```
#          Doc Number: Korea JTC1/SC2 k2677_882_export_vba_chunks_to_png.py
# Date: 2026.06.11.
# Author: SHIN, SangHyun
import win32com.client
import os
import sys
import time
import pythoncom
from pathlib import Path
from PIL import Image
# CopyPicture Appearance: xlScreen=1, xlPrinter=2 / Format: xlBitmap=2
_XL_SCREEN, _XL_PRINTER = 1, 2
_XL_BITMAP = 2
# 기호표 N열(0.13)과 동일한 캡처용 더미 열 너비 – PNG 바깥 테두리만 제거하고 데이터 오른쪽 선 유지
_SPACER_COL_WIDTH = 0.13
_MM_TO_PT = 72.0 / 25.4
def _create_excel_app(visible=True):
    """Excel 인스턴스 생성. 기존 프로세스에 붙을 때 Visible 설정이 실패할 수 있어 예외 무시."""
    try:
        excel = win32com.client.DispatchEx("Excel.Application")
    except Exception:
        excel = win32com.client.Dispatch("Excel.Application")
    if visible:
        try:
            excel.Visible = True
        except Exception:
            pass
    try:
        excel.DisplayAlerts = False
    except Exception:
        pass
    return excel
def export_vba_chunks(excel_path, output_dir):
    print("================================================")
    print("    VBA 매크로 분할 로직(페이지 조각) 기반 PNG 추출")
    print("================================================")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
        print(f"저장 폴더 생성 완료: {output_dir}")
    excel = _create_excel_app(visible=True)
    try:
        excel.EnableEvents = False
        excel.ScreenUpdating = True
    except Exception:
        pass
    excel.ErrorCheckingOptions.BackgroundChecking = False
    excel.DisplayCommentIndicator = 0
```

```
    try:
        wb = excel.Workbooks.Open(excel_path)
        print("\n[1] 기호테이블 조각 추출 시작...")
        ws_symbol = wb.Sheets("기호표")
        ws_symbol_name = wb.Sheets("기호명")
        ws_symbol.Activate()
        excel.ActiveWindow.Zoom = 150
        ws_symbol_name.Activate()
        excel.ActiveWindow.Zoom = 150
        pages = [4, 2, 2, 2, 2, 2, 4, 4, 2]
        i1 = 1
        j1 = 1
        img_idx = 1
        for i in range(9):
            i2 = i1 + 35
            ws_symbol.Activate()
            rng = ws_symbol.Range(ws_symbol.Cells(i1, 1), ws_symbol.Cells(i2 - 1, 13))
            export_range_to_png(rng, ws_symbol, os.path.join(output_dir,
f"01_기호_{img_idx:03d}_기호표_파트{i+1}.png"))
            img_idx += 1
            for j in range(1, pages[i] + 1):
                j2 = j1 + 50
                ws_symbol_name.Activate()
                rng = ws_symbol_name.Range(ws_symbol_name.Cells(j1, 1),
ws_symbol_name.Cells(j2 - 1, 5))
                export_range_to_png(rng, ws_symbol_name, os.path.join(output_dir,
f"01_기호_{img_idx:03d}_기호명.png"))
                img_idx += 1
                j1 = j1 + 51
            i1 = i1 + 37
        print("\n[2] 한글테이블 조각 추출 시작...")
        ws_hangul = wb.Sheets("한글표")
        ws_hangul.Activate()
        excel.ActiveWindow.Zoom = 150
        i1 = 1
        for i in range(9):
            i2 = i1 + 35
            rng = ws_hangul.Range(ws_hangul.Cells(i1, 1), ws_hangul.Cells(i2 - 1, 19))
            export_range_to_png(rng, ws_hangul, os.path.join(output_dir,
f"02_한글_{i+1:03d}_한글표.png"))
            i1 = i1 + 37
        print("\n[3] 한자테이블 조각 추출 시작...")
