THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 10, December 1981
Printed in U.S.A.

Microprocessor Firmware Update Inventory Model

By S. M. BRECHER

(Manuscript received June 10, 1981)

Microprocessor-based systems are used in many applications of modern telecommunications. The controlling program of most microprocessor systems is stored in firmware which is usually coded into erasable programmable read-only memory chips (EPROM). As new services are implemented, there is a continuing need to update the firmware, a potentially expensive process. In this paper, a method is presented for determining the resources necessary for updating EPROM firmware from a centralized location by using a rotating inventory scheme.

I. INTRODUCTION

Microprocessor-based systems are used in many applications of modern telecommunications. The controlling program (firmware) of most microprocessor systems, is coded into either read-only memory (ROM), programmable read-only memory (PROM), or erasable programmable read-only memory (EPROM) chips, which are mounted on circuit boards. As new services are implemented in the microprocessor-based systems, there is a continuing need to update the firmware. Frequent updating of firmware is more effectively accomplished by the use of EPROM chips, because they can be repeatedly erased by exposure to ultraviolet light and reprogrammed by the use of a specially designed unit. Typical EPROM circuit packs may require 0.5 hours for erasing and 1.5 hours for programming.

One serious drawback to using EPROM firmware is that it cannot be altered by means of a data link. A change in firmware entails the removal and reinstallation of the memory circuit boards or packs. Because of this manual process, updating a large number of microprocessor systems may involve a long and costly procedure.

The object of this paper is to provide a quantitative method to

determine the level of spare boards and programming units necessary to achieve a predeterminated time span for updating all the microprocessor-based systems.

1.1 Firmware update process

Typically, there are three conditions that could affect a firmware module: (*i*) Program updates because of new feature introduction, (*ii*) program changes caused by fixing of "bugs," and (*iii*) program changes because of hardware updates. In general, a firmware update process begins with a notice of a change sent by the microprocessor manufacturer to centralized programming sites where ultraviolet light programming units and a spare inventory of circuit packs are kept. Over a dial-up connection, the programming unit receives the latest program version and writes it onto spare circuit packs taken from inventory. When all spare circuit packs are rewritten they are shipped to specific distribution sites associated with a subset of the microprocessor systems. From each distribution site, craft persons are dispatched to install the updated boards. The removed circuit packs are then returned to the centralized programming site for updating. The recycling process continues until all the microprocessor systems in the field, plus their allocated maintenance spares, are updated.

1.2 Basic model

The flow and assumptions of the basic update model, for a typical process schedule, are shown in Fig. 1. The process of updating begins when a firmware change message is received at the centralized programming site. An initial period is used for administrative procedures, unpacking of inventoried circuit packs, erasing, and reprogramming, packing and crating, and shipping of the circuit packs to the distribution sites. Out of this initial time interval, it can be assumed that one day will be needed for reprogramming the spare EPROM circuit packs. If there are not enough programming units, a delay in the update process could be incurred.

Once the rewritten boards arrive at the distribution sites, the associated microprocessor systems are scheduled for update. The process of coordinating the installation forces necessary for the update is not immediate, and the following distribution can be assumed to characterize the update interval: a proportion, i_1, of the updated boards are installed and an equal amount of outdated boards are returned to the centralized location for reprogramming in one week; a proportion, i_2, of the updated boards are recycled in two weeks; a proportion, i_3, in three weeks, and so on until all the boards are recycled.

Once the outdated boards arrive back at the programming site, the process of reprogramming begins again. This procedure continues until all of the microprocessor systems in the field have been updated.

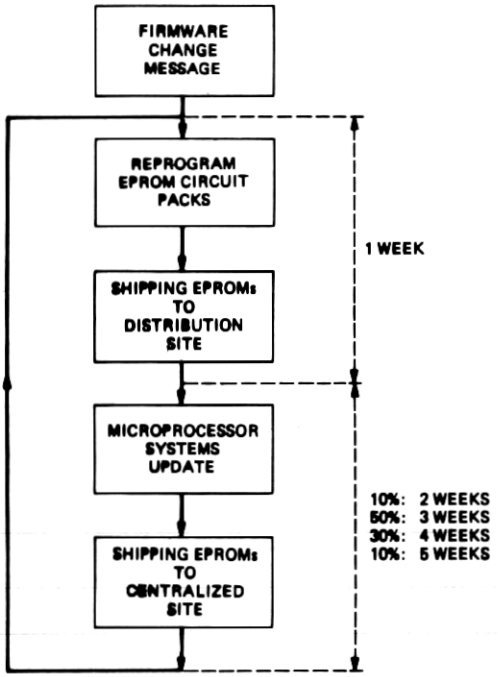

Fig. 1—Typical firmware update process.

