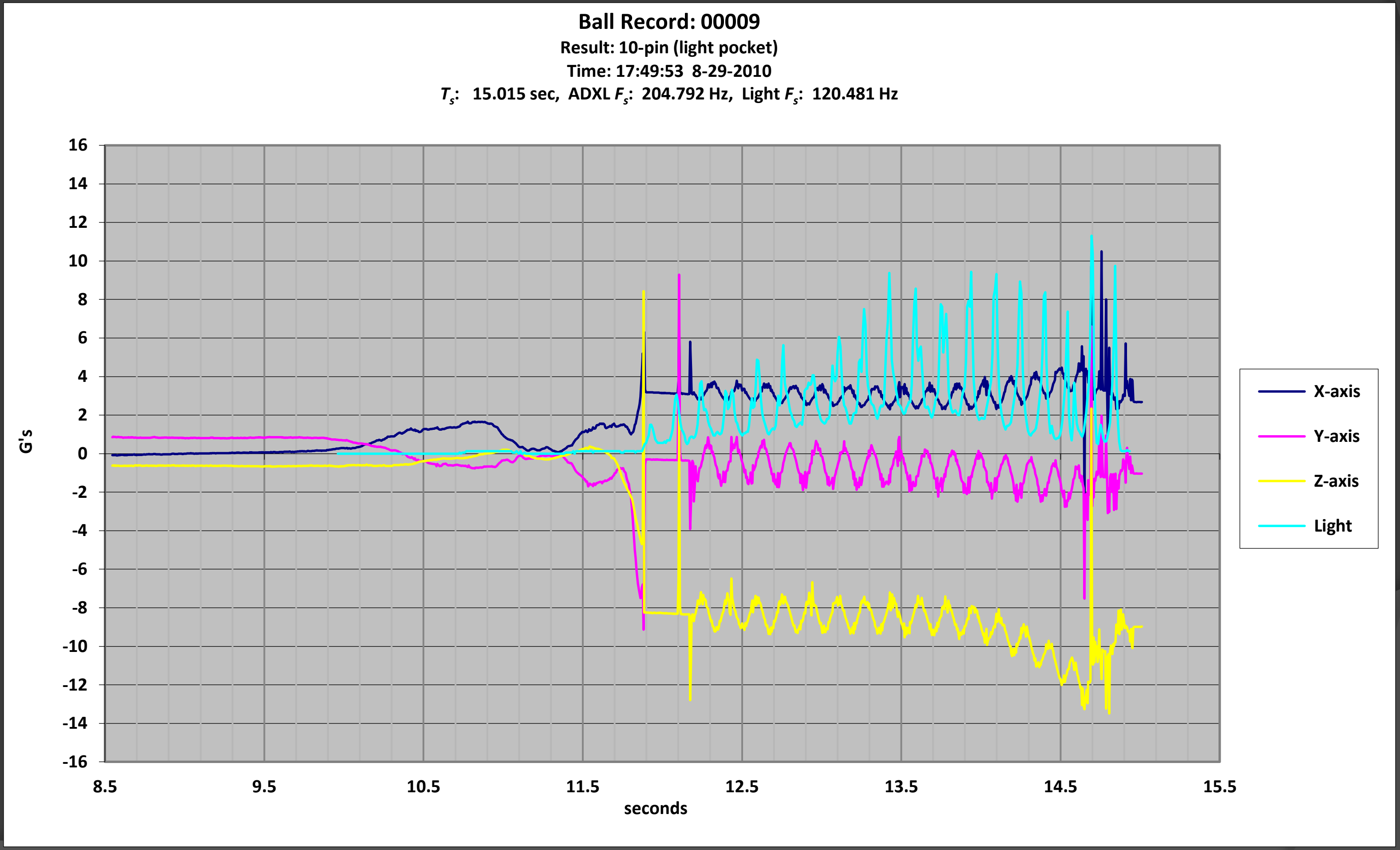
Waveform Analysis

- Focus on Loft and Reaction regions (from release to pin impact)
- FIR filters combined with *biorthogonal* wavelet decomposition and reconstruction extract 3rd-level approximation
- Additional filtering isolates tilt-response sinusoidal “chirp” signal indicative of ball “revving up” as it rolls down lane
- *Biorthogonal* 5th-level approximation recovers centripetal acceleration of ball
- Extrapolation techniques obtain meaningful data at segment fringes
- Metrics are then extracted from filtered, isolated waveforms

So what does that look like…?

Waveform Analysis

- We'll work with this data – ambient light and 3-axis acceleration:



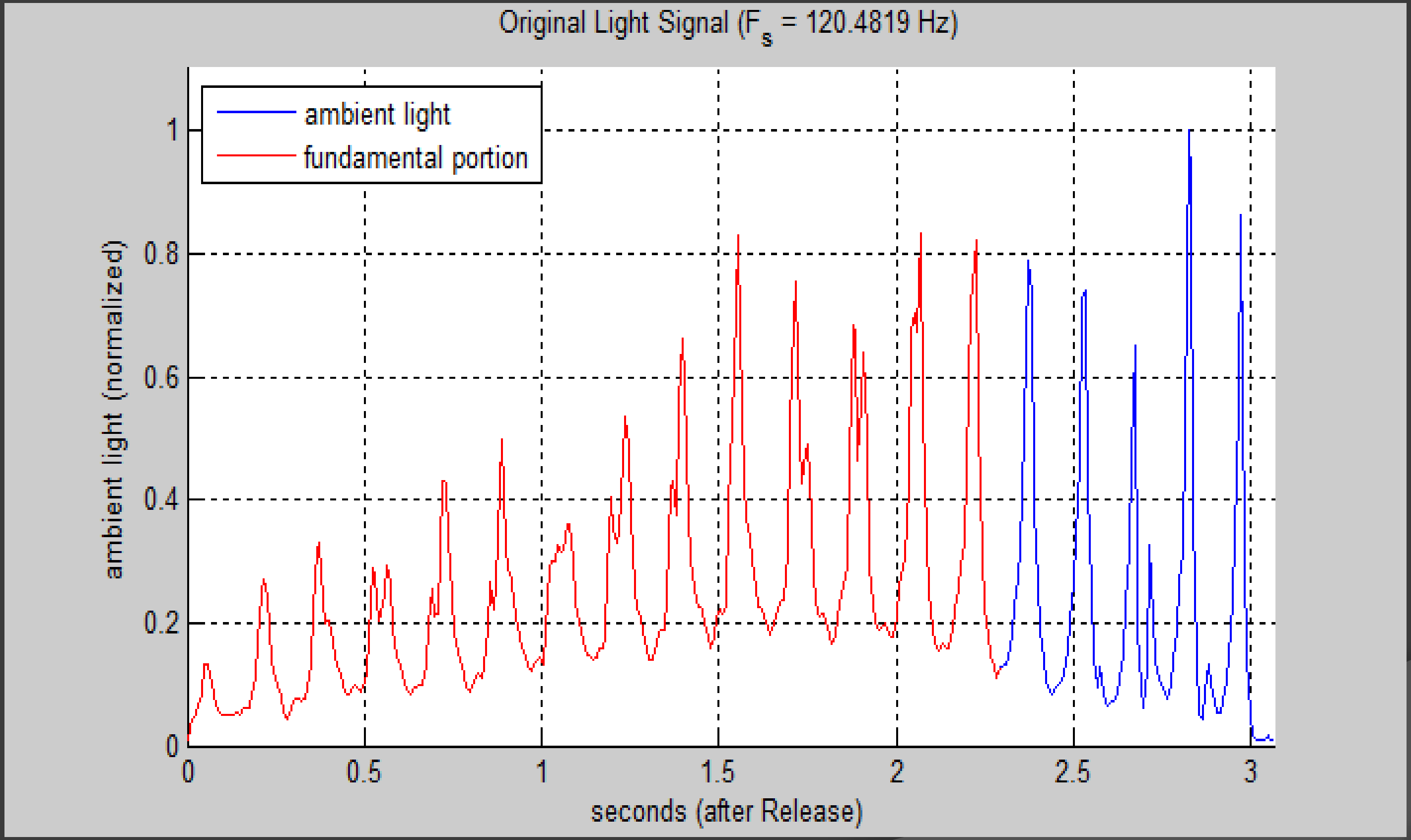
Waveform Analysis

Fundamental Frequency of Rotation (F_R)

- Ball is released with “spin” - initial angular velocity A_{v0}
- Ball generally spends at least $^1/_3$ time at or near A_{v0}
- Results in fundamental frequency of rotation – F_R
- Light waveform used to detect F_R
- Original F_s = 120 ± 1 Hz

Waveform Analysis
Fundamental Frequency of Rotation (F_R)

Find the fundamental from the ambient light waveform:



Waveform Analysis

Fundamental Frequency of Rotation (F_R)

- Normalize to 1
- Interpolate to F_s = 1000 Hz (1 ms resolution) for consistency and increased resolution
- Apply Hamming window to lessen edge effects
- Pad with sufficient 0's to achieve 0.5 RPM (0.00833 Hz) resolution in FFT

