NOAA Technical Memorandum ERL WPL-104

AUTOMATIC GAIN CONTROL FOR AN HF DOPPLER RADAR RECEIVER

J. D. Forberg
B. L. Weber

Wave Propagation Laboratory
Boulder, Colorado
November 1982

noaa NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION / Environmental Research Laboratories

NOAA Technical Memorandum ERL WPL-104

AUTOMATIC GAIN CONTROL FOR AN HF DOPPLER RADAR RECEIVER

J. D. Forberg
B. L. Weber

Wave Propagation Laboratory
Boulder, Colorado
November 1982

UNITED STATES
DEPARTMENT OF COMMERCE
Malcolm Baldrige,
Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION
John V. Byrne,
Administrator

Environmental Research
Laboratories
George H. Ludwig
Director

NOTICE

Mention of a commercial company or product does not constitute an endorsement by NOAA Environmental Research Laboratories. Use for publicity or advertising purposes of information from this publication concerning proprietary products or the tests of such products is not authorized.

CONTENTS

1. INTRODUCTION

In recent years, our laboratory* has developed and operated an HF Doppler radar system to remotely measure surface currents and waves on the ocean. The basic ideas and principles of that radar system are discussed by Georges (1983), Barrick and Evans (1976), and Barrick (1974). One important hardware component is the HF receiver, which must have a large dynamic range of sensitivity. We shall discuss here the Automatic Gain Control (AGC) algorithm which is used to adjust that sensitivity.

2. THE CONCEPT

The AGC concept is really rather simple. The output of the receiver is examined to determine if the sensitivity of the receiver needs to be altered. If necessary, a gain change is computed and then introduced in the receiver automatically under computer control. Therefore, this AGC procedure is nothing more than a feedback loop, which is analogous to those in some common electronics circuits.

However, the AGC cannot be enabled all of the time. When sea echo data are being collected for Doppler processing, the receiver sensitivity must be held constant. Otherwise, the Doppler spectra are contaminated with spurious components created by the changing sensitivity. The AGC is implemented, therefore, briefly at the beginning of data collection. After the gains have been set to suitable values, the AGC is disabled and sea echo data are recorded.

There is a minor complication because the radar employs a pulsed/ground-wave propagation HF signal. As a result, the sea echo varies dynamically with time delay after the transmitted pulse is initiated. For example, the sea echo returned from 60 km distance may be more than 60 dB weaker than the sea echo

---

* The Wave Propagation Laboratory/ERL/NOAA, U.S. Dept. of Commerce, Boulder, Colorado.

power from a couple of kilometers off shore. Therefore, the receiver sensitivity must be increased with time delay after the pulse is transmitted. Furthermore, the sensitivity at any given time delay must be repeated exactly for every pulse. The AGC is adjusting, then, a whole profile of gain versus time delay (or range).

3. SOFTWARE ALGORITHM

A schematic of the AGC algorithm is given in Fig. (3-1). We shall now discuss in detail each step of that algorithm. The computer software code itself is contained in an appendix at the end of this report.

3.1 Initialization

Just prior to the start of AGC, the receiver gains for every range cell are retrieved from an array in memory. These gains are usually the ones that were used during the previous period of data acquisition. However, the array may contain meaningless numbers. Whatever their values, they will be loaded into the receiver.

Step (1) in Fig. (3-1) involves setting the gain G_n for each range cell to its array value. The subscript n does not refer to the range-cell counter. It is an iteration index. At this point, G_n is the initial gain at the start of AGC. After AGC is completed, the new receiver gain will replace this one in the memory array and the receiver.

Two iteration counters, i and j, are initialized to zero. The first one increments by 1 up to i = I, which is the number of gain estimates that are averaged together. The second counter steps by 1 up to j = J, which is the number of times an estimate is made before averaging begins. The averaging is delayed because the initial gains could be much different from their optimum values. The J loops permit the gains to approach reasonable values before

