PharmaSUG China 2026 - Paper AA156-2026

A Practical Framework for Applying Generative AI in Clinical SAS Programming: A Human-in-the-Loop Approach for Chinese Clinical Studies

Siyu Yu, Clinical Biostatistics Department, CHIA TAI TIANQING PHARMACEUTICAL GROUP CO., LTD

ABSTRACT

Objective: This paper proposes a practical framework for applying generative artificial intelligence in clinical SAS programming. It does not position AI as a replacement for clinical programmers, but embeds it in a controlled, specification-driven, human-reviewed workflow for protocol interpretation, specification review, SAS development, log checking, and output validation. The framework defines a risk-scaled release gate: the evidence needed before an AI-assisted draft is review-ready.

Background: Clinical SAS programmers are required to work with multiple sources of information, including protocols, statistical analysis plans, SDTM and ADaM specifications, TFL shells, annotated CRFs, define.xml, study data review guides, project macros, and study-specific folder structures. In Chinese clinical study environments, additional challenges often arise from Chinese CRF content, Chinese text values, UTF-8 encoding, SAS Viya compatibility, special characters, and the need to maintain consistency between Chinese source documents and English submission deliverables. Traditional programming workflows rely heavily on programmer experience and manual review, which may lead to repetitive work, inconsistent coding styles, environment-specific issues, and time-consuming log troubleshooting. Generative AI provides new opportunities to support these tasks, but its use must be standardized and carefully reviewed to avoid logic errors, traceability gaps, and quality risks.

Methods: A four-component AI-assisted clinical programming framework was developed. First, common programming tasks were classified into practical categories, such as specification interpretation, SDTM and ADaM derivation, TFL programming, batch execution, log review, and submission document support. Second, a standardized prompt and context package was designed to provide AI with the study background, input datasets, variable specifications, folder structure, macro requirements, encoding rules, and expected output format. Third, AI was used to generate draft SAS programs or review suggestions, with explicit requirements for UTF-8 encoding, SAS Viya compatibility, clean log expectations, project-specific paths, and reusable macro structures. Fourth, a human-in-the-loop review process was applied, including programmer review, SAS execution, log checking, dataset and output validation, and feedback-based prompt refinement.

Results: The framework was demonstrated through representative scenarios, including SDTM EC/EX mapping, supplemental qualifier handling, ADaM derivation, TFL generation, batch log review, and Define-XML/SDRG issue triage. A fully de-identified operational ADLB1 case showed that a clean log can conceal a non-usable output when an approved rule or required dependency is not explicit. Programmer review, output-level QC, correction, and reusable feedback made the draft review-ready. Critical analysis decisions, endpoint definitions, statistical assumptions, regulatory interpretation, and final validation still require experienced programmer review.

Conclusions: Generative AI is most valuable in clinical SAS programming as a controlled, reviewable, and iterative assistant rather than an independent code generator. A clean log is necessary but insufficient: a draft is review-ready only when approved rules, dependencies, execution, and output-level QC agree. Standardized prompts, project context, SAS execution, and human validation provide a traceable path from draft to review-ready artifact, including for Chinese studies using SAS Viya, UTF-8, Chinese source documents, and complex project standards.

Keywords: Generative AI; Clinical SAS Programming; SAS Viya; UTF-8; SDTM; ADaM; TFL; Human-in-the-Loop; Quality Control

INTRODUCTION

A practical clinical programming day rarely starts with a blank SAS editor. It often starts with an incomplete set of documents: a protocol, a statistical analysis plan (SAP), an annotated CRF, SDTM and ADaM specifications, TFL shells, reviewer comments, Pinnacle 21 findings, project macros, study folders, and sometimes bilingual meeting notes. The programmer must convert those materials into traceable datasets, programs, logs, outputs, and submission-support documents that remain aligned with CDISC SDTM, ADaM, Define-XML, and validation expectations (CDISC 2021a; CDISC 2021b; CDISC 2019; Pinnacle 21 2024).

Generative AI can help with this conversion, but the value of AI in clinical SAS programming is not simply whether it can generate a piece of code. The more important question is whether AI can work inside the real project rhythm: according to standards, study rules, validation paths, and human review. Without that control layer, AI may create fluent but unsupported answers, inconsistent coding styles, unreviewed assumptions, or environment-specific programs that fail in SAS Viya.