II. SYSTEM PARAMETERS

The system parameters to be determined are the update time, the total update spares, and the number of programming units. Update time, T, is defined as the interval in weeks between a change notice and the time at which all the systems in the field, including maintenance spares, have been updated. To determine T, it is necessary to define the concepts of update spare, total update spares, total systems, and spare ratio.

An update spare is defined as one complete set of EPROM circuit packs for one microprocessor system. Since an update spare is defined as one full set of boards, an update spare may be equated to a microprocessor system and vice versa. Total update spares is defined as the number of EPROM circuit packs at the centralized programming site available for initiating the update process. Total systems is defined as the total number of microprocessor-based systems in service and their maintenance spares to be updated when a program change is issued.

The spare ratio, S, is defined as the ratio of total update spares to total systems; that is,

$$S = \frac{\text{Total Update Spares}}{\text{Total Systems}}.$$

A spare ratio of one, $S = 1$, says that for each microprocessor system in the field and its maintenance spares there is one set of update spares at the programming site. In this case, T is the time required for one pass through the basic flow shown in Fig. 1. Similarly, if S is 0.5, one spare for every two systems in the field and their maintenance spares, T is the time required for approximately two passes through the basic flow. By following this reasoning, it becomes clear that T is a function of S.

III. MODEL FORMULATION

3.1 Update time

The update time can be determined by considering the following installation distribution mentioned in Section I and shown in Table I. In Table I, i_j = proportion of boards updated in week j, w = number of weeks required for returning all spares updated in week 0; $i_j = 0$, for $j = 0$ and $j > w$, i.e., no system can be updated in week 0 with spares reprogrammed in that week. Then,

$$\sum_{j=0}^{w} i_j = 1,$$

which implies that after w weeks all the spares reprogrammed in week 0 are returned. Define $q(n)$ as the quantity of spares updated in week n, with $n = 0$, week of change message, and $n = T$, week by which all systems are updated. The update distribution can be modeled by

$$\begin{aligned}
q(0) &= \text{Total Update Spares} = S \times \text{Total Systems},\\
q(1) &= i_1 q(0),\\
q(2) &= i_2 q(0) + i_1 q(1),\\
q(3) &= i_3 q(0) + i_2 q(1) + i_1 q(2),\\
&\vdots \qquad\qquad , \quad \text{and}\\
q(n) &= \sum_{j=0}^{n-1} i_{n-j} q(j). \qquad (1)
\end{aligned}$$

Equation (1) can be rewritten as:

$$q(n) = \sum_{j=0}^{w} i_j q(n-j), \qquad (2)$$

since $i_o = 0$, and with initial conditions

$$\begin{aligned}
q(0) &= \text{Total Update Spares} = S \times \text{Total Systems},\\
q(n) &= 0, \qquad \text{for} \quad n < 0. \qquad (3)
\end{aligned}$$

Table I—Distribution of microprocessor systems update

Week of Update Process	Proportion Updated
1	i_1
2	i_2
3	i_3
⋮	⋮
w	i_w

Equations (2) and (3) are the difference equations describing the microprocessor update process. To obtain the solution of the difference equation, define the ratio of total update spares updated in week n:

$$a(n) = \frac{q(n)}{\text{Total Update Spares}}, \tag{4}$$

where

$$0 \leq a(n) \leq 1.$$

Substituting $q(n)$ from eq. (4) into eq. (2)

$$a(n) = \sum_{j=0}^{w} i_j a(n-j), \tag{5}$$

and

$$a(0) = 1. \tag{6}$$

Before solving eq. (5), notice that the update process ends when the total number of updated spares equals the number of total systems; that is,

$$\sum_{n=1}^{T} q(n) = \text{Total Systems}, \tag{7}$$

where T is the update time.

Substituting eq. (4) into eq. (7) and using the definition of spare ratio,

$$\sum_{n=1}^{T} a(n) = \frac{1}{S}. \tag{8}$$

Expressions (5), (6), and (8) are the difference equations for the weekly ratio of updated spares and T.

