Y-12

**OAK RIDGE
Y-12
PLANT**

LOCKHEED MARTIN

RECEIVED

FEB 27 1998

OSTI

**SMARTSHELF™: Report of Activities for
Fiscal Year 1997**

Z. W. Bell

Data Systems Research and Development

R. L. Lawson

Instrumentation & Characterization

September 26, 1997

LOCKHEED MARTIN

MASTER

Prepared by the
Oak Ridge Y-12 Plant
managed by

LOCKHEED MARTIN ENERGY SYSTEMS, INC.
for the

U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MANAGED BY
LOCKHEED MARTIN ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

UCN-13672 (26 6-85)

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

**Portions of this document may be illegible
electronic image products. Images are
produced from the best available original
document.**

SMARTSHELF™: Report of Activities for Fiscal Year 1997

Z. W. Bell
R. L. Lawson

Executive Summary

This report covers activities relating to the SmartShelf™ project during the period October 1, 1996, through September 30, 1997. During this year, project team members have advanced the state of both the hardware and software through a six-month-long test that exercised all hardware and nearly all software components. Second-generation node hardware was constructed and tested with the system and it was found that components supplied by Dallas Semiconductor did not meet the manufacturer's specifications and were unusable. However, it was possible to work around this problem by rapidly redesigning the second-generation printed circuit board to use proven first-generation electronics and still fit inside the custom designed second-generation enclosure. Thus, the benefits realized by adopting the custom enclosure was not compromised. The software was improved by moving the user interface from modules developed with LabView to forms, queries, and reports developed with Microsoft Access and the structure of the software was modified to take better advantage of the dynamic data exchange (DDE) client-server architecture built into the Windows95 operating system and Access.

System Testing: October 2, 1996 - April 16, 1997

A test that would exercise the node computer firmware and Microsoft Access 2.0 database software was designed near the end of fiscal year 1996. Software was prepared to develop a schedule of activity in an array of 49 first-generation (1-G) SmartShelf™ node involving the insertion and removal of members of a set of 49 containers fitted with identification buttons. The entire test encompassed 2922 transactions including 1517 connections, 1405 disconnects, and 257 forced errors. The latter were included to exercise the error handling procedures in the code. Each container was used, on average, in 60 transactions and the same was true of nodes.

The SmartShelf™ system was realized as a STD-bus-based node computer that monitored 49 1-G nodes for activity and a PC running Microsoft Access 2.0 and LabView. LabView was used to communicate with the node computer because it combined a simple software interface to the PC's RS-485 port and a number of flexible diagnostic indicators. Access was responsible for the maintenance of the accumulated database of node activity. The node computer was programmed in C and incorporated the TICS real time operating system.

Nodes were arranged as seven chains of seven nodes each attached to seven serial ports of a node computer. The schedule of transactions was developed as a set of 400 sequences of transactions. The algorithm, initialized to have all containers disconnected, began by selecting a random number of transactions between 1 and 14, inclusive, representing up to 28% of the containers in the collection. Then for each transaction, a container was chosen at random. If that container was already connected, the transaction was designated a disconnect, the node at which the container had been connected was marked empty, and the container was marked disconnected. On the other hand, if the container was not already connected, the transaction was designated a connection and an empty node (one at which no container was connected) was selected at random for the site of connection.

A transaction with protocol was defined as the removal or insertion of a container with the presentation of an operator identification fob to the system within the 15-second time limit. That is, a proper insertion consisted of the operator presenting his or her fob, waiting for the lamp on the node to flash, and removing the fob and attaching the node's connector to the container's button within 15 seconds. A proper removal consisted of removing the connector from the container, waiting for the node's lamp to flash, presenting an operator fob within 15 seconds, and then removing the fob within 15 seconds of the lamp ceasing to flash.

