984r

PSEUDO-RANDOM NUMBER GENERATOR

by

CLEMENT C.Y. LAM, B.ENG. (McMaster)

PART B: ON-CAMPUS PROJECT*

A project report submitted in partial fulfillment of the requirement for the degree of Master of Engineering

Dept. of Engineering Physics

McMaster University

Hamilton, Ontario

September, 1977

*One of the two project reports: The other part is designated PART A:McMASTER (Off-Campus) PROJECT.

MASTER OF ENGINEERING (1977) MCMASTER UNIVERSITY

Department of Engineering Physics Hamilton, Ontario

TITLE: PSEUDO-RANDOM NUMBER GENERATOR

AUTHOR: Clement C.Y. Lam, B. Eng. (McMaster)

SUPERVISOR: Dr. T.J. Kennett

NUMBER OF PAGES: vi, 45

ABSTRACT

A simple and inexpensive pseudo-random number generator has been designed and built using linear feed-back shift registers to generate rectangular and gaussian distributed numbers. The device has been interfaced to a Nova computer to provide a high speed source of random numbers.

The two distributions have been checked with the following tests: (i) Frequency test (ii) Autocorrelation test and (iii) d^2-test. Results of each test have been compared with the expected theoretical values. Finally, a comparison of the generating speed has been made between this new generator and the existing old software generators.

This 28-bit generator is especially desirable in random simulation and Monte Carlo application if randomness, speed and cost are the main consideration in the design.

ACKNOWLEDGEMENT

I wish to thank my supervisor, Dr. T.J. Kennett, for his valuable guidance and suggestions throughout the course of this project. I also wish to thank Mr. K. Chin for the helpful discussions I had with him on many occasions.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Purpose of the project

In a wide variety of simulation and modelling studies where digital computers are used, a need usually arises for random number sequences. It is for such applications that most pseudo-random number(PRNG) are designed and built.

The PRNG's existing in the laboratory are either too slow or they are not as random as the user wants them to be. The need for a new generator which is both fast and has a longer non-repeating sequence has led to the design and implementation of the device reported in this project, namely, a PRNG which could give rectangular distributed or gaussian distributed numbers.

The objective of the project was to build such a PRNG using hardware and interface it to the Nova computer.

1.2 Theory(1,2,3)

The term "random" when applied to numbers or sequences relates to their origion rather than the numbers or sequences themselves.(1) By this definition, one could never be able to construct any sequence which is absolutely random. However, a pseudo-random number sequence can be constructed quite easily using a PRNG. The sequence so produced will be regarded as random if it can pass certain tests on the properties of randomness.

It has been proved that a n-stage shift register will cycle through all its possible states if a suitable feedback element is used.(2)(Fig. 1.1)

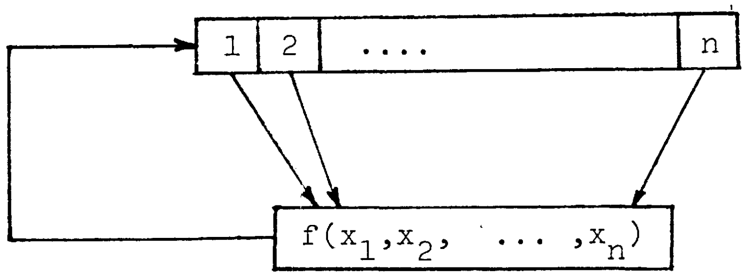


Fig. 1.1 General feedback shift register(FSR)

A special subclass which employs modulo-2 additions of the register stages is very important because such function can be shown to be linear.(1) In general, to get a pseudo-random number of m-binary digits involves taking m elements out from the n-stage FSR and have them packed together. It may be interpreted as a number when suitable weighting is assigned to the various digit position. In this way, a number N which lies in the range $0<N\leqslant 2^m-1$ may be created. Zero is a forbidden state because it will make the sequence stationary.

The numbers coming out from the FSR will be uniformly distributed in the range of $(1,2^m-1)$. Every number (or possible state) in this range has equal probability of occurance.

Repeated convolutions of a uniform distribution will give rise to an approximation to a gaussian distribution.(3) Convolution of two distributions corresponds to finding the resultant probability density function of the two independent random variables being added together. Using convolution theory, the gaussian distribution in this project is generated by adding twelve consecutive rectangular numbers together.

1.3 Implementation

All the required logic to build the PRNG is done using hardware. The device is interfaced to the Nova to facilitate software control. An overall view of the system is shown in Fig. 1.2.

The user can select distribution and set a starting number(seed) in the FSR if desired. This option allows one to reproduce a random number sequence. Then, the 28-stage FSR will shift 28 times to the right before it settles to give a 24 bit number selected from a rectangular distribution.(extracted from stages 0 through 23) At this moment, the computer will fetch the number into the specified

