マイクロコンピュータによる Fire 符号のシミュレーションプログラム†

——データ通信への応用——

岡野 博一††

ディジタル通信の高信頼度化のための主要な方法の一つである符号理論の研究は多大な成果を上げ，種々の誤り検出ならびに誤り訂正符号が実用されている．
また，マイクロコンピュータが飛躍的に発展し，データ通信の端末としても用いられている．
したがって，従来ハードによっていた符号理論の手法をマイクロコンピュータのソフトによって実現することができれば，経済的で柔軟なデータ通信システムが構成され得る．
本論文において，Fire 符号のシミュレーションプログラムとその良好な結果を示す．
シフト演算法を用いた場合，3バースト誤り訂正，$n_{max}=35$ の Fire 符号を用いたときの復号時間の上限は約 4.0 ms である．さらに 256 語の復号表（剰余と誤りパターンの対応表）を用いるテーブル・ルックアップ法によれば復号時間を約 1/2 とすることが可能である．
なお，プログラム作成に用いた言語はマイクロコンピュータ PANAFACOM L-16A のアセンブリ言語である．

1. まえがき

ディジタル通信の高信頼度化のための主要な方法の一つである符号理論の研究は多大な成果を上げ，種々の誤り検出ならびに誤り訂正符号が衛星通信回線，データ通信回線等に実用されている．符号理論の手法の実現にはハードによる方法とソフトによる方法が考えられる．従来はハードによる方法が主流を占めたが，最近のコンピュータ技術の進歩はソフトによる方法を実現可能にしている．つまり，計算速度の高速化，メモリの経済化およびマイクロコンピュータの発展は通信制御の形態に一つの変化をもたらしている．すなわち，ミニコンピュータを内蔵したインテリジェントターミナルは一定規模以上のシステムではその効果を遺憾なく発揮していたが，LSI マイクロコンピュータの出現はほとんどすべての端末のハードウェアゲートロジックの端末と同程度，あるいはそれ以下の価格でインテリジェントターミナルのメリットを享受できるようになった[9]．したがって，端末におけるハードウェア設計の仕事の大部分がプログラム作成作業という容易に見える作業におきかえられる[8]．

また，大型コンピュータにおいても，CPU と Disk，MT，メモリ間等のデータ転送の誤り制御をソフトによって行うことも可能である．

さて，プログラム言語としては，ビットハンドリングな論理演算機能のある言語であれば符号理論のソフト化が可能である．筆者はすでに，PL/1 および HPL によって Fire 符号演算をソフト化した[13),14)]．これらは大型機用の高級言語である．一方，ミニコンピュータおよびマイクロコンピュータにおいてはアセンブリ言語を用いる必要がある．

本論文において，マイクロコンピュータ PFL-16 Aのアセンブリ言語によって作成した Fire 符号のシミュレーションプログラムについて述べる．なお，シフト演算法およびテーブル・ルックアップ法の2つの方法を用いている．

2. Fire 符号[1)~4),6)]

Fire 符号は巡回符号の一種であり，バースト誤りに適用される．生成多項式 $P(x)$ は次式となる．

$$P(x)=P_1(x)\cdot(1+x^e) \qquad (1)$$

ここで，1) $P_1(x)$ は次数 m の既約多項式，2) c は e で割り切れない．但し e は $P_1(x)$ が x^e-1 を割り切るような最小の正整数，3) 最大符号長 n_{max} は c と e の最小公倍数であって，$x^{n_{max}}-1$ は $P(x)$ で割り切れるが x^n-1（但し，$n<n_{max}$）は割り切れない．

このとき，次の誤りを検出および訂正する．すなわち，$b_1+b_2-1\leq c$，$m\geq b_1(b_2\geq b_1)$ の条件のもとで，1) 長さ b_1，b_2 の二つのバースト誤りを検出する．また $(m+c)$ の長さの単一バースト誤りを検出する．

† Simulation Programs of Fire Codes by a Microcomputer ——Application to Data Communication—— by HIROKAZU OKANO (Department of Information Electronics, Tokuyama Technical College).