10% of the transactions were selected at random to have one of six different protocol errors: Connecting a container without presenting an operator fob (impatient or untrained operator), presenting a fob but failing to attach a container (change of mind), abandoning an operator fob instead of connecting a container (distracted operator), removing a container without presenting an operator fob (theft), removing a container but presenting an operator fob before the node's lamp flashed (impatient operator), and abandoning an operator fob presented after removing a container (forgetful operator). Other errors, such as the failure of node hardware or identification buttons, were not included among the forced errors because these cannot be forced by operator actions. However, such errors occurring during the course of the test would have been detected by the firmware and software.

Table 1. Frequency of Node Selection

	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7
Node 0	62	68	64	45	61	45	49
Node 1	64	50	52	52	63	63	64
Node 2	64	47	59	75	53	61	51
Node 3	77	62	59	59	64	57	58
Node 4	57	79	56	70	50	66	68
Node 5	57	60	65	60	54	75	65
Node 6	46	58	50	64	57	48	69

Table 2. Distribution of forced errors

Container connected without fob	37
Connect transaction aborted	46
Operator abandoned fob during connect	44
Container stolen	45
Fob presented before lamp flashed	49
Operator abandoned fob after disconnect	36

Table 3. Frequency of container selection

Container		Container		Container	
0	49	17	57	34	64
1	57	18	67	35	66
2	59	19	49	36	75
3	60	20	65	37	62
4	58	21	53	38	59
5	59	22	58	39	53
6	59	23	63	40	73
7	62	24	55	41	65
8	53	25	55	42	70
9	53	26	53	43	65
10	59	27	57	44	56
11	68	28	60	45	62
12	63	29	63	46	67
13	63	30	60	47	61
14	52	31	62	48	63
15	45	32	61		
16	58	33	46		

Tables 1, 2, and 3 above give the frequencies with which nodes were selected, errors were selected, and containers were selected. The program that produced the schedule was run on a PC and is included as an appendix to this document.

The schedule of transactions was begun on October 2, 1996. Except weekends and holidays, each morning, noon, and evening a transaction sequence was performed and irregularities were noted. Within 24 hours, the most egregious errors in the firmware were discovered and corrected. Within 96 hours, a bad cable connecting nodes on serial port 7 was discovered and repaired. By October 21, 1996, a sufficient number of unexplainable system software failures, during which the nodes did not respond to the presentation of operator fobs, had been noted and an effort was launched to discover the cause.

Investigation showed that the node computer acted as if its interrupt software were still operating properly (it was able to communicate with the PC running Access, and kept the correct time of day), but otherwise seemed to be stuck waiting for data from a serial port. The problem was traced to the combination of an error in the C code that under rare circumstances forced the software to wait for data at a serial port even though no data would be forthcoming and a feature of the TICS operating system that disabled task switching if the next task ready to run was the same as the current task. Because of some unfortunate coincidences related to setting software timers with TICS, it was possible to satisfy the condition for TICS, and because of the electrical characteristics of the addressable switches (DS2405) and identification buttons (DS1982 and DS1990A), it was possible to confuse the firmware into expecting a character when there would be none. The problem was eliminated by having the software wait either for a character at the serial port or for a timer to expire. In this way, the software no longer stopped because of the lack of data at a serial port. From this time onward (October 22, 1996), no major software problems were encountered.

To investigate the effects of surface preparation, some the stainless steel containers were left bare while others had swatches of duct tape, cellophane tape, and mylar tape applied to them. Adhesive pads were pressed onto identification buttons and then pressed onto either the bare steel or the tape swatch. During the course of the testing, it was found that two (of eight) duct tape swatches began to separate. The problem appeared to be caused by the duct tape being fabricated as an adhesive layer (that remained attached to the container) and a plastic non-adhesive top layer (that remained attached to the identification button via the adhesive pad). No damage or deterioration of the other attachments were noted, despite unscheduled flooding of the containers on November 26, 1996. During the flooding, standing water was allowed to remain on the container lid until the container was scheduled for a transaction. The water covered the tape swatch (if any), the adhesive pad, and most of the identification button. No electrical problems were noted because of the flooding.