Waveform Analysis

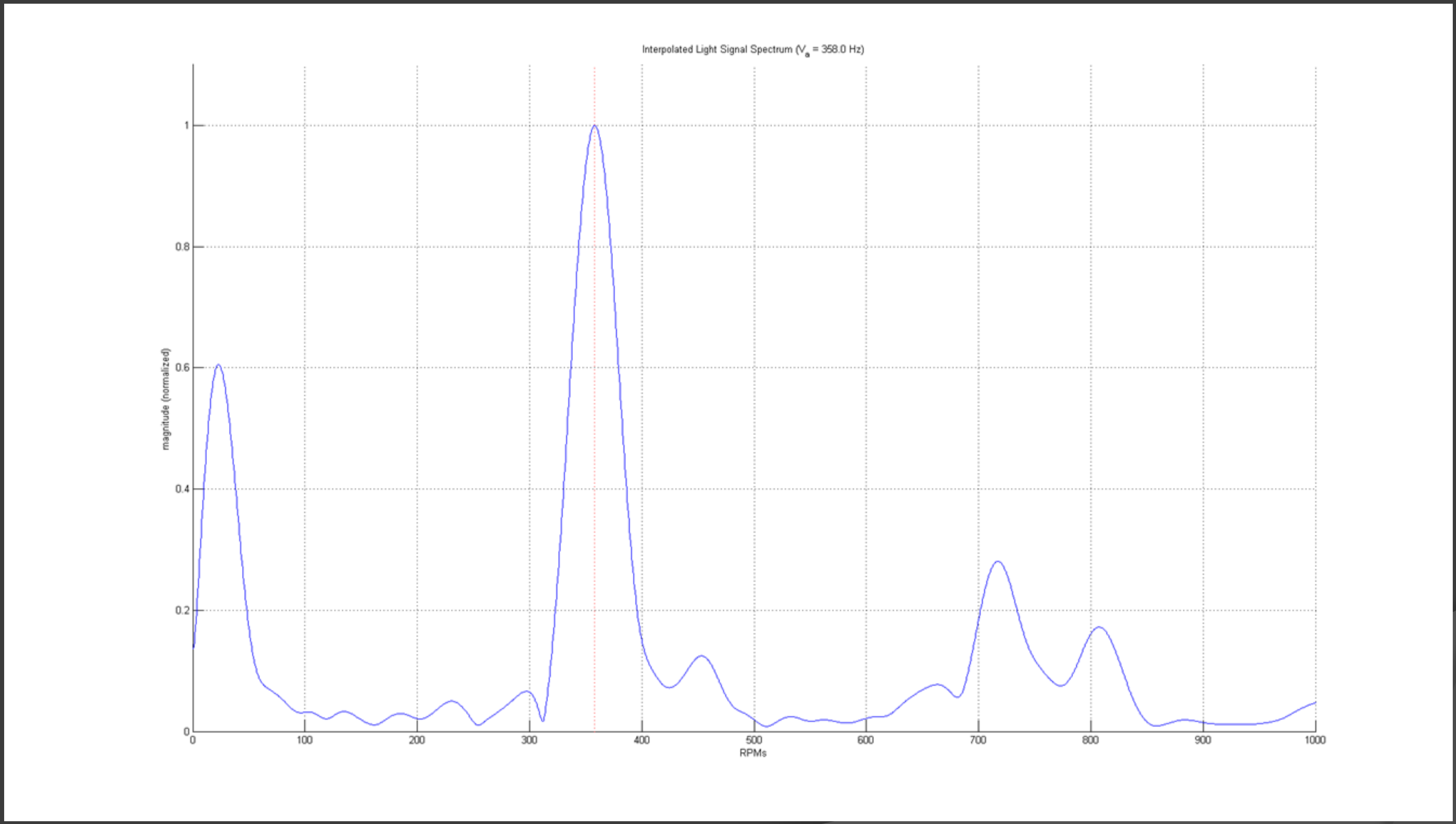
Fundamental Frequency of Rotation (F_R)

- Apply FFT
- Find max spectral component (ignoring DC and low-frequencies (<1 Hz)
- Now have F_R
- F_R is used to set FIR bandpass cut-off frequencies
- F_R is used to set interpolation level for ADXL waveforms – 2^n samples per Hz – for wavelet decomposition

Waveform Analysis

Fundamental Frequency of Rotation (F_R)

Frequency spectrum of interpolated ambient light waveform:



Waveform Analysis

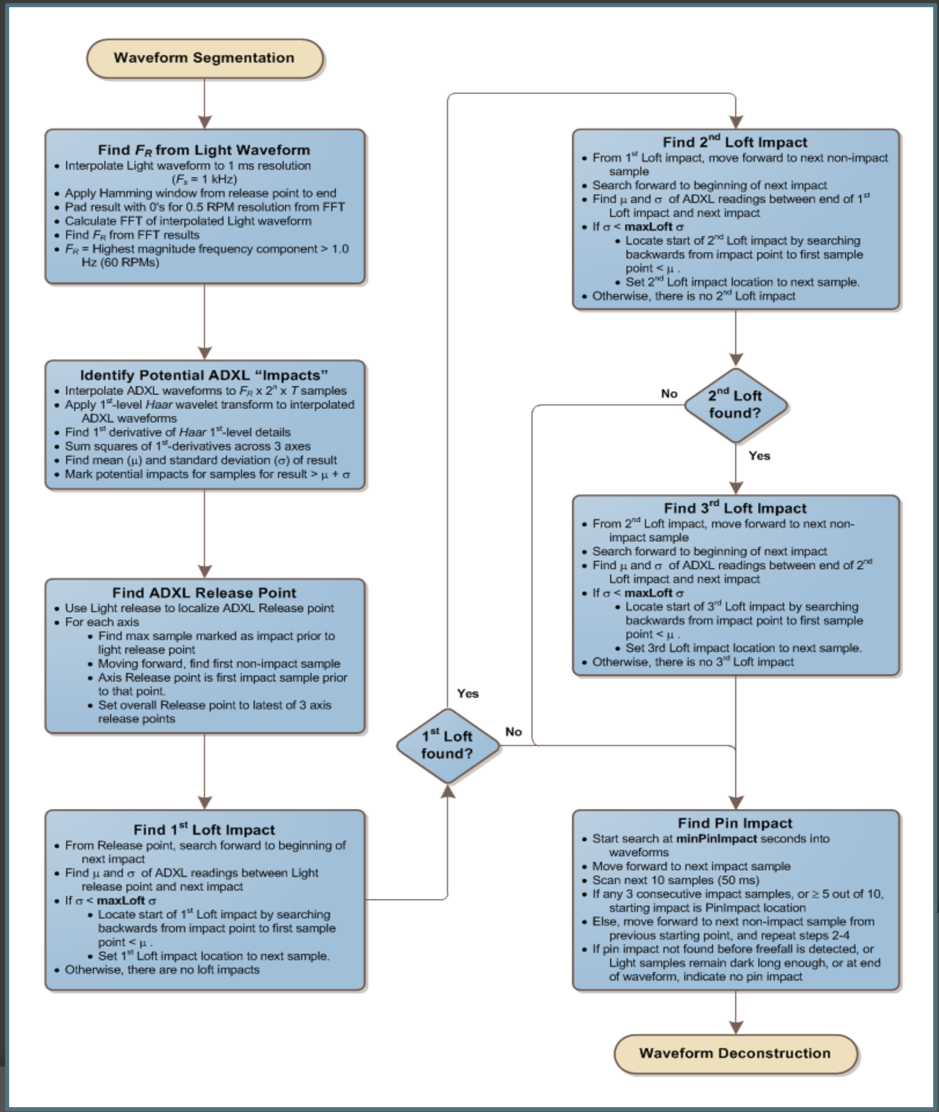
Finding Segment Boundaries

- Four segments
  - Approach: Time during which bowler delivers ball to lane
  - Release/Loft: From moment of Release until end of last loft "bounce"
  - Reaction: Time that ball is in continuous contact with lane, until impact with pins
  - Pin Impact: From 1^{st} impact w/pins to end of waveform
- Three segment boundaries
  - Release (Approach-Loft boundary)
  - Last Loft Impact (Loft-Reaction boundary)
  - First Pin Impact (Reaction-Pin Impact boundary
- Each segment boundary is marked by an "impact"

How do we find the segment boundaries?

Waveform Analysis

Finding Segment Boundaries



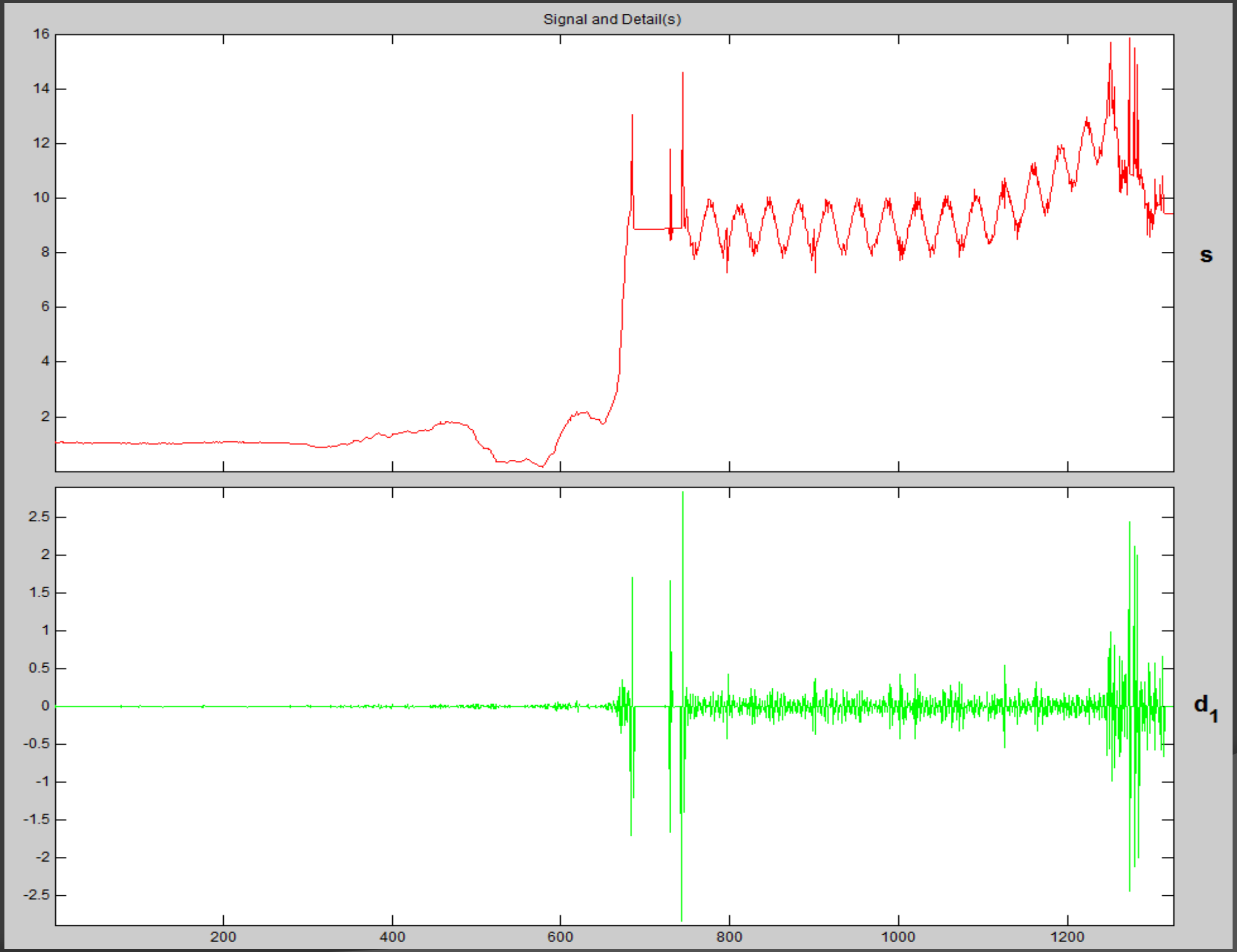
Waveform Analysis

Find Impacts Using Haar Wavelet

- Find vector magnitude of 3-axis waveform
- Wavelet theory is based on repeated 2-level decimation
- Interpolate so that there are 2^n samples per Hz of F_R
- Used $n = 5$, but could be higher
- Guarantees that some level of wavelet decomposition will hit integral harmonic of F_R
- *Haar* wavelet handles combination of flat components along with sudden transitions and impact spikes