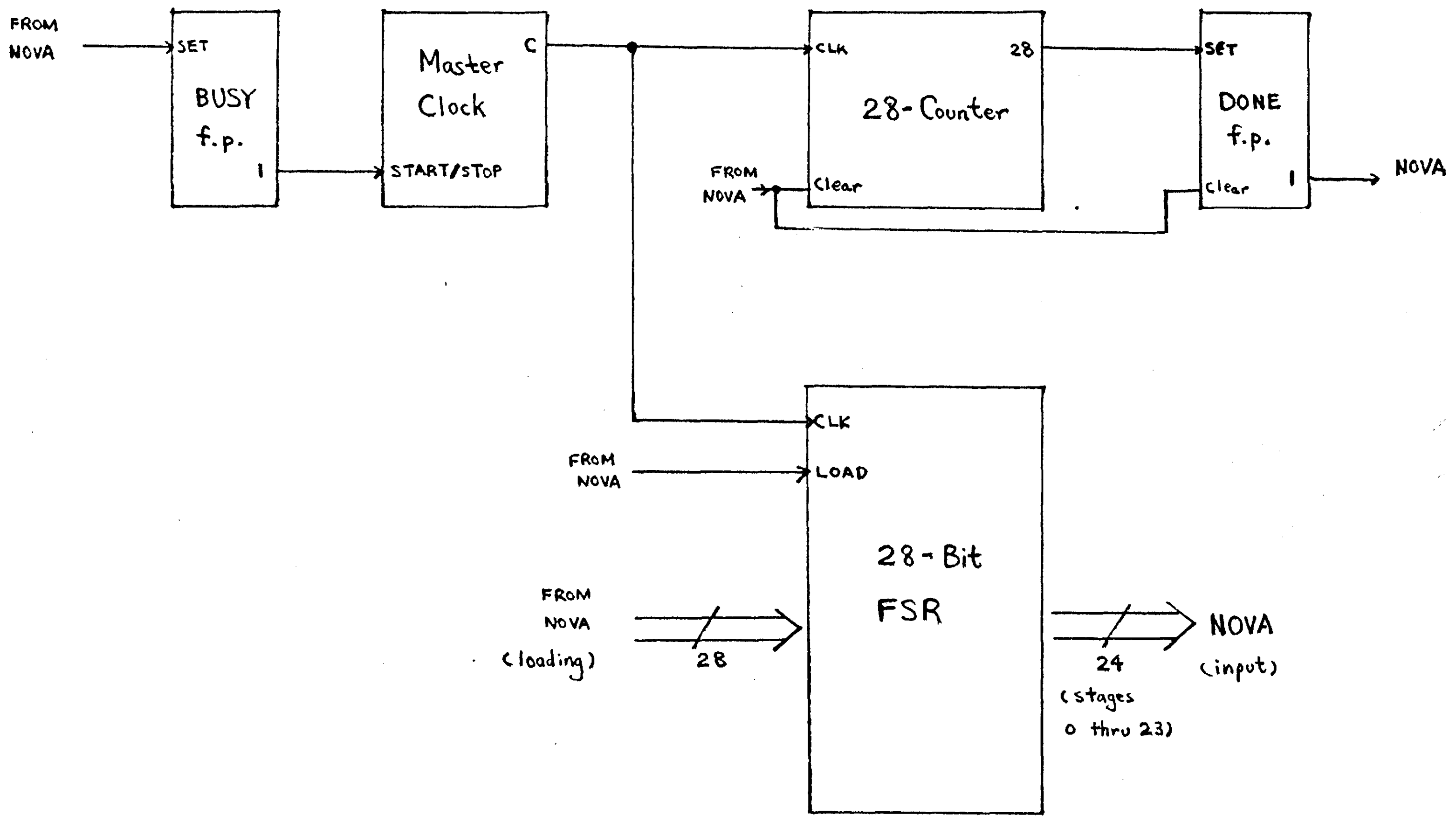


FIG. 1.2(a) Rectangular Distribution

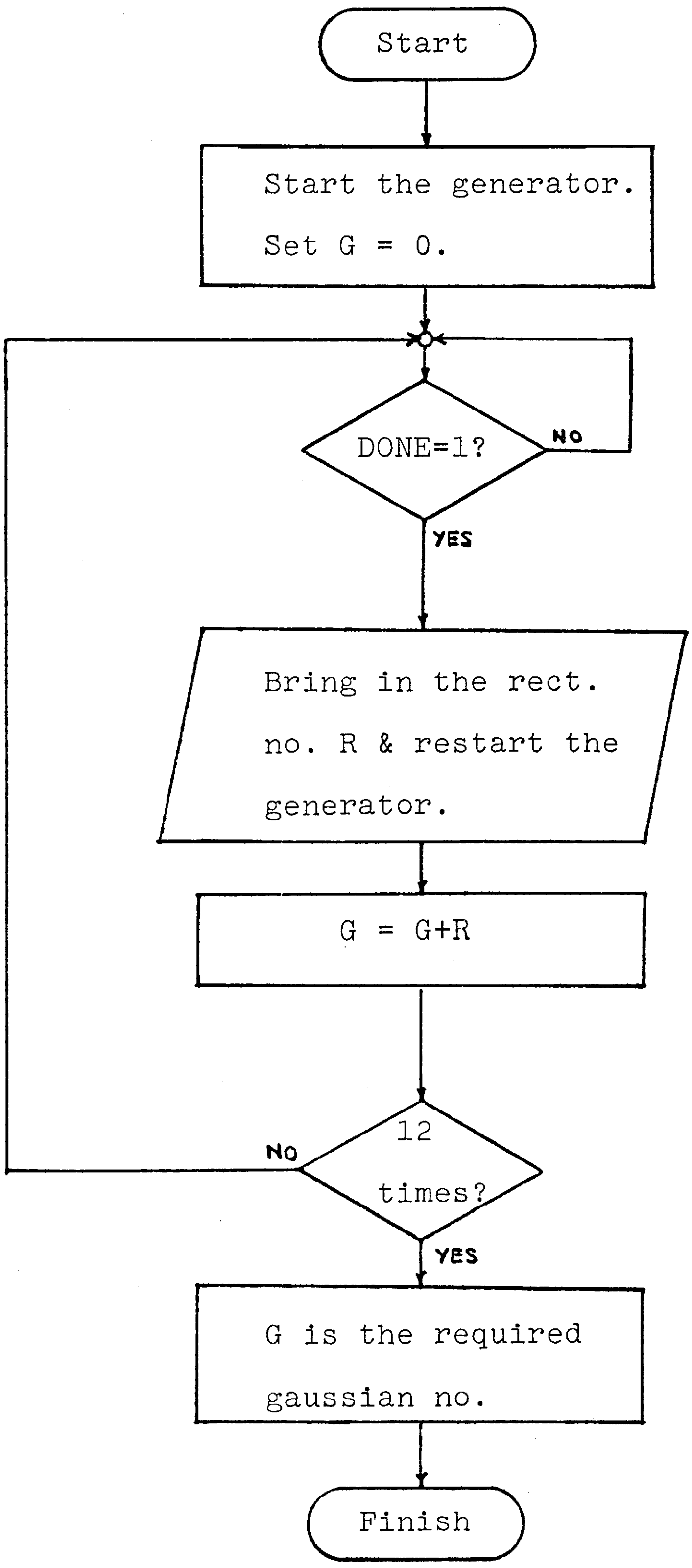

Start
Start the generator.
Set G = 0.
DONE=1?
NO
YES
Bring in the rect. no. R & restart the generator.
G = G+R
12 times?
NO
YES
G is the required gaussian no.
Finish

Fig. 1.2(b) <u>Gaussian Distribution</u>

accumulators.

If a gaussian number is selected, twelve constcutive rectangular numbers from the FSR will be added together to give a sample from a gaussian distribution. The addition is done by software in the Nova. Only 20 bits from the 24-bit rectangular sample(stage 4 through 23) are used in each addition.

Rate of clock pulse used is about 2.5 Mhz. It would mean a minimum of 11.2 microseconds is needed to generate a sample from a rectangular distribution.

The reason to shift the FSR 28 times is to ensure that maximum length can be obtained while inter-correlation between consecutive numbers is minimized.

CHAPTER 2

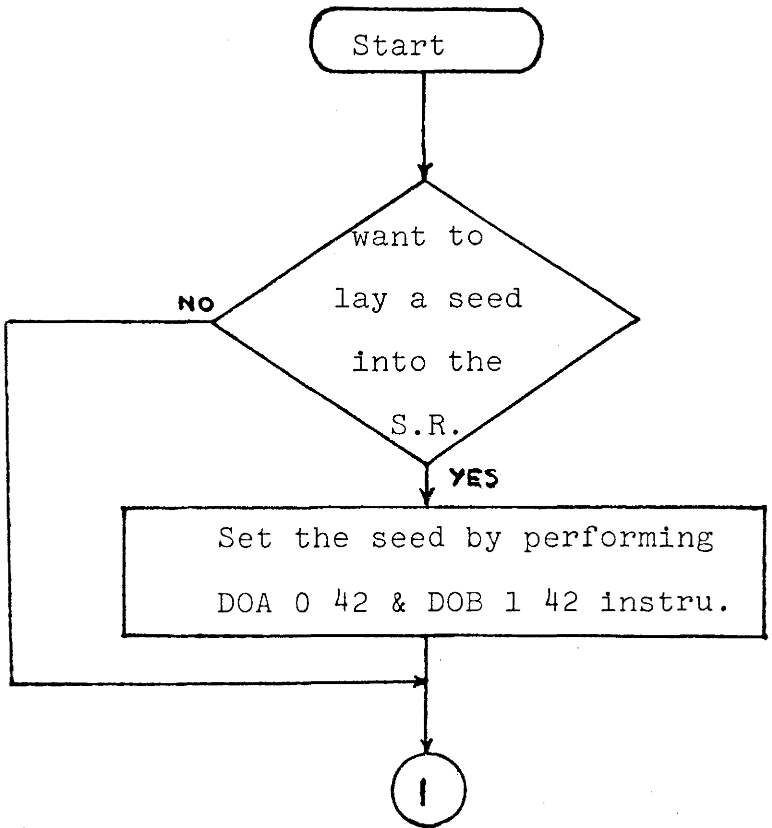
HARDWARE DESIGN

The purpose of the hardware is to generate uniformly distributed numbers. Using this source, gaussian distributed numbers can be created through addition.

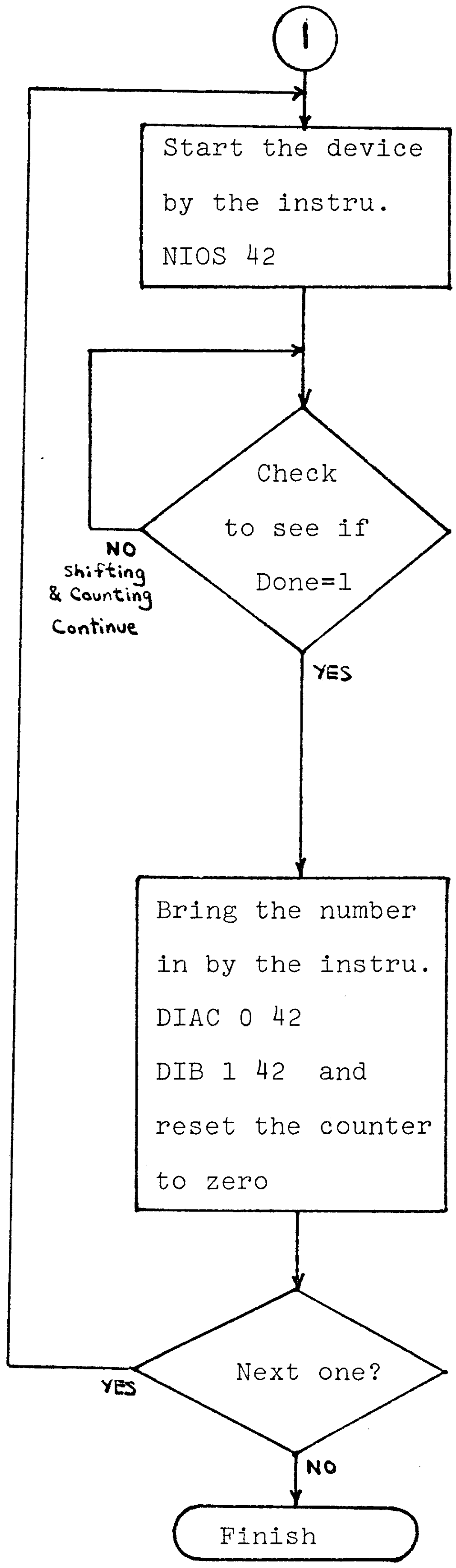
2.1 Rectangular distribution

Detailed hardware diagrams are shown in Fig. 2.1 to Fig. 2.5.

The following is a flowchart showing the procedure to get a rectangular distributed number.


Start
want to lay a seed into the S.R.
NO
YES
Set the seed by performing DOA 0 42 & DOB 1 42 instru.
1

The generator has been given a device code of 42.


1
Start the device by the instru. NIOS 42
Check to see if Done=1
NO
Shifting & Counting Continue
YES
Bring the number in by the instru. DIAC 0 42 DIB 1 42 and reset the counter to zero
Next one?
YES
NO
Finish

The S pulse will set Busy to 1(Fig. 2.1). It also initiates the Master Clock(Fig. 2.2) which will start shifting the FSR(Fig. 2.3) and incrementing the 28-counter(Fig. 2.2).

Shift the FSR to the right 28 times. The Done flag will be set if the counter has counted up to 28. By testing the Done flag with an SKPDN 42 instru., one will know whether there is a rectangular no. ready to be input.

The 24-bit rectangular number will be inside AC1 and AC0(low order). (Fig. 2.5) The C pulse will clear the counter , Done and Busy flag.