This paper therefore treats AI as a controlled, reviewable assistant. The framework is organized into four operational components: first classify the programming task; second provide a standardized prompt and context package; third allow AI to generate only a draft program, checklist, or review suggestion; and fourth complete programmer review, SAS execution, log checking, dataset/output validation, and prompt refinement.

Compared with prior AI work that often focuses on a single tool, model, or automation use case, this paper contributes a workflow-level control framework for clinical SAS programming. Its novelty is not autonomous programming performance or general model accuracy; it is a minimum, risk-scaled release gate linking task scope, approved rules, execution, output-level evidence, and feedback reuse in one auditable operating model for Chinese clinical studies.

A FOUR-COMPONENT HUMAN-IN-THE-LOOP FRAMEWORK

The framework is designed around the actual deliverables of clinical SAS programming rather than around the model itself. Each component has a specific work product and a specific validation point. Practice-focused assets such as prompt cards, rule IDs, reusable examples, and AI use logs support the framework, but they do not replace the four-component workflow.

Figure 1 presents the operational workflow, which corresponds to Layer 2 of the three-layer view shown in Table 1. The workflow starts with task classification, then moves to a standardized prompt and context package, controlled AI draft generation, human validation, and feedback to reusable prompts or rules.

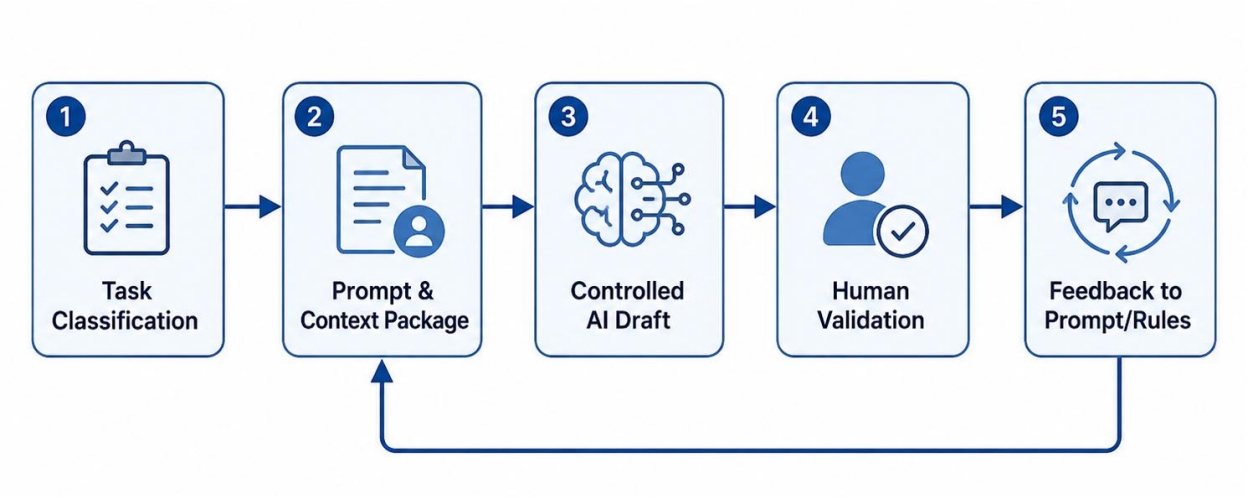

Figure 1. Operational Human-in-the-Loop Workflow, Corresponding to Layer 2 of the Framework

Layer 3 - Governance and Novelty: task boundaries, comparison with prior tool-centered automation, risk-based use, data protection, auditability, and organizational policy alignment.
Layer 2 - Operational Human-in-the-Loop Workflow: task classification, standardized context package, controlled AI draft generation, human review and validation, and feedback refinement.
Layer 1 - Toolkit and Reuse Scope: lightweight prompt cards, rule IDs, worked examples, reusable code snippets, issue cards, terminology maps, and log triage patterns.