Waveform Analysis
Find Impacts Using Haar Wavelet

1[st]-level *HAAR* wavelet details:

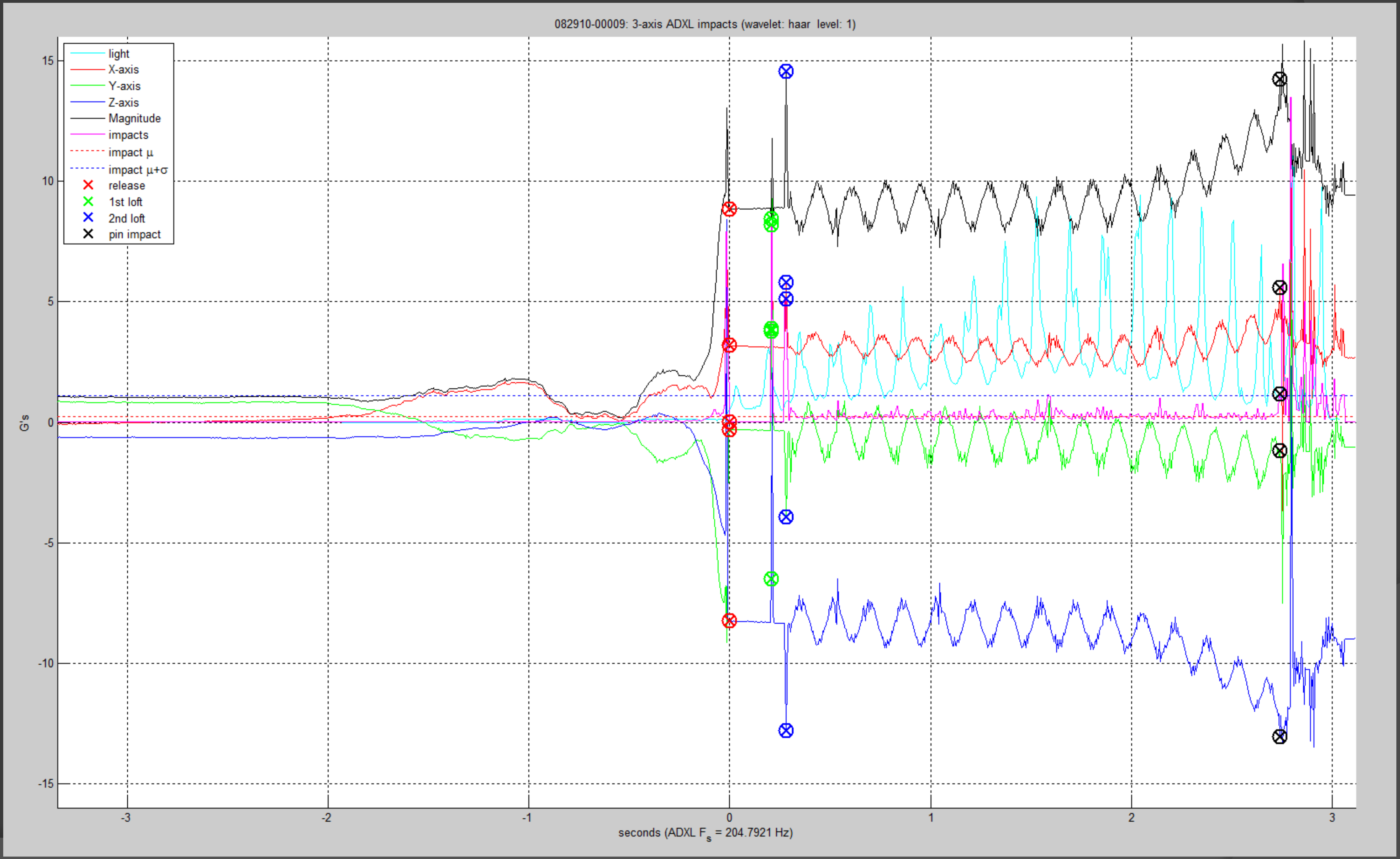


Find Segment Boundaries From Impacts

- Detection methods:
  - $\mu + \sigma$ across impact details
  - 1st derivative threshold detection
  - Duration also used for pin impact
  - Loft requires start and end – looks for "flat" portion of curve – far below mean (μ)

