Voting System Standards

This document is part of the Draft Voting System Standards approved for public comment at the December 13, 2001, meeting of the Federal Election Commission. Use the links above to find the other parts.

Volume I, Section 4
Table of Contents

4 Software Standards 4-1
4.1 Scope 4-1
4.1.1 Software Types 4-1
4.1.2 Software Sources 4-2
4.1.3 Location and Control of Software and Hardware on Which it Operates 4-2
4.1.4 Exclusions 4-3
4.2 Software Design and Coding Standards 4-3
4.2.1 Selection of Programming Languages 4-3
4.2.2 Software Integrity 4-4
4.2.3 Software Modularity and Programming 4-4
4.2.4 Control Constructs 4-4
4.2.5 Naming Conventions 4-9
4.2.6 Coding Conventions 4-10
4.2.7 Comments Conventions 4-12
4.2.8 COTS Software 4-12
4.3 Data Quality Assessment 4-13
4.4 Data and Document Retention 4-13
4.5 Audit Record Data 4-13
4.5.1 Pre-election Audit Records 4-14
4.5.2 System Readiness Audit Records 4-14
4.5.3 In-Process Audit Records 4-15
4.5.4 Vote Tally Data 4-16
4.6 Vote Secrecy (DRE Systems) 4-17

4 Software Standards

4.1 Scope

This section describes essential design and performance characteristics of the software embodied in voting systems, addressing both system level software, such as operating systems, and voting system application software, including firmware. The requirements of this section are intended ensure that voting system software is reliable, robust, testable, and maintainable; and supports system accuracy, logical correctness, privacy, security and integrity.

This section recognizes that there is no single "best" way to design software. Many programming languages are available for which modern programming practices are applicable, such as the use of rigorous program and data structures, data typing, and naming conventions. Other programming languages exist for which such practices are not easily applied.

4.1.1 Software Types

The more general requirements of this section apply to software used to support the entire range of voting system activities described in Section 2. More specific requirements are defined for ballot counting, vote processing, creating an unalterable, non-bypassable audit trail, and generating output reports and files. Although this section emphasizes software, the standards described also influence hardware design considerations.

The standards are intended to guide the design of software written in any of the programming languages commonly used for mainframe, mini-computer, and microprocessor systems. They are not intended to preclude the use of other languages or environments, such as those that exhibit "declarative" structure, "object-oriented" languages, "functional" programming languages, or any other combination of language and implementation that provides appropriate levels of performance, testability, reliability, and security. The specific software selections are made by the vendor. However, the use of widely recognized and proven software design methods

will facilitate the analysis and testing of voting system software in the qualification process.

4.1.2 Software Sources

The requirements of this section apply generally to all software developed for use in voting systems, including:

- Software provided by the voting system vendor and its component suppliers;
- Software furnished by an external provider (for example, providers of COTS operating systems and web browsers) where the software may be used in any way during voting system operation; and
- Software developed by the voting jurisdiction.

Compliance with the requirements of the software standards are assessed by several formal tests, including code examination. However, unmodified COTS software is not subject to code examination.

4.1.3 Location and Control of Software and Hardware on Which it Operates

The requirements of this section apply to all software used in any manner to support any voting-related activities, regardless of the ownership of the software or the ownership and location of the hardware on which the software is installed or operates. These requirements apply to:

- Software that operates on voting devices and vote counting devices installed at polling places under the control of the voting jurisdiction;
- Software that operates on ballot printers, vote counting devices, and other hardware typically installed at central or precinct locations (including contractor facilities); and
- Election management software.

However, some requirements apply only in specific situations as indicated in this section. In addition to the requirements of this section, all software used in any manner to support any voting-related activities shall meet the requirements for security described in Section 6 of the Standards.

4.1.4 Exclusions

Some voting systems use equipment, such as personal computers, that may be used for other purposes and have resident on the equipment general purpose software such as operating systems, programming language compilers, database management systems, and Web browsers. Such software is governed by the Standards unless:

- The software provides no support of voting system capabilities;
- The software is removable, disconnectable, or switchable such that it cannot function while voting system functions are enabled; and
- Procedures are provided that confirm that the software has been removed, disconnected, or switched.

4.2 Software Design and Coding Standards

The software used by voting systems is selected by the vendor and not prescribed by the Standards. This section provides standards for voting system software with regard to:

- Selection of programming languages;
- Software integrity;
- Modularity and programming;
- Control constructs;
- Naming conventions;
- Coding conventions;
- Comments;
- COTS Software; and
- Content of Executable Modules.