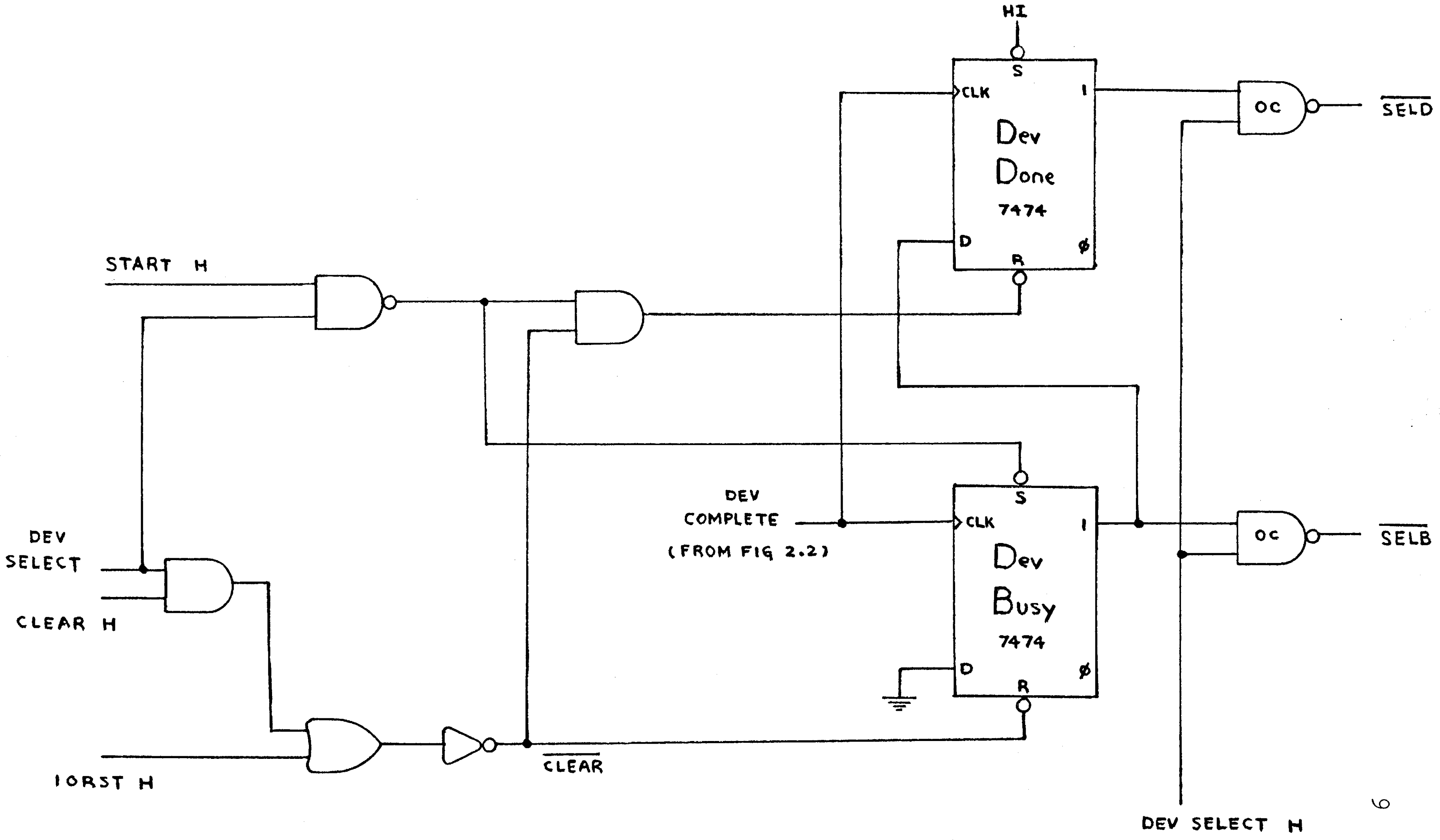


FIG. 2.1 Interfacing Network

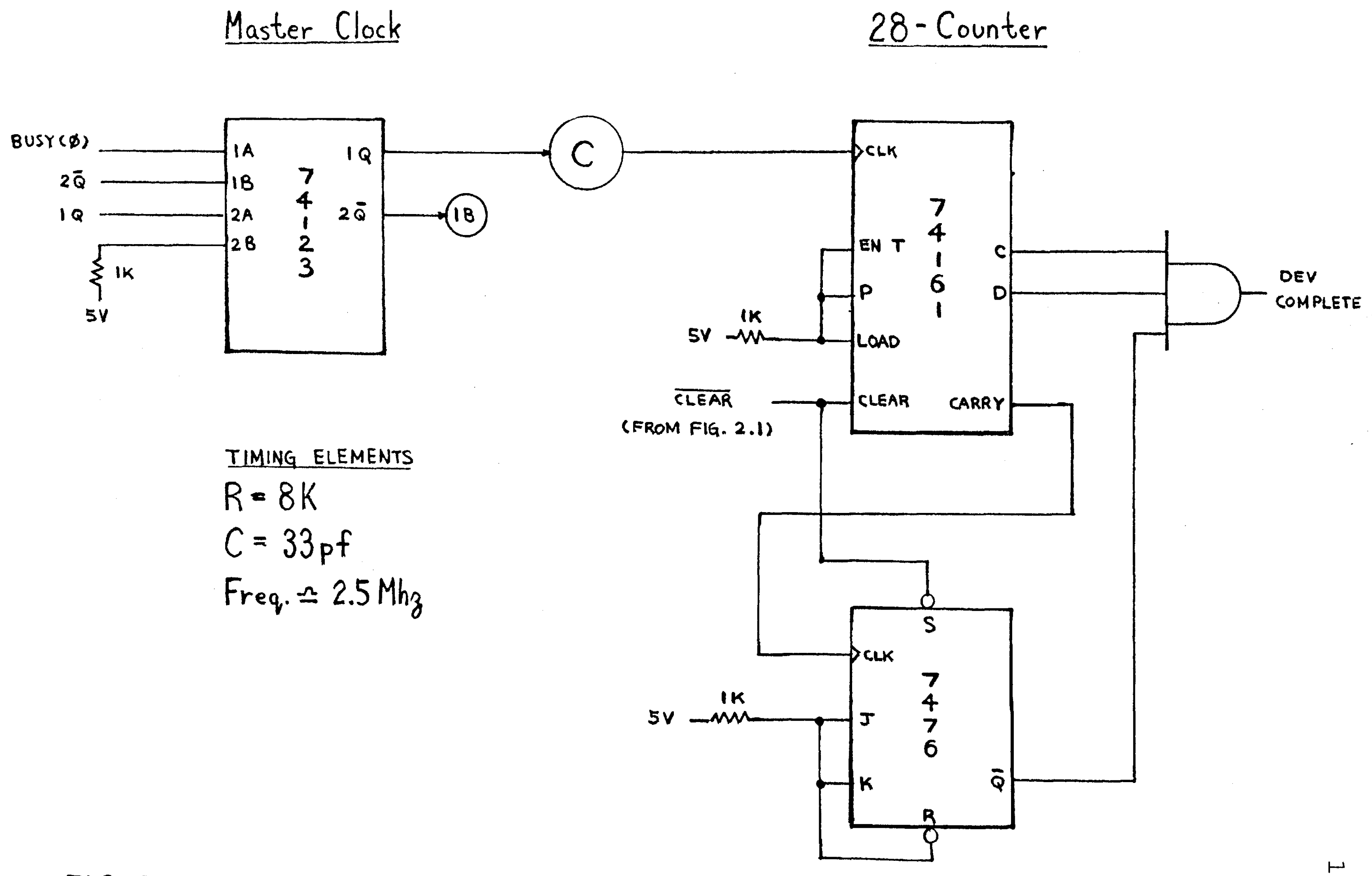


FIG. 2.2 Timing Device

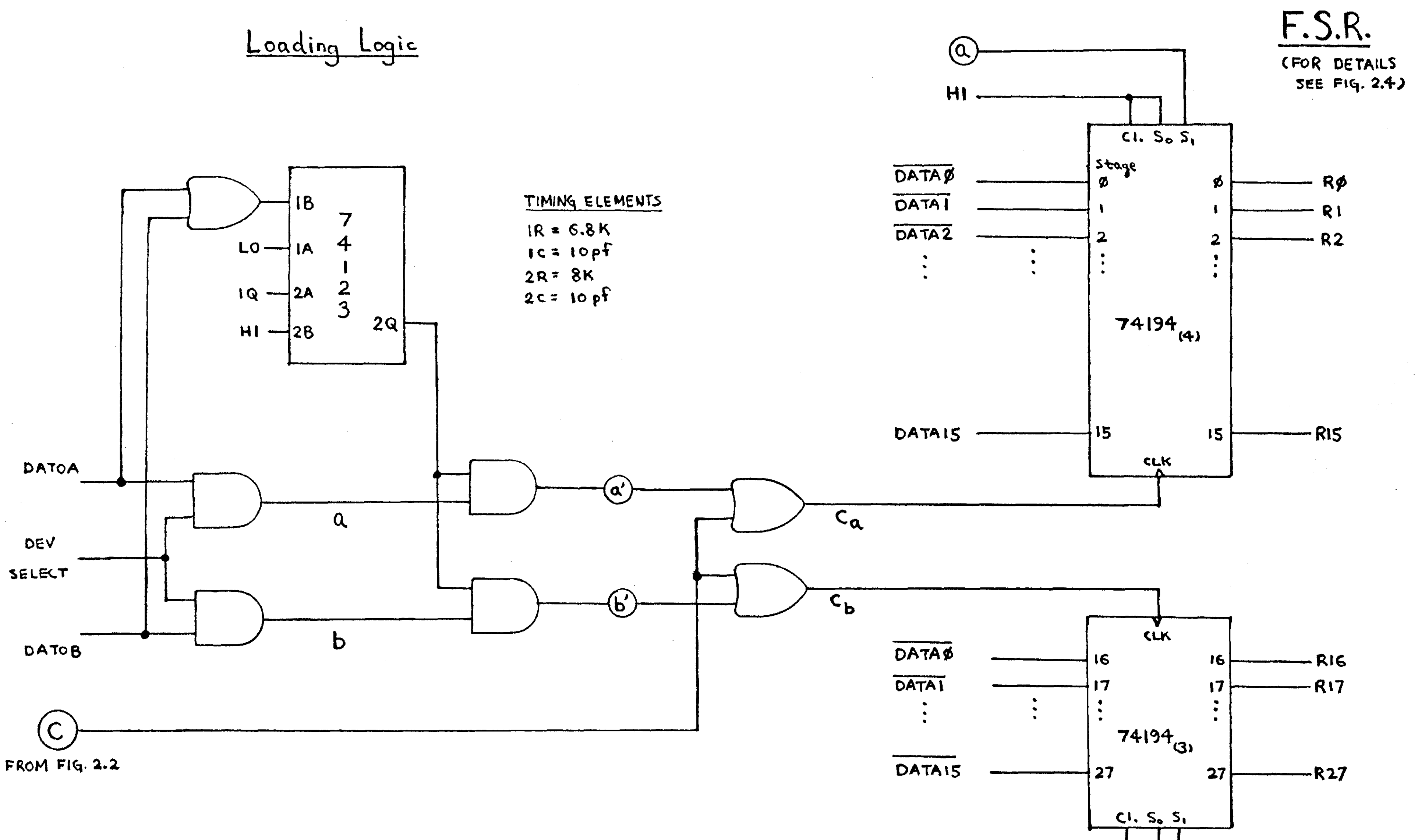


FIG. 2.3 28-Bit F.S.R.