Between December 1996 and April 1997, the test ran smoothly without further unscheduled environmental mishaps. Only a few minor errors in the software were discovered and these were rapidly fixed. In addition, a few enhanced diagnostics were added in the firmware on the node computer to detect and recover from unexpected software and hardware interrupts. Scheduled testing was completed with transaction sequence 400 having been performed on the morning on April 16, 1997.

Second Generation Nodes

During the testing described above, a second generation (2-G) node was designed. This node used a single DS2407 dual addressable switch in place of two DS2407 addressable switches, and incorporated a solid-state relay to convert the connection to the identification button from an open drain to an open source configuration. The latter was necessary to eliminate the possibility of two identification buttons being accessed from a single node if they happened to be attached to metal containers that also happened to be electrically shorted together.

Aside from the reduction in the number of components, the use of the DS2407 also removed the requirement that Access maintain a table telling which of the DS2405s in the 1-G nodes ran the node

lamp and which connected to the identification button. Because of this data requirement, the system cannot use a 1-G node as soon as it is connected to the node computer, but rather must be "taught" about the two switches in the node box. However, a node constructed with a DS2407 is immediately usable because only the one address need be learned, and that address will be found by the same software that checks that nodes are still on-line. Thus, the use of the DS2407s will cause configuration data to flow from the node computer to the Access database rather than from the Access database to the node computer. In addition, since only a single addressable switch would be in each 2-G node, the number of nodes that could be attached to a node computer could be doubled.

A quantity of DS2407 switches was procured and a number of 2-G nodes were assembled. It was found immediately that the new nodes were not compatible with the 1-G nodes. Whenever even a single 2-G node was added to the existing network, all nodes appeared to be lost. The problem was eventually traced to the DS2407. It was found that after only a few data bits were sent to the device, it issued a reset pulse that also reset the remainder of the switches on the line. It was also found that the DS2407 drew approximately 50 times the current specified by Dallas Semiconductor (80 μ amps instead of 1.6).

After contacting Dallas Semiconductor, it was learned that the manufacturer was aware of this problem and it was a result of a design error in the integrated circuit. Replacements were promised in early spring 1997. However, to be safe, the 1-G node electronics were redesigned to fit into the 2-G enclosure and operate with the solid-state relay. This proved to have been a wise course of action, since as of this writing, redesigned DS2407s are not yet available from Dallas Semiconductor. Further activity relating to the DS2407-based 2-G node has been suspended pending the availability of functioning parts from Dallas Semiconductor.

Migration of the Software

The testing during the first half of the fiscal year demonstrated the need for a simpler user interface than that provided by LabView. In addition, the system as fielded in October 1996 used Access 2.0, which had been orphaned by Microsoft and superseded by Access 7.0 for Windows 95. Even that product already has a successor, Access 97.

The Access 2.0 database was converted to Access 7.0 without major difficulty; only a few changes were necessitated because of differences between the Visual Basic programming languages of the two products. After conversion, it was verified that the Access 7.0 database performed identically to the Access 2.0 database.

Far more major changes were made in the functionality of the LabView software and the user interface that would now be provided by Access 7.0. The functionality of the single LabView program that interrogated node computers and also was responsible for learning new identification buttons and nodes was split between two LabView programs: A LEARN server for the latter functions, and a node computer client to monitor node computer activity. The LEARN server, designed to be a DDE server to code running through Access, is only accessed when the user needs to register an operator fob, container identification button, or node with Access. The node computer client (simply called the client below) is designed to be a true DDE client to Access and uses Access to update the transaction data tables. This division of functionality allows the node computer client to monitor node activity independent of user activity and LEARN server activity.

The database was split into two databases: One database, SMRTSHLF.MDB, contains only the transaction table, node table, container table, node computer table, and operator table. The second database, SMRTCODE.MDB, contains all the forms, reports, queries, macros, and code modules, as well as tables of transaction descriptions and container status. Tables in the SMRTSHLF.MDB are linked to SMRTCODE.MDB so only the latter database need be opened by Access. This division of information permits the modification of the forms, queries, macros, descriptions, and modules without affecting the

transaction tables. That is, if new functionality is desired, it will only be necessary to replace SMRTCODE.MDB.