4.2.1 Selection of Programming Languages

Software associated with the logical and numerical operations on vote data shall use a high level programming language, such as: Pascal, Visual Basic, Java, C and C++. The requirement for the use of high level language for logical operations does not preclude the use of assembly language for hardware-related segments, such as device

controllers and handler programs. Also, operating system software may be designed in assembly language.

4.2.2 Software Integrity

To ensure that the software tested and approved during the qualification process remains unchanged and retains its integrity, all voting system software shall not be self-modifying. External modification of code during execution shall be prohibited.

4.2.3 Software Modularity and Programming

Voting system application software, with the exception of COTS software, shall be designed in a modular fashion in accordance with the following rules:

a. Each module shall have a specific function that can be tested and verified more-or-less independently of the remainder of the code.

b. Each module shall be uniquely and mnemonically named, using unit names that differ by more than a single character. Modules shall follow a standard format consisting of header, declarative statements, and executable statements or comments, in that order. Headers are optional for modules of fewer than ten executable lines.

c. Except for code generated by commercial code generators, code shall be written in relatively small and easily identifiable modules, with no more than 50% of all modules exceeding 60 lines of in length, no more than 5% of all modules exceeding 120 lines in length, and no modules exceeding 240 lines in length. Lines in this context are defined as executable statements or flow control statements. The vendor shall justify the need for the excessive length of each module larger than 120 lines in comments in its header.

d. Each module shall have a single entry point, and a single exit point, for normal program flow. In the event of an abnormal error condition, the error condition shall be handled as close to the point of detection as possible. Abnormal error conditions are defined as device write, device read, file open, file close, or operating system errors module. Conditions that are simply not what was expected or desired are not “abnormal”.

4.2.4 Control Constructs

Voting system software shall use the control constructs, where applicable, as follows:

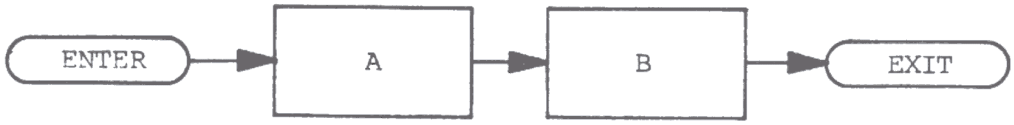
a. If the language does not contain these control constructs, the vendor shall use suitable assembly language constructs, or these constructs shall be simulated by code that follows their logic. If these constructs are simulated, the same form of simulation shall be used throughout the code. No other constructs shall be used to control the logic of program execution.

b. The redirection of control by means of operator intervention or data-driven logic shall not be allowed during the execution of any program unit. The redirection of control resulting from the calling of subroutines, procedures and functions, or by the action of exception handlers (on abnormal error conditions) and interrupt service routines, is allowed. Intentionally thrown exceptions used as GOTOs are prohibited, as are do-while (FALSE) constructs.

Illustrations of control construct techniques are provided in Figures 4-1 through 4-5.

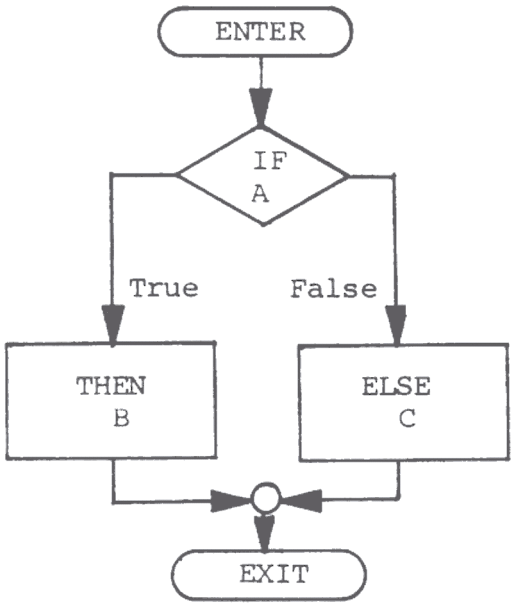
- Fig. 4-1 Sequence
- Fig. 4-2 If -Then -Else
- Fig. 4-3 Do -While
- Fig. 4-4 Do -Until
- Fig. 4-5 Case

As an alternative to the Do-While and Do-Until constructs, the Loop construct shown in Figure 4-6 may be used.