To obtain T, eqs. (5) and (8) must be solved. The solution can be obtained using a computer simulation, or a closed-form technique such as the z-transform. To use the z-transform method, recall the following properties:[1]

(*i*) *Translation property*:

$$Z[f(n+k)] = z^k F(z) - z^k \sum_{j=0}^{k-1} f(j) z^{-j},$$

where $F(z) = Z[f(n)]$.

(*ii*) *Summation property*:

$$Z\left[\sum_{n=-\infty}^{N} f(n)\right] = \frac{z}{z-1} F(z) + \frac{z}{z-1} \sum_{n=-\infty}^{-1} f(n).$$

(*iii*) *Final value property*:

$$\lim_{t \to \infty} f(t) = \lim_{z \to 1} (z-1) F(z).$$

Using property (*i*) in eqs. (5) and (6) gives:

$$A(z) = \frac{z^w}{z^w - \sum_{j=1}^{w} i_j z^{w-j}}, \tag{9}$$

where i and w were previously defined, and

$$A(z) = Z[a(n)].$$

Using property (*ii*) in eq. (8), and the initial conditions for $a(n)$, gives

$$Z\left[\sum_{n=0}^{T} a(n)\right] = \frac{z}{z-1} A(z). \tag{10}$$

Substituting eq. (9) into eq. (10) and considering that $i_o = 0$, gives

$$Z\left[\sum_{n=1}^{T} a(n)\right] = \frac{z}{z-1} \frac{\sum_{j=1}^{w} i_j z^{w-j}}{z^w - \sum_{j=1}^{w} i_j z^{w-j}}.$$

The inverse z-transform of eqs. (9) and (6) gives the following relationship between T and S:

$$\frac{1}{S} = Z^{-1}\left\{\frac{z}{z-1} \frac{\sum_{j=1}^{w} i_j z^{w-j}}{z^w - \sum_{j=1}^{w} i_j z^{w-j}}\right\}. \tag{11}$$

The right-hand term of eq. (11) can be inverted using classical techniques and z-transform tables.

An example can be used to show an application of the z-transform technique. Consider the firmware update distribution shown in Fig. 1 and given in Table II.

Table II—Firmware update distribution for a six-week turnaround interval (w = 6)

Week of Update Process	Proportion Updated i_j
3	0.10
4	0.50
5	0.30
6	0.10

Substituting the above values of i_j in eq. (11),

$$\frac{1}{S} = Z^{-1}\left\{\frac{z}{z-1}\,\frac{0.1z^3 + 0.5z^2 + 0.3z + 0.1}{z^6 - 0.1z^3 - 0.5z^2 - 0.3z - 0.1}\right\}. \tag{12}$$

Using the partial-fraction expansion technique for inverting eq. (12), the closed-form expression for S as a function of T is

$$\begin{aligned}\frac{1}{S} &= 0.22727T - 0.36983 \\ &\quad + 0.09374(-0.67495)^T \\ &\quad + 0.03228(0.45089)^T \sin(2.27567T + 1.23494) \\ &\quad + 0.32472(0.85369)^T \sin(1.41837T + 0.85769).\end{aligned} \tag{13}$$

A closed form solution to this transcendental equation in T is difficult to obtain. An approximate solution is obtained when T is large. In this case

$$\frac{1}{S} \approx 0.22727T - 0.36983, \tag{14}$$

or

$$T \approx \frac{4.4}{S} + 1.62725, \tag{15}$$

which is the expression for T. Equation (15) satisfies all update times with an error of less than 2 percent for $T \geq 10$, and with a maximum error of 7 percent occurring in week 7.

An integer solution can be obtained by making:

$$T = \left\lceil \frac{4.4}{S} + 1.6 \right\rceil, \tag{16}$$

with a maximum positive error of 1 week. The $\lceil \cdot \rceil$ operator denotes rounded-up approximation. Figure 2 shows the representation of the update time as a function of the spare ratio for this example.