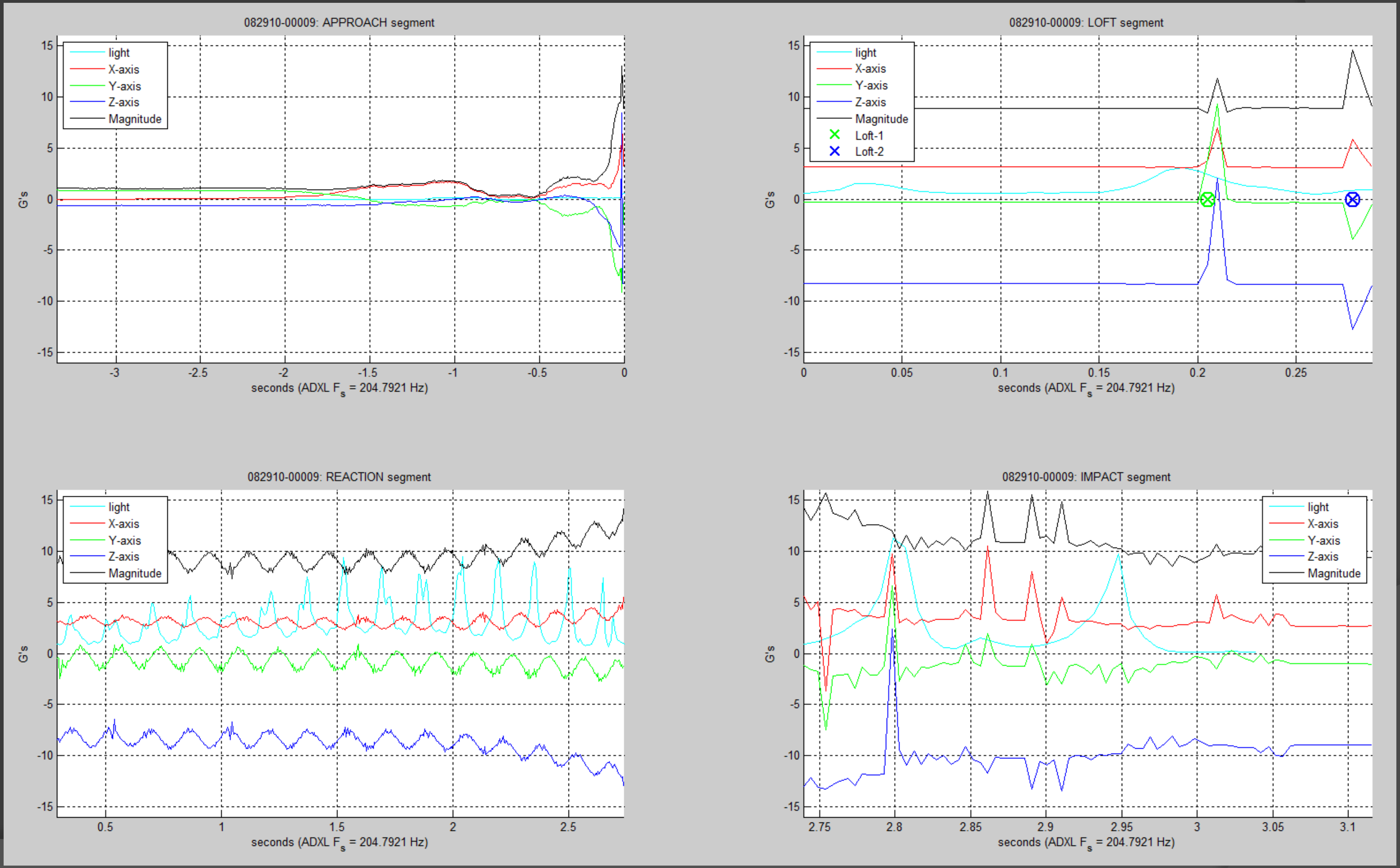
Waveform Analysis

Finding Segment Boundaries



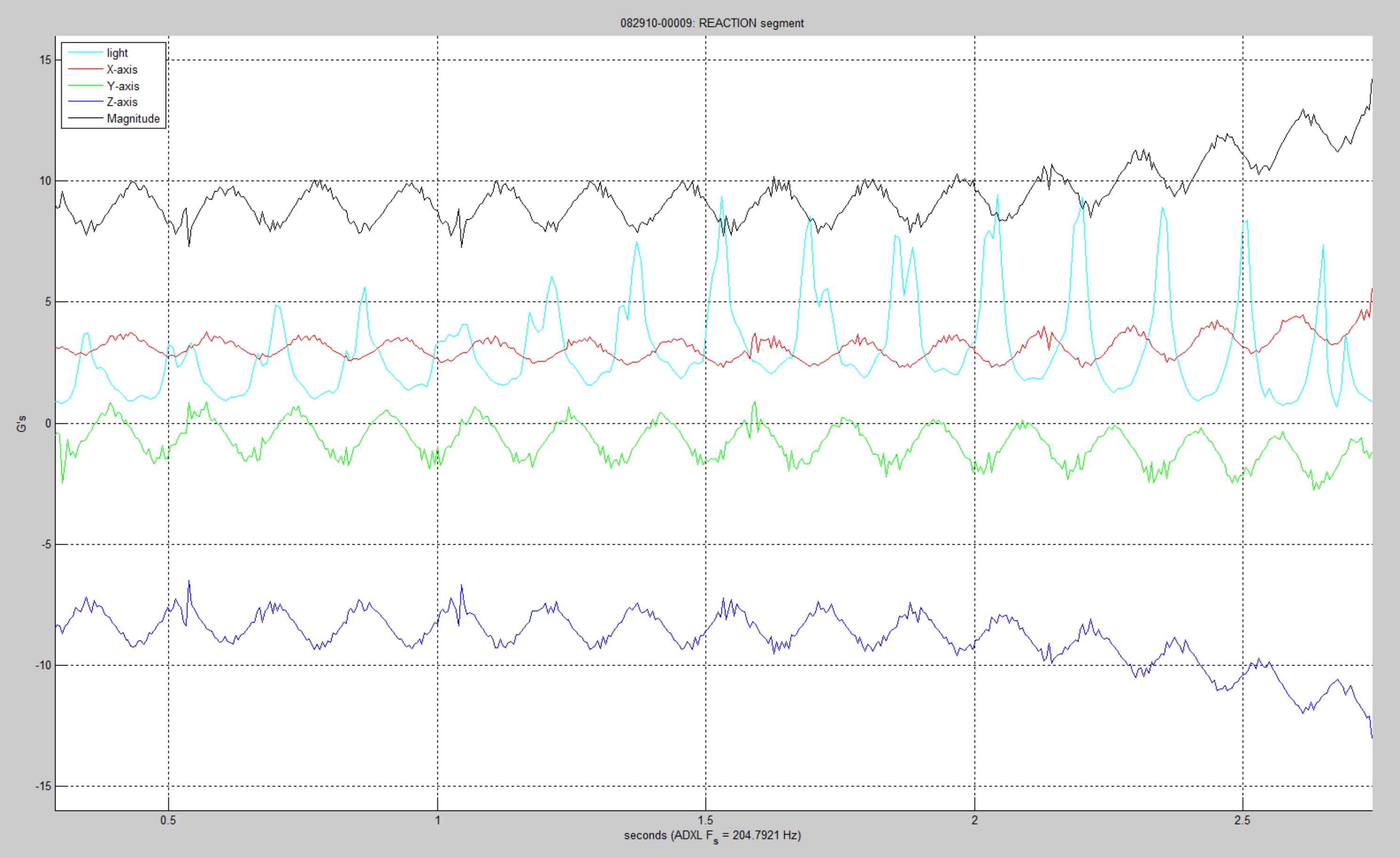
Waveform Analysis

ADXL Segments



Waveform Analysis

Reaction Segment



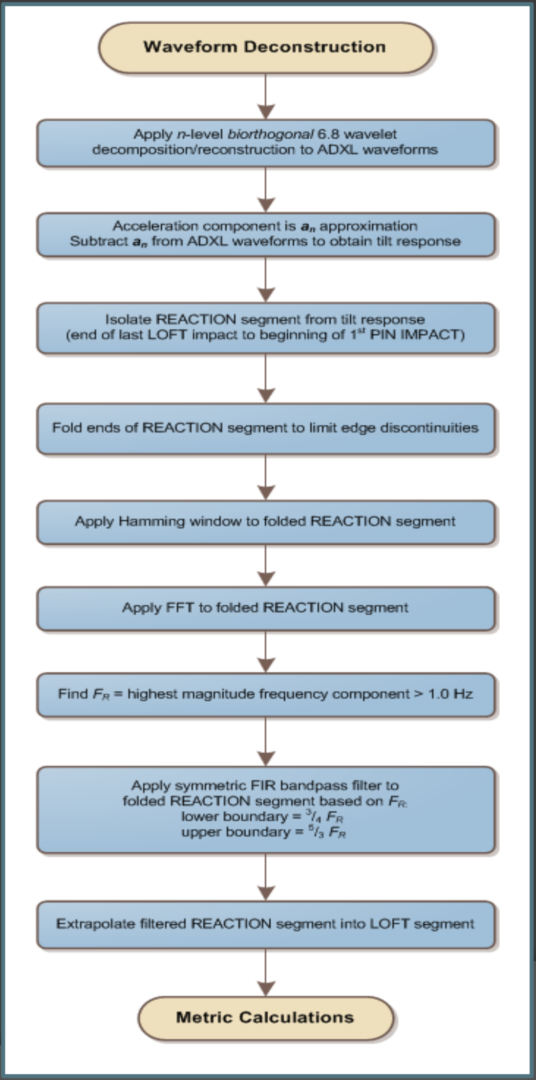
Waveform Analysis

Reaction Segment

- Comprised of three distinct components:
  - Centripetal acceleration: Generated by centripetal force due to ball’s rapid rotation
  - Sinusoidal tilt response: *SenseModule* rotating through gravitational field
  - High frequency noise: Irregularities in contact surfaces between ball and lane, *SenseModule* vibration in finger hole, and digital noise infiltrating ADXL345
- Must isolate centripetal acceleration from tilt-response, while filtering out noise
- Wavelets are perfect for this

Waveform Analysis

Reaction Segment Deconstruction

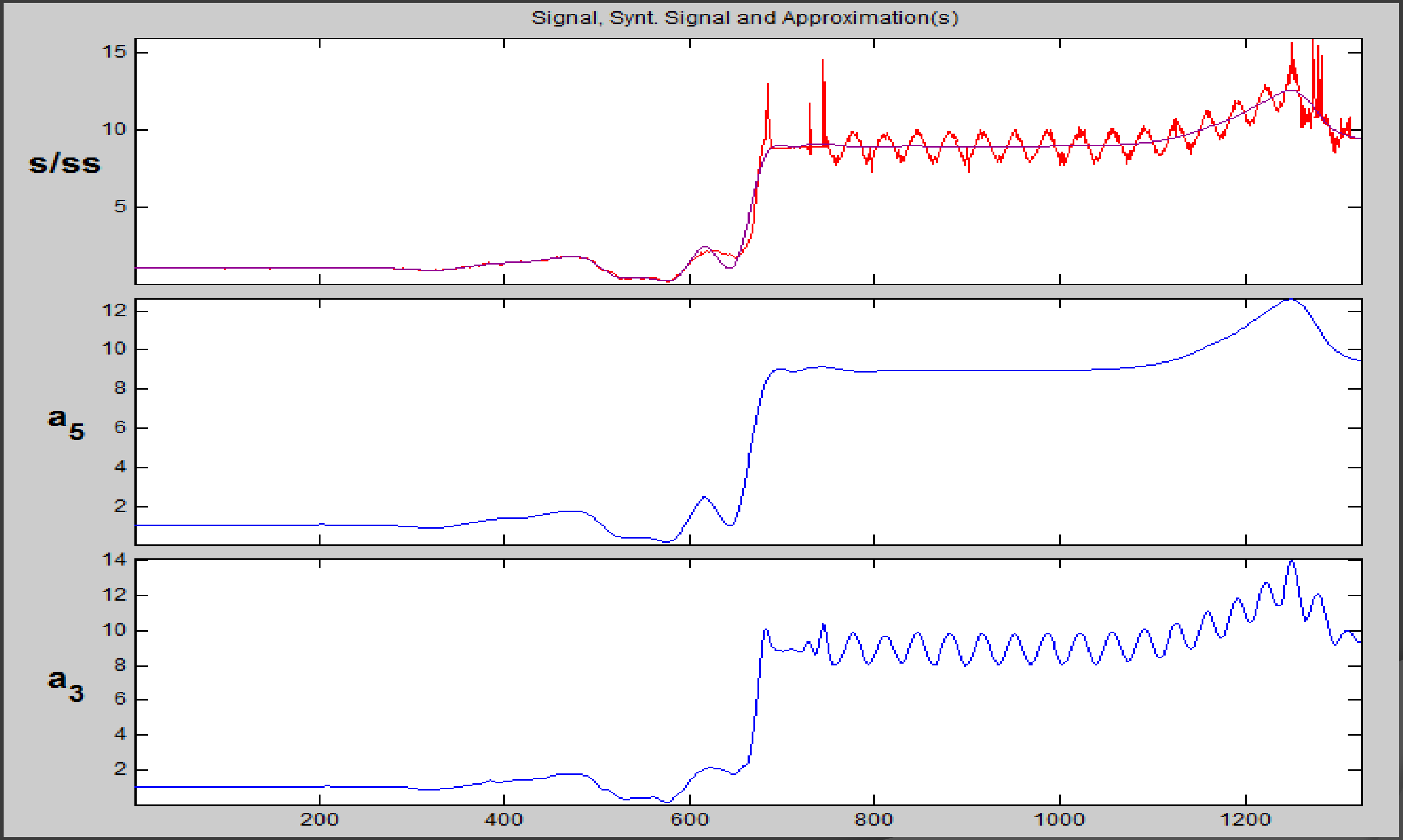


Reaction Segment Deconstruction

- Results of 3rd and 5th-level *biorthogonal* 6.8 wavelet decomposition and reconstruction
- 3rd-level approximation (a_3) yields tilt-response superimposed on centripetal acceleration
- 5th-level approximation (a_5) isolates centripetal acceleration
- $a_3 – a_5$ isolates tilt-response from centripetal acceleration