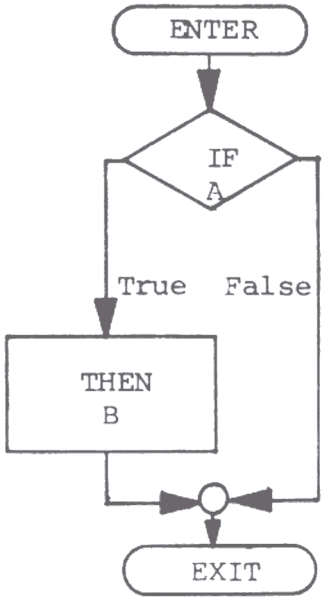


Control flows from Process “A” to the next in sequence, Process “B.”

Figure 4-1, “SEQUENCE”

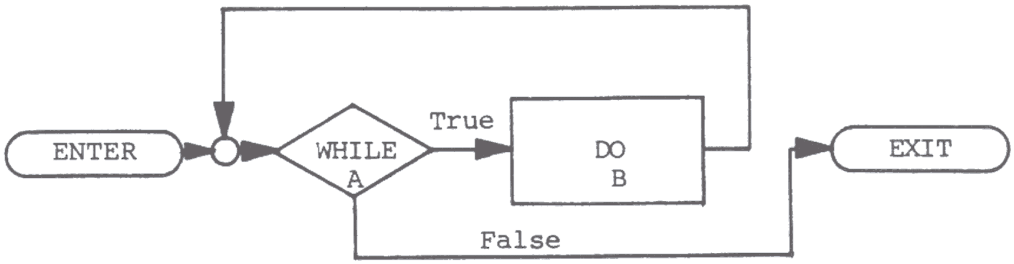


Basic - Flow of control will return to common point after executing Process “B” or “C”. “A” predicates the conditional execution.



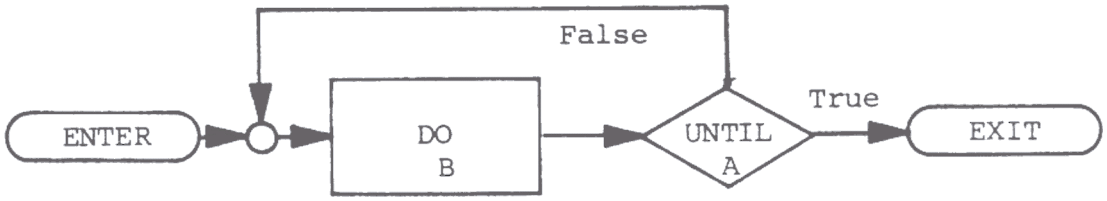
Option - Flow of control will skip a process pending the condition of “A."

Figure 4-2, “IF-THEN-ELSE”



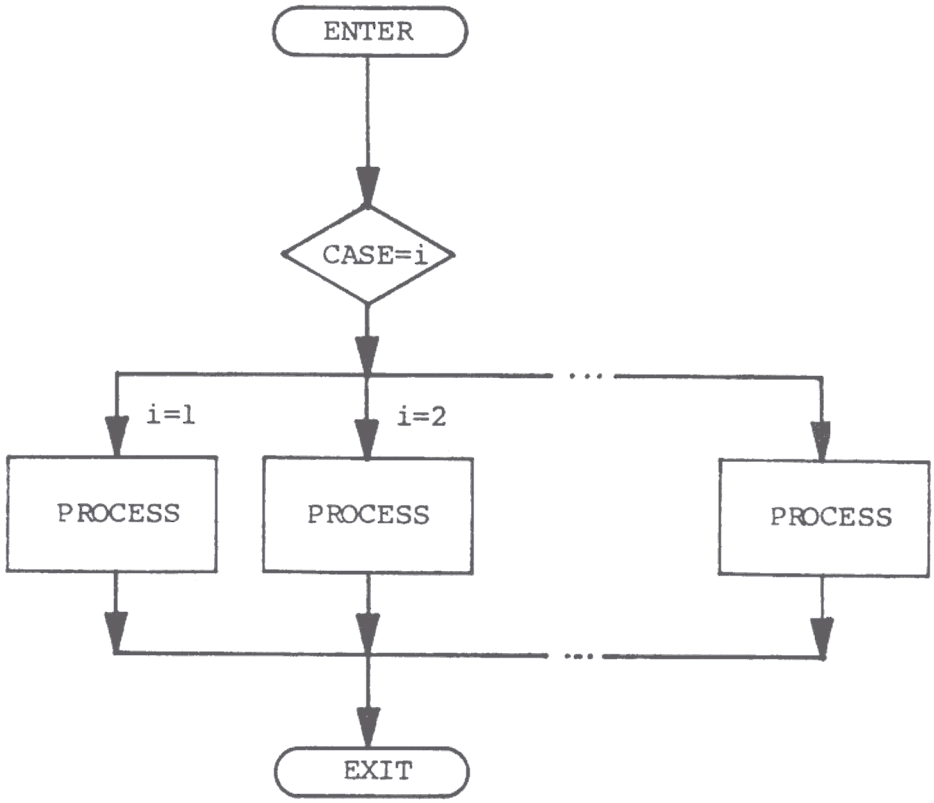
Condition “A” is evaluated. If found to be true, then control is passed to Process “B” and condition “A” is reevaluated. If condition “A” is found to be false, then control is passed out of the loop.