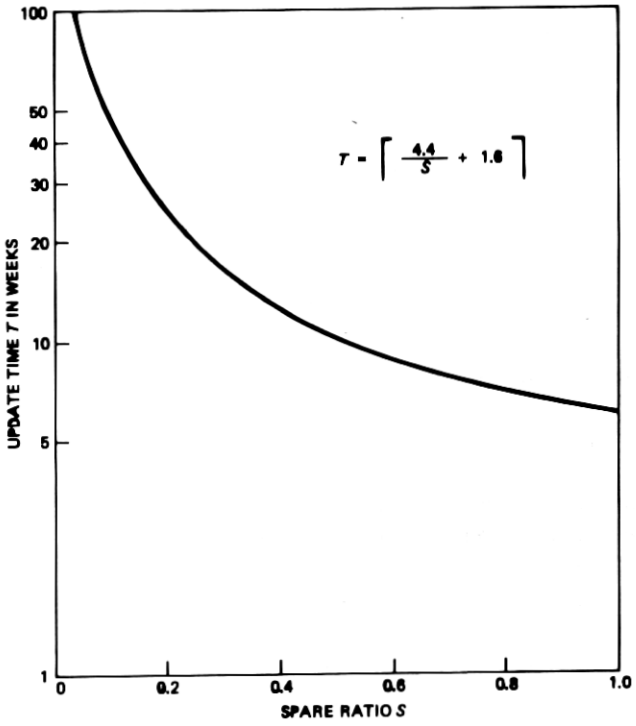

Fig. 2—Update time.

A more simplified approach was used for solving the original real-life update time problem. From the given installation distribution, the expected number of weeks required to program, ship, and install a set of spares, i.e. one pass through the basic flow of Fig. 1, is

$$\bar{T} = 0.1 \times 3 + 0.5 \times 4 + 0.3 \times 5 + 0.1 \times 6 = 4.4 \text{ weeks.}$$

This means that one reprogramming process for the average microprocessor system takes 4.4 weeks. The update time is determined by the number of times this process will be repeated, plus the error introduced by using the expected value approach. The number of times the process is repeated depends on the spare ratio. Therefore, the update time can be represented by the following equation:

$$T = \frac{4.4}{S} + \text{error},$$

which agrees with eq. (16) for an error of 1.6.

The update time algorithm shows the dependency between T, S, and the microprocessor's update distribution. For an illustrative purpose, assume that there are as many spare boards at the programming

site as microprocessor systems in the field, i.e., $S = 1$. In this case, $T = 6$ weeks. This period is the minimum time possible under the given basic assumptions. On the other hand, if we have spare boards available for only 50 percent of microprocessor systems in the field, $S = 0.5$, the update time according to the algorithm is $T = 11$ weeks. The conclusion is that T is directly proportional to the installation distribution and inversely proportional to S.

3.2 Total update spare requirements

The number of update spares necessary at the centralized programming site is determined by the update time and the total number of microprocessor systems. The update time determines the spare ratio which, when combined with the number of systems, gives the number of update spares as indicated in Section II:

$$\text{Total Update Spares} = \lceil S \times \text{Total Systems} \rceil. \tag{17}$$

For the example of Section 3.1, the update spare requirements for implementing a firmware update system for 150 total systems in 20 weeks is computed as follows:

First, the spare ratio for 20 weeks from Fig. 2 is;

$$S = 0.24.$$

Then, the number of update spares is given by

$$\text{Total Update Spares} = \lceil 0.24 \times 150 \rceil = 36.$$

3.3 Programming units

The number of programming units necessary for a centralized operation can be determined as a function of their capacity. To determine the number of units, two concepts must be used: weekly spares and programming days.

Weekly spares, $q(n)$, already defined in Section III, is the number of spares returned each week to the site for reprogramming. The number of weekly spares is a function of the installation distribution. Programming days, P, is the number of days per week allocated to the erasing and programming of EPROM circuit packs.

The number of programming units is the following function of the weekly spares, the number of programming days, and the unit capacity:

$$\text{Programming Units} = \left\lceil \frac{q(n)}{P} \times \frac{1}{C} \right\rceil, \qquad \text{for all } n,$$

where C, microprocessor systems per day per unit, is the capacity factor of the unit. The number of programming units can be computed using the three following approaches.

An upper bound for the weekly spares can be used as a starting point

for determining the number of programming units. The number of total update spares available at the site can be used for this purpose. In this case, the number of units is:

$$\text{Programming Units} = \left\lceil \frac{q(o)}{P \times C} \right\rceil. \tag{18}$$

This algorithm overestimates the number of programming units because only in the initial week of the update process are weekly spares equal to update spares. In the following weeks, the number of spares returning vary according to the assumed installation distribution.

A second approach for determining programming units is to make the weekly spares equal to the largest number of spares updated in any week after the initial week of the update process. In this case, the number of units is:

$$\text{Programming Units} = \max_{n \geq 1} \left\lceil \frac{q(n)}{P \times C} \right\rceil. \tag{19}$$

This algorithm reduces the number of units required and introduces a delay in the first week of the update process.