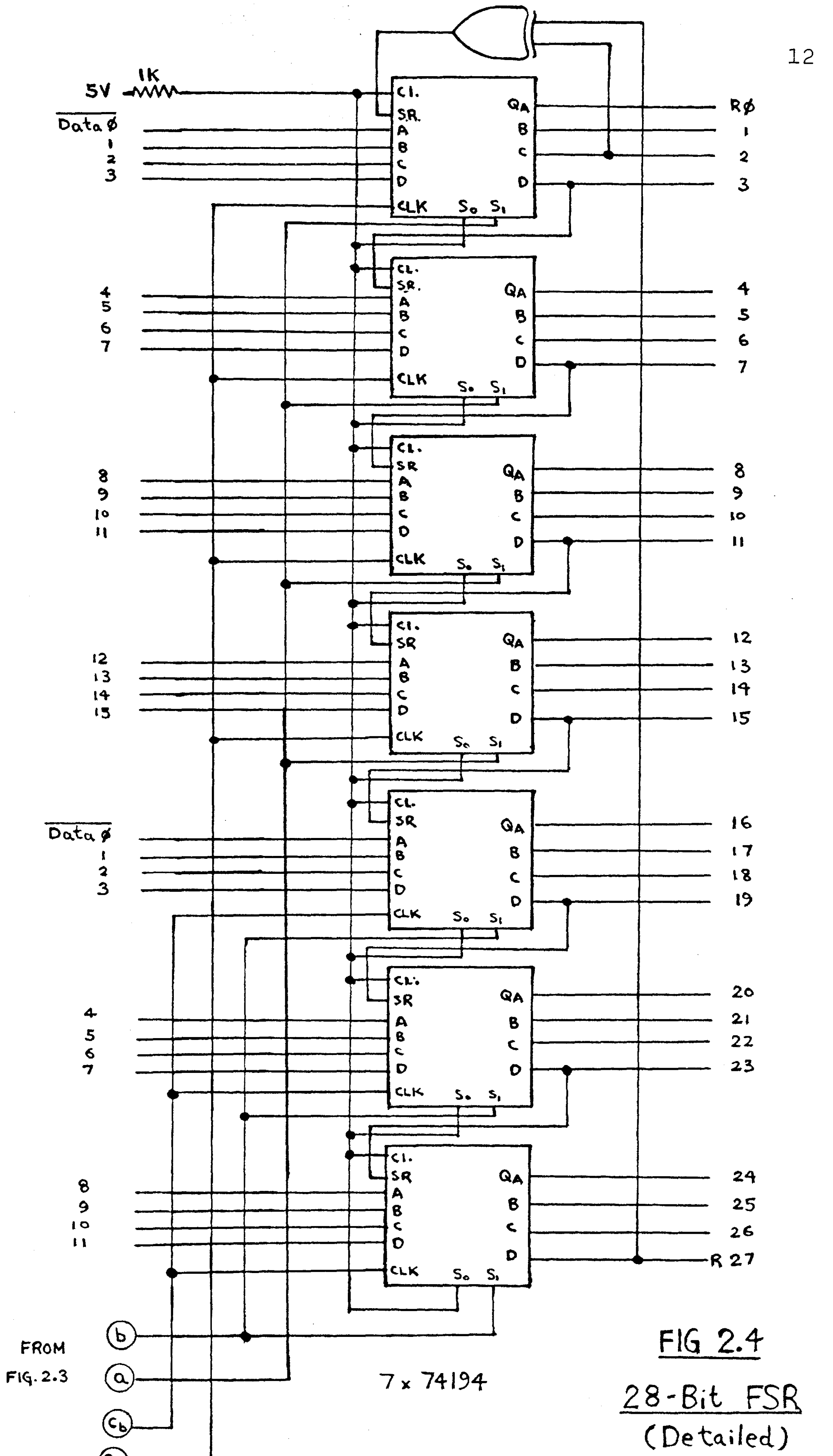


FIG 2.4

28-Bit FSR

(Detailed)

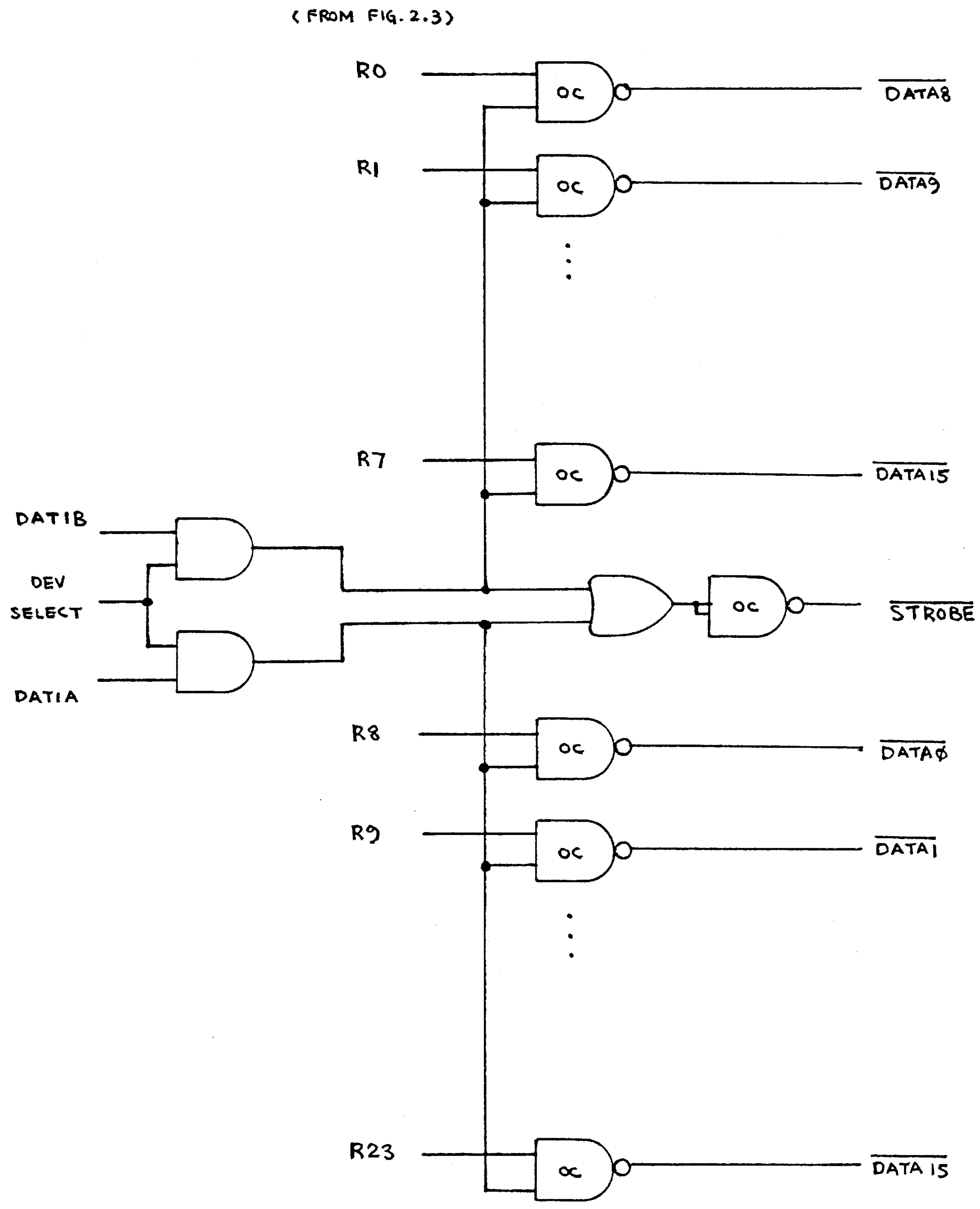

(FROM FIG. 2.3)
R0
R1
R7
R8
R9
R23
OC
DATA8
DATA9
DATA15
DATA∅
DATA1
DATA 15
DATIB
DEV SELECT
DATIA
STROBE

FIG 2.5 Input

There are three stages in getting a rectangular number:

1. Laying of a seed (Output from computer)
2. Generation by shifting the FSR 28 times to the right.
3. Input (Input to computer)

In each step the command is coming from the Nova.

<u>Laying of a seed</u>

The random sequence can start at a certain state by setting a number into the FSR at the beginning. The number should be inside AC1 and AC0(lower order) before the following instructions are performed:

```
DOA 0 42
DOB 1 42
```

These two instructions will load

bit 0 through 15 of AC0 into stage 0 through 15 of FSR(Fig. 2.3). and bit 0 through 11 of AC1 into stage 16 through 27 of FSR.

Null state is avoided by software described in chapter 3. Fig. 2.6 shows a timing diagram of the signals during loading.

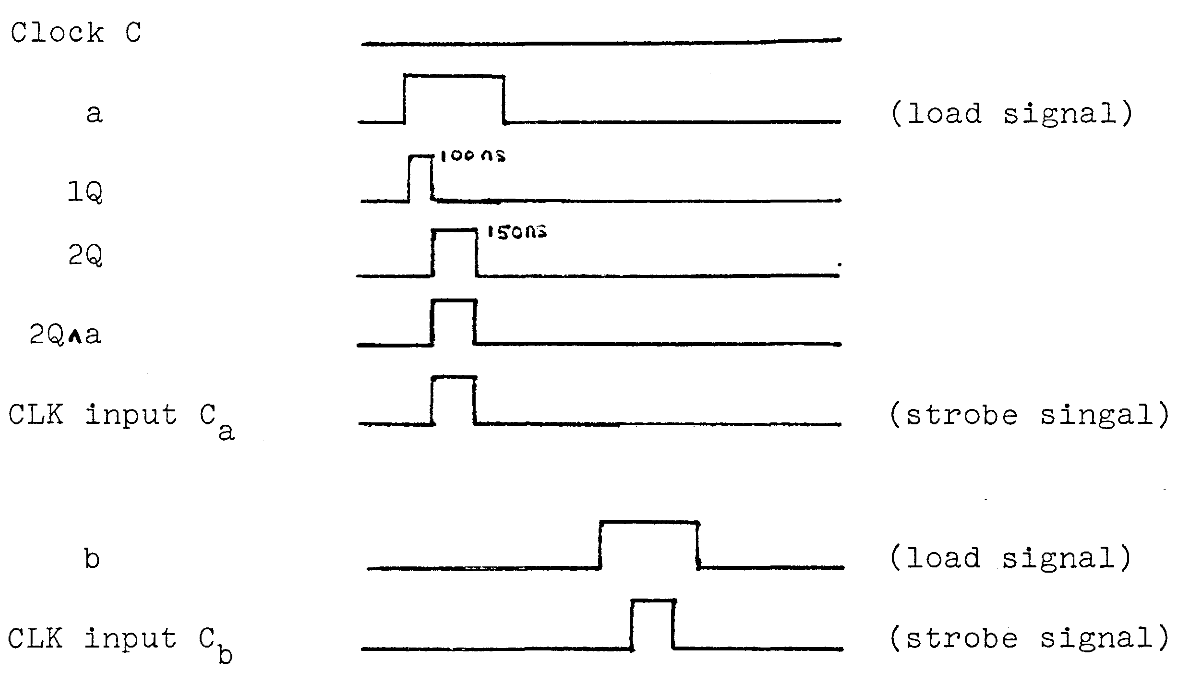


Fig. 2.6 Timing signals when a seed is laid
(Symbols referred to Fig. 2.3)

Generation of the number

After a seed is provided, the device will be started by the instruction NIOS 42. The S pulse will set Busy to 1 and initiate the Master Clock. Clock C will start shifting the FSR while at the same time incrementing the 28-counter. When the counter reaches 28, the Done flag will be set. The Master Clock will be stopped because Busy is clear at this moment. The present state of stages 0 through 23 of the FSR will be taken as the desired rectangular number in binary. If another number is required, the whole generating procedure may be repeated. Fig. 2.7 gives a timing picture of the generation of a rectangular number.