Table 1. Three-Layer Governance, Workflow, and Toolkit View

Component	Main Purpose	Typical Work Product	Human Control
Task classification	Separate different kinds of work so AI is used only for appropriate draft or review tasks.	Task intake note: spec interpretation, SDTM/ADaM derivation, TFL programming, batch/log review, or submission-support task.	Confirm whether the task involves key endpoints, analysis decisions, identifiable data, or regulated interpretation.
Prompt and context package	Give AI the study background and constraints it cannot infer by itself.	Prompt card with standards, input datasets, variable specifications, folder paths, macros, encoding rules, output format, and refusal conditions.	Check data sensitivity, source trace, standard version, and whether rules come from SAP/spec rather than the model.
Controlled AI draft generation	Produce a reviewable draft rather than a final deliverable.	Mapping table, SAS skeleton, log triage, QC checklist, issue list, or document-support text.	Require AI to state assumptions, open questions, and items that require programmer confirmation.
Human-in-the-loop validation	Turn an AI draft into a controlled programming result.	Validation note, clean log, dataset/output comparison, reviewer sign-off, and prompt feedback.	Experienced programmers remain responsible for final logic, execution, and deliverable quality.

Table 2. Four Components of the Human-in-the-Loop Framework

COMPONENT 1: TASK CLASSIFICATION

The first control is deciding what kind of task is being assigned to AI. A request such as “help me with ADaM” is too broad. The framework classifies tasks into practical categories that map directly to clinical programming work products. This avoids asking AI to make decisions that belong to a statistician, lead programmer, or submission owner.

Task Category	AI-Suitable Draft Output	Not Suitable for AI Final Decision
Protocol, SAP, and analysis-rule interpretation	Extract variables, timing windows, candidate flags, rule traces, and open questions.	Endpoint definitions, estimands, population rules, missing-date rules, and statistical strategy.
SDTM, aCRF, SUPP, and Define-XML review	Draft mappings, SUPP candidates, value-level metadata notes, controlled terminology risks, and Pinnacle 21 (P21) review points.	Final domain structure, CT approval, QNAM placement, reviewer-facing metadata, and sponsor-specific modeling decisions.
ADaM derivation drafting	Pseudo-code, SAS skeletons, parameter-level handling, trace variables, and focused QC checks.	Baseline rule, treatment-emergent window, same-day pre-dose handling, tie-breakers, and partial date handling.
TFL and batch execution support	Shell decomposition, PROC	Final denominator, display wording,

	REPORT skeletons, output dataset checks, log triage, and issue summaries.	statistical method, warning acceptability, and output approval.

Table 3. Task Classification and AI Boundary Setting

COMPONENT 2: STANDARDIZED PROMPT AND CONTEXT PACKAGE

A clinical programming prompt should behave like a compact requirement document. It should tell AI what role to play, what task to perform, which standards and versions apply, what study context matters, which inputs are approved for use, what output format is expected, how the draft will be verified, and what the model must not invent. For Chinese clinical studies, the prompt should also define bilingual handling.

DATA PROTECTION AND MINIMUM-NECESSARY CONTEXT

The context package should follow a minimum-necessary principle. Prompts should use approved and de-identified excerpts whenever possible, and should avoid direct identifiers, patient narratives, investigator information, unapproved interim results, or confidential sponsor strategy unless the AI environment has been approved for that data classification. For public or general-purpose AI tools, only synthetic, generalized, or de-identified examples should be used. The prompt card should record the input sensitivity level and whether the draft output may be stored as a reusable asset.

Prompt Field	Content to Provide	Example
Role	Expected expertise and domain.	Senior clinical SAS programmer familiar with CDISC, SAS Viya, and Chinese clinical study delivery.
Task	A specific programming or review task.	Draft SDTM EC/EX mapping logic from approved dose administration CRF and source metadata.
Standards	CDISC, controlled terminology, sponsor standard, SAP/spec version.	SDTMIG/ADaMIG version, CT package, project spec version, sponsor TFL style guide.
Context	Study phase, therapeutic area, data source, system environment, bilingual requirements.	Phase II oncology study; SAS Viya on Linux; UTF-8; Chinese CRF and English submission deliverables.
Inputs	Approved excerpts only.	CRF fields, source variables, spec rows, SAP paragraph, shell section, or SAS log fragment.
Output	Expected structure.	Mapping table, SAS skeleton, open questions, QC checklist, and source trace.
Verification	Required checks.	Clean log, PROC COMPARE, P21, metadata review, shell comparison, independent QC.
Refusal condition	What AI must not guess.	Do not invent CT terms, imputation rules, endpoint definitions, or population rules.
Bilingual policy	How Chinese and English content should be handled.	Preserve Chinese source labels; use English CDISC variables and terms; follow project glossary.