A third approach is to make the weekly spares equal to the expected number of spares returned each week for reprogramming. This is equivalent to the steady-state of the discrete time process. This approach satisfies the programming assumption of one day in 50 percent of the weeks and incurs a one-day delay during the other 50 percent. Therefore, on the average, programming will require 1.5 days per week. As a result, the nondelayed update time, which was derived assuming one day per week for programming, must be increased by 10 percent (one-half a day per five-day week). In this case, the number of units is:

$$\text{Programming Units} = \left\lceil \frac{\bar{Q}}{P \times C} \right\rceil, \tag{20}$$

where $\bar{Q}$ = expected number of spares returned each week. To evaluate $\bar{Q}$, the final value theorem can be used. Property (*iii*) of the z-transform methodology gives:

$$\begin{aligned} \bar{Q} = q(\infty) &= \lim_{z \to 1} (z - 1) Q(z) \\ &= \text{Total Update Spares} \times \lim_{z \to 1} (z - 1) A(z). \end{aligned} \tag{21}$$

Substituting eq. (9) into eq. (21) yields:

$$\bar{Q} = \text{Total Update Spares} \times \lim_{z \to 1} \frac{(z - 1) z^w}{z^w - \sum_{j=0}^{w} i_j z^{w-j}},$$

which is indeterminate of the type 0 over 0, because $\sum_{j=0}^{w} i_j = 1$. The application of l'Hopital's rule solves this problem.

For the distribution given in the example of Fig. 1, the expected number of spares is:

$$\bar{Q} = q(\infty) = \text{Total Update Spares} \times \lim_{z \to 1} \frac{(z-1)z^6}{z^6 - 0.1z^3 - 0.5z^2 - 0.3z - 0.1}$$

$$= \text{Total Update Spares} \times 0.22727$$

$$= \text{Total Update Spares} \times \bar{A},$$

where $\bar{A}$ is the expected ratio of total update spares returned each week. It can be seen that the reciprocal of $\bar{A}$ represents the average number of weeks required for one pass through the basic flow of Fig. 1; that is,

$$\bar{T} = \frac{1}{\bar{A}} = \frac{1}{0.22727} = 4.4 \text{ weeks.}$$

For the example of Fig. 1, where one day per week is assumed for reprogramming, and the unit capacity is three systems per day, eqs. (18), (19), and (20) become, respectively:

$$\text{Programming Units} = \left\lceil \frac{\text{Total Update Spares}}{3} \right\rceil, \tag{22}$$

$$\text{Programming Units} = \left\lceil \frac{\text{Total Update Spares}}{6} \right\rceil, \quad \text{and} \tag{23}$$

$$\text{Programming Units} = \left\lceil \frac{\text{Total Update Spares}}{13.2} \right\rceil. \tag{24}$$

Comparing eqs. (22), (23), and (24), it can be seen that eq. (24) reduces the number of programming units by a factor of more than four, with respect to eq. (22), with the introduction of a 10-percent delay in the update time.

The algorithms discussed above are represented in Fig. 3. The number of required programming units are shown as a function of the number of total update spares. To contain the size of the plot, a scale factor, N, was introduced. Also included is a parameter D, which is the additional delay in the update time because of the reduced number of programming units at the centralized site.

3.4 Example

To have a better understanding of the determination of the resource requirements for the firmware update process, consider the following example.