†† 徳山工業高等専門学校情報電子工学科

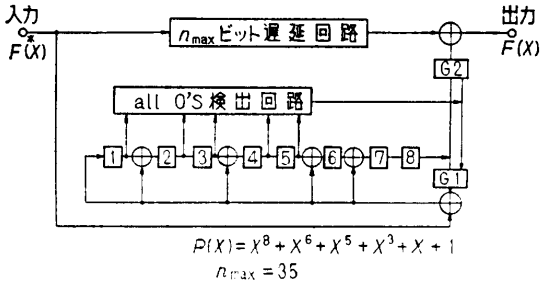

図 1 3バースト誤り訂正復号化回路

Fig. 1 An error-correction circuit for the Fire code with birst-correcting ability 3.

2) $b_1=b_2=b$ とすると b バースト以下の単一誤りを訂正できる.

まず,符号化の手順を説明する. 符号長を $n(<n_{max})$ とし, 入力符号長を k とし $G(x)$ で表わす. 生成多項式 $P(x)$ の次数を $n-k$ とする. 入力符号 $G(x)$ に x^{n-k} を乗じ, $P(x)$ で割った剰余を $R(x)$ とすると次式を得る. (演算は Modulo 2)

$$x^{n-k}\cdot G(x)=P(x)\cdot Q(x)+R(x) \qquad (2)$$

両辺に $R(x)$ を加え $F(x)$ とおく

$$F(x)=x^{n-k}\cdot G(x)+R(x)=P(x)\cdot Q(x) \qquad (3)$$

この $F(x)$ が送信符号, $R(x)$ がチェック符号である. 受信符号を $P(x)$ で割り, 剰余が0ならば誤りなし, 剰余が0でなければ誤りが発生していることになる.

本論文においては, 3バースト誤り訂正の場合について述べるので, まず, このときの生成多項式を求める. $b=m=3$, $c=2b-1=5$, $P_1(x)=x^3+x+1$ となるので, $P(x)$ は次式となる.

$$P(x)=(x^3+x+1)\cdot(1+x^5)$$
$$=x^8+x^6+x^5+x^3+x+1 \qquad (4)$$

次に, シフトレジスタを用いた復号化回路について説明する. **図1**に3バースト誤り訂正復号化回路を示す. シフトレジスタ回路は $P(x)$ に対応した Modulo 2 の割り算回路である. 受信符号を n_{max} ビット遅延回路に入力すると同時に, 割り算回路にも入力する. 誤りが3バースト以内の場合は割り算を続けていくと, all 0′s 検出回路が動作する. すなわち, 割り算回路の下位5ビットが all 0′s となり, 上位3ビットに誤り訂正可能パターンが生じる. この時, n_{max} ビット遅延回路の出力に割り算回路のパターンを加えれば誤りの訂正を行うことができる. (n_{max} は最大符号長であり, この場合は 35.)

復号化の詳細については参考文献 1)〜4), 6) を参照されたい.

3. シフト演算法によるシミュレーションプログラム

3.1 符 号 化

図1におけるシフトレジスタの演算動作は入力符号を $P(x)$ で割り算して行くことと等価であり, XOR (exclusive OR) の論理演算と同じである. この演算をシフト演算法とよぶこととする. マイクロコンピュータを使用する場合, 演算はレジスタ間において行われるので, 符号長がレジスタ長より長い場合には若干の工夫を要する. すなわち, 入力符号を順次レジスタ

信号 (ESR セット)	111100000000
XOR の結果	010001011
シフト	100010110
XOR の結果	001111101
シフト	011111010
シフト	111110100
XOR の結果	010011111
右へ3シフト	000010011111

図 2 符号化の演算例

Fig. 2 The encoding calculation of a Fire code.