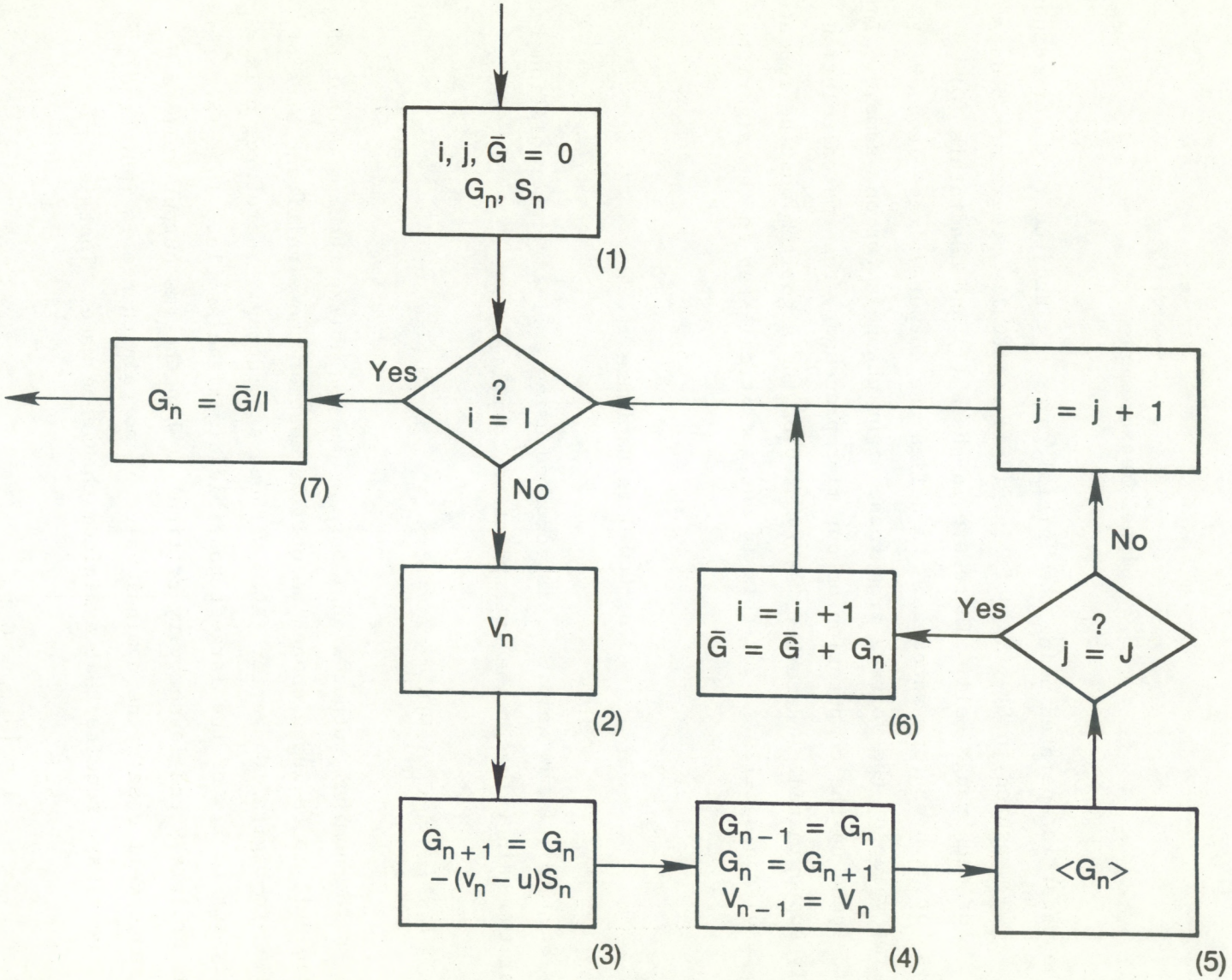

Figure (3-1). The AGC flow diagram shows seven fundamental steps. Each is discussed in order in Sections 3.1 through 3.6.

averaging is done to reduce noise. In addition, the average gain array $\bar{G}$ is initialized to zero and the slope $S_n = (.1u)^{-1}$, where u is the nominal output signal of the receiver. More will be said later concerning these parameters.

3.2 Signal Measurement

The radar is operated using the initial gains G_n, and the receiver output signal V_n is measured in Step (2) of Fig. (3-1). Actually, there are 8 data channels for each range cell. There are in-phase (I) and quadrature (Q) channels for each of four antennas. We define the output voltage level V_n to be the rms value of the voltage from these eight channels for one sample. In order to speed up the computation on our microprocessor, the rms calculation was only approximated. (Remember, every step of the algorithm must be repeated for up to 64 range cells.) But this is not a critical step in the algorithm.

3.3 New Gain Estimate

In Step (3), a new gain G_{n+1} is computed using a linear interpolation from the last two gain values. The interpolation formula is

$$G_{n+1} = G_n - (V_n - u)\, S_n,$$

where u is the nominal output signal level. We would like the rms output of any range cell to be large enough to utilize most of the significant bits of the Analog-to-Digital Converter (ADC) without saturating. Therefore, u is typically set to 1/4 of the largest possible digitized sample. That factor is somewhat arbitrary and is not very critical. When the rms output voltage V_n in any range cell equals the nominal value u, no gain change is indicated. However, we always require that a minimum change be made. That is,

$$1 \leq |G_{n+1} - G_n|,$$

where the gain values are integers limited between 1 and 255. When the gain equals 1 the receiver is least sensitive, and when the gain equals 255 it is most sensitive.

The line slope is calculated according to

$$S_n = \frac{\Delta G_n}{\Delta V_n} = \frac{|G_n - G_{n-1}|}{|V_n - V_{n-1}|}$$

where

$$1 \leq |G_n - G_{n-1}| \leq 32$$

and

$$.1\ u \leq |V_n - V_{n-1}| \quad .$$

The above constraints on the slope were made in order to stabilize the AGC in the presence of noise.