        ws_hanja = wb.Sheets("한자표")
        ws_hanja.Activate()
        excel.ActiveWindow.Zoom = 150
        i1 = 1
        for i in range(26):
            i2 = i1 + 35
```

```
            rng = ws_hanja.Range(ws_hanja.Cells(i1, 1), ws_hanja.Cells(i2 - 1, 25))
            export_range_to_png(rng, ws_hanja, os.path.join(output_dir,
f"03_한자_{i+1:03d}_한자표.png"))
            i1 = i1 + 37
        print("\n[4] 부속서 조각 추출 시작...")
        ws_annex_a = wb.Sheets("부속서")
        ws_annex_a.Activate()
        excel.ActiveWindow.Zoom = 150
        annex_a_ranges = [
            (3, 6), (7, 43), (44, 49), (50, 75), (76, 90),
            (91, 129), (130, 139), (140, 189), (190, 232), (233, 238),
            (239, 288), (289, 317), (318, 334), (335, 384), (385, 392),
        ]
        for i, (start_row, end_row) in enumerate(annex_a_ranges):
            rng = ws_annex_a.Range(ws_annex_a.Cells(start_row, 1), ws_annex_a.Cells(end_row,
5))
            # 부속서는 사용자가 원하는대로 값 있는 마지막 행까지만 엄격 trim
            rng = _trim_range_to_last_value_row_strict(rng, keep_tail_rows=0)
            export_range_to_png(
                rng,
                ws_annex_a,
                os.path.join(output_dir, f"04_부속서_{i+1:03d}_부속서.png"),
            )
        # 부속서는 011 파일의 하단 여백을 기준으로 전체 하단 여백을 통일
        _normalize_annex_bottom_margins(output_dir)
        print("\n[5] 동형표 조각 추출 시작...")
        ws_annex_b = wb.Sheets("동형표")
        ws_annex_b.Activate()
        excel.ActiveWindow.Zoom = 150
        # 마지막 조각: B93:S100 – 부호 3개/4개 한자 표(테두리·더미열 조작 없음)
        annex_b_chunks = [
            {"rows": (1, 42), "cols": (1, 21), "keep_bottom_line": True},
            {"rows": (47, 88), "cols": (1, 21), "keep_bottom_line": True},
            {
                "rows": (93, 100),
                "cols": (2, 19),
                "strip_borders": False,
                "spacer": False,
                "prepare": True,
                # 3번 청크는 현재 결과 유지
                "keep_bottom_line": None,
            },
        ]
        for i, chunk in enumerate(annex_b_chunks):
            if chunk.get("prepare"):
                _prepare_donghyeong_last_table(ws_annex_b)
            r1, r2 = chunk["rows"]
            c1, c2 = chunk["cols"]
```

```
                rng = ws_annex_b.Range(
                    ws_annex_b.Cells(r1, c1), ws_annex_b.Cells(r2, c2)
                )
                export_range_to_png(
                    rng,
                    ws_annex_b,
                    os.path.join(output_dir, f"05_동형표_{i+1:03d}_동형표.png"),
                    use_spacer_column=chunk.get("spacer", True),
                    strip_outer_borders=chunk.get("strip_borders"),
                    keep_bottom_line=chunk.get("keep_bottom_line"),
                )
            print("\n모든 시트(부속서/동형표 포함) 조각(Chunk) 보내기 완벽 성공!")
