Concept2 PM5 BLE Communication Specification

Overview

This document provides the technical requirements for communicating with Concept2 Performance Monitor 5 (PM5) devices via Bluetooth Low Energy (BLE) in iOS applications.

Base Configuration

UUIDs

- **Base UUID**: `CE06XXXX-43E5-11E4-916C-0800200C9A66`
- Replace `XXXX` with the specific service/characteristic identifier

Device Discovery

- **Device Name**: `PM5 [SerialNumber]` (e.g., "PM5 430000000")
- **Appearance**: `0x0000`

Core BLE Services

1. GAP Service (`0x1800`)

Characteristic	UUID	Type	Permissions	Description
Device Name	`0x2A00`	String	READ	PM5 device name with serial
Appearance	`0x2A01`	UInt16	READ	Device appearance (0x0000)
Privacy Flag	`0x2A02`	UInt8	READ/WRITE	Privacy disabled (0x00)
Reconnect Address	`0x2A03`	Data	READ/WRITE	00:00:00:00:00:00
Connection Parameters	`0x2A04`	Data	READ	30ms intervals, 10s timeout

2. Device Information Service (`0x0010`)

Characteristic	UUID	Permissions	Data Type	Description
Model Number	`0x0011`	READ	String(16)	"PM5"
Serial Number	`0x0012`	READ	String(9)	Device serial number
Hardware Revision	`0x0013`	READ	String(3)	Hardware version
Firmware Revision	`0x0014`	READ	String(20)	Firmware version
Manufacturer	`0x0015`	READ	String(16)	"Concept2"
Erg Machine Type	`0x0016`	READ	UInt8	Machine type enum
ATT MTU	`0x0017`	READ	UInt16	23-512 bytes
LL DLE	`0x0018`	READ	UInt16	27-251 bytes

3. PM Control Service (`0x0020`)

Characteristic	UUID	Permissions	Description
PM Receive	`0x0021`	WRITE	Send CSAFE commands (max 20 bytes)
PM Transmit	`0x0022`	READ	Receive CSAFE responses (max 20 bytes)

4. Rowing Service (`0x0030`)

Core Data Characteristics

General Status (`0x0031`) - NOTIFY (19 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Bytes 3-5:   Distance (0.1m LSB)
Byte 6:      Workout Type (enum)
Byte 7:      Interval Type (enum)
Byte 8:      Workout State (enum)
Byte 9:      Rowing State (enum)
Byte 10:     Stroke State (enum)
Bytes 11-13: Total Work Distance (1m LSB)
```

Additional Status 1 (`0x0032`) - NOTIFY (17 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Bytes 3-4:   Speed (0.001m/s LSB)
Byte 5:      Stroke Rate (strokes/min)
Byte 6:      Heart Rate (bpm, 255=invalid)
Bytes 7-8:   Current Pace (0.01s LSB)
Bytes 9-10:  Average Pace (0.01s LSB)
Bytes 11-12: Rest Distance
Bytes 13-15: Rest Time (0.01s LSB)
Byte 16:     Erg Machine Type
```

Additional Status 2 (`0x0033`) - NOTIFY (20 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Byte 3:      Interval Count
Bytes 4-5:   Average Power (watts)
Bytes 6-7:   Total Calories
Bytes 8-9:   Split/Interval Avg Pace (0.01s LSB)
Bytes 10-11: Split/Interval Avg Power (watts)
Bytes 12-13: Split/Interval Avg Calories (cals/hr)
Bytes 14-16: Last Split Time (0.1s LSB)
Bytes 17-19: Last Split Distance (1m LSB)
```

Sample Rate Control (`0x0034`) - READ/WRITE (1 byte)

```
0: 1 second intervals
1: 500ms intervals (default)
2: 250ms intervals
3: 100ms intervals
```

Detailed Data Characteristics

Stroke Data (`0x0035`) - NOTIFY (20 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Bytes 3-5:   Distance (0.1m LSB)
Byte 6:      Drive Length (0.01m, max 2.55m)
Byte 7:      Drive Time (0.01s, max 2.55s)
Bytes 8-9:   Recovery Time (0.01s, max 655.35s)
Bytes 10-11: Stroke Distance (0.01m, max 655.35m)
Bytes 12-13: Peak Drive Force (0.1 lbs)
Bytes 14-15: Average Drive Force (0.1 lbs)
Bytes 16-17: Work Per Stroke (0.1 Joules)
Bytes 18-19: Stroke Count
```