Figure 4-3, “DO-WHILE”



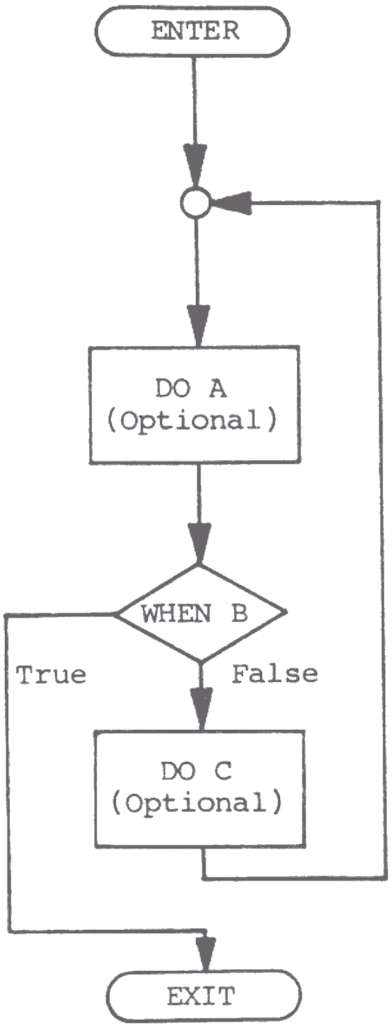
Similar to DO-WHILE, except that the test of condition A is performed after Process B has executed. If condition A is true, control is passed out of the loop.

Figure 4-4, “DO-UNTIL”



Control is passed to a Process based on the value of i.

Figure 4-5, "CASE"



Optional process A is executed. Condition B is then evaluated. If found to be false, optional process C is executed and control is passed to process A. Condition B is then evaluated again. If condition B is true, then control is passed out of the loop.

Figure 4-6, "LOOP"

4.2.5 Naming Conventions

Voting system software shall use the following naming conventions:

a. Object, function, procedure, and variable names shall be chosen so as to enhance the readability and intelligibility of the program. Insofar as possible, names shall be selected so that their parts of speech represent their use, such as nouns to represent objects, verbs to represent functions, etc.

b. Names used in code and in documentation shall be consistent.

c. Names shall be unique. Names shall differ by more than a single character. All single-character names are forbidden except those for variables used as loop indexes.

d. Language keywords shall not be used as names of objects, functions, procedures, variables, or in any manner not consistent with the design of the language.

4.2.6 Coding Conventions

For coding conventions, voting system software shall:

a. In developing source code, be consistent among all units. Uniform calling sequences shall be used, and all parameters shall be validated for type and range on entry into each unit;

b. Be indented consistently and clearly to indicate logical levels;

c. Have no line of code shall exceeding 160 columns in width (including comments).

d. For each line of source code, contain no more than one executable statement and no more than one flow control statement. A function call inside an if() condition is deemed to be an executable statement.

e. If() statements shall have their scopes explicitly delimited. For example:

```
if ( flag ==  true )
        counter = counter + 1;
```

shall be rewritten as

```
if ( flag == true )
{
        counter = counter + 1;
}
```

f. If() statements shall have their scopes explicitly delimited.

g. Use consistent scope specification and indentation throughout the code.

h. Avoid mixed-mode operations. If it is necessary to use them, then all uses shall be identified and clearly explained by comments.

i. The program may exit() at any point, although it should exit under controlled conditions from main(). All exit()s shall result in a message to the user indicating the reason for the exit().