        finally:
            excel.Quit()
_XL_EDGE_LEFT, _XL_EDGE_TOP, _XL_EDGE_BOTTOM, _XL_EDGE_RIGHT = 7, 8, 9, 10
_RANGE_EDGES = (_XL_EDGE_LEFT, _XL_EDGE_RIGHT, _XL_EDGE_TOP, _XL_EDGE_BOTTOM)
_XL_NONE = -4142
def _border_snapshot(border):
    return {"LineStyle": border.LineStyle, "Weight": border.Weight, "Color": border.Color}
def _restore_border(border, snap):
    if snap["LineStyle"] == _XL_NONE:
        border.LineStyle = _XL_NONE
        return
    border.LineStyle = snap["LineStyle"]
    try:
        border.Weight = snap["Weight"]
    except Exception:
        pass
    try:
        border.Color = snap["Color"]
    except Exception:
        pass
def _strip_outer_borders_for_capture(rng, keep_bottom_line=False):
    saved = []
    edges = list(_RANGE_EDGES)
    if keep_bottom_line and _XL_EDGE_BOTTOM in edges:
        edges.remove(_XL_EDGE_BOTTOM)
    for edge in edges:
        b = rng.Borders(edge)
        saved.append(("range", rng, edge, _border_snapshot(b)))
        b.LineStyle = _XL_NONE
    nrows, ncols = rng.Rows.Count, rng.Columns.Count
    for c in range(1, ncols + 1):
        row_edges = [(1, _XL_EDGE_TOP), (nrows, _XL_EDGE_BOTTOM)]
        if keep_bottom_line:
            row_edges = [(1, _XL_EDGE_TOP)]
        for row, edge in row_edges:
            cell = rng.Cells(row, c)
```

```
                b = cell.Borders(edge)
                saved.append(("cell", cell, edge, _border_snapshot(b)))
                b.LineStyle = _XL_NONE
    for r in range(1, nrows + 1):
        for col, edge in ((1, _XL_EDGE_LEFT), (ncols, _XL_EDGE_RIGHT)):
            cell = rng.Cells(r, col)
            b = cell.Borders(edge)
            saved.append(("cell", cell, edge, _border_snapshot(b)))
            b.LineStyle = _XL_NONE
    return saved
def _restore_outer_borders(saved):
    for kind, target, edge, snap in saved:
        _restore_border(target.Borders(edge), snap)
def _spacer_col_is_usable(ws, col, first_row, last_row):
    """데이터 오른쪽 열이 비어 있으면 삽입 없이 캡처 범위만 확장."""
    rng = ws.Range(ws.Cells(first_row, col), ws.Cells(last_row, col))
    v = rng.Value
    if v is None:
        return True
    if not isinstance(v, tuple):
        return v in (None, "")
    for row in v:
        if isinstance(row, tuple):
            if any(cell not in (None, "") for cell in row):
                return False
        elif row not in (None, ""):
            return False
    return True
def _spacer_sides_for_sheet(ws_name):
    """더미 열: 기호명=양쪽, 기호표 등=오른쪽만, 부속서=없음."""
    if ws_name == "부속서":
        return "none"
    if ws_name == "기호명":
        return "both"
    return "right"
def _strip_outer_borders_for_sheet(ws_name):
    """부속서는 상하좌우 테두리를 그대로 둔다."""
    return ws_name != "부속서"
def _keep_bottom_line_for_sheet(ws_name):
    """요청 시트는 PNG 하단선이 반드시 보이도록 외곽선 제거 시 bottom은 유지."""
    return ws_name in ("기호표", "한글표", "한자표", "기호명")
def _trim_range_to_last_value_row(rng, keep_tail_rows=0):
    """범위 하단의 빈 행을 줄인다.
    값이 없는 행이라도 '하단 테두리'가 있으면 표의 마지막 선이므로 유지한다.
    """
    ws = rng.Worksheet
    r1 = rng.Row
    r2 = rng.Row + rng.Rows.Count - 1
```

```
    c1 = rng.Column
    c2 = rng.Column + rng.Columns.Count - 1
    last_val_row = r1
    for r in range(r1, r2 + 1):
        row_has_bottom_border = False
        for c in range(c1, c2 + 1):
            v = ws.Cells(r, c).Value
            if v is not None and str(v).strip() != "":
                last_val_row = r
                break
            try:
                if ws.Cells(r, c).Borders(_XL_EDGE_BOTTOM).LineStyle != _XL_NONE:
                    row_has_bottom_border = True
            except Exception:
                pass
        if row_has_bottom_border:
            last_val_row = r
    keep_until = min(r2, last_val_row + max(0, keep_tail_rows))
    if keep_until < r2:
        return ws.Range(ws.Cells(r1, c1), ws.Cells(keep_until, c2))
    return rng
def _trim_range_to_last_value_row_strict(rng, keep_tail_rows=0):
    """값 있는 마지막 행까지만 유지(테두리-only 행 제외)."""