Waveform Analysis

Reaction Segment Deconstruction



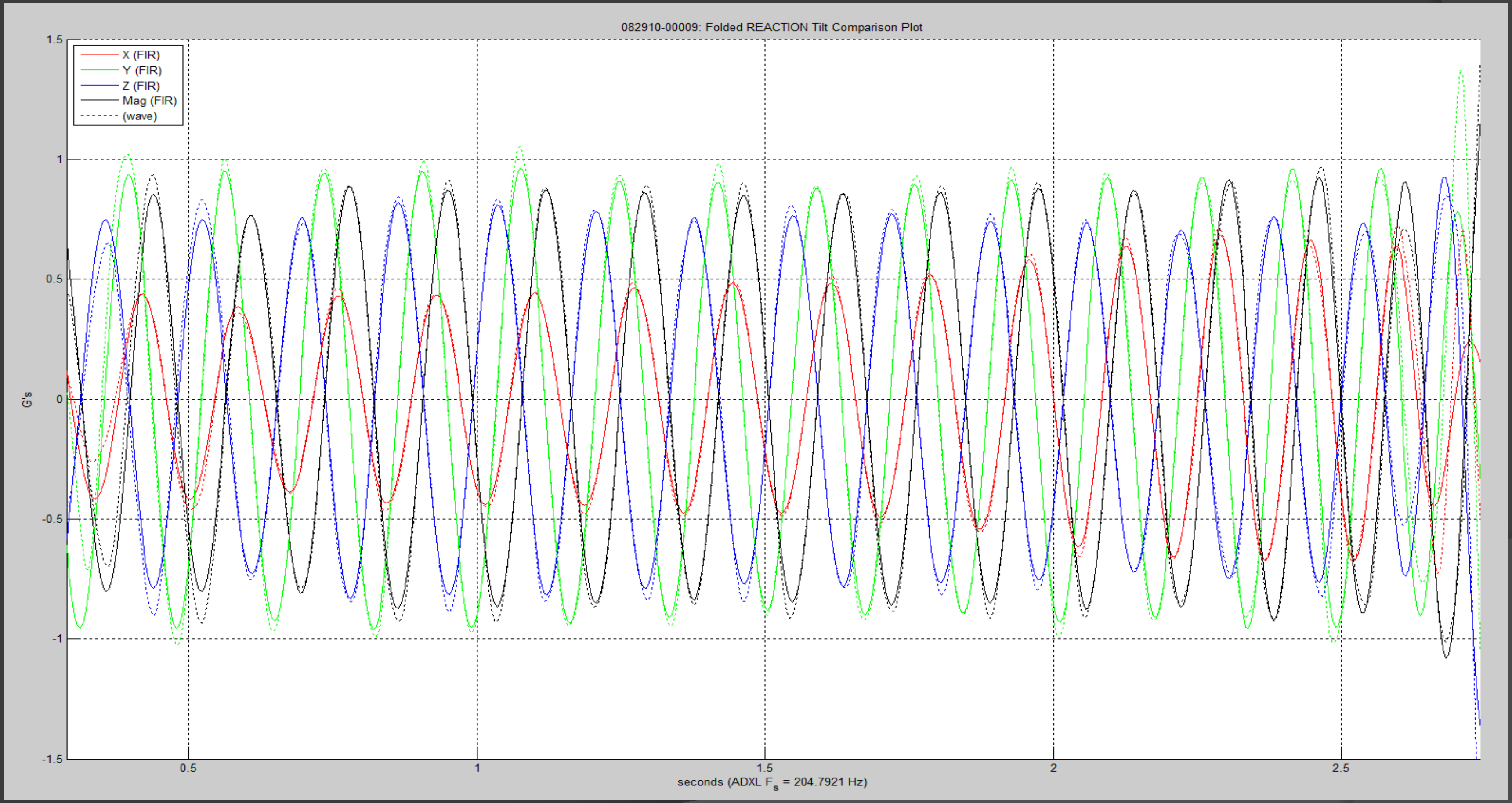
Reaction Tilt Response

- Compared filtering performance between FIR and wavelet techniques for tilt response
- FIR bandpass filtering uses F_R to establish pass band
  - Low band = 0.75 * F_R
  - High band = 1.67 * F_R
- Better results from FIR filter, with fringes “folded” into loft and pin impact regions before applying Hamming window
- Expected, since tilt response is highly sinusoidal
- Wavelets used to isolate segments and REACTION segment components (centripetal acceleration, tilt response), while FIR filter used to remove noise from tilt-response
- FIR results used for remainder of analysis for tilt response
- Following graph shows differences between FIR results and wavelet results

Waveform Analysis

Reaction Tilt Response

Difference between FIR results and Wavelet results
Wavelet results shown with dotted lines



Waveform Analysis

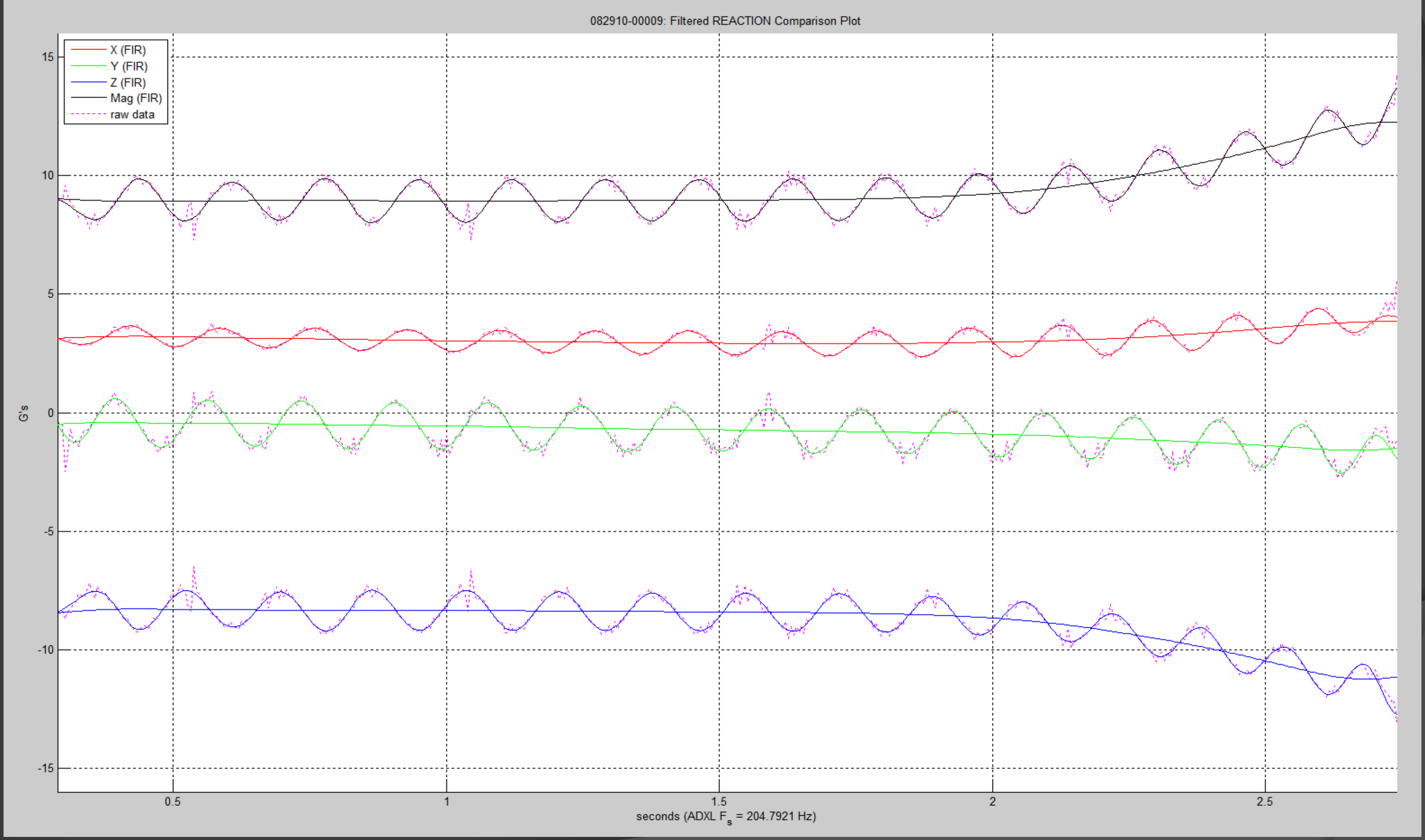
Centripetal Acceleration vs Tilt Response

- Filtered tilt response closely follows raw data
- Centripetal acceleration response should closely correspond with peak-to-peak tilt response
- Can use either/both to find instantaneous angular velocity
- For tilt response, peak-to-valley, and valley-to-peak times will give discrete angular velocity during each half-revolution
- Centripetal acceleration curve is continuous, but does not reflect true centripetal acceleration at surface of ball, since *SenseModule* is at bottom of finger hole
- Need to know depth of *SenseModule* to find angular velocity from centripetal acceleration

Waveform Analysis

Centripetal Acceleration vs Tilt Response

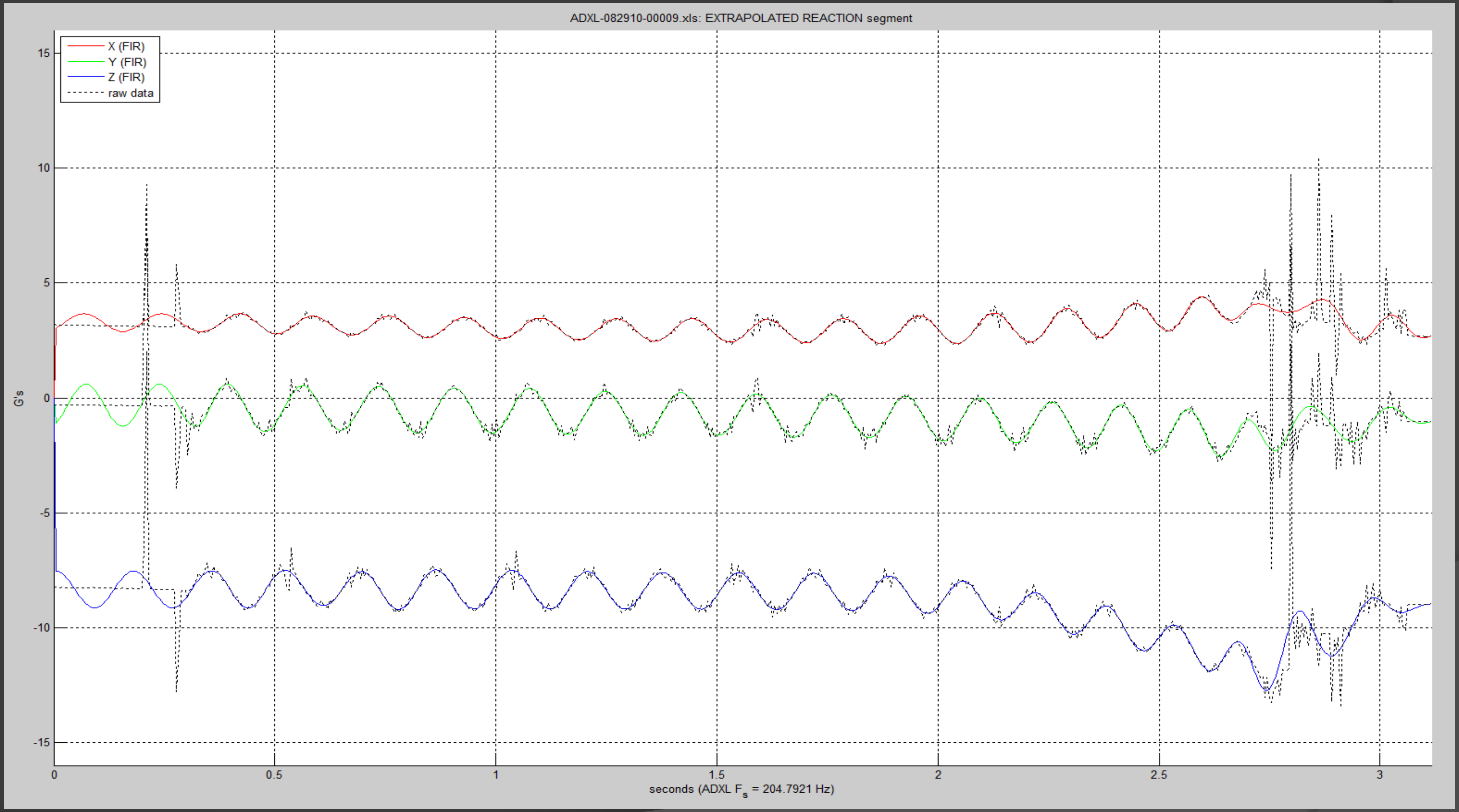
Raw data shown with dotted burgundy lines