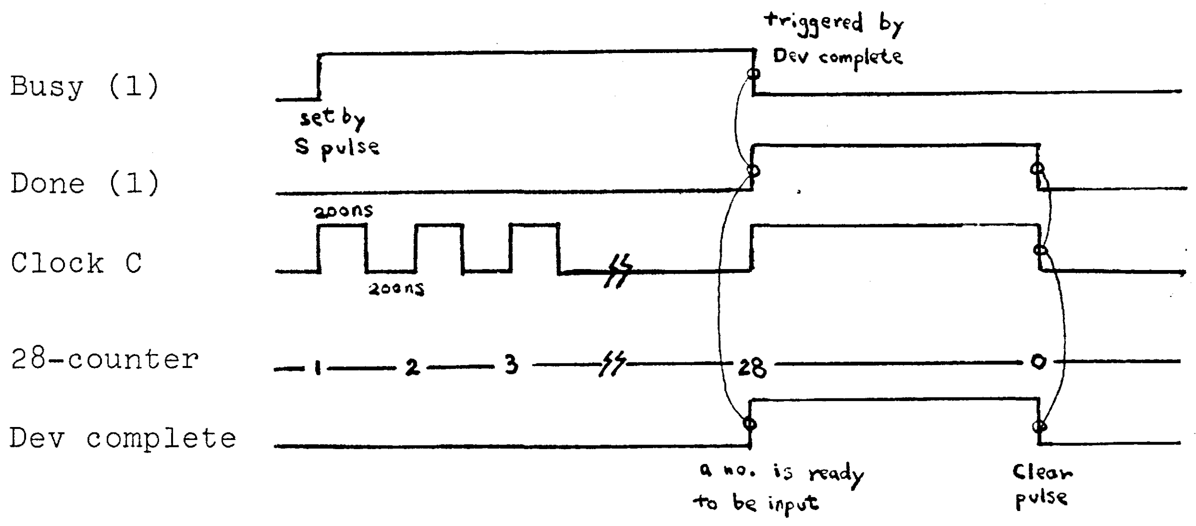


Fig. 2.7 Timing synchronization in generating a rectangular number

Input

The computer can check whether there is a number ready for input by testing the Done flag of the device. If Done is set to 1 input procedure may be initiated. The input sequence is as follow:

```
SKPDN 42        ;test to see if Done=1
JMP .-1         ;no, keep testing
DIAC 0 42       ;yes, input and clear counter
DIB 1 42        ;input
```

After input, bit 0 through 15 of AC0(lower order word) will contain stage 8 through 23 of the FSR and bit 8 through 15 of AC1 will contain stage 0 through stage 7 of the FSR.

2.2 Gaussian distribution

The gaussian number is created by adding twelve consecutive rectangular numbers together. Details of generation is depicted in Fig. 1.2(b). Instead of adding the whole 24 bit number, only the least significant 20 bits are added together each time. It is done to suit the floating point notation of the Nova. Hence, the addition result will never exceed 24 bits in length.

CHAPTER 3

SOFTWARE CONTROL

The PRNG can be called from a Basic or a Fortran environment under the Real-time Disk Operating System(RDOS) of the Nova.

3.1 Fortran callable subroutines(5,6,7)

Three subroutines have been implemented in the Fortran environment. Their function and calling sequence are as follow.

(a) CALL RAND(X,XMEAN,STD) for rectangular distribution.
where X = returned rectangular distributed number.
XMEAN = mean of the distribution.
STD = standard deviation of the distribution.
X,XMEAN and STD are numeric variables.

(b) CALL RANG(X,XMEAN,STD) for gaussian distribution.
Everything will be similar to (a) except that X will be the returned gaussian distributed number.

(c) CALL SEED(S) for laying a seed in the generator.
where S = the starting value(seed) which would be loaded into the FSR of the generator. Arbitrary seed will be used if S is zero.
S is a numeric variable.

A listing of each subroutine is attached at the end of the chapter.

3.2 <u>Basic callable subroutines</u>(5,6,7)

Similar to the Fortran calls, there are three options available in BasicG.

(a) CALL 5,X,M,S for rectangular distribution.

where X = returned rectangular distributed number.

M = mean of the distribution.

S = standard deviation of the distribution.

X is a numeric variable. M and S are numeric expressions.

(b) CALL 6,X,M,S for gaussian distribution.

Everything will be similar to (a) except that X will be the returned gaussian number.

(c) CALL 7,S for laying of a seed.

where S = seed which would be loaded into the FSR of the generator. Arbitrary seed will be used if S is zero.

S is a numeric expression.

A listing of each subroutine is not provided because it is very similar to the corresponding Fortran subroutine except for the linkage between Basic and the assembly language subroutine.

```
;*****************************************************************
;       THIS FORTRAN CALLABLE ROUTINE WILL RETURN A RECTANGULAR
;DISTRIBUTED NUMBER TO THE CALLING PROGRAM.
;       THE CALLING SEQUENCE IS

;                 CALL RAND(X,XMEAN,STD)

;WHERE
;                 X = RETURNED RANDOM NO.
;            XMEAN  = MEAN FOR THE DISTRIBUTION
;              STD  = STANDARD DEVIATION WANTED

;WRITTEN BY CLEMENT LAM
;DATE:  JULY 28,77.

;*****************************************************************

        .TITL RAND
        .ENT RAND
        .EXTD .CPYL
        .EXTN FRET
        .NREL
        5
RAND:   JSR @.CPYL                             ;ENTER ROUTINE
        NIOS 42                                ;START DEVICE
        SUB 1 1
        DOA 1 76                               ;NORMAL MODE
        LDA 2 EXPR                             ;EXPR=EXPONENT TO BE ADDE
        SKPDN 42
        JMP .-1
        DIAC Ø 42                              ;INPUT AND CLEAR DEVICE
        DIB 1 42
        ADD 2 1                                ;FLOAT THE NUMBER
        STA 1 TEMP
        STA Ø TEMP+1
        LDA Ø .TEMP
        DOBP Ø 74                              ;LOAD FPH WITH NUMBER

;CONVERT THE NUMBER TO STANDARD DEVIATE WITH MEAN=Ø AND STD=1

        LDA 1 RM
        DOAS 1 74                              ;SUB. SINGLE
        LDA Ø RSTD
        DOAP Ø 74                              ;MULT. SINGLE
```

```
;ADD IN USER DEFINED MEAN AND STD

          LDA 2,-165,3
          DOAP 2 74                         ;MULT. SINGLE WITH STD
          LDA 1,-166,3
          DOA 1 74                          ;ADD IN MEAN

          LDA Ø,-167,3
          DOBS Ø 74                         ;STORE NUMBER
          FRET

RM:       .+1                               ;RM=INTRINSIC MEAN
          Ø.5
RSTD:     .+1                               ;(1/RSTD)=INTRINSIC STD
          3.4641Ø16
.TEMP:    .+1
TEMP:     .BLK 2
EXPR:     Ø4ØØØØ
          .END
```

```
;*************************************************************

;       THIS FORTRAN CALLABLE ROUTINE WILL RETURN A GAUSSIAN
;DISTRIBUTED NUMBER TO THE CALLING PROGRAM.
;       THE CALLING SEQUENCE IS

;                   CALL RANG(X,XMEAN,STD)

;WHERE
;                   X = RETURNED GAUSSIAN NO.
;              XMEAN  = MEAN FOR THE DIST.
;                STD  = STANDARD DEVIATION WANTED

;WRITTEN BY CLEMENT LAM
;DATE: JULY 28,77.

;*************************************************************

        .TITL RANG
        .ENT RANG
        .EXTD .CPYL
        .EXTN FRET
        .NREL
        5
RANG:   JSR @.CPYL
        NIOS 42                  ;START DEVICE
        STA 3 POINT              ;SAVE STACK POINTER
        SUB 3 3
        STA 3 TEMP+1
         DOA 3 76                ;WRITE STATUS,NORMAL MODE
        LDA 1 CONST              ;INITIALIZE COUNT=12
        STA 1 COUNT
        SKPDN 42
        JMP .-1

;LOOP IS A ROUTINE TO GENERATE THE GAUSSIAN NO.

LOOP:   DIAC Ø 42                ;INPUT AND CLEAR DEVICE
        DIBS 1 42                ;INPUT AND RESTART DEVICE
        LDA 2 M17                ;SAVE BITS 12 TO 15
        AND 2 1
        LDA 2 TEMP+1

;DOUBLE PRECISION ADDITION. RESULT IS IN AC2 & 3

        ADDZ Ø 2 SZC
        INC 3 3
        ADD 1 3

        STA 2 TEMP+1
```

```
          DSZ COUNT                   ;COUNT=Ø ?
          JMP LOOP                    ;NO, CONTINUE TO ADD

          LDA 1 EXPG                  ;YES, FLOAT THE GAUSSIAN NO.
          ADD 1 3                     ;ADD IN EXPONENT
          STA 3 TEMP
          LDA Ø .TEMP
          DOBP Ø 74                   ;LOAD SINGLE

;CONVERT TO STANDARD DEVIATE, M=Ø & STD=1
;INTRINSIC STD=1

          LDA 1 GM
          DOAS 1 74                   ;SUB SING

;ADD IN USER DEFINED MEAN AND STD

          LDA 2 POINT
          LDA 3,-165,2
          DOAP 3,74                   ;MULT SING
          LDA 1,-166,2
          DOA 1 74                    ;ADD SING

          LDA 2,-167,2
          DOBS 2 74                   ;STORE SINGLE
          FRET                        ;RETURN

GM:       .+1                         ;GM=INTRINSIC MEAN OF DIST
          6.Ø

POINT:    Ø
CONST:    14
COUNT:    Ø
M17:      17
EXPG:     Ø4Ø4ØØ
.TEMP:    .+1
TEMP:     Ø
          Ø
          .END
```

```
;*****************************************************************

;       TO LAY A SEED IN THE RANDOM NUMBER GENERATER

;                   CALL SEED(S)
;WHERE
;       S = DESIRED STARTING NUMBER.
;           IF S=Ø., SEED WILL BE ARB.
;         S COULD BE ANY REAL NO.

;WRITTEN BY CLEMENT LAM
;DATE: JULY 28,77.

;*****************************************************************

        .TITL SEED
        .ENT SEED
        .EXTD .CPYL
        .EXTN FRET
        .NREL
        3
SEED:   JSR @.CPYL
        LDA 2,-167,3
        LDA Ø,Ø,2
        MOV Ø Ø SNR              ;TEST FOR Ø SEED
        FRET                     ;SEED=Ø., RETURN.
        LDA 1 1 2
        DOA Ø 42                 ;LOAD SEED INTO THE GENERATER
        DOB 1 42
        FRET

        .END
```

CHAPTER 4

TESTS AND OBSERVATION

The PRNG was subjected to three statistical tests. They were

(i) Frequency test

(ii) Correlation test

(iii) d^2-test

4.1 Frequency test(1,3)

In this test, one divides the possible existence interval of the numbers in equal non-overlapping intervals and tallies the amount of numbers in each interval. The probability density function and the distribution function of the generated numbers can be obtained by examining the tally in each interval.