j. Use separate and consistent formats to distinguish between normal status and error or exception messages. All messages shall be self-explanatory and shall

not require the operator to perform any look-up to interpret them, except for error messages which require resolution by a trained technician.

k. Reference variables by fewer than five levels of indirection (i.e. a.b.c.d or a[b].c->d).

l. Have functions with fewer than six levels of indented scope, counted as follows:

```
type function()
{
        if (a = true)
1       {
                if ( b = true )
2               {
                        if ( c = true )
3                       {
                                if ( d = true )
4                               {
                                        while(e > 0 )
5                                       {
                                                code
                                        }
                                }
                        }
                }
        }
}
```

m. Initialized every variable.

n. For all if() statements, be implemented with comparisons in their conditions. For instance,

```
if(flag)
```

is prohibited, and shall be written in the format

```
if (flag = TRUE)
```

in both single and multiple conditions.

o. All constants other than 0 and 1 shall be defined or enumerated, or shall have a comment explaining clearly what the constant means in the context of its use. Enumerations and defines shall be mnemonic and not simply a restatement of the constant (e.g. definitions like “#define 7 SEVEN” are prohibited).

p. In C and C++, only the minimum implementation of the “a = b ? c : d” shall be allowed. Expansions such as “j=a?(b?c:d):e;” are prohibited.

q. All calculations that are of a form more complicated than “a = b * c” shall be commented clearly.

r. Spelling and grammar errors in messages that are part of the user interface are sufficient grounds for non-compliance.

4.2.7 Comments Conventions

Voting system software shall use the following comments conventions:

a. All modules longer than ten lines shall contain headers. Header comments shall provide the following information:

   1) the purpose of the unit and how it works;

   2) other units called and the calling sequence;

   3) a description of input parameters and outputs;

   4) file references by name and method of access (read, write, modify , append, etc.);

   5) global variables used; and

   6) date of creation and a revision record.

b. Descriptive comments shall be provided to identify objects and data types. All variables shall have comments at the point of declaration clearly explaining their use.

c. In-line comments shall be provided to facilitate interpretation of functional operations, tests, and branching.

d. At least 20% of the executable lines in assembly code shall have in-line comments. These comments shall be descriptive and informative.

e. All comments shall be formatted in a uniform manner.

4.2.8 COTS Software

Vendors shall provide information specifying which code, if any, is COTS or public domain code, as well as which portions of COTS software, if any, have been changed by the vendor. Software changed by the vendor in any way must adhere to the Standards.

4.3 Data Quality Assessment

To aid in data quality assessment, all systems shall:

a. Provide real-time monitoring of system status and data quality. The vendor will determine the methods of assessment. Implementation options include but are not limited to:

   1) Hardware monitoring of redundant processing functions that are carried out in parallel or serially; and

   2) Statistical assessment and measures of system operation; and

b. Measure the relative frequency of entry to program units and the frequency of exception conditions.

4.4 Data and Document Retention

All systems shall:

a. Maintain the integrity of voting and audit data during an election, and for at least 22 months thereafter, a time sufficient in which to resolve most contested elections and support other activities related to the reconstruction and investigation of a contested election; and

b. Protect against the failure of any data input or storage device at a location controlled by the jurisdiction or its contractors, and against any attempt at improper data entry or retrieval.

4.5 Audit Record Data

Election audit trails are essential to ensure the integrity of a voting system. Operational requirements for audit trails are described in Section 2.2.5.2 of the Standards. Audit record data are generated by these procedures. The audit record requirements listed in the following subsections are considered essential to the complete recording of election operations and reporting of the vote tally. This list of audit records may not reflect the design constructs of some systems. Therefore, vendors shall supplement it with information relevant to the operation of their specific systems.

4.5.1 Pre-election Audit Records

The following minimum requirements apply to pre-election audit records*:*

a. During election definition and ballot preparation phases, the system shall maintain an audit log of the preparation of the baseline ballot formats and modifications to them, a description of these modifications, and corresponding dates. The log shall include:

   1) The allowable number of selections for an office or issue;

   2) The combinations of voting patterns permitted or required by the jurisdiction;

   3) The inclusion or exclusion of offices or issues as the result of multiple districting within the polling place; and

   4) Any other characteristics that may be peculiar to the jurisdiction, the election, or the polling place's location.

b. The data is required to verify the election-specific database has been correctly prepared and maintained throughout subsequent modifications to the baseline format.

c. The pre-election audit log shall include manual data maintained by election personnel, samples of all final ballot formats, and the ballot preparation edit listings associated with them.