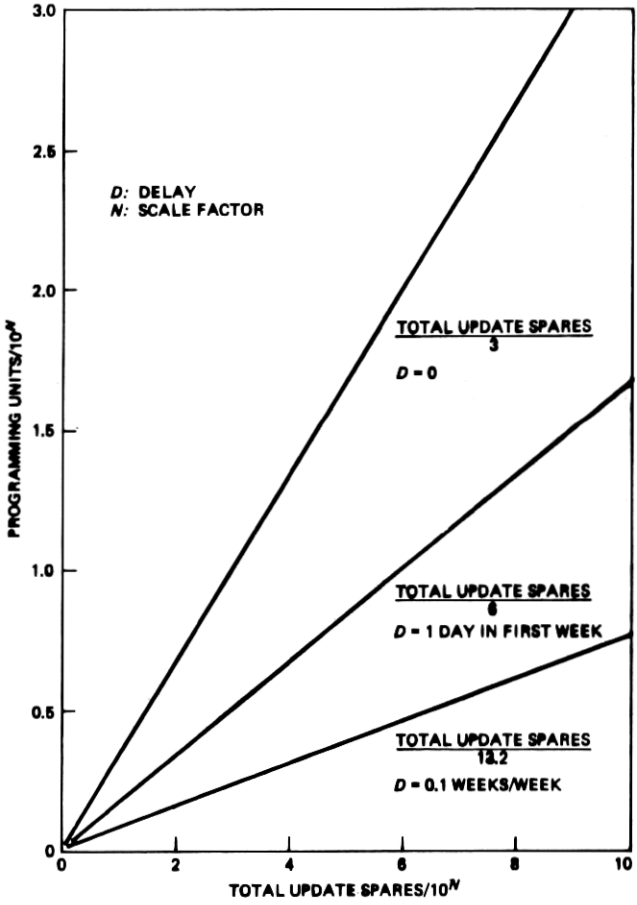

Fig. 3—Programming units.

For a projection of microprocessor systems, it is desired to determine the total update spares and programming unit requirements for the years 1982 and 1983, for an overall update time of 10 weeks in 1982 and 15 weeks in 1983, under the assumption of providing the fewest number of programming units. (See Tables III, IV, and V.)

IV. ECONOMIC IMPACT OF FIRMWARE UPDATE SCHEMES

4.1 Economic dependencies

Based on the obtained algorithms, an economic analysis of an update scheme can be performed if the capital and expense costs associated with the process are known.

The capital expenditures for the firmware update process are due to

Table III—Forecast of microprocessor systems

Year	Total Systems	Update Time Weeks
1982	26	10
1983	116	15

Table IV—Update spares from Fig. 2

Year	Nondelayed Update Time	Spare ratio	Total Systems	Total Update Spares
1982	9	0.59	26	15
1983	13	0.39	116	45

Table V—Programming units from Fig. 3 ($D = 0.1$)

Year	Programming Units
1982	2
1983	4

the total update spares and programming unit requirements. Therefore, the capital is a function of the desired update time, the total number of microprocessor systems, and the delay tolerated. If the capital budget is exceeded, the total update spares and programming units must be decreased to meet the dollar constraints, increasing the update time. The update time is independent of the number of microprocessor firmware changes; therefore, the capital is also independent of the change rate.

The update expenses are due to the labor costs associated with the programming unit operation, the installation efforts, and the costs of crating and shipping. Since these tasks are performed for each microprocessor system each time a program change occurs, the expenses are dependent on the system market and program change rate. The economic dependencies are shown in Fig. 4.

Using standard techniques, we can determine the economic feasibility of alternate memory schemes for microprocessor-based systems. One alternative, for example, could be to determine the benefits of replacing EPROM memory with random access memory (RAM) and

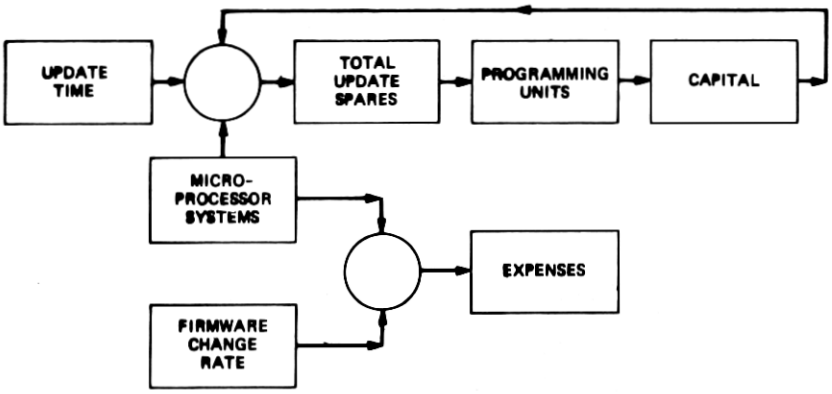

Fig. 4—Economic dependencies.

magnetic bubble store as backup. This possibility, even though initially more expensive, has the capability of being updated electrically via a data link, and may offer savings over the life cycle of the system.

V. SUMMARY

Algorithms related to the EPROM firmware update process for microprocessor systems have been developed in this paper. They give the update time, the number of update spares, and the numer of programming units required at a centralized site for a given firmware installation distribution. The algorithms presented do not render policy decisions, but enable the user to determine the economics and responsiveness of alternative administrative schemes.

REFERENCE

1. E. I. Jury, "Theory and Application of the *z*-Transform Method," New York: John Wiley, 1964.