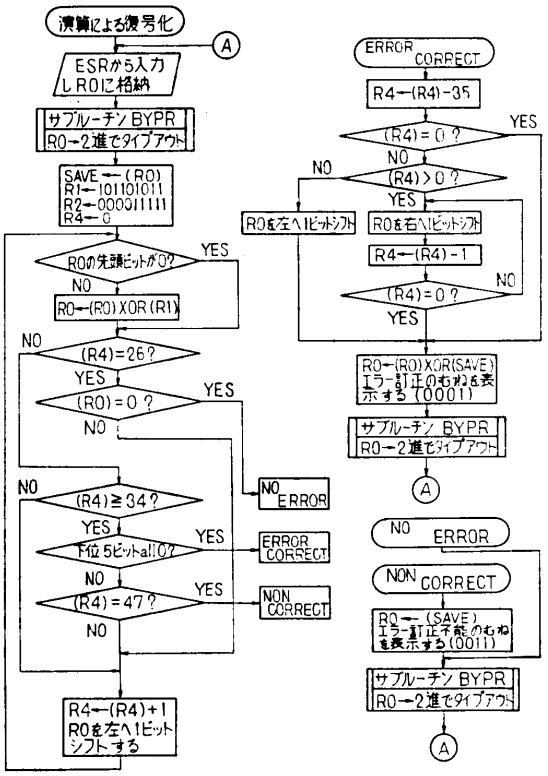

図 3 復号化フローチャート (シフト演算法)

Fig. 3 Flow chart of the decoding program using shift calculation method.

```
STNO.  LABEL  OP.    OPERANDS/COMMENT
  1           BGN    FUKUGOKA
  2           L      SP, SPAD
  3    A      H
  4           MVI    R0, X'0A'
  5           BAL    (TYP)
  6           BAL    (TYP)
  7           RD     R0, X'04'
  8           ST     R0, SAVE
  9           BAL    (BYPR)
 10           MVI    R0, X'0A'
 11           BAL    (TYP)
 12           L      R0, SAVE
 13           L      R1, JOSU
 14           L      R2, KAI6
 15           CLEAR  R4
 16    *
 17    J0     TBIT   R0, 0, Z
 18           EOR    R0, R1
 19           L      R3, N26
 20           S      R3, R4, Z
 21           B      J1
 22           SKIP   R0, NZ
 23           B      NOE
 24    J2     AI     R4, 1
 25           SL     R0, RE
 26           B      J0
 27    *
 28    J1     L      R3, N34
 29           S      R3, R4, MZ
 30           B      J2
 31           MV     R3, R0
 32           AND    R3, R2, NZ
 33           B      CORRE
 34           L      R3, N47
 35           S      R3, R4, Z
 36           B      J2
 37    * NON CORRECT
 38           L      R3, NCORE
 39           L      R0, SAVE
 40           EOR    R0, R3
 41           B      PRI
 42    * ERR CORRECT
 43    CORRE  L      R3, N35
 44           S      R4, R3
 45    B1     SKIP   R4, NZ
 46           B      E1
```

```
STNO.  LABEL  OP.    OPERANDS/COMMENT
 47           SKIP   R4, P
 48           B      SSL1
 49    SSR    SR     R0, RE
 50           SI     R4, 1
 51           SKIP   R4, NZ
 52           B      E1
 53           B      SSR
 54    SSL1   SL     R0, RE
 55    E1     L      R3, ECORE
 56           EOR    R0, R3
 57           L      R3, SAVE
 58           EOR    R0, R3
 59           B      PRI
 60    NOE    L      R0, SAVE
 61    PRI    BAL    (BYPR)
 62           B      A
 63           DS     XL24
 64    SPAD   DC     A(*-1)
 65    TYP    EXTRN  TYP
 66    BYPR   EXTRN  BYPR
 67    *
 68    SAVE   DS     XL2
 69    JOSU   DC     X'B580'
 70    KAI6   DC     X'0F80'
 71    N26    DC     F'26'
 72    N34    DC     F'34'
 73    N47    DC     F'47'
 74    N35    DC     F'35'
 75    NCORE  DC     X'0003'
 76    ZERO   DC     F'0'
 77    ECORE  DC     X'0001'
 78           END
```