Waveform Analysis

Reaction Tilt Response Extrapolation

Loft region extrapolated from start of Reaction region



RevMetrix Application

Waveform Metric Calculations

Physics of Bowling

- Ball is released with initial linear velocity greater than what its initial angular velocity (rate of rotation) indicates – the ball is skidding
- Friction acts to resolve discrepancy, transferring linear kinetic energy to angular kinetic energy
- As ball rolls down lane, it “revs” up – angular velocity increases as linear velocity decreases
- If linear and angular velocities reach equilibrium, the ball has stopped skidding – it has *rolled out* (angular velocity now drops along with linear velocity)
- Roll out is undesirable – ball is losing more energy
- Goal is to hit pocket at a sharp angle, while the ball is still skidding

Physics of Bowling

- Following release, only force acting on ball is friction between ball and lane
  - Wind resistance is negligible for dense sphere traveling at 15-20 mph
  - Lane is level (within 40/1000”), so no potential energy
- Kinetic energy has two components – linear kinetic energy and angular kinetic energy
- As ball grabs lane, angular kinetic energy increases, linear kinetic energy drops, in monotonic fashion
- If ball rolls out, both linear and kinetic energy drop

Waveform Metric Calculations

- Having isolated acceleration components, can now extract metrics of interest to bowler:
  - **RPMs:** Release, impact, and instantaneous angular velocity
  - **Revolutions:** Revolution count from release through pin impact, revolution location
  - **Ball Speed:** Release, impact, average, and instantaneous linear velocity
  - **Loft:** Distance (and approximate height)
  - **Breakpoint:** Location where ball starts to hook
  - **Axis Tilt:** Release, impact, and instantaneous deviation of axis from parallel with lane surface

Waveform Metric Calculations

- Known quantities:
  - Instantaneous angular velocity
  - Length of lane (60 feet)
  - Travel time from foul line to head pin
- Assumptions:
  - No loss of energy, energy is conserved
  - Friction acts solely to transfer linear kinetic energy to angular kinetic energy
  - No other forces act upon ball

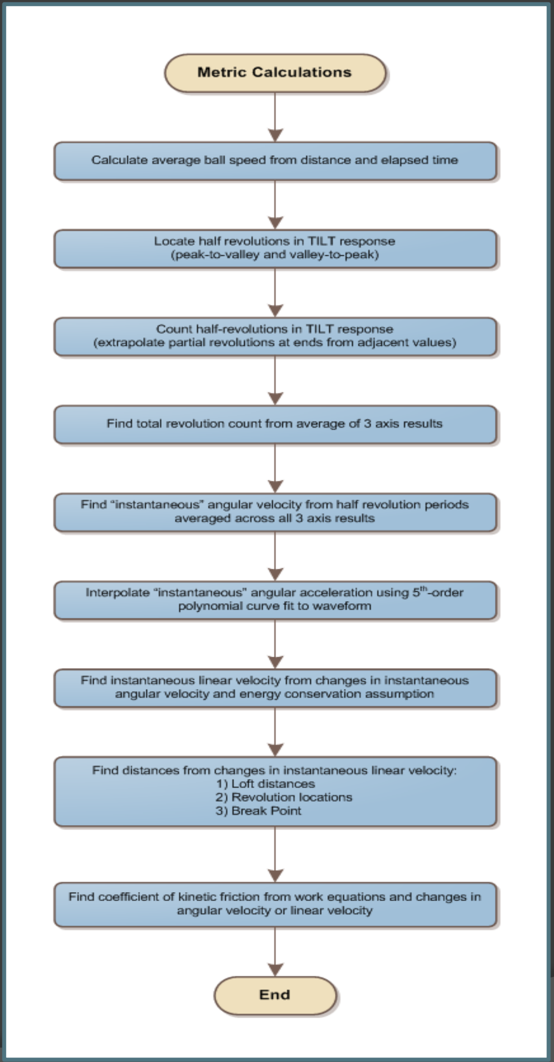
Waveform Metric Calculations

- Strategy – Part I:
  - Average linear velocity can be found from length of lane and transit time of ball
  - Changes in linear velocity are inversely proportional to changes in angular velocity, with no frictional loss
  - Find release linear velocity (v_0) from average velocity and changes in linear velocity
  - Develop converging iterative algorithm to find v_0

Waveform Metric Calculations

- Strategy – Part II:
  - Find instantaneous linear velocities (v_i) from release linear velocity (v_0)
  - Find distances (D_i) by integrating from $v_0 t_s$ to $v_i t_s$, for sample period t_s
  - Loft distance and revolution locations follow
  - Breakpoint distance follows from revolution locations
  - Coefficient of friction follows from changes in linear velocity or angular velocity

Waveform Metric Calculations



Waveform Metric Calculations

Some Physics and Math

- Total kinetic energy of ball:

$$K = K_{\omega} + K_{v}$$

- Linear kinetic energy of an object with mass *m* and linear velocity *v* is given by

$$K_v = \frac{1}{2}mv^2$$

- Angular kinetic energy of an object with mass *m*, moment of inertia *I*, and angular velocity ω is

$$K_{\omega} = \frac{1}{2}I\omega^2$$

Waveform Metric Calculations

Some Physics and Math

- Moment of inertia *I* of a sphere with mass *m* and radius *r* has the form

$$I = kmr^2$$

- Substituting

$$K_\omega = \frac{1}{2} I\omega^2 = \frac{1}{2} kmr^2\omega^2$$

- *k* can be determined from mass distribution within sphere
- USBC imposes limits on diameter and radius of gyration (*RoG*) of ball which limits *k* to

$$0.3201 \leq k \leq 0.4340$$

- Thus, *I* is restricted to

$$0.0402m \leq I \leq 0.0556m, \qquad (\mathrm{lb} - \mathrm{ft}^2)$$

Some Physics and Math

- Substituting the moment of inertia into the equation for angular kinetic energy, we get

$$K_\omega = \frac{1}{2} I\omega^2 = \frac{1}{2} kmr^2\omega^2$$

- Therefore, the total kinetic energy *K* of the ball is given by

$$K = \frac{1}{2} mv^2 + \frac{1}{2} kmr^2\omega^2$$

$$= \frac{m}{2}\left(v^2 + kr^2\omega^2\right)$$

Average Ball Speed

- Finding average ball speed is simple(?):

$$v_{ave} = \frac{D}{T_s} = \frac{60}{T_s}$$

T_s is transit time from release to first pin impact

- Not really that simple (assumptions):
  - Release occurs at foul line
  - Head pin is first impact
  - Ball travels from center of foul line and hits head pin head on
  - None of those assumptions is precisely true