3.4 History

Since new gains are estimated from previous ones, arrays must be set aside to store some information for each range cell. Step (4) accomplishes this by moving the previous gains G_n to the G_{n-1} array. Then, the gains G_{n+1} (just computed) are placed in the G_n array. Similarly, the previous output levels V_n are moved to V_{n-1} to make room for the new measurements.

3.5 Filter

The AGC is not applied for every range cell. During the time the transmitted RF pulse is being formed, the receiver is blanked (or shut off) at its front end to prevent overloading its components. Hence, no sea echo gets

through to the output stage. The AGC would be ineffective at best. Furthermore, because the output signal level at this time is very low, the AGC would attempt to increase the receiver sensitivity to its upper limit. That can cause the receiver to be unstable in its early range cells. Therefore, the AGC is applied only in the region illustrated in Fig. (3-2). The transition region at either end is where the receiver electronics respond to the blanking command. The computed gains from the AGC region are then extended through the transistion regions and the blanking region. This results in smoother gain profiles, so the receiver is stable.

The above gain extension, along with some filtering, is done in Step (5) of Fig. (3-1). After the extension, a simple three-point average is done on all of the gains. Thus, the gains are forced to change smoothly with range (or time delay). We found that the smoothing process is always a necessary safe-guard against the "picket-fence" effect. These wide-frequency-bandwidth and large-dynamic-range receivers exhibit a finite time delay from when you set the gain to when you see a change in output. That delay may be on the order of one or two range cells. Therefore, the gain array and signal array must be aligned before the AGC is initiated each pass. But the time delay does not correspond to exactly an integral number of range cells. Thus, a shift of array indices can not align the gains and signals sufficiently well. Without filtering, the AGC often raises the gain in every other range cell and lowers it in-between. That is because the AGC is not really seeing the signal that it changes in the feedback loop. The result is alternating high and low signals that continue to get higher and lower, respectively. Hence, the "picket-fence" name came into being.

3.6 Averaging

Steps (2) through (5) are repeated a small number of times (J = 8), amounting to about four seconds. This allows the AGC time to get closer to optimum gain settings if the initial ones are poor. After those iterations, the gains for each range cell are accumulated in Steps (2) through (6). We