    ws = rng.Worksheet
    r1 = rng.Row
    r2 = rng.Row + rng.Rows.Count - 1
    c1 = rng.Column
    c2 = rng.Column + rng.Columns.Count - 1
    last_val_row = r1
    for r in range(r1, r2 + 1):
        for c in range(c1, c2 + 1):
            v = ws.Cells(r, c).Value
            if v is not None and str(v).strip() != "":
                last_val_row = r
                break
    keep_until = min(r2, last_val_row + max(0, keep_tail_rows))
    if keep_until < r2:
        return ws.Range(ws.Cells(r1, c1), ws.Cells(keep_until, c2))
    return rng
def _expand_range_with_bottom_spacer_row(rng, spacer_mm=1.0):
    """범위 하단에 더미 1행을 삽입하고(기본 1mm), 그 행까지 캡처 범위를 확장."""
    ws = rng.Worksheet
    r1 = rng.Row
    r2 = rng.Row + rng.Rows.Count - 1
    c1 = rng.Column
    c2 = rng.Column + rng.Columns.Count - 1
    spacer_row = r2 + 1
    ws.Rows(spacer_row).Insert()
```

```
    ws.Rows(spacer_row).RowHeight = max(1.0, spacer_mm * _MM_TO_PT)
    expanded = ws.Range(ws.Cells(r1, c1), ws.Cells(spacer_row, c2))
    return expanded, {"ws": ws, "spacer_row": spacer_row}
def _restore_bottom_spacer_row(state):
    if not state:
        return
    try:
        state["ws"].Rows(state["spacer_row"]).Delete()
    except Exception:
        pass
_XL_CENTER = -4108
def _prepare_donghyeong_last_table(ws):
    """동형표 마지막 표: '부호값이 4개인 한자' 제목을 M93:S93 병합으로 전부 노출."""
    try:
        title_val = None
        for addr in ("M93", "P93", "N93"):
            v = ws.Range(addr).Value
            if v is not None and str(v).strip():
                title_val = v
                break
        if title_val is None:
            return
        dst = ws.Range("M93:S93")
        if dst.MergeCells:
            dst.UnMerge()
        dst.Merge()
        dst.Value = title_val
        dst.HorizontalAlignment = _XL_CENTER
        dst.WrapText = False
    except Exception:
        pass
def _narrow_spacer_column(ws, col, first_row, last_row, insert_if_occupied=True):
    inserted = False
    if not _spacer_col_is_usable(ws, col, first_row, last_row):
        if not insert_if_occupied:
            raise ValueError(f"spacer column {col} is not empty")
        ws.Columns(col).Insert()
        inserted = True
    width_before = ws.Columns(col).ColumnWidth
    ws.Columns(col).ColumnWidth = _SPACER_COL_WIDTH
    return {"col": col, "width_before": width_before, "inserted": inserted}
def _insert_left_spacer(ws, at_col):
    ws.Columns(at_col).Insert()
    width_before = ws.Columns(at_col).ColumnWidth
    ws.Columns(at_col).ColumnWidth = _SPACER_COL_WIDTH
    return {"col": at_col, "width_before": width_before, "inserted": True}
def _expand_range_with_spacers(rng, sides="right"):
    """캡처 범위 앞·뒤에 0.13 더미 열을 붙여 PNG 바깥 테두리만 제거한다.