(出力例)

```
1111 1001 1111 0000
1111 1001 1111 0000   } NO ERROR
1000 1110 0101 0000
1000 1110 0101 0000

1100 0001 1111 0000
1111 1001 1111 0001   } ERROR CORRECT
1011 1111 1000 0000
1011 0101 1000 0001

1000 0101 1111 0000
1000 0101 1111 0011   } NON CORRECT
1101 0101 1111 0000
1101 0101 1111 0011
```

図 4 復号化プログラムとその出力例（シフト演算法）

Fig. 4 The decoding program using shift calculation method and its output data.

表 1 種々の Fire 符号の性能

Table 1 Performance of some Fire codes.

訂正可能バースト長	3	5	7	9
生成多項式	$(x^3+x+1)(1+x^5)$	$(x^5+x^2+1)(1+x^9)$	$(x^7+x^3+1)(1+x^{13})$	$(x^9+x^4+1)(1+x^{17})$
チェックビット長	8	14	20	26
最大符号長	35	279	1651	8687
演算時間の上限	4.0 ms	31.6 ms	186.6 ms	981.7 ms

に格納して行く必要がある．本プログラムでは符号長はレジスタ長より短かいものとしている．

まず，$P(x)$ に対応する JOSU＝101101011 をレジスタ R1 に格納し固定する．そして，入力符号をエントリースイッチ ESR にセットし，それを R0 に移す．R1 と R0 の XOR 演算を行い，結果を R0 に格納する．次に R0 の内容を左にシフトして演算を繰返す．**図2**に符号化の演算例を示す．入力符号(1111)が JOSU で割り算されて，チェックビット (10011111) を生じ，送信符号(111110011111)が求められている．なお，プログラムリスト等は省略する．

3.2 復 号 化

図3に復号化のフローチャートを示す．まず，受信符号を ESR にセットし，R0 に移す．符号化の場合と同じ JOSU を R1 に格納する．R2 には XOR の演算結果の下位5ビットを検出するためのマスク 000011111 を格納する．そして R0 と R1 の XOR の演算結果を R0 に格納し，R0 の内容を左シフトしながら演算していく．誤りがなければ，符号長 n＝35，$P(x)$ の長さが9ビットであるから，26回目の演算で余りが0となる．誤りが訂正可能パターンの場合はさらに演算していくと，XOR の結果の下位5ビットが，all 0′s となる．このとき上位3ビットが誤りパターンである．この誤りパターンは入力符号に対して周期 n_{max}＝35 ビット遅れているから，演算回数より 35 を減じた値を K とすると，K＝0 ならば R0 に発生した誤りパターンはそのままとする．K が負ならば最初のビットに誤りが生じているから，R0 の内容を1ビット左シフトする．もし，K が正ならば，K ビットだけ R0 の内容を右シフトする．以上で R0 の誤りパターンが入力符号に対して正しい位置に移動したので，R0 の内容と SAVE に格納しておいた入力符号との XOR により訂正が行われる．

もし，演算を続けても下位5ビットが all 0′s とならない場合は誤り訂正ができないと判断し，47 回で演算を打ち切る．**図4**にマイクロコンピュータのアセンブリ言語による復号化プログラムリストとその出力例を示す．

表1に種々の Fire 符号の性能を示す．演算時間の上限は入力符号長よりレジスタ長が長いものとして算出している．3バースト誤り訂正，n_{max}＝35 の場合，復号時間の上限は約 4.0 ms である．