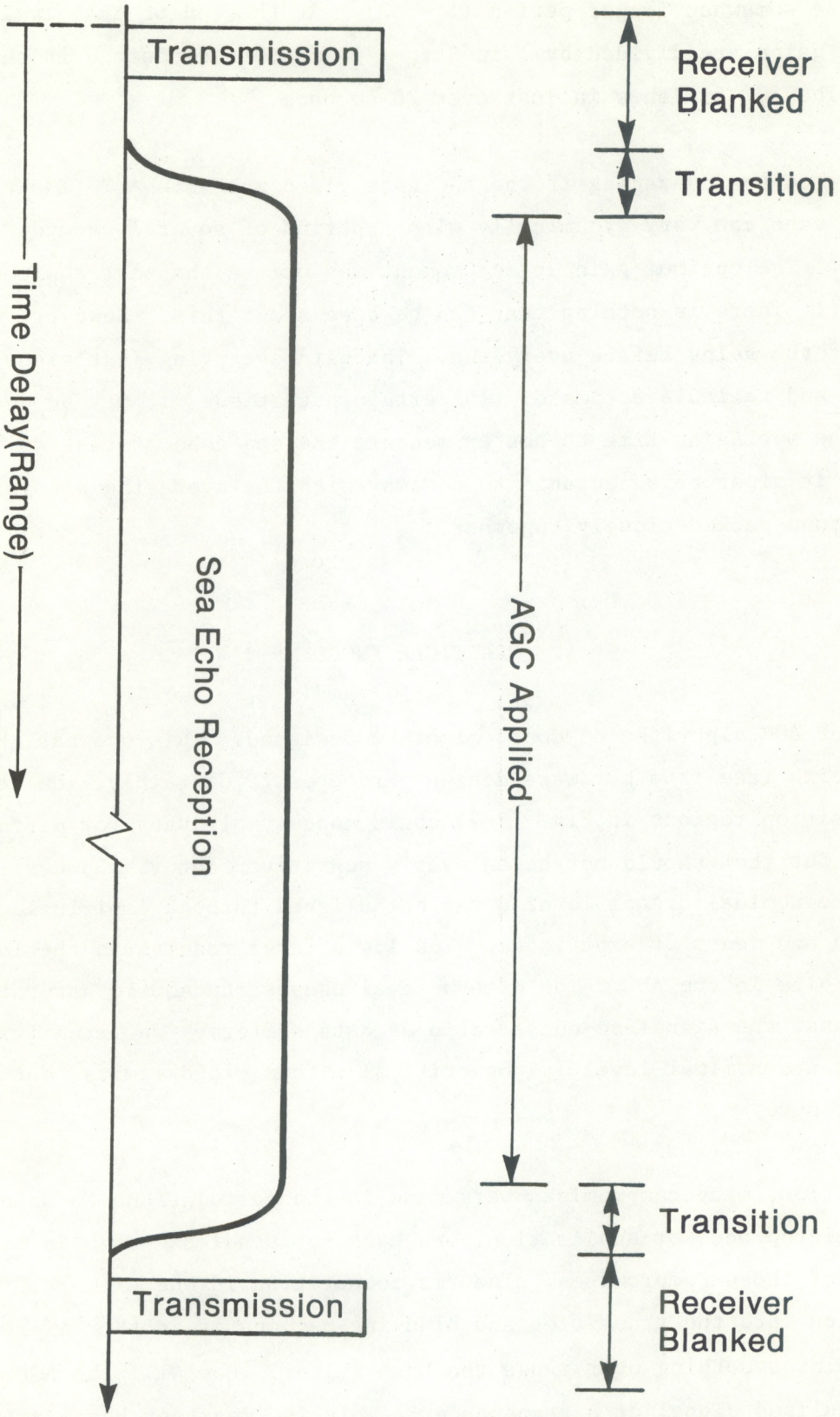

Figure (3-2). This diagram indicates where the AGC operates relative to the transmitted pulse in time. The AGC does not work properly in the blanking or transition regions.

do this for a somewhat longer period (I = 32). At the end of that cycle, the accumulated gains are divided by I in Step (7) and then deposited in the gain array G_n. The AGC finishes in just over 20 seconds.

But why do the averaging if the AGC gets close much sooner? The answer is that sea echo can vary dynamically with a period of several seconds or even much longer. The optimum gain at one moment may not be the best choice one minute later. There is nothing that can be done about this. However, we can and do reset the gains before every run. The gain averaging just gives a more stable and reliable estimate. In certain situations, it may be desirable to extend the averaging time to better measure the sea echo statistics. On the other hand, it might be acceptable to do away with the averaging altogether for short data runs packed closely together.

4. PRACTICAL PROBLEMS

A better AGC algorithm no doubt might be designed. This one has the virtue of being free from hardware design parameters. Certainly, the length of the transition regions in Fig. (3-2) could conceivably vary for different receivers. But that should not be an overly sensitive factor. However, the choice of the nominal signal level u was established through empirical observations and years of experience. Too low a level results in the loss of significant bits in the ADC. Too high a level causes the ADC to saturate. In either case, the signal-to-noise ratio of data suffers. We intuitively believe that the nominal level is not critical within broad limits, but we do not know for sure.

In addition, many compromises were made in the calculations to maintain speed in a microprocessor application. We have not studied enough to find the total price of those compromises. The rms computation is one example. The gain extension into the transition and blanking regions is another example. Is the three-point smoothing over range the best filter? How will the AGC deal with the localized signal of a transponder? This is important because the

analog output of the receiver does not necessarily correspond to the digitized signal. The receiver has such a short response time that it is possible for signals to vary significantly in a period shorter than the digitization interval. For example, we have synthesized signals in the laboratory with pulse widths smaller than the time delay between samples. Hence, the pulse was not always "seen" by the AGC algorithm. While this is not a problem with sea echo, it could be a problem if a transponder pulse is so narrow it can fit between two range samples. These are a few of the practical problems that remain, although none seem to be serious. Perhaps we will be forgiven our oversight of these problems since the AGC is not an end in itself. More important problems elsewhere have commanded attention.

Meanwhile, some problems are resolved by fiat. The choice of averaging time in Step (6) is somewhat a subjective matter. It could be limited by the available time. Also, the interpolation method is simple and stable. But it could have some pathological conditions of which we are unaware. Some analysis of the response of this AGC algorithm to critical signals should be considered. Computer simulations are preferable to field testing because a wider range of conditions can be created. However, the simulations are only as good as the model of the real world. The ultimate judgment must come, then, after radar operation. We have tried to design the AGC such that the above questions and problems (along with others we do not see) should not impair radar applications.

5. REFERENCES

Barrick, D. E.; Sea backscatter at HF: Interpretation and utilization of the echo, preceedings IEEE, Vol. 62, No. 6, pp. 673-680, June, 1974.

Barrick, D. E. and M. W. Evans; Implementation of Coastal Current-Mapping HF Radar System - Progress Report No. 1, NOAA Tech. Report, ERL 373-WPL 47, July, 1976.

Georges, T. M. (ed.); Coastal Ocean Dynamics Applications Radar - A Users' Manual, NOAA Special Report, to be published, 1983.