Additional Stroke Data (`0x0036`) - NOTIFY (15 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Bytes 3-4:   Stroke Power (watts)
Bytes 5-6:   Stroke Calories (cal/hr)
Bytes 7-8:   Stroke Count
Bytes 9-11:  Projected Work Time (seconds)
Bytes 12-14: Projected Work Distance (meters)
```

Split/Interval Data (`0x0037`) - NOTIFY (18 bytes)

```
Bytes 0-2:   Elapsed Time (0.01s LSB)
Bytes 3-5:   Distance (0.1m LSB)
Bytes 6-8:   Split/Interval Time (0.1s LSB)
Bytes 9-11:  Split/Interval Distance (1m LSB)
Bytes 12-13: Interval Rest Time (1s LSB)
Bytes 14-15: Interval Rest Distance (1m LSB)
Byte 16:     Split/Interval Type
Byte 17:     Split/Interval Number
```

End of Workout Summary (`0x0039`) - NOTIFY (20 bytes)

```
Bytes 0-1:   Log Entry Date
Bytes 2-3:   Log Entry Time
Bytes 4-6:   Elapsed Time (0.01s LSB)
Bytes 7-9:   Distance (0.1m LSB)
Byte 10:     Average Stroke Rate
Byte 11:     Ending Heart Rate
Byte 12:     Average Heart Rate
Byte 13:     Min Heart Rate
Byte 14:     Max Heart Rate
Byte 15:     Drag Factor Average
Byte 16:     Recovery Heart Rate
Byte 17:     Workout Type
Bytes 18-19: Average Pace (0.1s LSB)
```

Heart Rate Belt Info (`0x003B`) - NOTIFY (6 bytes)

```
Byte 0:      Manufacturer ID
Byte 1:      Device Type
Bytes 2-5:   Belt ID (32-bit)
```

Multiplexed Information (`0x0080`) - NOTIFY (up to 20 bytes)

- First byte is identifier (0x31-0x3F)
- Remaining bytes contain characteristic data
- Only multiplexes when specific characteristic notifications are disabled

Data Type Conversions

Multi-byte Data Construction

Two-byte values (Little Endian)

```
swift
```

```swift
let value = UInt16(data[0]) | (UInt16(data[1]) << 8)
```

Three-byte values (Little Endian)

```swift
let value = UInt32(data[0]) | (UInt32(data[1]) << 8) | (UInt32(data[2]) << 16)
```

Four-byte values (Little Endian)

```swift
let value = UInt32(data[0]) | (UInt32(data[1]) << 8) | (UInt32(data[2]) << 16) | (UInt32(data[3]) << 24)
```

Time Conversions

- **Display Time**: 1 second resolution (truncated from 0.01s)
- **Storage Time**: 0.1 second resolution (rounded from 0.01s)
- **BLE Time**: 0.01 second LSB for elapsed time, 0.1s LSB for split times

Distance Conversions

- **Meters**: 1m resolution (no rounding)
- **BLE Distance**: 0.1m LSB for distance, 1m LSB for splits

Pace Conversions

```swift
// Watts to Pace (seconds per meter)
func wattsToPace(_ watts: Double) -> Double {
    return pow(2.8 / watts, 1.0/3.0)
}

// Pace to 500m split time
func paceTo500m(_ pace: Double) -> Double {
    return pace * 500.0
}

// Calories/hr to Pace
func caloriesToPace(_ calories: Double) -> Double {
    let watts = (calories - 300.0) / (4.0 * 0.8604)
    return wattsToPace(watts)
}
```

CSAFE Protocol (Control Service)

Frame Structure

```
Standard Frame: [0xF1] [Frame Contents] [Checksum] [0xF2]
Extended Frame: [0xF0] [Dest] [Src] [Frame Contents] [Checksum] [0xF2]
```

Byte Stuffing

Replace these bytes in frame contents:

- `0xF0` → `0xF3, 0x00`
- `0xF1` → `0xF3, 0x01`
- `0xF2` → `0xF3, 0x02`
- `0xF3` → `0xF3, 0x03`

Common Commands

Get Status