4. テーブル・ルックアップ法によるシミュレーションプログラム

誤りの復号化は高速に行うことが望ましい．シフト演算法の場合は n_{max}～$2n_{max}$ 回の演算が必要である．しかし，テーブル・ルックアップ法によれば非常に高速に復号化することができる．すなわち，誤りパターンとその剰余（シンドローム）との対応表を作成しておき，入力符号を $P(x)$ で割り剰余を求め，対応する誤りパターンをテーブル・ルックアップして誤りの訂正を行う．剰余を求めてから誤りの訂正を行うために，数ステップを要するのみである．

復号表は前項と同じシフト演算法によってマイクロコンピュータによって計算しタイプアウトする．さらに紙テープに作成しておく．復号表は 16 進で表示している．復号表作成のためのプログラムの説明は省略する．**図5**に復号化フローチャートを示す．前半は復号表をメモリに格納するアルゴリズム，後半は符号の誤り処理を行うアルゴリズムである．符号表は先頭番地を (AA) とすると，(AA)+255 番地まで格納され

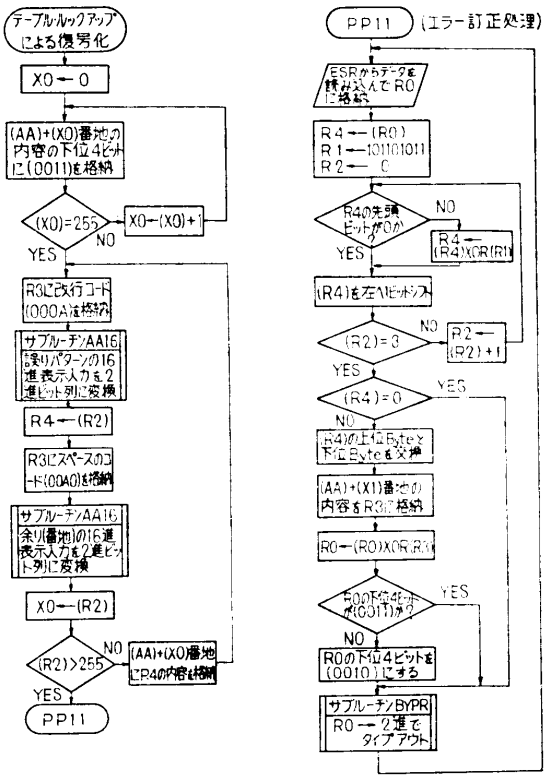

図5 復号化フローチャート（テーブル・ルックアップ法）

Fig. 5 Flow chart of the decoding program using table look-up method.

```
STNO.  LABEL   OP.     OPERANDS/COMMENT
  1            BGN     CL 11,, X' 0'
  2            L       SP, SPAD
  3            CLEAR   X 0
  4            MVI     R 0, X' 0003'
  5    C       ST      R 0, (AA)(X 0)
  6            L       R 1, BB
  7            S       R 1, X 0, P
  8            B       KAKU
  9            AI      X 0, 1
 10            B       C
 11    KAKU    MVI     R 3, X' 000 A'
 12            BAL     (AA 16)
 13            MV      R 4, R 2
 14            MVI     R 3, X' 00A0'
 15            BAL     (AA 16)
 16            MV      X 0, R 2
 17            L       R 1, BB
 18            S       R 2, R 1, MZ
 19            B       PP 11
 20            ST      R 4, (AA)(X 0)
 21            B       KAKU
 22    *
 23    PP 11   H
 24            MVI     R 0, X' 0 A'
 25            BAL     (TYP)
 26            BAL     (TYP)
 27            RD      R 0, X' 04'
 28            MV      R 4, R 0
 29            BAL     (BYPR)
 30            MVI     R 0, X' 0 A'
 31            BAL     (TYP)
 32            MV      R 0, R 4
 33            L       R 1, JOSU
 34            CLEAR   R 2
 35    *
 36    J 0     TBIT    R 4, 0, Z
 37            EOR     R 4, R 1
 38            SL      R 4, RE
 39            MV      R 3, R 2
 40            SI      R 3, 3, NZ
 41            B       J 1
 42            AI      R 2, 1
 43            B       J 0
 44    J 1     SKIP    R 4, NZ
 45            B       NOE
 46            BSWP    R 4, R 4
 47            L       R 3, (AA) (X 1)
 48            EOR     R 0, R 3
 49            TBIT    R 0, 15, NZ
 50            SBIT    R 0, 14
 51    NOE     BAL     (BYPR)
 52            B       PP 11
 53    *
 54    JOSU    DC      X' B580'
 55            DS      XL 20