The user interface was shifted to Access 7.0. A switchboard form has been added to allow the user to select between several functional areas: system control functions, node computer, node, operator, container functions, and reports. Under control functions are found selections to allow the user to stop the system, change timing parameters, enable or disable the display of errors, and enable or disable periodic downloading of database information to the node computers, and to force the client to reload the current system configuration. Under node computer, node, operator, and container functions are found selections to register an item, declare an item inactive, reactivate an item, and change information about an item. Reports include a list of presently attached containers, presently detached containers, invalid transactions, transactions by a specified operator or at a specified node or involving a specified container. All user input is via mouse clicks on screen buttons and user-filled text boxes. Invalid or missing information is detected and flagged. The user interface, as it is presently implemented, integrates into a single set of menus all the functionality of the system.

Logic has been added to SMRTCODE.MDB to require the registration of operator fobs, containers, and nodes before they are used. The use of unregistered items are flagged as invalid and an alert box is displayed on the computer screen.

The system is started by launching the client which, in turn, launches the LEARN server and then Access 7.0. As the client executes its cycle of interrogating the node computers and sending them the current time of day, it notifies Access of its progress and causes Access to display its status on the switchboard screens. It also checks that the LEARN server is still running and restarts it if it is not found. At 10-second intervals, the client notifies Access that it is still running by executing a macro in SMRTCODE.MDB. In addition, once per day, at a random time between midnight and 4:00 a.m., the client closes Access, re-opens it, compacts the two databases, closes it again, and then re-opens SMRTCODE.MDB and SMRTSHLF.MDB. This allows Windows 95 to recover system resources, and forces the disk files to reflect all data that have been acquired in the previous 24 hours. During this time, the node computers continue to monitor their nodes.

In addition to acting as the user interface, Access has a schedule of activities. Periodically (with the length of the interval selected by the user) Access signals the client that it should download the current system configuration. This is done to ensure that the node computer data and the database match. Although it is not likely, it is possible for transaction data to be lost between the PC and a node computer because of noise on the serial line, causing the node computer configuration data not to be synchronized with the Access database. Access also checks that the client program has communicated with it at least once during the past minute. If more than a minute passes without any messages from the client, Access displays a warning message instructing the user to restart the entire system.

Anticipated Future Activities

The coming fiscal year will see SmartShelf™ fully integrated with the Graphical Facility Information Center (GraFIC) system. The database will be shared so that a GraFIC operator may query SmartShelf™ from the GraFIC workstation. This will allow all storage system information to be accessible from a single point.

2-G node hardware and software will be perfected if Dallas Semiconductor makes good on its promise to provide on-specification DS2407s. 2-G nodes will double the number of items that can be monitored by a single node computer thereby decreasing the costs associated with the system. In addition, 2-G nodes will be immediately usable and will not require a learning procedure. This will simplify the software.

The possibility of wireless operation will be considered. At present, the most distant node computer can be up to 3000 feet from the PC, a limitation imposed by the electrical specifications of RS-485, and node

computers require installation of cable. Wireless operation offers the advantage of eliminating the expense of running cable and extending the range of the system. It remains to be seen what are the most promising technologies to implement wireless communication and where it is most advantageous to use them.

It is expected to deploy a SmartShelf™ system at the Y-12 Plant. This deployment will give operational feedback from personnel not involved in the development of the system. From this feedback improvements to the user interface, training material, reporting system, and operational protocols will be made to make SmartShelf™ easier to use and a more valuable computer tool.