4.1.1 Rectangular distribution

The interval examined was (0,1). It was divided into 1000 channels or bins and 10^6 numbers were sampled and sorted into the corresponding channel.

If the numbers are uniformly distributed in (0,1), then one would expect each channel to contain $1000 \pm \sqrt{1000}$ numbers. If the numbers of elements inside a channel is plotted against the channel number, a horizontal line should be obtained.

Fig. 4.1 shows the actual frequency curve obtained by the above mentioned method. It is quite a close approximation of the uniform distribution. Further look at Fig. 4.3 indicates that the distribution obtained is statistically acceptable. The normal standard deviation is equal to $\sqrt{1000}$. It can be seen that all non-zero channels lie within the 3σ limit. The integral distribution curve shown in Fig. 4.2 further reinforces the indication that the distribution so obtained is uniform.

4.1.2 <u>Gaussian distribution</u>

The interval being examined was (0,12). It was again divided into 1000 channels and 10^6 numbers were sampled.

From Fig. 4.4, the frequency curve obtained from plotting the tally in each channel against the channel number appears like a gaussian distribution. Almost all counts are inside the 3σ limit. For comparison the analytical description of a gaussian distribution is plotted on the same graph. This reveals good agreement suggesting the random source is statistically acceptable. The integral distribution curve in Fig. 4.5 further supports this inference.

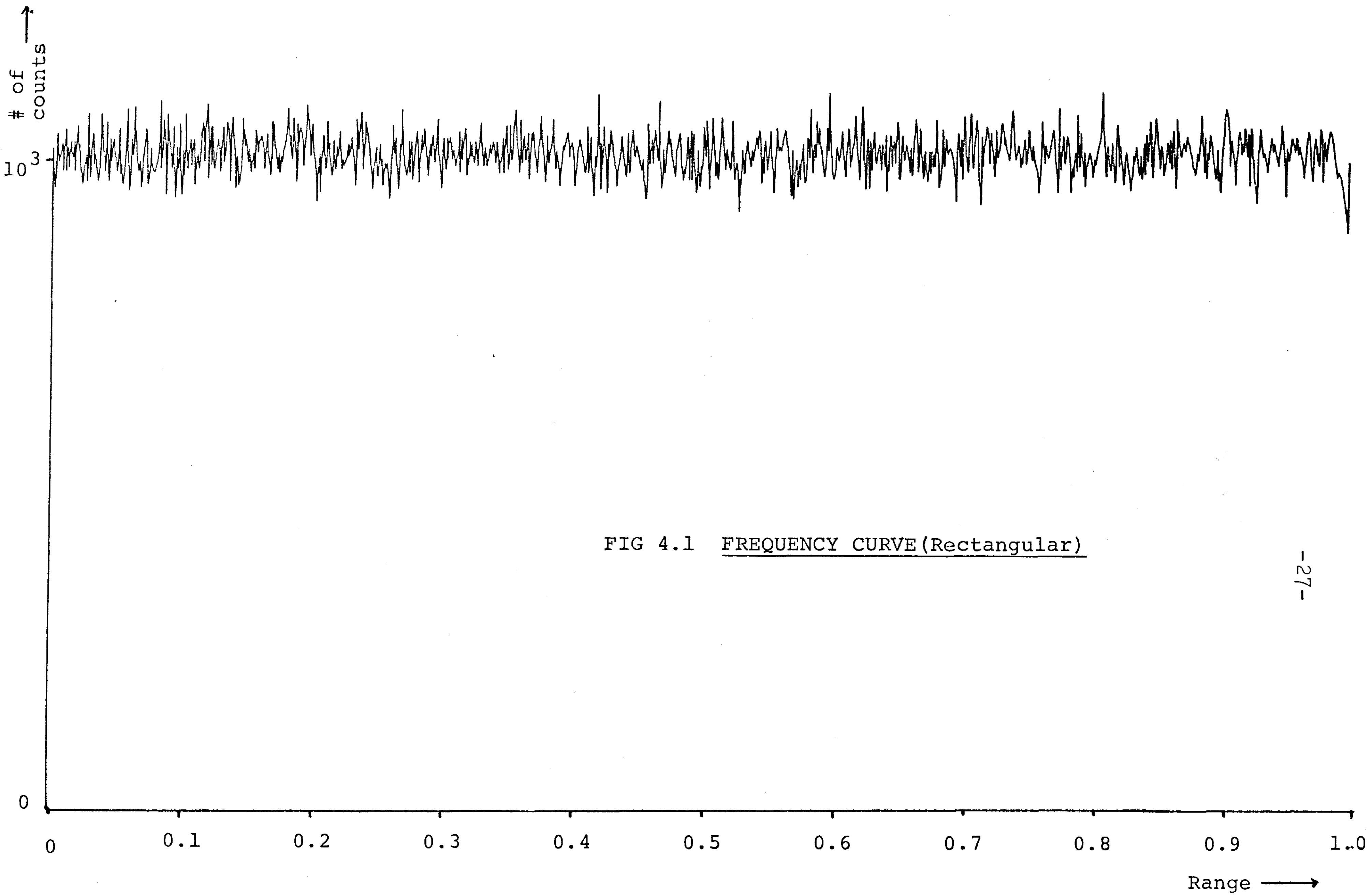


FIG 4.1 FREQUENCY CURVE(Rectangular)