 56    SPAD    DC      A(*−1)
 57    BB      DC      F' 255'
 58    AA 16   EXTRN   AA 16
 59    TYP     EXTRN   TYP
 60    BYPR    EXTRN   BYPR
 61    AA      DC      A(*+1)
 62            DS      2 XL 256
 63            END
```

```
STNO.  LABEL   OP.     OPERANDS/COMMENT
  1            BGN     AA 16
  2    RDD     BAL     (PTR 16)
  3            CB      R 0, R 3, Z
  4            B       RDD
  5    *
  6            BAL     (PTR 16)
  7            BSWP    R 0, R 0
  8            MV      R 2, R 0
  9            BAL     (PTR 16)
 10            MVB     R 2, R 0
 11            BAL     (PTR 16)
 12            BSWP    R 0, R 0
 13            MV      R 1, R 0
 14            BAL     (PTR 16)
 15            MVB     R 1, R 0
 16            DSWP    R 2, R 2
 17            BSWP    R 2, R 2
 18            DSWP    R 1, R 1
 19            MVB     R 2, R 1
 20            RET
 21    *
 22    PTR 16  EXTRN   PTR 16
 23            END
```

```
STNO.  LABEL   OP.     OPERANDS/COMMENT
  1            BGN     PTR 16
  2            MVI     R 0, X' 21'
  3            WT      R 0, X' 14'
  4    P 1     RD      R 0, X' 15'
  5            TBIT    R 0, 8, Z
  6            TBIT    R 0, 9, NZ
  7            B       P 1
  8            WT      R 0, X' 15'
  9            RD      R 0, X' 16'
 10            TBIT    R 0, 9, Z
 11            AI      R 0, 9
 12            RET
 13            END
```

(出力例)

```
1111 1001 1111 0000
1111 1001 1111 0000   } NO ERROR
1000 1110 0101 0000
1000 1110 0101 0000

1100 0001 1111 0000
1111 1001 1111 0010   } ERROR CORRECT
1011 1111 1000 0000
1011 0101 1000 0010

1000 0101 1111 0000
1000 0101 1111 0011   } NON CORRECT
1101 0101 1111 0000
1101 0101 1111 0011
```

図 6 復号化プログラムとその出力例 (テーブル・ルックアップ法)

Fig. 6 The decoding program using table look-up method and its output data.

る．剰余に対応する番地にその誤りパターンが格納される．それ以外の番地には，誤り訂正不能の表示 0011 が格納される．

さて，次に誤り訂正処理に移る．受信符号を ESR にセットし，シフト演算により剰余（シンドローム）を求めレジスタ R 4 に格納する．剰余が 0 であれば誤りは生じていない．R 4 が 0 でないならば，剰余に対応する番地に格納されている誤り訂正可能パターンをテーブル・ルックアップする．すなわち，(AA)+(X1) 番地（但し，R 4=X 1，テーブルの先頭番地に剰余の値を加算した番地）の内容をレジスタ R 3 に格納する．この操作により，受信符号に含まれている誤りパターンが R 3 に格納される．したがって，別に R 0 に格納していた受信符号と R 3 の誤りパターンとを XOR により加えることによって，誤りの訂正を行うことができる．**図 6** にテーブル・ルックアップ法による Fire 符号のシミュレーションプログラムのアセンブリプログラムリストとその出力例を示す．なお，ERROR CORRECT 出力の下位 4 ビットの (0010) によって訂正が行われたことを表示する．図 4 と異なるが，これはプログラム作成時統一しなかったためである．