Appendix Scheduler Software

```

#include <stdio.h>
#include <stdlib.h>

#define NPORTS 8
#define NPARTS 49
#define NTRANS 14
#define MAXPARTS_PER_PORT 7
#define FIRST_PORT 1
#define LAST_PORT 7

#define FRAND() (((float)rand())/((float)RAND_MAX))

#define ERROR_RATE 0.10
#define NERRORS 6

main(int argc, char *argv[])
{
    int array[NPORTS][MAXPARTS_PER_PORT];
    int parts[NPARTS];
    int niter, trans, i, j, k, l, m;
    int part_hits[NPARTS];
    int array_hit[NPORTS][MAXPARTS_PER_PORT];
    int nconnects = 0;
    int ndisconnects = 0;
    int ntransactions = 0;
    int nerrors[NERRORS];
    char *errtypes[NERRORS] = {"Part connected without ID (0100)",
                                "Connect transaction aborted (0200)",
                                "Operator left ID connected on connect (0300)",
                                "Part taken without ID (0800)",
                                "ID presented before prompt (0c00)",
                                "Operator left ID connected on disconnect (0f00)"};

    int toterrors = 0;
    FILE *fptr;

    if (argc < 2)
    {
        printf("\n\nHow many iterations? ");
        scanf("%d", &niter);
    }
    else
    {
        sscanf(argv[1], "%d", &niter);
    }
    printf("\n\n%d iterations will be calculated.\n\n", niter);
    fptr = fopen("c:\\schedule.dat", "w");
    for(i = 0; i < NPORTS; i++)
    {
        for(j = 0; j < MAXPARTS_PER_PORT; j++)
        {
            array[i][j] = -1; // each node is empty
            array_hit[i][j] = 0;
        }
    }
    for(i = 0; i < NPARTS; i++)
    {
        parts[i] = -1; // each container is disconnected
        part_hits[i] = 0;
    }
    for(i = 0; i < NERRORS; i++) nerrors[i] = 0;
    for(k = 1; niter != 0; niter--)
    {
        trans = (int)(FRAND() * (float)(NTRANS)) + 1; // select number of transactions for this set
        trans = (trans < 1) ? 1 : (trans > NTRANS ? NTRANS : trans);
        fprintf(fptr, "Sequence %d contains %d transactions\n\n", k++, trans);
        while (trans != 0)

```

```

{
    j = (int)(FRAND() * (float)(NPARTS)); // pick the container
    j = (j < 0) ? 0 : (j > NPARTS - 1 ? NPARTS - 1 : j);
    part_hits[j]++;
    if (parts[j] >= 0) // container is connected
    {
        i = parts[j] % MAXPARTS_PER_PORT; // decode port and node numbers
        l = parts[j] / MAXPARTS_PER_PORT;
        array[l][i] = parts[j] = -1; // mark node and container disconnected
        array_hit[l][i]++;
        ndisconnects++;
        if (FRAND() < ERROR_RATE) // there should be an error
        {
            m = (int)(3.0 * FRAND()); // select the error
            if (m < 1)
            {
                fprintf(fp, "Disconnect container %3d @ port %2d, box %2d\n", j, l, i);
                fprintf(fp, "DO NOT PRESENT OPERATOR ID\n\n");
                nerrors[3]++;
            }
            else if (m < 2)
            {
                fprintf(fp, "Disconnect container %3d @ port %2d, box %2d\n", j, l, i);
                fprintf(fp, "PRESENT OPERATOR ID BEFORE LED FLASHES\n\n");
                nerrors[4]++;
            }
            else
            {
                fprintf(fp, "Disconnect container %3d @ port %2d, box %2d\n", j, l, i);
                fprintf(fp, "PRESENT OPERATOR ID FOR 1 MINUTE\n\n");
                nerrors[5]++;
            }
            toterrors++;
        }
        else
            fprintf(fp, "Disconnect container %3d @ port %2d, box %2d\n", j, l, i);
    }
    else // the container is currently disconnected
    {
        i = (int)(FRAND() * (float)(MAXPARTS_PER_PORT)); // find an empty node
        i = (i < 0) ? 0 : (i >= MAXPARTS_PER_PORT ? MAXPARTS_PER_PORT - 1 : i);
        l = (int)(FRAND() * (float)(LAST_PORT - FIRST_PORT + 1)) + FIRST_PORT;
        l = (l < FIRST_PORT) ? FIRST_PORT : (l >= LAST_PORT ? LAST_PORT : l);
        while(array[l][i] >= 0)
        {
            i = (int)(FRAND() * (float)(MAXPARTS_PER_PORT));
            i = (i < 0) ? 0 : (i >= MAXPARTS_PER_PORT ? MAXPARTS_PER_PORT - 1 : i);
            l = (int)(FRAND() * (float)(LAST_PORT - FIRST_PORT + 1)) + FIRST_PORT;
            l = (l < FIRST_PORT) ? FIRST_PORT : (l >= LAST_PORT ? LAST_PORT : l);
        }
        array[l][i] = j;
        parts[j] = l * MAXPARTS_PER_PORT + i; // mark container as at this node
        array_hit[l][i]++;
        if (FRAND() > ERROR_RATE) // there should not be an error now
        {
            fprintf(fp, "Connect container %3d @ port %2d, box %2d\n", j, l, i);
        }
        else // this transaction should have an error
        {
            m = (int)(3.0 * FRAND()); // select the error
            if (m < 1)
            {
                fprintf(fp, "Connect container %3d @ port %2d, box %2d\n", j, l, i);
                fprintf(fp, "DO NOT PRESENT OPERATOR ID\n\n");
                nerrors[0]++;
            }
            else if (m < 2)
            {
                fprintf(fp, "PRESENT OPERATOR ID @ PORT %2d BOX %2d, WAIT FOR LED TO FLASH, DISCONNECT ID\n\n", l, i);
                array[l][i] = parts[j] = -1;
            }
        }
    }
}