Waveform Metric Calculations

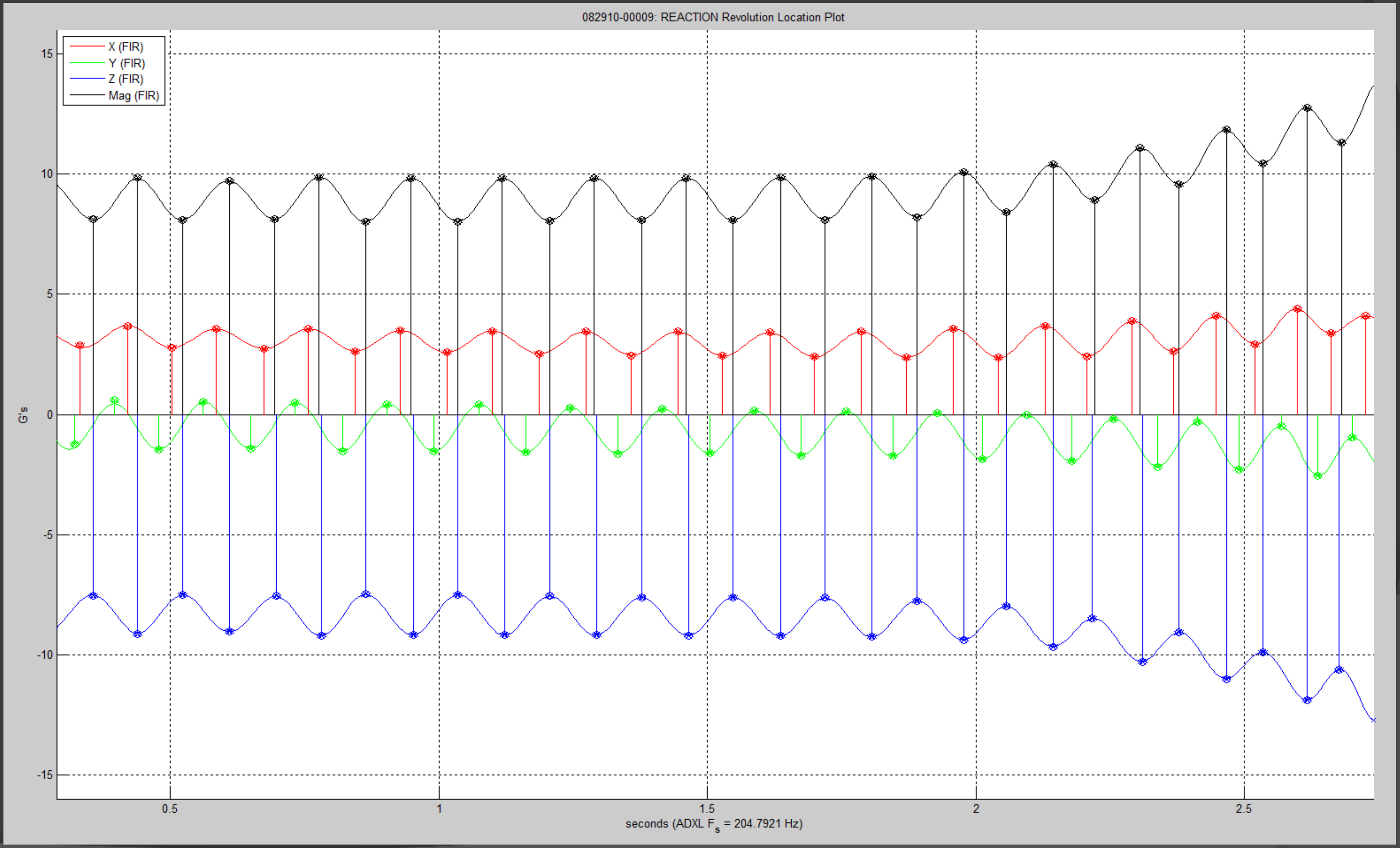
Revolution Location and Count

- Finding the revolutions is relatively simple:
  - Locate peaks and valleys from filtered tilt response waveform
  - Count peaks and valleys
  - Measure time between locations (difference in time stamps)
  - Find RPMs for each peak-valley, valley-peak pair
  - Average across all 3 axes yields better resolution
  - Can extrapolate from ½-revolutions adjacent to release and pin impact fringes

Waveform Analysis

Revolution Location

Automated Revolution Location Results



Waveform Metric Calculations

Instantaneous Angular Velocity (RPMs)

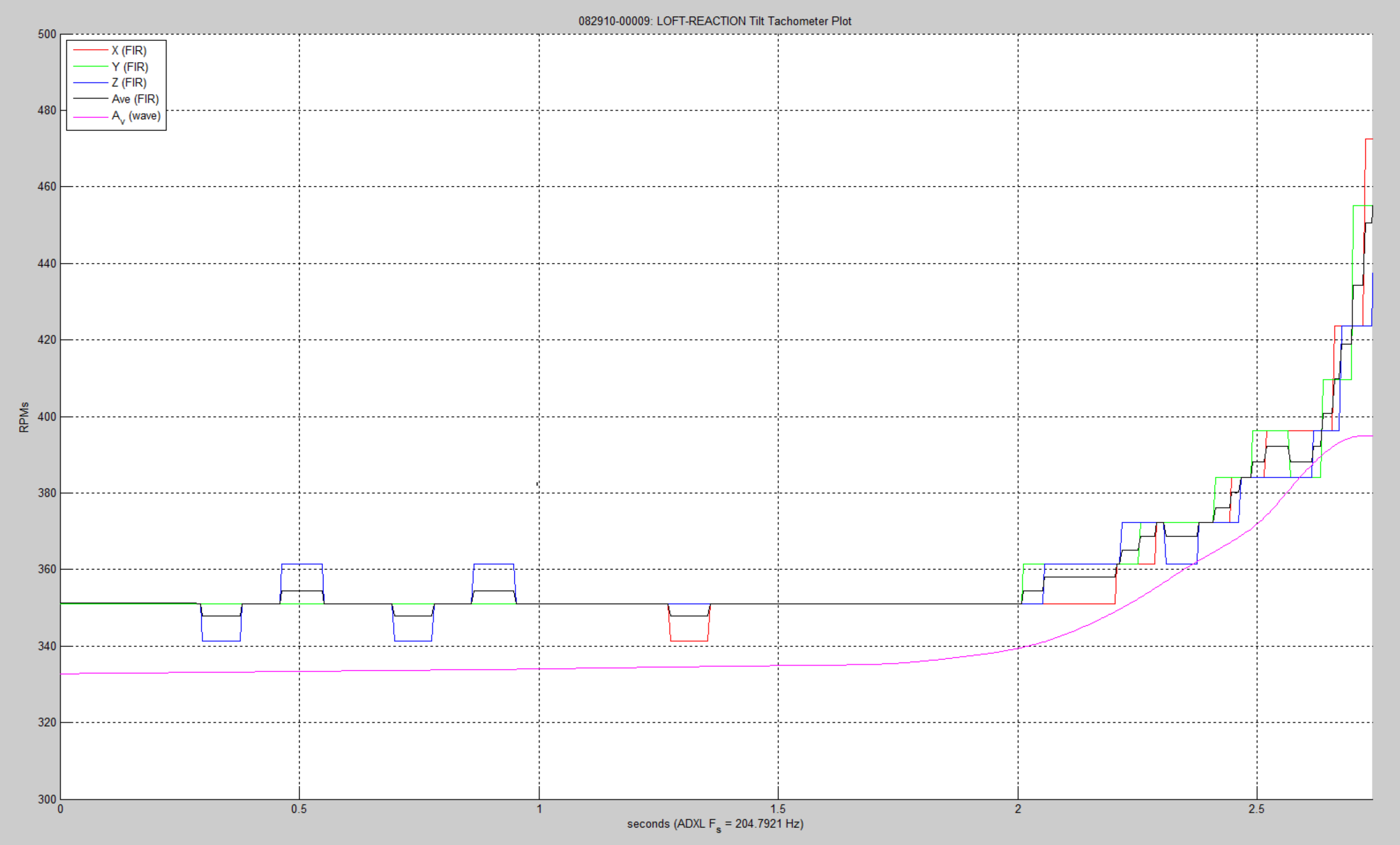
- Half-revolution angular velocities (in RPMs) for revolution containing peak *p* and valley *v* are given by

$$f_{(v-1,p)} = \frac{2}{t_p - t_{v-1}}(60) \quad \text{(valley − to − peak)}$$

$$f_{(p,v)} = \frac{2}{t_v - t_p}(60) \quad \text{(peak − to − valley)}$$

Waveform Analysis

Angular Velocity



Waveform Metric Calculations

Instantaneous Angular Velocity (RPMs)

- Earlier interpolation of $2^5 = 32$ samples per Hz of F_R yields resolution of ~10 RPMs
- Jitter of a single sample time translates to step change of 10 RPMs
- Interpolation of $2^8 = 256$ would increase resolution to ~1 RPM
- Could interpolate with 3rd to 5th-order polynomial curve-smoothing routine

Waveform Metric Calculations

Instantaneous Angular Velocity (RPMs)

- Extract angular velocity *f* from centripetal acceleration A_c

$$A_c = \frac{v^2}{r} = \frac{\omega r^2}{r} = \omega^2 r, \text{ where } v = \omega r = 2\pi f r$$

$$\omega = \sqrt{\frac{A_c}{r}}$$

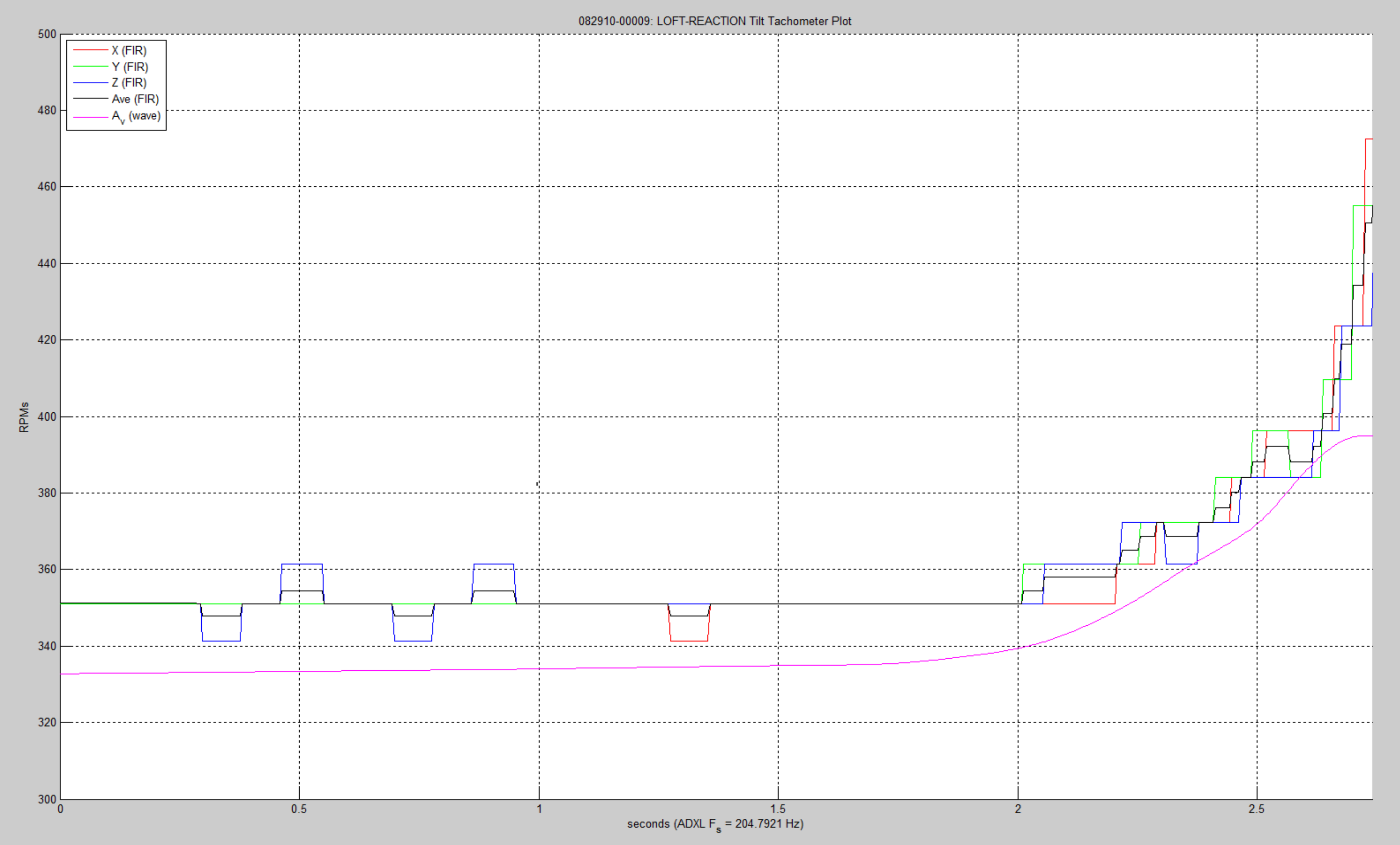
- Recall that *r* is not radius of ball, but distance of *SenseModule* from center of ball (*SenseModule* is at bottom of finger hole)
- The revolution rate *f* in RPMs is