Table 4. Recommended Fields in a Standardized Prompt and Context Package

```
Role: Senior clinical SAS programmer.
Task: Draft reviewable logic for <domain/dataset/output> based on approved
specifications.
Standards: <CDISC / CT / sponsor standard / project spec version>.
Context: SAS Viya on Linux; UTF-8 encoding; bilingual China study.
```

```
Inputs: <approved CRF/SAP/spec/shell/log excerpts>.
Alignment Rules: <global/team/study/task rule IDs>.
Output: <mapping table / SAS skeleton / checklist / issue triage>.
Verification: List clean log, P21, metadata, source trace and independent QC checks.
Refusal: Do not invent analysis rules, CT terms, missing-date rules, or population
definitions.
Bilingual Policy: Preserve Chinese source labels; use English CDISC variables and
standard terms.
```

Display 1. Reusable Prompt Card Skeleton

COMPONENT 3: CONTROLLED AI DRAFT GENERATION

AI output should be deliberately positioned as a draft. The expected output is not “final SAS code.” It is a reviewable artifact: a mapping proposal, derivation skeleton, program outline, log triage, checklist, or open-question list. This reduces the risk that fluent AI output is treated as validated logic.

In SAS Viya environments, prompt instructions should explicitly include UTF-8 encoding, Linux path sensitivity, project macro dependencies, expected folder structure, and clean log expectations. A draft program should identify required libraries, macro assumptions, parameter inputs, sort keys, and trace variables so that programmers can execute and test it in the target environment.

COMPONENT 4: HUMAN REVIEW AND VALIDATION

Human-in-the-loop review is the central quality mechanism. A programmer reviews the AI draft, executes the SAS program, checks the log, compares datasets and outputs, reviews metadata and traceability, and refines the prompt or rule card when a recurring problem is found. The review tests whether the draft survives ordinary programming controls, including output-level QC. This evidence-focused control aligns with, but does not claim regulatory compliance with, current FDA and EMA principles for risk-based use, context of use, data governance, and lifecycle management (ICH 2025; FDA 2025; FDA and EMA 2026).

Quality Gate	Evidence Expected Before the AI-Assisted Work Can Be Used
Source trace	Key derivations link to protocol, SAP, CRF/aCRF, specification row, CT package, reviewer comment, or project rule ID.
Execution	Program runs in the target local or SAS Viya environment with correct paths, libraries, macro dependencies, permissions, and encoding.
Log quality	ERROR, WARNING, and suspicious NOTE messages are fixed, explained with impact assessment, or documented as acceptable by the programmer.
Dataset or output comparison	Derived datasets, TFL outputs, or metadata are compared against the specification, shell, independent QC output, or prior approved version.
Metadata and submission readiness	Labels, lengths, formats, origins, methods, value-level metadata, Define-XML content, and P21 findings are reviewed.
Human correction record	Any AI draft error is converted into an issue card or rule card so the same failure is less likely to recur.

Table 5. Quality Gates for Human Validation

REPRESENTATIVE SCENARIOS

SDTM EC/EX Mapping

Exposure collection and administered treatment data are common targets for AI assistance because source fields, dose units, routes, start and end dates, and treatment epochs can be difficult to align

across CRF, protocol, and SDTM specifications. AI can draft EC/EX mapping logic and list ambiguity, but it must not decide final exposure modeling or dose derivation without the approved specification.

Step	AI Assistance	Human Validation
Task context	Summarize dose administration CRF fields, source variables, route, dose amount, unit, frequency, start/end date-time, and visit context.	Confirm that the source fields are approved, de-identified if needed, and sufficient for EC/EX mapping.
Context package	Use protocol treatment description, SDTM spec, source metadata, and study rules for planned vs actual dosing.	Check SDTMIG, controlled terminology, sponsor conventions, and any study-specific derivation rules.
Draft output	Produce a mapping table for ECTRT/EXTRT, ECDOSE/EXDOSE, units, route, timing, --DY, EPOCH, and open questions.	Verify dose units, actual vs collected exposure distinction, partial/missing dates, sequence logic, and traceability.
Reusable output	Save EC/EX prompt card, dose-unit rule, date/time handling rule, and issue card for ambiguous source fields.	Only save after project lead review.