6. APPENDIX

The AGC algorithm was designed for and implemented in a specific software package, although the concept and approach should be generally applicable to a larger body of control problems. The DEC* assembly language, MACRO-II, was used with the RSX11M operating system (Version 3.2). This program runs on a PDP 11/23 micro-computer, but it is compatible with all PDP 11 minicomputers. The software assumes a 16 bit word size and uses the floating point instruction set. The code itself is completely documented and the task-building procedure is explained.

---

* Digital Equipment Corporation, Maynard, Massachusetts.

```
.TITLE  AGC
;                                                                    ( 1)
.IDENT  /SV02.01/
.ENABL  LC
;
;       'AGC.TSK'       Automatic gain setting task.
;
;
;       Ownership:
;
;       Written for the 'CODAR' HF Radar system being developed by
;       the Sea State Studies group of the U.S. Dept. of Commerce,
;       N.O.A.A.-E.R.L.-W.P.L., Boulder, Colo.
;
;
;       Authorship:
;
;       By John Forbers and Bob Weber.  15-AUG-82.
;
;       Partially based on previous work by B.Weber, K.Sutterfield, M.Barth.
;
;       Written on a PDP 11/23 using MACRO II, running on RSX11M,
;       version 3.2.  This program is a small part of the 'CODAR'
;       data acquisition system, and the main 'CODAR' software
;       contains more complete documentation on the overall system.
;
;
;       Change History:
;
;       Modified last on 20-AUG-82 J.Forbers
;
;
;       General Function:
;
;       This routine does nothing but 1) acquire data for; 2) set the gains as
;       a result of; the auto gain subroutine.
;
```

```
;
;.;     AGC.CMD          /V02.00/                19-Aug-82
;.;
;.; This command file is used to create AGC.TSK, which is the task that as part
;.; of the remote-site CODAR data-acquisition and processing system, performs
;.; automatic-gain-control for the receiver.
;.;
;.; NOTES:
;.;     1) You must already have COMMON.TSK installed in the COMMON partition.
;.;
; ;     Creation of AGC.TSK:
; ;
; ;     1) Deletion of previous files:
; ;
; SET /UIC=[201,24]
; PIP AGC.OBJ;*/DE/NM
; SET /UIC=[201,34]
; PIP AGC.LST;*/DE/NM,AGC.MAP;*
; SET /UIC=[201,54]
; PIP AGC.TSK;*/DE/NM
; SET /UIC=[201,1]
; ;
; ;     2) Assembly of object module(s):
; ;
; MAC [201,24]AGC,[201,34]AGC/-SP=[201,10]AGC
; ;
; ;     3) Task building:
; ;
; TKB @[201,24]AGCBLD
; ;
; ;     4) Installation:
; ;
; REM AGC
; INS [201,54]AGC
;


;
; ;
; ;     AGCBLD.CMD
; [201,1]AGC/-CP/FP=[201,1]AGC,LB:[1,1]F4POTS/LB
; /
; UNITS =2
; ASG =SI:1       ; SI: is the SPI-board.
; ASG =TT2:2      ; TT2: is the A-scope.
; PRI=159
; RESCOM=[201,54]COMMON/RW
; TASK=AGC
; //
;
```

```
;
;       Executive references:
;
        .MCALL  SETF$C,QIOW$C,DIR$,SPND$S,CLEF$C,EXST$S,QIO$,QIOW$
        .MCALL  RSUM$,ASTX$S,DSAR$S,RDAF$,RSUM$C,WTSE$C
        .MCALL  QIO$C,RQST$C,WTSE$,QIOW$S

;
;
;       Global symbol definitons:
;

SI=1                                    ; SI: LUN and EFN
DBSY=34.                                ; "DATA.TSK BUSY" EF
                                        ; (We usurp it's use to keep everyone else
                                        ;       out of SI world)
GR=2                                    ; A-SCOPE LUN and EFN
ASE=62                                  ; Code to enable A-SCOPE.
ASD=65                                  ;         Disable.
IO.RGR=1004
IO.LGR=404
IO.RGD=1005
IO.LGD=405
IO.RDR=1006
IO.LDR=406
IO.RDD=1007
IO.LDD=407
IO.RCS=2400
IO.SFQ=IO.RCS+1
IO.SPW=IO.RCS+2
IO.SRM=IO.RCS+3
IO.STP=3000

;
;       In addition to the above definitions, some additional symbols are
;       referenced by this program, and reside in the installed partition
;       called 'COMMON', which is used for storage of parameters and for
;       inter-task communication.
;


;       The symbols used here that reside in COMMON:
;
;       DBSY    Flag to signify AGC task is active.
;       E$RF    Flag for various RF errors.
;       RCVCSR  Command/status register for receiver.
;       ENAB    Sets on-line graphics processor to analog mode.
;       H$PW    Ascii receiver pulse width(currently either 8 or 16 micro-sec.
;       H$FREQ  Four digit(ascii) receiver frequency.
;       SITOP   Location at which data from each sample from SPI is deposited.
;       SIBUF   Starting location of real data in SITOP.
;       GAINS   Location of gain values to be used by receiver.
;       GN      Scratch array for use by AGAIN subroutine.
;       SLOP      "     "   "   "   "   "       "
;
```

```
;
;       External Routines:
;
;               $CDTB   Converts ascii to binary(Fortran OTS).
;               SUMS    Subroutine to compute voltage.
;               GAIN    Subroutine to compute new gain values.
;
;
;       Local data blocks:
;

SIATT:  QIOW$   IO.ATT,SI,SI,,SIIOSB
;
;       SPI read puts data into the COMMON partition starting at
;       SITOP, but due to hardware design, actual data starts at
;       SITOP+2, or at SIBUF. (for more information, see the doc-
;       mentation in the 'DATA' task).
;
SIRLB:  QIO$    IO.RLB,SI,,,SIIOSB,AST,<SITOP,1024.>
SILGR:  QIOW$   IO.LGR,SI,SI,,SIIOSB,,<GAINS,128.>
SISPW:  QIOW$   IO.SPW,SI,SI,,SIIOSB,,<0>
SISRM:  QIOW$   IO.SRM,SI,SI,,SIIOSB,,<0>