```

```
    - right: 데이터 오른쪽 열 1개 (기호표·한글표 등)
    - both: 데이터 양옆 열 1개씩 (기호명 – 좌·우 세로선 유지)
    """
    ws = rng.Worksheet
    first_row = rng.Row
    last_row = rng.Row + rng.Rows.Count - 1
    first_col = rng.Column
    last_col = rng.Column + rng.Columns.Count - 1
    pads = []
    capture_first = first_col
    capture_last = last_col
    if sides == "both":
        if first_col > 1 and _spacer_col_is_usable(ws, first_col - 1, first_row, last_row):
            pads.append(_narrow_spacer_column(ws, first_col - 1, first_row, last_row,
insert_if_occupied=False))
            capture_first = first_col - 1
        else:
            pads.append(_insert_left_spacer(ws, first_col))
            capture_first = first_col
            last_col += 1
    if sides in ("right", "both"):
        right_col = last_col + 1
        pads.append(_narrow_spacer_column(ws, right_col, first_row, last_row,
insert_if_occupied=True))
        capture_last = right_col
    capture_rng = ws.Range(
        ws.Cells(first_row, capture_first),
        ws.Cells(last_row, capture_last),
    )
    return capture_rng, {"ws": ws, "pads": pads}
def _restore_spacer_columns(state):
    if not state:
        return
    ws = state["ws"]
    for pad in sorted(state["pads"], key=lambda p: p["col"], reverse=True):
        try:
            if pad["inserted"]:
                ws.Columns(pad["col"]).Delete()
            else:
                ws.Columns(pad["col"]).ColumnWidth = pad["width_before"]
        except Exception:
            pass
def _hide_chart_frame(chart):
    for area_name in ("ChartArea", "PlotArea"):
        try:
            area = getattr(chart, area_name)
            area.Format.Line.Visible = 0
        except Exception:
```

```
            try:
                area = getattr(chart, area_name)
                area.Border.LineStyle = _XL_NONE
            except Exception:
                pass
def _scroll_range_into_view(rng):
    try:
        rng.Application.Goto(rng, True)
        time.sleep(0.08)
    except Exception:
        pass
def _purge_chart_objects(ws):
    for _ in range(50):
        try:
            if ws.ChartObjects().Count == 0:
                return
            ws.ChartObjects(1).Delete()
        except Exception:
            return
def _chart_has_pasted_picture(chart):
    try:
        return chart.Shapes.Count > 0
    except Exception:
        return False
def _copy_range_picture(rng, appearance=_XL_PRINTER):
    app = rng.Application
    _scroll_range_into_view(rng)
    was_updating = app.ScreenUpdating
    app.ScreenUpdating = True
    try:
        rng.CopyPicture(Appearance=appearance, Format=_XL_BITMAP)
    finally:
        app.ScreenUpdating = was_updating
    time.sleep(0.35)
    pythoncom.PumpWaitingMessages()
def _paste_into_chart(chart, chart_obj, rng):
    appearances = (_XL_PRINTER, _XL_PRINTER, _XL_SCREEN)
    last_err = None
    for i in range(5):
        appearance = appearances[min(i, len(appearances) - 1)]
        try:
            try:
                chart_obj.Activate()
            except Exception:
                pass
            time.sleep(0.12)
            chart.Paste()
            time.sleep(0.15)
```

```
            if _chart_has_pasted_picture(chart):
                return
            last_err = RuntimeError("차트에 붙은 그림 없음")
        except Exception as e:
            last_err = e
        time.sleep(0.25)
        try:
            _copy_range_picture(rng, appearance)
        except Exception:
            pass
    raise last_err
def _png_looks_valid(png_path, min_bytes=8000):
    if not os.path.exists(png_path):
        return False
    return os.path.getsize(png_path) >= min_bytes
def _detect_bottom_blank_rows(img, white_threshold=245):
    gray = img.convert("L")
    w, h = gray.size
    px = gray.load()
    last_nonwhite = -1
    for y in range(h - 1, -1, -1):
        has_nonwhite = False
        for x in range(w):
            if px[x, y] < white_threshold:
                has_nonwhite = True
                break
        if has_nonwhite:
            last_nonwhite = y
            break
    if last_nonwhite < 0:
        return h
    return h - 1 - last_nonwhite
def _normalize_png_bottom_blank_rows(png_path, target_blank_rows):
    with Image.open(png_path) as img:
        img = img.convert("RGB")
        w, h = img.size
        current_blank = _detect_bottom_blank_rows(img)
        if current_blank == target_blank_rows:
            return
        delta = target_blank_rows - current_blank
        if delta < 0:
            # 여백이 더 크면 하단을 잘라낸다.
            new_h = max(1, h + delta)
            out = img.crop((0, 0, w, new_h))
        else:
            # 여백이 더 작으면 흰색 행을 하단에 붙인다.
            out = Image.new("RGB", (w, h + delta), "white")
            out.paste(img, (0, 0))
```

```
        out.save(png_path)
def _normalize_annex_bottom_margins(output_dir):
    annex_files = [
        os.path.join(output_dir, f"04_부속서_{i:03d}_부속서.png")
        for i in range(1, 16)
    ]
    annex_files = [p for p in annex_files if os.path.exists(p)]
    if not annex_files:
        return
    ref_path = os.path.join(output_dir, "04_부속서_011_부속서.png")
    if os.path.exists(ref_path):
        with Image.open(ref_path) as ref_img:
            target_blank = _detect_bottom_blank_rows(ref_img)
    else:
        blanks = []
        for p in annex_files:
            with Image.open(p) as img:
                blanks.append(_detect_bottom_blank_rows(img))
        blanks.sort()
        target_blank = blanks[len(blanks) // 2]
    for p in annex_files:
        _normalize_png_bottom_blank_rows(p, target_blank)
def _export_range_once(
    rng,
    ws,
    png_path,
    use_spacer_column=True,
    strip_outer_borders=None,
    keep_bottom_line=None,
):
    app = rng.Application
    ws.Activate()
    try:
        app.ActiveWindow.Zoom = 150
    except Exception:
        pass
    aw = app.ActiveWindow
    grid_was_on = aw.DisplayGridlines
    aw.DisplayGridlines = False
    _purge_chart_objects(ws)
    spacer_state = None
    capture_rng = rng
    if use_spacer_column:
        sides = _spacer_sides_for_sheet(ws.Name)
        if sides != "none":
            capture_rng, spacer_state = _expand_range_with_spacers(rng, sides=sides)
    if strip_outer_borders is None:
        strip_outer_borders = _strip_outer_borders_for_sheet(ws.Name)
```

```
    saved_borders = []
    if keep_bottom_line is None:
        keep_bottom_line = _keep_bottom_line_for_sheet(ws.Name)
    if strip_outer_borders:
        saved_borders = _strip_outer_borders_for_capture(
            capture_rng,
            keep_bottom_line=keep_bottom_line,
        )
    chart_obj = None
    try:
        copied = False
        for appearance in (_XL_PRINTER, _XL_PRINTER, _XL_SCREEN):
            for _ in range(3):
                try:
                    _copy_range_picture(capture_rng, appearance)
                    copied = True
                    break
                except Exception:
                    time.sleep(0.3)
            if copied:
                break
        if not copied:
            return False
        # 차트 캔버스에 여유 패딩 추가 (하단 선 짤림 방지)
        # CopyPicture 이미지가 rng 치수보다 미세하게 클 수 있어
        # 캔버스가 정확히 같으면 하단/우측 테두리 선이 잘림
        _H_PAD = 4    # pt (하단 선 짤림 방지 - 주요 원인)
        _W_PAD = 2    # pt (우측 선 짤림 방지 - 보조)
        chart_obj = ws.ChartObjects().Add(
            capture_rng.Left, capture_rng.Top,
            capture_rng.Width + _W_PAD,
            capture_rng.Height + _H_PAD,
        )
        chart = chart_obj.Chart
        _hide_chart_frame(chart)
        try:
            _paste_into_chart(chart, chart_obj, capture_rng)
        except Exception:
            return False
        if os.path.exists(png_path):
            os.remove(png_path)
        chart.Export(png_path, "PNG")
        return _png_looks_valid(png_path)
    finally:
        if chart_obj is not None:
            try:
                chart_obj.Delete()
            except Exception:
```

```
                pass
        if saved_borders:
            _restore_outer_borders(saved_borders)
        _restore_spacer_columns(spacer_state)
        aw.DisplayGridlines = grid_was_on
        try:
            app.CutCopyMode = False
        except Exception:
            pass
        _purge_chart_objects(ws)
        pythoncom.PumpWaitingMessages()
def export_range_to_png(
    rng,
    ws,
    png_path,
    use_spacer_column=True,
    strip_outer_borders=None,
    keep_bottom_line=None,
):
    print(f"  -> 추출 중: {os.path.basename(png_path)} ... ", end="", flush=True)
    app = rng.Application
    for attempt in range(3):
        if _export_range_once(
            rng,
            ws,
            png_path,
            use_spacer_column=use_spacer_column,
            strip_outer_borders=strip_outer_borders,
            keep_bottom_line=keep_bottom_line,
        ):
            print(f"완료 ({os.path.getsize(png_path):,} bytes)")
            return
        try:
            app.CutCopyMode = False
        except Exception:
            pass
        _purge_chart_objects(ws)
        time.sleep(0.4 + attempt * 0.3)
    print("실패 (하얀 공백 이미지 발생)")
def export_spacer_test(excel_path, output_dir, sheet_name, row_start, row_end, last_col, tag):
    os.makedirs(output_dir, exist_ok=True)
    excel = _create_excel_app(visible=True)
    try:
        wb = excel.Workbooks.Open(excel_path)
        ws = wb.Sheets(sheet_name)
        ws.Activate()
        excel.ActiveWindow.Zoom = 150
        rng = ws.Range(ws.Cells(row_start, 1), ws.Cells(row_end, last_col))
```

```
            sides = _spacer_sides_for_sheet(sheet_name)
            strip = _strip_outer_borders_for_sheet(sheet_name)
            out = os.path.join(output_dir, f"TEST_{tag}_{sides}.png")
            export_range_to_png(rng, ws, out, use_spacer_column=True)
            print(f"시험({sheet_name}, spacer={sides}, strip={strip}): {out}")
            wb.Close(False)
        finally:
            excel.Quit()
if __name__ == "__main__":
    base_dir = Path(__file__).resolve().parent
    excel_file = os.path.join(base_dir, "KSX1001_excel.xls")
    output_folder = os.path.join(base_dir, "vba_chunk_images")
    if not os.path.exists(excel_file):
        print("엑셀 파일을 찾을 수 없습니다.")
    elif len(sys.argv) > 1 and sys.argv[1] == "--test-giho":
        export_spacer_test(excel_file, output_folder, "기호표", 1, 35, 13, "기호표")
    elif len(sys.argv) > 1 and sys.argv[1] == "--test-both":
        export_spacer_test(excel_file, output_folder, "기호명", 1, 49, 5, "기호명")
        export_spacer_test(excel_file, output_folder, "부속서", 7, 43, 5, "부속서")
    elif len(sys.argv) > 1 and sys.argv[1] == "--test-donghyeong":
        os.makedirs(output_folder, exist_ok=True)
        excel = _create_excel_app(visible=True)
        try:
            wb = excel.Workbooks.Open(excel_file)
            ws = wb.Sheets("동형표")
            ws.Activate()
            excel.ActiveWindow.Zoom = 150
            _prepare_donghyeong_last_table(ws)
            rng = ws.Range(ws.Cells(93, 2), ws.Cells(100, 19))
            out = os.path.join(output_folder, "TEST_동형표_마지막표.png")
            export_range_to_png(
                rng, ws, out, use_spacer_column=False, strip_outer_borders=False
            )
            print(f"시험 출력: {out}")
            wb.Close(False)
        finally:
            excel.Quit()
    else:
        export_vba_chunks(excel_file, output_folder)
```