この方法によれば，剰余を求めてから誤りの訂正まで数ステップを要するのみであるから復号時間は剰余を求める時間でほぼ決まり，シフト演算法と比較して復号時間を約 1/2 とすることができる．但し，3 バースト訂正の場合に 256 語の復号表が必要である．一般にチェックビットの長さを k とすると，2^k 語の復号表が必要となる．

使用したマイクロコンピュータ PFL-16 A について考察すると，最大メモリ容量 64 k 語なので，5 バースト誤り訂正，チェックビット長 14，復号表 16 k 語までが可能であり，シフト演算法よりテーブル・ルックアップ法が有効である．

さらに，エラーパターンのみを復号表に記憶する等の圧縮が可能となればテーブル・ルックアップ法の有効範囲が広がると思われる．

5. むすび

本稿によりマイクロコンピュータは符号処理を行う機能を充分持っていることが示された．

4800 ボーのデータ通信回線を例にとると，符号長 n_{max} とした時の符号伝送時間約 208 μs$\times n_{max}$ に比して，シフト演算法による復号時間は約 113 μs$\times n_{max}$ となる．さらにテーブル・ルックアップ法によれば復号時間はシフト演算法の約 1/2 となる．

今後は，プログラムの効率化，高速化，テーブルの圧縮等の検討を行う必要がある．さらに BCH 符号等のソフト化を進めたい．

なお，符号理論の適用分野としては，データ通信の他に，衛星通信，短波無線通信回線，ポケットベル等の多くの応用分野がある．

最後に，日頃御指導頂く，広島大学工学部川野董教授，協力頂いた本学学生，池田博君，山田良雄君の諸氏に深甚なる謝意を表する．

参考文献

1) Peterson, W. W. and Weldon, E. J.: Error-Correcting Codes, 2 nd Edition, MIT press, Mass (1972).
2) Berlekamp, E. R.: Algebraic Coding Theory, McGraw-Hill Book Co., New York (1968).
3) 宮川，岩垂，今井：符号理論，昭晃堂 (1973).
4) 嵩，都倉，岩垂，稲垣：符号理論，コロナ社 (1975).
5) 猪瀬，山本：データ通信，産報 (1971).
6) 星子訳：データ通信の原理，ラテイス刊 (1975).
7) 野口他：CRC 演算の実現方法について，信学全大 (1977).
8) 元岡達：マイクロコンピュータの展望，情報処理，Vol. 17, No. 4 (1976).
9) 桜井，田沼：マイクロコンピュータの端末装置への応用，同上.
10) 桑原，高村，佐藤，岡野：誤り訂正符号を用いた選択呼出方式，信学全大 (1967).
11) 桑原，西野，岡野，米岡：Fire Code を用いた無線印刷電信方式，信学会通信方式研資，CS 72-104 (1972).
12) 桑原，西野，岡野，米岡：Fire Code を用いた無線印刷電信方式，信学論 (B), 56-B, 7 (1973).
13) 岡野：ソフトプログラム (HPL) による Fire 符号の演算とその解析，電子通信学会技術研究報告，AL 76-33 (1976-07).
14) 岡野：PL/1 による Fire 符号のシミュレーションプログラム，情報処理，Vol. 18, No. 12 (1977).
15) 岡野：マイクロコンピュータによる Fire 符号演算，電気四学会中国支部大会，No. 22309 (1978).
16) 岡野：マイクロコンピュータによる Fire 符号のシミュレーションプログラム—データ通信への応用—，電子通信学会技術研究報告，CS 78-145, No. 8 (1978-11).
17) PANAFACOM L-16 A ソフトウェア文法書，パナファコム.

(昭和 53 年 12 月 4 日受付)
(昭和 54 年 6 月 21 日採録)