SISTP:  QIOW$   IO.STP,SI,SI
GRWAL:  QIOW$   IO.WAL,GR,GR,,,,<ENAB,1,0>
ENAB:   .BLKB   1
        .EVEN
RSUM:   RSUM$   AGC
SIIOSB: .BLKB   4                       ; SI: I/O status block.

EXST:   .BLKW   1                       ; EXIT-WITH-STATUS word.
RDAF:   RDAF$   RDAFBF
RDAFBF:                                 ; RDAF$ buffer.
LOCEF:  .BLKW   2                       ; Local event flags.
GBLEF:  .BLKW   2                       ; Global event flags.
VMEAS:  .BLKW   64.                     ;
PCNT:   .WORD   0                       ; Pass counter.
;
```

```
;
;       Task entry point:
;
;       This task is spawned by the 'CODAR' task to set the receiver
;       gains to an optimum (or nominal) level. It receives data from
;       the receiver through the Signal Processing Interface(SPI),
;       sums the data over all 256 channels, and calls the auto gain
;       subroutine which then computes the next gain page values to
;       be used. Upon return, this routine loads the gain page with
;       the values returned by the auto gain subroutine and then issues
;       a 'QIO' to the SPI board to set a new set of voltages. Presently,
;       the auto gain routine will make 32 attempts at setting the
;       gains to the correct level, and then force this routine to
;       exit. Upon exit, 'CODAR' will then spawn the data aquistion
;       task to proceed with a run.
;
;
START:  SETF$C  DBSY            ; Set "DATA.TSK BUSY".
        CLRB    E$RF            ; Clear RF ERROR flag.
        CLR     RCVCSR          ; Clear receiver CSR in COMMON.
        CLR     PCNT            ; Clear pass counter.

        MOVB    #ASE,ENAB       ; Enable A-SCOPE.
        DIR$    #GRWAL
        DIR$    #SIATT          ; Attach SI:.
        BCS     BADSPI          ; Is it there?
        TSTB    SIIOSB          ; Bad news?
        BMI     BADSPI          ; If MI YES.
        DIR$    #SISRM,ERRSI    ; Set normal read mode.
        TSTB    SIIOSB
        BMI     BADSPI

        MOV     #H$PW,R0        ; Get pulse width from header area in COMMON.
        CALL    $CDTB
        MOV     R1,SISPW+Q.IOPL ; Load pulse width in DPB.
19$:    DIR$    #SISPW,ERRSI    ; Do an IO.SPW

        MOVB    H$FREQ,R0       ; Set the frequency.
        MOVB    H$FREQ+1,R1
        MOVB    H$FREQ+3,R2
        MOVB    H$FREQ+4,R3
        QIOW$S  #IO.SFQ,#SI,#SI,,,,<#'0,R0,R1,R2,R3>

        DIR$    #SILGR,ERRSI    ; Load initial gains.
        TSTB    SIIOSB          ;
        BMI     BADSPI          ;

        DIR$    #SIRLB,ERRSI    ; Start SI:
        MOV     #EX$SUC,EXST    ; and assume all will be OK when we exit
        SPND$S                  ; and go to sleep.
```

```
;                                                              ( 6)
;
;       Exit points:
;

;
;       Exit from Directive error.
;
ERRSI:  TST     (SP)+           ; Fake an "RTS PC".

;
;       Exit from SPI error.
;
BADSPI: MOVB    SIIOSB,E$RF

;
;       Exit from operator abort.
;

ABORT:  DSAR$S                  ; Disable AST'S.
        MOV     #EX$ERR,EXST    ; Let CODAR know it's not to take a data run.
        QIOW$C  IO.KIL,SI,SI    ; Stop SI: and kill queued requests.

;
;       Exit in normal manner.
;

FINIS:  QIOW$C  IO.DET,SI,SI    ;
        MOVB    #ASD,ENAB       ; Disable A-SCOPE.
        DIR$    #GRWAL
        TSTB    E$RF            ; If E$RF is in warning state, clear it.
        BLE     EXIT            ; (If it's in error state, we want OP to know).
        CLRB    E$RF
EXIT:   EXST$S  EXST
```

```
;
        .ENABL  LSB
;       AST service routine.
;
AST:    MOV     R0,(SP)         ; (Don't need IOSB address).
        MOV     R1,-(SP)        ;
        MOV     R2,-(SP)        ;
        MOV     R3,-(SP)        ;
        MOV     R4,-(SP)        ;
        MOV     R5,-(SP)        ;
        DIR$    #RDAF           ; Read event flags.
        BIT     #5,GBLEF        ; Have we aborted or killed?
        BEQ     OK              ; EQ -- NO
        JMP     ABO             ; YES -- Resume main code to exit.
OK:     CMPB    SIIOSB,#1       ; Was SI: successful?
        BGE     2$              ; If PL YES.
1$:     MOV     #BADSPI,16(SP)  ; NO.
        JMP     ASTOUT          ;
2$:     MOV     #3742,R0        ; Check receiver status.
        BIC     SIIOSB+2,R0     ; Are all appropriate bits set in receiver CSR?
        BEQ     6$              ; EQ -- YES.
        MOV     SIIOSB+2,RCVCSR ; NO -- Let CONSOLE know what the problem is.
                                ;       (By placing the CSR in COMMON).
        MOV     #3542,R0        ; Bit pattern for fatal errors.
        BIC     SIIOSB+2,R0     ; Was this a fatal error condition?
        BEQ     5$              ; EQ -- NO, We can continue (limping along...)
        MOVB    #-5,SIIOSB      ; YES, Set IO.ERR byte to -5 for CONSOLE'S use
        BR      1$              ; and abort.
5$:     MOVB    #1,E$RF         ; Set E$RF to warning and tell operator.
        BR      7$
6$:     CLRB    E$RF            ; (In case a warning condition has ended).

7$:     JSR     PC,SUMS         ; Update sums of absvals for 256 channels.
        JSR     PC,GAIN         ; Call AUTO gain routine.
        BCS     ASTFIN          ; CS -- Success!
        DIR$    #SISTP          ; Else, stop SI, reload gain page,