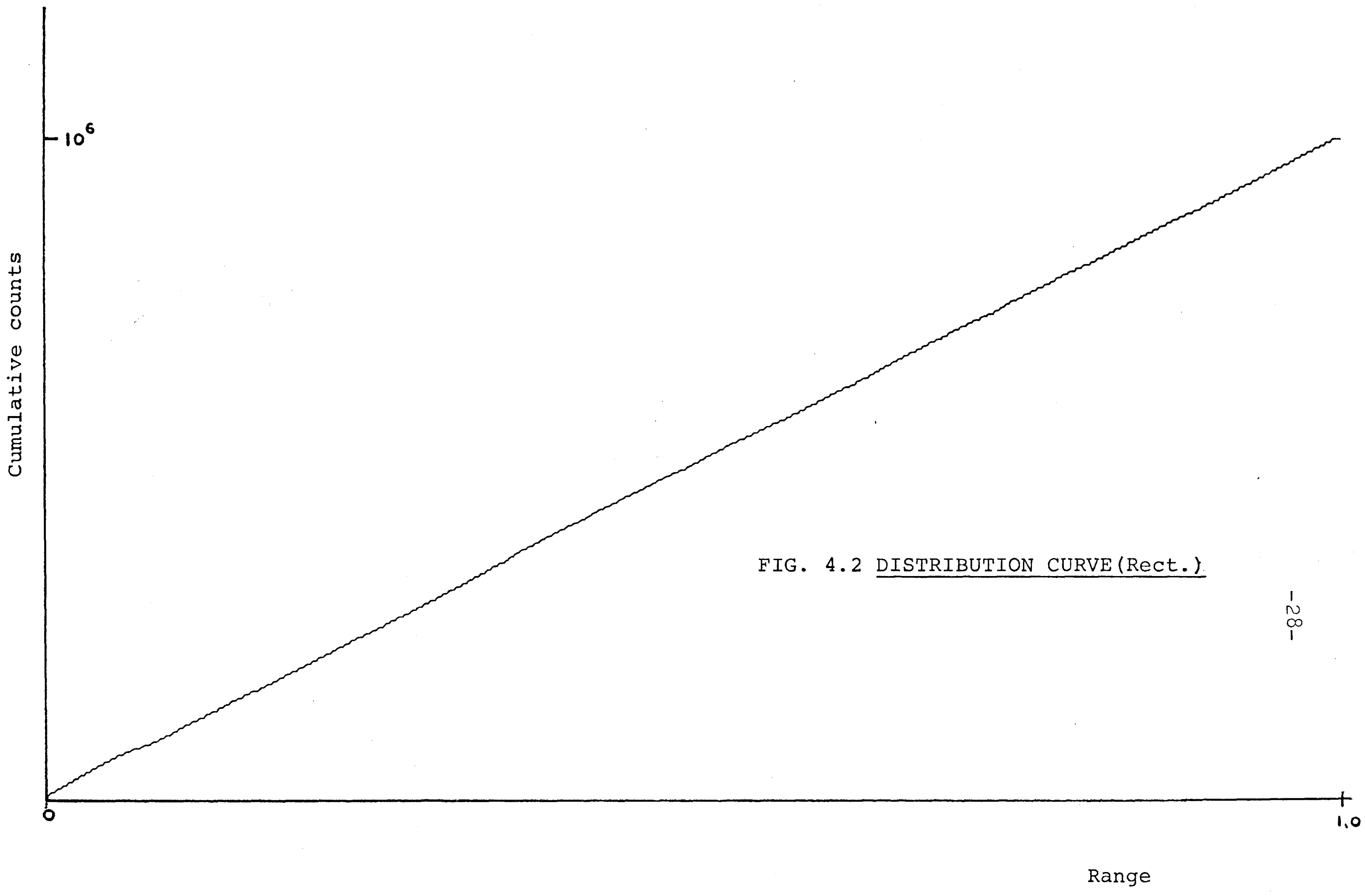


FIG. 4.2 <u>DISTRIBUTION CURVE(Rect.)</u>

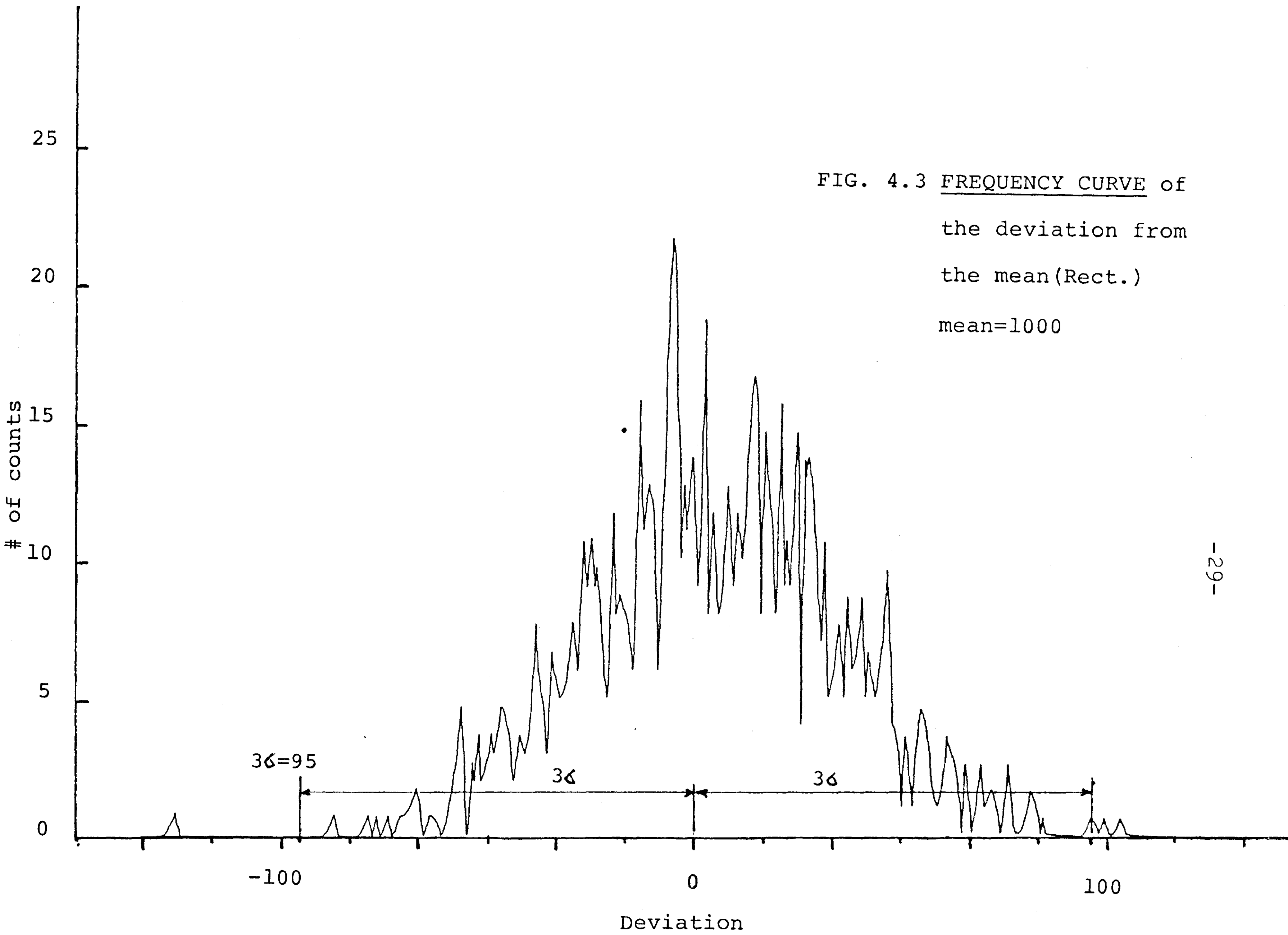


FIG. 4.3 FREQUENCY CURVE of the deviation from the mean (Rect.) mean=1000

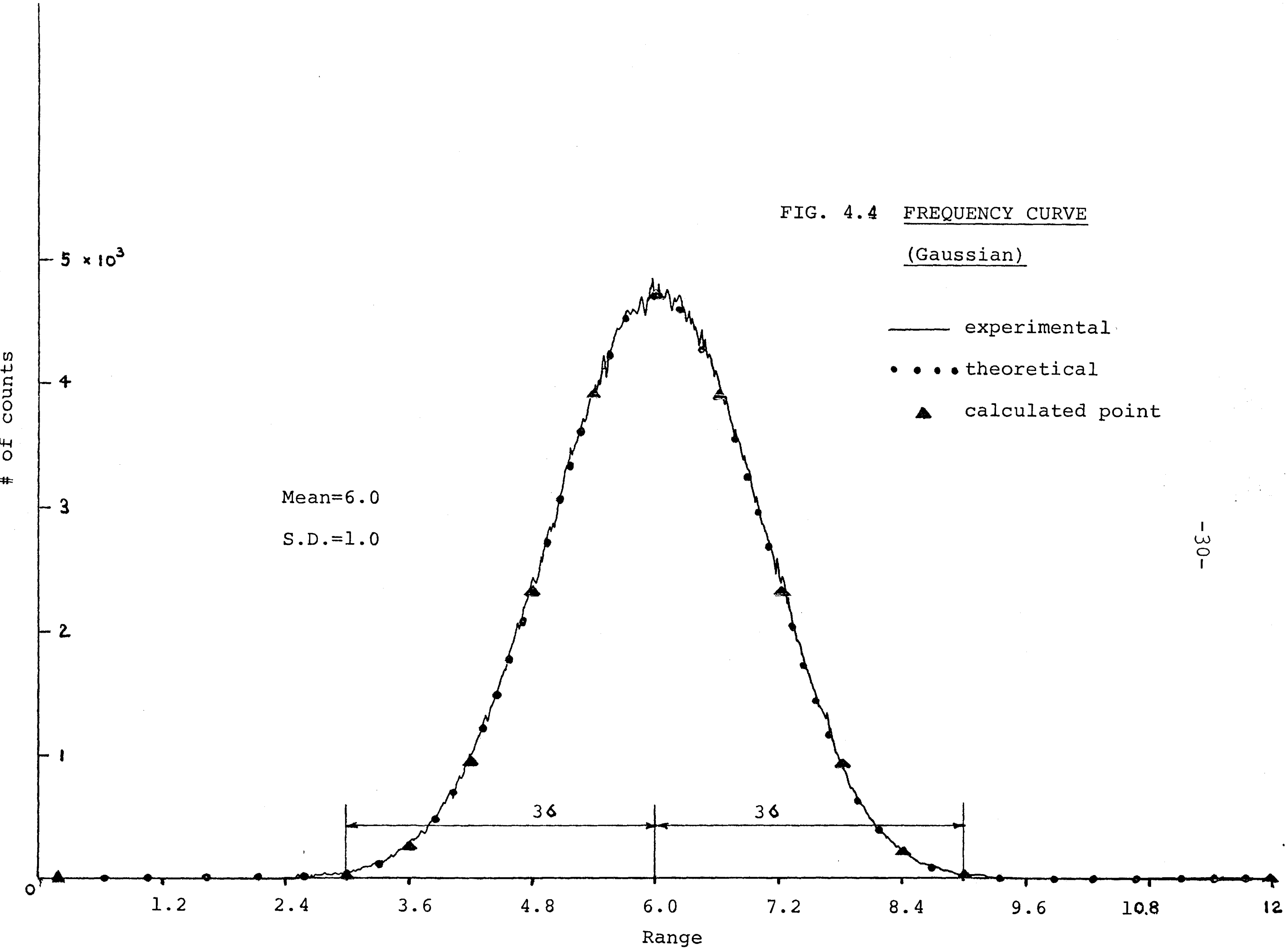


FIG. 4.4 FREQUENCY CURVE (Gaussian)

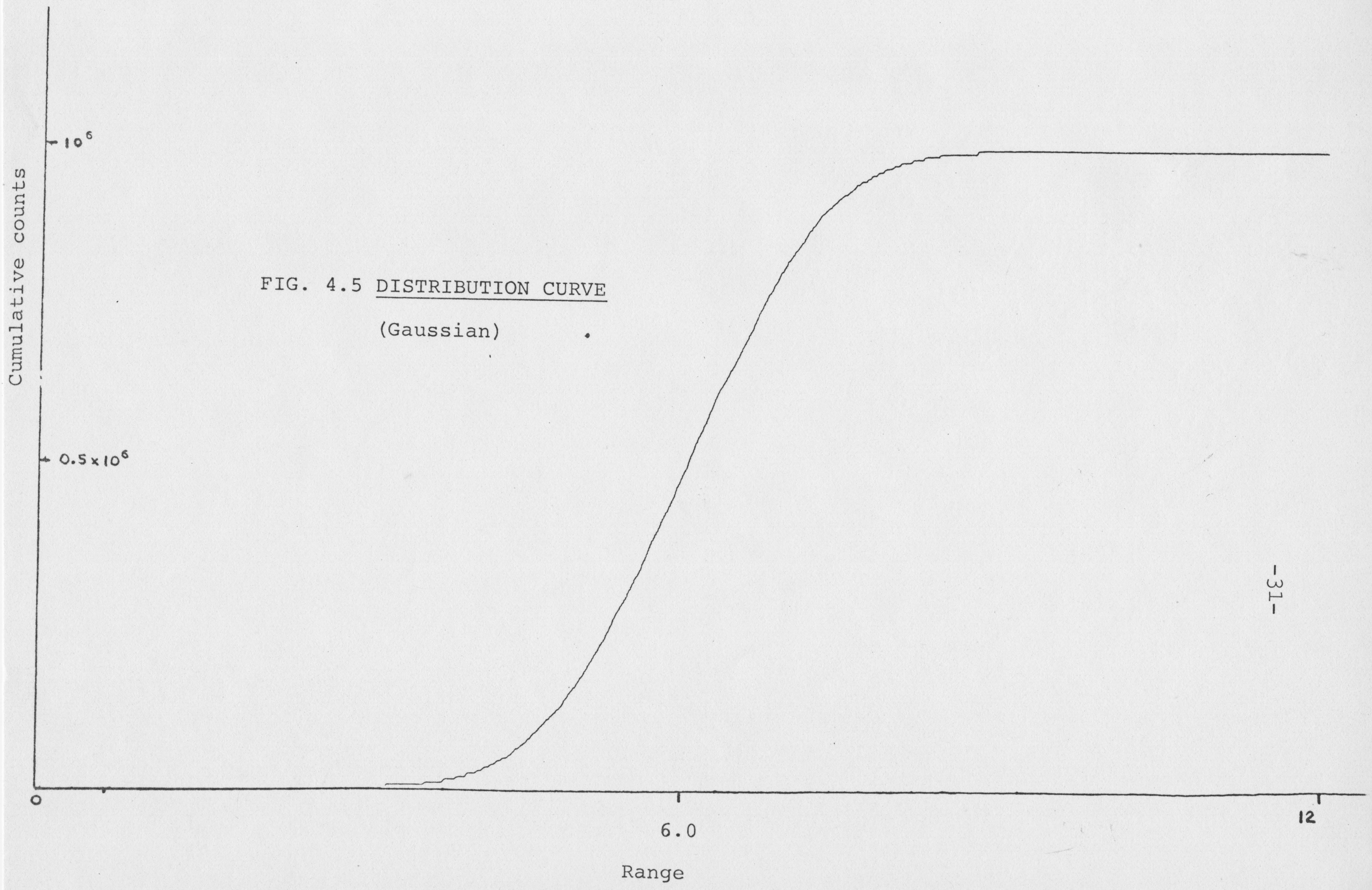


FIG. 4.5 DISTRIBUTION CURVE

(Gaussian)

4.2 Correlation test(1,3)

The autocorrelation function of a function f(t) is defined as

$$R(k) = \int_{-\infty}^{\infty} f(t)f(t-k)dt$$

To check whether a sequence X is random, one can find the autocorrelation between x_n and x_{n+k} where k is the lag in the generation order and compare them with the expected values.

