# Nano Ardule Drum System – PC 전처리 툴체인 가이드 (ADT v2.2)

## 1. 개요

이 문서는 Nano Ardule Drum Player용 드럼 패턴 데이터를 준비하기 위한 전체 전처리 과정과 Python 도구들의 사용법을 설명합니다. MIDI → ADT(Ardule Drum Text) 변환, ADT → ADP(바이너리 캐시) 변환, 인덱스 생성, ARR(곡 구성) 파일 작성 과정을 포함합니다.

## 2. 디렉토리 구조

PC/  
 ├─ MID/ # 2마디 길이의 MIDI 세그먼트  
 ├─ ADT/ # 변환된 ADT (백업/편집용)  
 ├─ SD/  
 │ ├─ PATTERNS/ # 최종 ADP 파일 (장치에서 읽음)  
 │ ├─ SONGS/ # ARR/APT 곡 구성 파일  
 │ └─ SYSTEM/ # INDEX.TXT, SETTINGS.CFG 등  
 └─ tools/ # Python 스크립트 (midi2adt.py, adt2adp.py, mkindex.py, arr\_make.py)

## 3. 처리 파이프라인

1) 세분화 분석(Subdivision Analysis): 트리플렛 또는 스트레이트 판별  
 python mid2report\_integrated.py 6ROCK.MID  
2) Type-0 MIDI 파일을 2마디 단위로 분할  
 python split\_drums\_2bar\_save\_v4a.py 6ROCK.MID  
3) MIDI → ADT 변환  
 python mid2adt.py --in-dir ./MID --out-dir ./ADT --recursive  
 (트리플렛 세트의 경우: --grid 16T --length 48)  
4) ADT → ADP 변환  
 python adt2adp.py --in-dir ./ADT --out-dir ./SD/PATTERNS --recursive --overwrite  
5) INDEX.TXT 생성  
 python mkindex.py --patterns ./SD/PATTERNS --out ./SD/SYSTEM/INDEX.TXT  
6) ARR 파일 작성  
 python arr\_make.py --title 'Rock Demo' --tempo 120 --in ./lists/rock\_demo.csv --out ./SD/SONGS/ROCK\_DEMO.ARR

## 4. ADT v2.2 사양

ADT(Ardule Drum Text) v2.2는 다양한 박자 세분화를 지원하는 사람이 읽을 수 있는 드럼 패턴 포맷입니다.  
주요 항목:  
 - LENGTH: 24, 32, 48 스텝 (2마디)  
 - GRID: 16(스트레이트), 8T(8분 트리플렛), 16T(16분 트리플렛)  
 - SLOTS: 최대 12개의 드럼 악기 슬롯  
 - ORIENTATION: STEP 또는 SLOT (자동 정규화)  
  
예시 헤더:  
; ADT v2.2  
NAME=ROCK\_MAIN  
TIME\_SIG=4/4  
GRID=16T  
LENGTH=48  
SLOTS=12  
KIT=GM\_STD  
ORIENTATION=STEP

## 5. ADP v2.2 바이너리 캐시 포맷

ADP(Ardule Drum Pattern) 파일은 ADT를 파싱한 결과를 바이너리 형태로 저장한 것으로, 장치에서 빠르게 읽어 들일 수 있습니다.  
헤더 구조 (<리틀엔디언>):  
 magic[4]='ADP2', version=22, grid(0=16,1=8T,2=16T), length, slots, ppqn=96, swing, tempo, reserved, adt\_crc16, payload\_bytes  
Payload: 각 스텝별로 count(1B) + (slot<<2|acc) × count

## 6. 각 스크립트 설명

### midi2adt.py

2마디 단위 MIDI 드럼 세그먼트를 ADT v2.2 형식으로 변환합니다.  
노트 이벤트와 velocity, 타이밍을 12슬롯 GM 드럼 킷에 매핑합니다.

### adt2adp.py

ADT 텍스트를 ADP 바이너리 캐시로 변환합니다.  
STEP/SLOT 자동 정규화, CRC16 기록, GRID=16/8T/16T 및 LENGTH=24/32/48 지원.

### mkindex.py

ADP 파일들을 스캔하여 /SYSTEM/INDEX.TXT를 생성합니다.  
헤더 정보를 추출하고, 파일명 접두어로 장르를 분류하여 요약합니다.

### arr\_make.py

패턴 이름과 반복 횟수로 구성된 CSV나 명령행 입력으로 ARR 곡 구성 파일을 만듭니다.  
예: RCK\_P001,4 → P01=RCK\_P001.ADP, REPEAT=4

## 7. '16T' 그리드(트리플렛)에 대하여

'16T'는 '16분 셋잇단음표(16th-note triplet)'의 약어로, 한 박(quarter note)을 6등분한 세밀한 리듬 단위를 의미합니다.  
셋잇단 계열의 스윙/셔플 리듬에서 사용되며, 4/4 한 마디를 24스텝(8T) 또는 48스텝(16T)으로 표현할 수 있습니다.

## 8. SD 카드 구성 예시

/PATTERNS/ — ADP 패턴 파일  
/SONGS/ — ARR 재생 시퀀스  
/SYSTEM/ — INDEX.TXT, SETTINGS.CFG, TEMPO.CFG  
  
장치는 부팅 시 INDEX.TXT를 읽어 장르별 카운트와 패턴 목록을 표시하고, 선택된 ADP/ARR 파일을 불러 재생합니다.