        DIR$    #SILGR          ; and try again.
        .DSABL  LSB
;
;       AST exit point.
;
AXIT:   DIR$    #SIRLB          ; Start SI read before suspending.
ASTX:   MOV     (SP)+,R5        ; Restore registers.
        MOV     (SP)+,R4        ;
        MOV     (SP)+,R3        ;
        MOV     (SP)+,R2        ;
        MOV     (SP)+,R1        ;
        MOV     (SP)+,R0        ;
        ASTX$S

ASTFIN: DIR$    #SISTP          ; Stop SI:
        DIR$    #SILGR          ; Reload hardware gain page.
        MOV     #FINIS,16(SP)   ; Awaken mainstream to finish a run.

ASTOUT: DIR$    #RSUM           ;
        DSAR$S                  ; Disable AST'S.
        DIR$    #SISTP          ;
        BR      ASTX            ;

ABO:    MOV     #ABORT,16(SP)   ; Resume main code to abort.
        BR      ASTOUT
```

```
;
        .SBTTL  Subroutine SUMS entry point.
;
;       Title:  Subroutine SUMS
;
;       Function:
;
;       This subroutine currently fills the 'Measured Voltage' array(VMEAS)
;       with an approximation to a true RMS value. The approximation used
;       is the sum of absolute values of the I-channel and Q-channel for
;       all four antennas in each of the 64. range cells. The sum is then
;       divide by 2 to prevent any overflow of values. In order to compen-
;       sate for this division, the nominal voltage levels in the AUTO-GAIN
;       subroutine have been set slightly lower.
;
;       Maximum error for using this approximation is 30%-40%(worst case).
;
;       Inputs: SIBUF   (256. channels- I&Q, 4 antennas, 64 range cells)
;                       SIBUF is filled through a QIO request.
;
;       Outputs: VMEAS
;                PCNT    (let's AUTO-GAIN know how many passes taken).
;
SUMS:   MOV     #64.,R0         ; Clear the 'measured voltage' array.
        MOV     #VMEAS,R1       ;
10$:    CLR     (R1)+           ;
        SOB     R0,10$          ;
        MOV     #SIBUF,R4       ; (ANT 1:RC 1:I),(ANT 1:RC 1:Q),(ANT 1:RC 2:I)
        MOV     #4,R2           ; ...(ANT 4:RC 64:I),(ANT 4:RC 64:Q)

3$:     MOV     #64.,R3         ; 64 * 4 RCA'S
        MOV     #VMEAS,R5       ;
1$:     TST     (R4)            ; Take absval of I-channel.
        BPL     40$             ;
        ADD     #1,(R4)         ;
        NEG     (R4)            ;
40$:    MOV     (R4)+,R0        ;
        ASH     #-3,R0          ; Divide by 8.
        ADD     R0,(R5)         ; Add to 'Voltage Measured' array.

        TST     (R4)            ; Take absval of Q-channel.
        BPL     41$             ;
        ADD     #1,(R4)         ;
        NEG     (R4)            ;
41$:    MOV     (R4)+,R0        ;
        ASH     #-3,R0          ; Divide by 8.
        ADD     (R5),R0         ;

        TST     R0              ; Make sure that 'Voltage' is not greater than
        BPL     32$             ; full scale(since not RMS).
31$:    MOV     #32000.,R0      ;
        BR      33$             ;
32$:    CMP     #32000.,R0      ;
        BGE     33$             ;
        BR      31$             ;
33$:    MOV     R0,(R5)+        ;

        SOB     R3,1$           ; Branch and continue for 64. range cells.
        SOB     R2,3$           ; Branch and continue for 4. antennas.
        INC     PCNT            ; Increment pass counter.
        RTS     PC
```

```
;
;       .SBTTL  Subroutine GAIN entry point.
;
;       Title:  Subroutine GAIN
;
;
;       Authors: John Forbers, Bob Weber          (18-AUG-82)
;
;       General function:
;
;       This routine looks at voltages returned from the CODAR receiver
;       and the attempts to adjust the gains(signal amplification) in
;       the receiver to provide the receiver and subsequent processing
;       with an 'optimum' signal level.
;
;
;       Inputs: Arrays 'VMEAS','GAINS','GN','SLOP'
;               Symbol 'PCNT'
;
;       Outputs: Array 'GAINS'
;
;       External routines:
;
;               $SAVAL  Saves R0-R5 (from Fortran Object Time System).
;               AVE3    Subroutine to do three point linear average.
;               MXMIN   Subroutine to limit gain values to 0 < gain < 256.
;
;
;       In order for this routine to work requires at least two
;       prior values, i.e. the last gain setting and the corres-
;       ponding voltage returned for that gain, and their present
;       values.  In order to get the first two, we use whatever
;       gains where last loaded in the receiver and their voltages.
;       The old gains and voltages are saved, and the same gains and
;       voltages are then used in the prediction algorithm to get a
;       new gain setting(in the first pass). For all subsequent passes
;       the oldest gain and voltage values are pushed out, and the
;       most recent are then used for the next prediction.
;
;       The method of predicting the new gain is the following:
;
;       First, notation:
;
;               V(N) = Voltage measured for present gain setting.
;
;               V(N-1) = Voltage for previous gain setting.
;
;               G(N) = Present gain setting.
;
;               G(N-1) = Previous gain setting.
;
;               G(N+1) = Gain to be predicted.
;
;               MU = Nominal voltage level.
;
;               NU = Dampening factor. Currently NU = (0.1)*MU
;
;               ABS = Signifies absolute value.
```