$$f = \frac{\omega}{2\pi}(60) \text{ RPMs}$$

- During LOFT segment, A_c directly indicates centripetal acceleration, with ball in free fall during that time
- 5% discrepancy with tilt response angular velocity
- Have not yet adequately accounted for that discrepancy

Waveform Analysis

Angular Velocity



Waveform Metric Calculations

Instantaneous Linear Velocity

- Assuming constant energy, any increase in angular kinetic energy k_{ω} must come from linear kinetic energy k_v

$$K_0 = K_i = K_n, \text{for } 0 \le i \le n$$

- Letting K_i be the kinetic energy of ball during sample time i, then

$$K_i = \frac{m}{2}\left(v_i^2 + k\omega_i^2\right)$$

- Combining above two equations yields

$$\frac{m}{2}\left(v_0^2 + k\omega_0^2\right) = \frac{m}{2}\left(v_i^2 + k\omega_i^2\right)$$

- Solving for v_i in terms of v_0 and angular velocities

$$v_i = \sqrt{\left(v_0^2 + k\left(\omega_0^2 - \omega_i^2\right)\right)}$$

Waveform Metric Calculations

Instantaneous Linear Velocity

- Distance is integral of velocity with respect to time

$$D = \sum_{i=0}^{n-1} v_i t_s$$

- Distance D = 60 feet, and we know sample interval t_s, $\left(t_s = \frac{1}{f_s}\right)$

$$D = \sum_{i=1}^{n} \sqrt{(v_0^2 + k(\omega_0^2 - \omega_i^2))} \cdot t_s$$

Waveform Metric Calculations

Instantaneous Linear Velocity

- Find v_0 through converging iteration, first guess

$$v_0 = v_{ave} = \frac{D}{T_s} = \frac{60}{T_s}$$

- Final value for v_0 must be greater than v_{ave}, since ball slows down after release
- Evaluating at v_{ave} yields $D' < 60$

Instantaneous Linear Velocity

- Next guess

$$v_0' = v_0 + \frac{(60 - D')}{T}$$

- Term (60 – *D*') / *T* represents error in initial linear velocity distributed across each sample point *i*

Waveform Metric Calculations

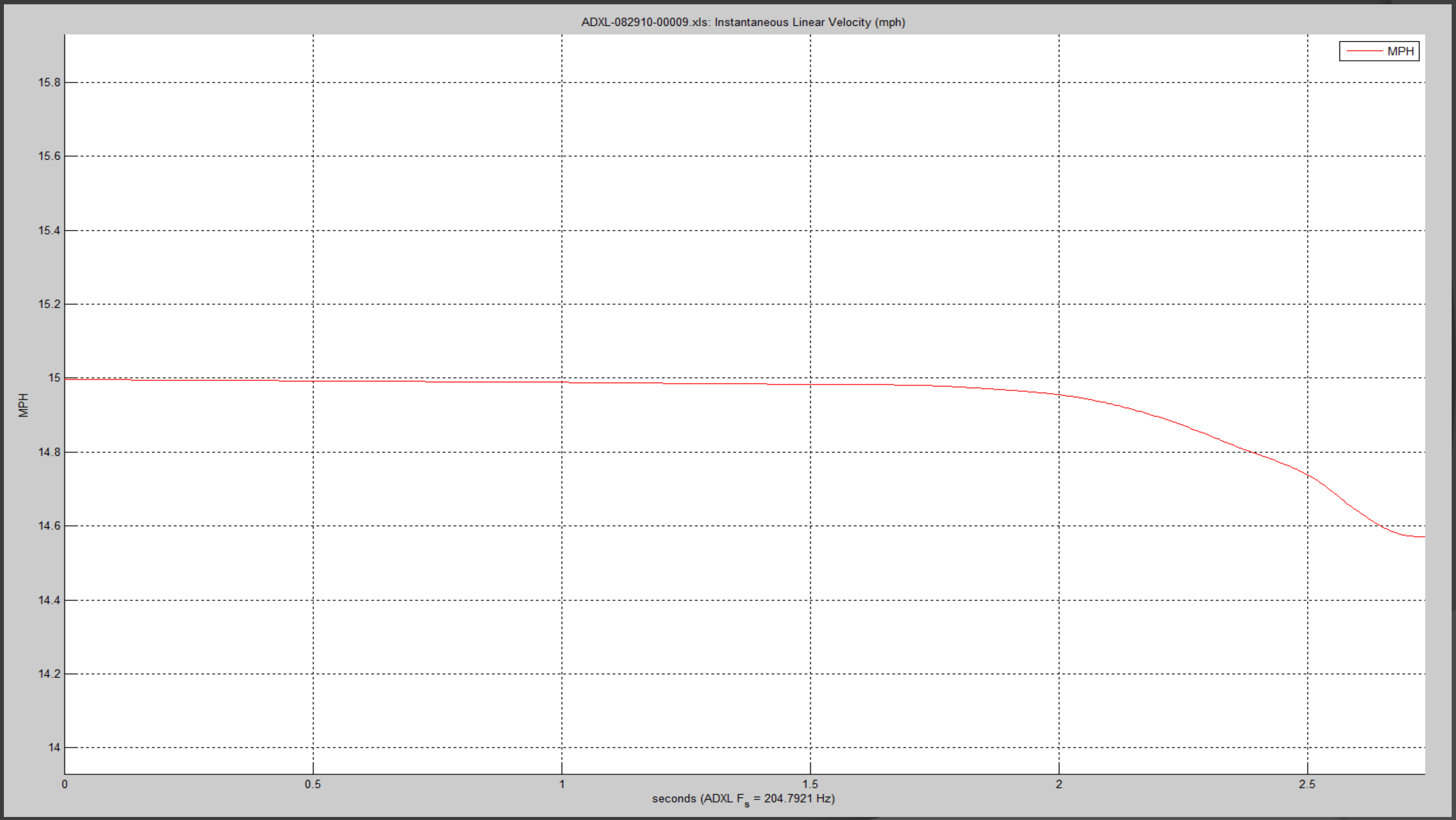
Instantaneous Linear Velocity

- Iterate until error $\varepsilon = 60 - D'$ is less than some acceptable error margin ε_0
- Now find all instantaneous linear velocities from v_0 and angular velocities

$$v_i = \sqrt{v_0^2 + k\left(\omega_0^2 - \omega_i^2\right)}$$

Waveform Analysis

Instantaneous Linear Velocity



Waveform Metric Calculations

Distance

- Obtain distance at any sample point from instantaneous linear velocity and sample period
- Find locations on lane for
  - Loft impacts
  - Revolutions
  - Breakpoint
- Distance D_k of ball from foul line at any time point k is

$$D_k = \sum_{i=0}^{k-1} v_i t_s$$

Waveform Analysis

- Validating Metric Calculations
  - Neither easy, nor cheap
  - "Simulate" with spinning fixture, and encoders
  - High speed video
  - C.A.T.S-instrumented facility
  - E.A.R.L. – USBC's bowling robot with C.A.T.S. instrumented lane
    - http://www.youtube.com/watch?v=s8yMFdPD68c
  - Throbot – Brunswick's version of the above
    - http://www.youtube.com/watch?v=QEeLNxlKRrU

Summary And Conclusions

Summary and Conclusions

- *SenseModule* is first of its kind
  - Fits unobtrusively in an existing finger hole
  - Autonomous and transparent operation
  - Does not effect balance of ball
  - Low cost, power, weight
- No similar commercial product on market – yet
  - "IMU" developed at U of Michigan around same time
  - Much more expensive – uses solid-state 3-axis gyroscopes
  - Uses separate hole for insertion
  - Much higher power requirements – rechargeable lithium-ion battery – 4 hours of run time

Summary and Conclusions

- Difficult problem
  - Dealing with power, size, and weight constraints
  - Creating robust autonomous operation
  - Extracting metrics from 3-axis acceleration data
  - How to use actual acceleration readings – rather than just tilt sensing aspect
  - Overall, more difficult than expected

Summary and Conclusions

- Commercialization
  - *SenseModule* platform appears to be viable as commercial product
  - To develop robust autonomous operation requires collecting data from a wide variety of bowlers and bowling styles
  - Involves major research effort
  - Development of *RevMetrixApp* is another major effort

Summary and Conclusions

- Intellectual property protection
  - Recent patent search revealed there is likely IP contained within the automated *SenseModule* functions:
    - Automated start-up and communications
    - False activation detection
    - Release detection and discrimination
    - Shutdown detection
  - Recent Supreme Court ruling on patenting algorithms most likely eliminates metric analysis algorithms from IP consideration

One Last Note

- As an engineer, there is nothing as satisfying as taking something of your own conception from idea to reality…
- You will likely spend most/all of your career getting paid to solve other people's problems…
- This effort has been more gratifying than any paycheck I have ever received…

One Last Note

- If you ever have the chance to do something similar…

TAKE IT

QUESTIONS…?