Table 6. SDTM EC/EX Mapping Scenario

Supplemental Qualifier Handling

Supplemental qualifiers are another scenario where AI can assist but should not decide alone. AI can list variables that appear to lack a parent-domain variable, identify potential QNAM/QLABEL candidates, and flag whether a value might belong in the main domain, SUPP, FA, RELREC, or a redesigned source mapping. Final modeling must remain a standards and sponsor decision.

ADaM Treatment and Baseline Derivations

ADaM treatment variables and baseline variables are frequently repetitive, but they are also analysis-critical. AI can help transform specification text into pseudo-code and skeleton code. It should be required to preserve trace variables and to flag missing rules rather than inventing imputation or tie-breaker rules. For baseline derivations, the prompt should explicitly state that missing time, same-day pre-dose records, tie-breaker order, and partial date handling must come from the SAP or approved ADaM specification. AI may draft the mechanics only after those rules are supplied; it should not invent these analysis decisions.

Derivation Area	AI-Assisted Draft	Required Human Review
Treatment variables	Draft logic for TRTSDT, TRTEDT, TRTP/TRTA, treatment duration, and exposure windows from ADSL/ADEx sources.	Confirm actual treatment records, date/time completeness, dose interruption handling, and study-specific treatment labels.
Population flags	Create skeleton logic for SAFFL/ITTFL/FASFL based on specification text.	Confirm definitions with SAP and project team; AI cannot define populations.
Treatment-emergent flags	Draft TRTEMFL logic for AE or concomitant medication datasets based on an approved window.	Confirm window, date-only vs datetime comparison, partial dates, and edge cases.
Baseline selection	Draft parameter-level baseline candidate and tie-breaker logic.	Confirm SAP rule, same-day pre-dose handling, missing time handling, and source trace.
Change variables	Draft CHG/PCHG only where numeric change is meaningful.	Confirm qualitative parameters, zero baseline, units, and rounding/display rules.

Table 7. ADaM Treatment and Baseline Derivation Scenario

```
/* Skeleton only: requires approved ADaM spec and SAP rule before use. */
/* Same-day pre-dose, missing time, partial date, and tie-breaker rules
   must come from the approved SAP or ADaM specification. */

proc sort data=lb_numeric(where=(not missing(aval) and baseline_candidate =
'Y'))
          out=base_candidates;
  by usubjid paramcd adt adtm lbseq;
run;

data base_select;
  set base_candidates;
  by usubjid paramcd;
  if last.paramcd then do;
    base = aval;
    basedt = adt;
    base_seq = lbseq;
    baseline_sort_reason =
      'Last nonmissing pre-dose candidate by approved sort key';
    output;
  end;
  keep usubjid paramcd base basedt base_seq baseline_sort_reason;
run;

/* Assumption: base_select contains no more than one record per
   USUBJID/PARAMCD. This must be verified before merge. */
proc sort data=base_select out=base_select_chk nodupkey dupout=base_dup;
  by usubjid paramcd;
run;
/* Programmer must confirm base_dup has zero records before using
   base_select_chk in the merge result. */

proc sort data=lb_numeric out=lb_all;
  by usubjid paramcd;
run;

/* This skeleton illustrates BASE/CHG/PCHG mechanics.
   ABLFL assignment and PARAM-specific exceptions require approved rules.
*/
data adlb_draft;
  merge lb_all(in=a) base_select_chk;
  by usubjid paramcd;
  if a;

  if not missing(aval) and not missing(base) then do;
    chg = aval - base;
    if base ne 0 then pchg = 100 * chg / base;
  end;
```

```
run;
```

Display 2. ADLB Baseline Skeleton That Must Be Validated Against the Approved Rule

MINI CASE: WHEN A CLEAN LOG WAS NOT ENOUGH

This fully de-identified ADLB1 case is process evidence, not a performance experiment. It shows that a clean log did not establish usability: only output-level QC, human correction, and reusable feedback made the draft review-ready.