4.5.2 System Readiness Audit Records

The following minimum requirements apply to system readiness audit records:

a. Prior to the start of ballot counting, software shall verify hardware and software status through a readiness audit record. This record shall include the identification of the software release, the identification of the election to be processed, and the results of software and hardware diagnostic tests.

b. In the case of systems used at the polling place, the record shall include the polling place's identification.

c. The ballot interpretation logic capability shall test for correct installation of ballot formats on voting devices.

d. The software shall perform checks of all data paths and memory locations to be used in actual vote recording to protect against contamination of voting data.

e. Upon the conclusion of the tests, the software shall provide evidence in the audit record that the test data have been expunged.

f. For paper-based systems only, the readiness audit capability shall evaluate the accuracy of the ballot reader and the arithmetic-logic unit. It shall allow the processing, or simulated processing, of sufficient test ballots to provide a statistical estimate of processing accuracy.

g. For DRE systems that use a public network, provide a report of test ballots that includes:

   1) Number of ballots sent;

   2) When each ballot was sent;

   3) Machine from which each ballot was sent; and

   4) Specific votes or selections contained in the ballot.

4.5.3 In-Process Audit Records

In-process audit records document system operations during diagnostic routines and the casting and tallying of ballots. At a minimum, the in-process audit records shall contain:

a. Machine generated error and exception messages to demonstrate successful recovery. Examples include, but are not necessarily limited to:

   1) The source and disposition of system interrupts resulting in entry into exception handling routines;

   2) All messages generated by exception handlers;

   3) The identification code and number of occurrences for each hardware and software error or failure;

   4) Notification of system login or access errors, file access errors, and physical violations of security as they occur, and a summary record of these events after processing;

   5) For paper-based systems, an event log of any ballot-related exceptions such as:

      i. Quantity of ballots that are not processable,

      ii. Quantity of ballots requiring special handling, and

      iii. In a central count environment, the quantity and identification number of aborted precincts.

6) Other exception events such as power failures, failure of critical hardware components, data transmission errors, or other type of operating anomaly.

b. Critical system status messages other than informational messages displayed by the system during the course of normal operations. These items include, but are not limited to:

   1) Diagnostic and status messages upon startup;

   2) The “zero totals” check conducted before opening the polling place or counting a precinct centrally;

   3) For paper-based systems, the initiation or termination of card reader and communications equipment operation; and

   4) For DRE machines at controlled voting locations, the event (and time, if available) of activating and casting each ballot (i.e., each voter's transaction as an event). This data can be compared with the public counter for reconciliation purposes.

c. Non-critical status messages that are generated by the machine's data quality monitor or by software and hardware condition monitors. This information is not required in real-time and may, instead, be reported in log form. The intent is to gauge the accuracy of the ballot data and adequacy of the system in monitoring and detecting system processing errors. For example, a cumulative or summary record of data read-write-verify, parity, or check-sum errors and retries is required.

d. System generated log of all normal process activity and system events that require operator intervention, so that each operator access can be monitored and access sequence can be constructed.

4.5.4 Vote Tally Data

In addition to the audit requirements described above, other election-related data is essential for reporting results to interested parties, the press, and the voting public, and is vital to verifying an accurate count.

Voting systems shall meet these reporting requirements by providing software capable of obtaining data concerning various aspects of vote counting and producing reports of them on a printer or at a terminal. At a minimum, vote tally data shall include:

a. Number of ballots cast, by each ballot configuration/type;

b. Candidate and measure vote totals for each contest;

c. The number of ballots read within each precinct, by type, including totals for each party in primary elections;

d. Separate accumulation of overvotes and undervotes for each race or issue (no overvotes would be indicated for DRE voting devices); and

e. For paper-based systems only, the total number of ballots both processed and unprocessable; and if there are multiple card ballots, the total number of cards read.

For systems that produce an electronic file containing vote tally data, the contents of the file shall include the same minimum data cited above for printed vote tally reports.

4.6 Vote Secrecy (DRE Systems)

All DRE systems shall ensure vote secrecy by:

a. Immediately after the voter chooses to cast his or her ballot, record the voter's selections in the memory to be used for vote counting and audit data (including ballot images), and erase the selections from the display, memory, and all other storage, including all forms of temporary storage; and

b. Immediately after the voter chooses to cancel his or her ballot, erase the selections from the display and all other storage, including buffers and other temporary storage.