4.2.1 Rectangular distribution

The sequence examined had a mean equal to 0 and standard deviation equal to 1.

Autocorrelation at lag k is given by

$$R(k) = \sum_{n=1}^{N} x_n x_{n+k} \qquad \text{where N is the number of elements}$$

$$\text{Expected value of } R(0) = \overline{\sum_{n=1}^{N} x_n x_n} = \sum_{n=1}^{N} \overline{x_n^{\,2}}$$

$$\text{But } \overline{x^2} = \int_{\text{lower limit}}^{\text{upper limit}} x^2 f(x)dx \qquad \text{where f(x) is the probability density function}$$

$$= \int_{l.l.}^{u.l.} (x-0)^2 f(x)dx$$

$$= \text{Variance of the distribution}$$

$$= 1$$

therefore, $\sum_{n=1}^{N} \overline{x_n^2} = N * \overline{x^2} = N$

Expected value of R(0) = N

Expected value of R(k), $k \neq 0$

$$= \overline{R(k)} = \overline{\sum_{n=1}^{N} x_n x_{n+k}} = \sum_{n=1}^{N} \overline{x_n x_{n+k}}$$

If x_n and x_{n+k} are independent of each other,in another words, they are uncorrelated, then

$$\overline{R(k)} = \sum_{n=1}^{N} \overline{x_n} * \overline{x_{n+k}}$$

But the expected value of the sequence is equal to its mean. Therefore,

$$\overline{R(k)} = \sum_{n=1}^{N} 0*0 = 0 \qquad k \neq 0$$

For N = 128, one would expect if the sequence is uncorrelated

$$R(0) = 128$$

$$R(k) = 0 \qquad k \neq 0$$

Table 4.1 shows the autocorrelations of the rectangular distribution with expected mean = 0 and expected standard deviation = 1.

Table 4.1 Autocorrelations of the rectangular distribution

Number of elements used (N) = 128

Number of repeated trials = 5000

lag k	R(k)	lag k	R(k)
0	128.02	11	0.02
1	-0.08	12	0.24
2	0.00	13	-0.04
3	0.11	14	0.28
4	-0.07	15	0.11
5	0.15	16	0.30
6	-0.11	17	-0.40
7	-0.03	18	0.16
8	-0.15	19	0.05
9	0.04	20	-0.11
10	-0.02		

Mean value of R(1) to R(20) = 0.02

The autocorrelations agree quite closely with the expected values. Should more trials were performed, correlations at higher lags would have come closer to zero.

4.2.2 Gaussian distribution

The sequence being examined has an expected mean of 0 and expected standard deviation equals 1.

128 numbers were used in each autocorrelation test. By calculation similar to that of section 4.2.1, the expected autocorrelations are as follow:

$R(0) = 128$

$R(k) = 0 \qquad k \neq 0$

Table 4.2 below shows the actual autocorrelations of the gaussian distribution with mean = 0 and standard deviation = 1.

Table 4.2 Autocorrelation of the gaussian distribution

Number of elements (N) = 128

Number of repeated trails = 10,000

lag k	R(k)	lag k	R(k)
0	128.02	10	-0.12
1	-0.10	11	0.12
2	-0.03	12	0.19
3	0.01	13	-0.10
4	-0.06	14	-0.09
5	0.06	15	-0.07
6	0.05	16	0.05
7	-0.06	17	0.04

8	0.09	18	0.13
9	-0.17	19	-0.08
		20	-0.09

Mean value of R(1) to R(20) = -0.01

The actual autocorrelations are very close to the expected values. This shows that the gaussian distribution so generated is quite a good approximation to the true gaussian which has the same mean and standard deviation.

4.3 d^2-test(1,3)

The frequency test and the autocorrelation test are developed for use in connection with random sampling and that Monte Carlo applications seem to require other tests. The d^2-test is designed for this purpose.

Assume that the random numbers(uniform distribution) lie within the interval (0,1) and regard four consecutive random numbers as the coordinates of two points in the unit square. Determine the square of distance between the two points (d^2). If the numbers are rectangularly distributed over (0,1), then the distribution function of d^2 is given by

$$p(d^2 \leqslant s^2) = \begin{cases} \pi s^2 - 8/3s^2 + s^4/2 & \text{when } s^2 \leqslant 1 \\ 1/3 + (\pi - 2)s^2 + 4(s^2 - 1)^{\frac{1}{2}} + 8/3(s^2 - 1)^{3/2} & \\ - s^4/2 - 4s^2 * \sec^{-1} s & \text{when } 1 < s^2 \leqslant 2 \end{cases}$$

The test consists in comparing the frequencies of a set of d^2-values obtained from a sequence generated by a random number generator with the theoretical probabilities.

Table 4.3 shows the calculated theoretical values and the actual values for a given s. The sample space is 10^4 d^2-values.

Table 4.3 <u>d^2-test</u>

s^2	Theoretical $p(d^2 \leqslant s^2)$	Actual $p(d^2 \leqslant s^2)$
0.1	0.235	0.236
0.3	0.549	0.552
0.5	0.753	0.752
0.7	0.882	0.886
1.0	0.975	0.973
1.01	0.976	0.974
1.1	0.986	0.987
1.5	0.999	0.999
1.8	1.000	1.000
2.0	1.000	1.000

As one can see, the experimental results agree closely with the theoretical values.

4.4 Speed consideration

One of the reasons to build this PRNG is because of time-saving consideration. Older generators are either too slow or the size of the sequence is too small. In building this PRNG, one of the objectives was to speed up the generation time, especially for the gaussian distribution.

Table 4.4 compares the speed of the new generator with the previous software generator.

Table 4.4 Speed of existing generators

Generator	Programming Environment	Time to generate 10^4 numbers: Rectangular	Time to generate 10^4 numbers: Gaussian
New (28 bits)	Fortran	3.85 sec.	5.70 sec.
	Basic	21.30 sec.	24.40 sec
	Assembly	0.15 sec.	1.90 sec.
Old (32 bits)	Basic	26.60 sec.	80 sec.
Data General's (16 bits)	Basic	16.40 sec	-----

It can be seen that a big improvement in speed is achieved for the gaussian distribution. As a matter of fact, the generator is capable of generating one rectangular number in about 11.2 microseconds. All the time has been wasted in program linkage and system commands when working in the high level language environment.

The generator provided by Data General is faster than the existing ones. However, it does not produce samples which obey any of the randomness tests. It is not desirable when numbers with a greater degree of randomness are required.

CHAPTER 5

CONCLUSION

The rectangular and gaussian distributions produced as a result of the PRNG have been proved to be a very good approximation of the corresponding theoretical distributions judging from the test results. It is relatively inexpensive and easy to construct such a PRNG using shift registers. When cost and time are the main concern, this type of PRNG is most suitable.

The following points are worth mentioning.

1. The clocking frequency could be increased considerably to speed up the generation time. The fact that a large portion of the time in getting a number is wasted in system linkage(Table 4.4) makes it quite meaningless to increase the clocking frequency unless the user is willing to work in an assembly language environment.
2. The gaussian distribution can be generated by hardware. The whole circuitry will become a lot more complex. As long as useful instructions can be squeezed in between generation time of a rectangular number, the time saved is not significant. In fact, the required hardware to generate the gaussian distribution has been implemented and it only improved the speed in a Fortran environment marginally while no significant improvement was observed in Basic. However,

the required circuitry became twice as complex as that for the rectangular distribution. In the light of maintenance of the hardware, software generation has been implemented for the generation of normally distributed variables.

APPENDIX A

MAXIMUM-LENGTH SHIFT-REGISTER-SEQUENCE(4)

The table below shows some of the maximum-length shift-register-sequence generators requiring a single modulo-2 adder. Feedback from shift-register stages n and m to modulo-2 adder, which feeds stage 1.

n	m or n-m	2^n-1
3	1	7
4	1	15
5	2	31
6	1	63
7	1 or 3	127
9	4	511
10	3	1023
11	2	2047
15	1,4 or 7	32,767
18	7	262,143
20	3	1,048,575
21	2	2,097,151
22	1	4,194,303
23	5 or 9	8,388,607

25	3 or 7	33,554.431
28	3,9 or 13	268,435,455
31	3,6,7 or 13	2,147,483,647
33	13	8,589,934,591

APPENDIX B

I. C. LAYOUT

1	2 7408	3 7403	4 74194 (stage 0)	5 7403 (I/O)
6 7476	7 7400	8 7474	9 74194	10 7403 (I/O)
11 74161	12 7411	13 7432	14 74194	15 7403 (I/O)
16 74123 (Clock	17 7408	18 7408	19 74194	20 7403 (I/O)
21 74123	22 7408	23 7486	24 74194	25 7403 (I/O)
26 74123	27	28 74194 (27th)	29 74194	30 7403 (I/O)

REFERENCES

1. Hartley,M.G., "Digital Simulation Method", IEE Monograph Series 15, Peter Peregrinus(1975).
2. Golumb,S.W., "Shift Register Sequences", Holden-Day(1967).
3. Brandt,S., "Statistical and Computational Methods in Data Analysis", North-Holland(1970).
4. Korn,G.A., "Random-Process Simulation and Measurement", McGraw Hill(1966).
5. Data General Corporation, "Introduction to Programming the Nova", 1972.
6. Data Gen. Corp., "User's Manual, Fortran 4", 1975.
7. Data Gen. Corp., "Extended Basic, User's Manual",1975.