Step	Example Evidence
Approved context	The scope was limited to ADLB1. Approved rules required a normal laboratory result to receive Grade 0; hepatic components had to be paired by the same collection event and repeat key; and the analysis-period cutoff included one specified Week 16 exception.
AI-assisted draft	The draft established a three-part SAS program but did not make the task boundary and every required dependency explicit.
Output-level QC	A clean execution log alone did not establish usability. Output QC identified that a required analysis dependency had not been retained, leaving the intended output non-usable.
Human correction and validation	The programmer locked the scope to ADLB1, restored the required dependency, and encoded the approved clinical rules and cutoff exception. After correction, the log was clean; checks for normal results with nonzero grade, incompatible hepatic event/repeat-key pairing, and records beyond the cutoff outside the exception all returned zero findings.
Reusable feedback	Rule cards recorded the Grade 0 and same-event pairing rules. An issue card recorded that a clean log must be supplemented by output-level QC before a draft can be used.

Table 8. Fully De-Identified ADLB1 Case: From Clean Log to Review-Ready Evidence

TFL Table and Figure Generation

For tables and figures, AI is useful for decomposing a shell into programming tasks: source dataset, variables, statistics, denominators, column structure, titles, footnotes, sorting and formatting. For Chinese clinical studies, the prompt should preserve approved Chinese shell text and use English technical terms for ADaM variables and SAS code.

Shell Element	AI-Assisted Output	Human Validation
Title and footnotes	Identify text to preserve and link footnotes to population/denominator rules.	Confirm approved bilingual wording; do not allow free translation.
Rows and columns	Map shell rows to ADaM variables, categories, and statistics.	Confirm source dataset, population flag, ordering, missing display, and denominator.
Table code	Draft PROC REPORT skeleton and macro parameter list.	Run and compare RTF output with shell; check formatting and footnotes.
Figures	Draft plotting data preparation and axis/legend specifications.	Confirm statistical intent, display convention, and review output visually.

Table 9. TFL Programming Scenario

Batch Log Review

Batch execution produces many logs. AI can help classify errors, warnings and suspicious notes, especially when logs contain repeated issues. The prompt should include program purpose, expected output, environment, and only relevant log excerpts. The AI output should be an issue triage list, not a decision that the log is acceptable.

Define-XML and SDRG Issue Triage

AI can help organize Define-XML and SDRG-related issues by comparing variable metadata, controlled terminology, value-level metadata, comments, and reviewer findings. It can also draft internal issue summaries. However, final metadata interpretation and submission wording require responsible human review. AI use itself should normally be recorded in internal logs or validation notes rather than by changing standard Define-XML Origin values.

Supplemental qualifier handling, batch log review, and Define-XML/SDRG issue triage were retained as summary use cases rather than expanded tables. Their common control point is the same: AI may organize candidate issues and likely investigation paths, but final metadata, impact assessment, and reviewer-facing decisions remain human responsibilities.

RESULTS AND OBSERVATIONS FROM ROUTINE APPLICATION

Most examples in this paper are generalized and de-identified. In addition, one fully de-identified operational ADLB1 case documented a visible correction loop: an AI-assisted draft was not accepted on the basis of clean execution alone, because output-level QC detected an incomplete result. Programmer correction made the scope, required dependency, and approved clinical rules explicit; the corrected result then passed predefined checks and was converted into reusable controls. The framework was not evaluated as a controlled productivity experiment. Routine application was therefore summarized semi-quantitatively. AI usefulness was high for TFL shell decomposition and batch log triage, medium to high for SDTM/aCRF/SUPP and metadata review, medium for ADaM skeleton drafting, and low for final clinical or statistical decisions. The following operational meanings were used for the semi-quantitative ratings.

Rating	Operational Meaning
High	AI draft could be converted into a usable review artifact after normal programmer validation and limited correction.
Medium	AI draft was useful for structure or checklist generation but required substantial logic, metadata, or programming correction.
Low	AI was useful mainly for brainstorming or open-question identification; final decision-making remained fully human-driven.
Not suitable	The task involved endpoint definition, statistical strategy, final regulatory wording, or other decisions outside AI draft scope.