```

```

        errors[1]++;
    }
    else
    {
        fprintf(fp, "\n\n PRESENT OPERATOR ID FOR 1 MINUTE @ PORT %2d BOX %2d\n\n",
            1, i);
        array[1][i] = parts[j] = -1;
        errors[2]++;
    }
    toterrors++;
}
nconnects++;
}
trans--;
ntransactions++;
}
}
fprintf(fp, "\n\n\nStatistics for %d transactions with %d Connects, %d Disconnects, and %d total errors\n\n",
    ntransactions, nconnects, ndisconnects, toterrors);
for(i = 0; i < NPARTS; i++) fprintf(fp, " Container %3d: %7d transactions\n", i, part_hits[i]);
fprintf(fp, "\n\n      Array Hits\n\n");
for(i = 0; i < NPORTS; i++) fprintf(fp, " Port %2d", i);
fprintf(fp, "\n\n");
for(i = 0; i < MAXPARTS_PER_PORT; i++)
{
    fprintf(fp, " Box %2d", i);
    for(l = 0; l < NPORTS; l++) fprintf(fp, " %8d", array_hit[l][i]);
    fprintf(fp, "\n\n");
}
fprintf(fp, "\n\n%d total errors as:\n\n", toterrors);
for(i = 0; i < NERRORS; i++) fprintf(fp, " Type %2d errors: %6d (%s)\n", i, errors[i], errtypes[i]);

    printf("\n\n%d Transactions with %d Connects, %d Disconnects, and %d total errors\n\n", ntransactions, nconnects, ndisconnects,
        toterrors);
return 0;
}

```

Distribution:

Z. W. Bell, 9203, MS-8084
S. W. Combs, 9706-1, MS-8213
M. M. Conger, 9113, MS-8207
S. O. Cox, 9113, MS-8207
G. A. Dailey, 1099 COM, MS-7615
T. A. Gafford, 9113, MS-8207
C. D. Key, 9119, MS-8233
D. L. Hanline, 9113, MS-8207
W. H. Hopwood, Jr., 9119, MS-8233
R. L. Lawson 9203, MS-8084
S. L. Packard, 1099 COM, MS-7622
C. A. Pickett (5), 9203, MS-8084
L. J. Satkowiak, 2020 COM, MS-8257
L. E. Seiber, 9203, MS-8084
S. P. Singh, 7601, MS-6306
J. M. Smith, Jr., 9103 MS-8142
C. E. Tilley, Jr., 9204-2E MS-8237
V. S. Tisdale/C. E. Walker, 9720-5, MS-8144
Y-12 Plant Records Services (3), 9711-5, MS-8169
[2 copies for OSTI and 1 copy for Central Files]