```
Command: [0x80]
Response: [Status Byte]
```

Set Workout Type

```
Command: [0x76] [0x02] [0x01] [0x01] [WorkoutType]
WorkoutType: 0=JustRow, 1=JustRow+Splits, 2=FixedDist, etc.
```

Go to Workout Screen

```
Command: [0x76] [0x04] [0x13] [0x02] [0x01] [0x01]
```

Enumerations

Workout Types

```swift
```

```
enum WorkoutType: UInt8 {
    case justRowNoSplits = 0
    case justRowSplits = 1
    case fixedDistNoSplits = 2
    case fixedDistSplits = 3
    case fixedTimeNoSplits = 4
    case fixedTimeSplits = 5
    case fixedTimeInterval = 6
    case fixedDistInterval = 7
    case variableInterval = 8
    case variableUndefinedRest = 9
    case fixedCalorieSplits = 10
    case fixedWattMinuteSplits = 11
    case fixedCalsInterval = 12
}
```

Workout States

swift

```
enum WorkoutState: UInt8 {
    case waitToBegin = 0
    case workoutRow = 1
    case countdownPause = 2
    case intervalRest = 3
    case intervalWorkTime = 4
    case intervalWorkDistance = 5
    case workoutEnd = 10
    case terminate = 11
    case workoutLogged = 12
    case rearm = 13
}
```

Rowing States

swift

```
enum RowingState: UInt8 {
    case inactive = 0
    case active = 1
}
```

Stroke States

swift

```
enum StrokeState: UInt8 {
    case waitingForWheelToReachMinSpeed = 0
    case waitingForWheelToAccelerate = 1
    case driving = 2
    case dwellingAfterDrive = 3
    case recovery = 4
}
```

iOS Implementation Notes

BLE Connection

swift

```
// Service UUIDs
let deviceInfoServiceUUID = CBUUID(string: "CE060010-43E5-11E4-916C-0800200C9A66")
let controlServiceUUID = CBUUID(string: "CE060020-43E5-11E4-916C-0800200C9A66")
let rowingServiceUUID = CBUUID(string: "CE060030-43E5-11E4-916C-0800200C9A66")

// Key characteristics
let generalStatusUUID = CBUUID(string: "CE060031-43E5-11E4-916C-0800200C9A66")
let strokeDataUUID = CBUUID(string: "CE060035-43E5-11E4-916C-0800200C9A66")
let controlReceiveUUID = CBUUID(string: "CE060021-43E5-11E4-916C-0800200C9A66")
```

Data Parsing Example

swift

```
func parseGeneralStatus(_ data: Data) -> GeneralStatus {
    let elapsedTime = UInt32(data[0]) | (UInt32(data[1]) << 8) | (UInt32(data[2]) << 16)
    let distance = UInt32(data[3]) | (UInt32(data[4]) << 8) | (UInt32(data[5]) << 16)
    let workoutType = WorkoutType(rawValue: data[6]) ?? .justRowNoSplits

    return GeneralStatus(
        elapsedTime: Double(elapsedTime) / 100.0, // Convert from 0.01s
        distance: Double(distance) / 10.0,       // Convert from 0.1m
        workoutType: workoutType,
        // ... parse remaining fields
    )
}
```

Important Considerations

1. **Connection Parameters**: Use 30ms connection intervals for optimal performance
2. **MTU Size**: Negotiate larger MTU (up to 512 bytes) for better throughput

3. **Notification Management**: Enable only needed characteristics to reduce bandwidth
4. **Sample Rate**: Set appropriate sample rate (default 500ms) based on application needs
5. **Data Validation**: Always validate data ranges and handle invalid values (e.g., HR = 255)
6. **Error Handling**: Implement robust error handling for BLE disconnections and data parsing

Battery Considerations

- PM5 uses 2xD cell batteries
- Monitor battery level through status notifications
- Implement power-efficient connection parameters
- Handle low battery disconnections gracefully

Testing Recommendations

- Test with various workout types and states
- Verify data parsing with extreme values
- Test connection stability during extended sessions
- Validate pace/power calculations against PM5 display