Table 10. Operational Meanings of Semi-Quantitative Ratings

Area	AI Draft	Human Correction	Reason
Baseline selection	Selected the last nonmissing pre-dose value.	Confirmed with SAP and ADaM specification; added same-day time, partial date, and tie-breaker rules.	AI cannot infer analysis rules that are not provided in approved documents.
CHG/PCHG code	Drafted CHG and PCHG after baseline selection.	Moved calculation into a valid DATA step after merging baseline back to all analysis records.	SAS syntax and data flow must be executable, not only logically described.
Controlled terminology	Suggested candidate terms.	Checked against the project CT package and sponsor convention.	AI must not invent or approve CT terms.
Log review	Classified a WARNING as low risk.	Required programmer impact assessment and rerun evidence.	Warning acceptability is a project decision.
Bilingual labels	Free-translated a Chinese source term.	Replaced it with the project glossary term and preserved source trace.	Translation consistency affects metadata, reviewer communication, and downstream outputs.

Table 11. AI Draft vs Human Correction

IMPLEMENTATION CONSIDERATIONS FOR CHINESE CLINICAL STUDIES

The framework is particularly relevant in Chinese clinical studies because teams often work across Chinese source documents and English submission deliverables. Prompts and validation checklists should address bilingual source text, UTF-8, special characters, SAS Viya compatibility, and traceability from Chinese business language to English CDISC metadata. China submission work is also shaped by NMPA/CDE expectations for standardized electronic study data and reviewer-facing documentation (Center for Drug Evaluation, NMPA 2020). The 2026 Drug GCP, effective 1 September 2026, further foregrounds data governance and quality management for new technologies; it is cited here as implementation context, not as a mandate for this workflow (NMPA et al. 2026). AI-assisted programming should preserve the chain from Chinese source documents to CDISC-style datasets, Define-XML metadata, SDRG/ADRG explanations, validated SAS logs, and final TFL outputs.

Consideration	Practical Control
Chinese CRF and bilingual terminology	Preserve Chinese source labels in trace columns or review tables; use English CDISC variables and standard terms; maintain a project glossary for approved translations.
UTF-8 and special characters	Check Chinese characters, full-width/half-width symbols, ±, μ, superscripts, line breaks, RTF rendering, and encoding-sensitive formats.
SAS Viya compatibility	State Linux paths, case sensitivity, macro library locations, file permissions, encoding, and output folders in the context package.
Project standards	Include sponsor macros, table style guide, decimal rules, naming conventions, and batch execution structure.
Submission consistency	Keep traceability between Chinese source documents, English SDTM/ADaM metadata, Define-XML, SDRG/ADRG, and TFL outputs.
NMPA/CDE electronic data submission context	Maintain traceable links among Chinese source documents, CDISC variables, Define-XML, SDRG/ADRG explanations, SAS logs, and final TFL outputs; do not let AI replace reviewer-facing documentation decisions.

Table 12. China-Specific Implementation Controls

IMPLEMENTATION TOOLKIT

The practice-focused PARK content is best used as an implementation toolkit under this framework, not as a replacement for the framework. PARK refers to Prompt template engineering, Agent specification design, Reuse architecture, and Knowledge feedback loop. The knowledge loop turns repeated programming corrections, log issues, specification decisions, and QC findings into reusable rule cards and issue cards. This knowledge feedback loop is consistent with the idea that repeated work experience can be converted into reusable organizational knowledge (Nonaka and Takeuchi 1995).

The toolkit is intentionally scoped as a lightweight implementation aid rather than a software product, enterprise platform, or replacement for sponsor SOPs. Teams can adopt only the elements that fit their governance model, such as a prompt card, a rule log, or a feedback register. Final programming decisions, validation evidence, and submission-ready outputs remain under the existing clinical programming and quality control process.

Toolkit Item	Minimum Fields or Content	Purpose
Prompt card	Role, task, standards, context, inputs, output, verification, refusal condition, bilingual policy.	Make each AI request complete and reviewable.
Rule ID	Rule text, source trace, owner, effective date, review date.	Prevent oral project rules from being lost.
Issue card	Issue, wrong output, why wrong, corrected rule, fixed prompt, validation, owner, status.	Turn errors into reusable prevention patterns.
AI use log	Task type, tool/model, input sensitivity, asset IDs, output use, reviewer, validation result.	Provide internal traceability for AI-assisted work.
Asset index	asset_id, type, task_type, standard_version, owner, status, validation_status, review dates, source_trace, successor_id.	Keep reusable materials searchable and maintainable.

Table 13. Minimal Implementation Toolkit

```
park_repo/
templates/
rules/
examples/
code_snippets/
issues/
term_map/
logs/
index.csv
```

Display 3. Minimal Folder Structure for Reusable Assets

Date	Omitted in public version
Task	ADLB1 rule completion and output-level QC
Prompt card	De-identified operational case card
Input sensitivity	No patient-level data; no study, product, or environment identifiers
AI draft type	Three-part SAS draft and open-question list
Human correction	Locked ADLB1 scope; made Grade 0, same-event pairing, cutoff exception, and required dependency explicit
Validation evidence	Clean log plus zero findings for the predefined output-level QC checks
Follow-up asset	Rule cards and issue card; internal identifiers omitted

Display 4. Example AI Use Log Entry

LIMITATIONS

This framework does not replace SOPs, validation plans, double programming, independent QC, or submission review. The fully de-identified operational case documents one correction loop, not comparative performance, productivity, accuracy, or generalizability. Organizations must adapt the

workflow to their tool policies, data classifications, audit expectations, and project standards. The FDA draft guidance is cited only as nonbinding, risk-based governance context, not as direct coverage of every internal programming activity.

CONCLUSION

Generative AI is most useful in clinical SAS programming as a controlled, reviewable, and iterative assistant. The workflow classifies the task, supplies context, generates only a draft, and validates code and output through human review. Its minimum release gate requires approved scope and rules, executable code, output-level evidence, and reusable correction before a draft is review-ready.

For Chinese clinical studies, the framework adds controls for SAS Viya, UTF-8, Chinese source documents, bilingual terminology, and complex project standards. The ADLB1 case reinforces one lesson: clean execution is necessary but insufficient; output-level evidence must confirm that approved clinical rules and required dependencies produced a usable result. The value is a more consistent, traceable path from AI-assisted work to human-validated deliverables, not autonomous code generation. Repetitive drafting and earlier issue identification remain potential rather than demonstrated benefits.

REFERENCES

CDISC. 2021a. Study Data Tabulation Model Implementation Guide: Human Clinical Trials, Version 3.4. Clinical Data Interchange Standards Consortium.

CDISC. 2021b. Analysis Data Model Implementation Guide, Version 1.3. Clinical Data Interchange Standards Consortium.

CDISC. 2019. Define-XML Specification, Version 2.1. Clinical Data Interchange Standards Consortium.

ICH. 2025. Good Clinical Practice Guideline E6(R3). International Council for Harmonisation.

Nonaka, I., and H. Takeuchi. 1995. The Knowledge-Creating Company. New York: Oxford University Press.

FDA. 2025. Considerations for the Use of Artificial Intelligence To Support Regulatory Decision-Making for Drug and Biological Products: Draft Guidance for Industry and Other Interested Parties. January 2025.

FDA. 2026. Study Data Technical Conformance Guide: Technical Specifications Document. June 2026.

Pinnacle 21. 2024. Community Validation Rules for CDISC Standards. Available at https://www.pinnacle21.com/validation-rules.

Center for Drug Evaluation, NMPA. 2020. 药物临床试验数据递交指导原则（试行） [Guideline on the Submission of Clinical Trial Data (Provisional)]. Notice No. 16, 2020.

FDA and European Medicines Agency (EMA). 2026. Guiding Principles of Good AI Practice in Drug Development. January 2026.

NMPA et al. 2026. 药物临床试验质量管理规范 [Good Clinical Practice for Drugs]. Effective 1 September 2026.

ACKNOWLEDGMENTS

The author gratefully acknowledges CHIA TAI TIANQING PHARMACEUTICAL GROUP CO., LTD and the Clinical Biostatistics Department for their support and encouragement during the preparation of this paper.

The examples, implementation considerations, and operational case presented in this paper are generalized or fully de-identified from routine programming experience. They do not contain patient-level data, study identifiers, product information, environment paths, or confidential project information. The views expressed in this paper are those of the author and do not necessarily represent the official position of the author's employer.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Siyu Yu

Clinical Biostatistics Department

CHIA TAI TIANQING PHARMACEUTICAL GROUP CO., LTD

E-mail: 1309145398@qq.com