Definitions:

$$Alpha(n) = \langle V(N)-MU \rangle / ABS\langle V(N)-MU \rangle$$

$$S(n) = ABS(\langle G(N)-G(N-1) \rangle * \langle V(N)-MU \rangle / \langle V(N)-V(N-1) \rangle)$$

$$G(N+1) = G(N) - Alpha(n)*S(n)$$

Combining and multiplying, we get

$$G(N+1) = G(N) - \langle V(N)-MU \rangle * ABS(\langle G(N)-G(N-1) \rangle / \langle V(N)-V(N-1) \rangle)$$

Constraints:

a) $ABS(V(N)-V(N-1)) > NU = (0.1)*MU$
b) $ABS(G(N)-G(N-1)) > 0.0$
c) $33. > ABS(G(N+1)-G(N)) > 0.0$

This formula is used to predict the new gain for each subsequent pass. A total of 40 passes are made through the subroutine, with the first eight just used for setting the gains near optimum in noisy situations, and the remaining 32 are averaged in range and smoothed in time.

Notes:

The mean value finally used is taken for each individual range cell, but in addition to this average, at the end of each pass a three point average across range cells is done to prevent the 'picket fence' effect from occuring.(This happens because of skewing of time sampling vs. where gain changes actually take effect in the receiver).

The dampening factor, NU, is used to prevent the denominator in the equation from getting too small and thereby throwing the calculations into oscillation. In addition, maximum gain change is bounded above by 32. to also prevent oscillation.

We use a flat gain value in the first eight ranges, in this case the gain value for range nine. This is due to the fact that the blanking in the receiver before, during, and after the transmit pulse causes non-linearities in voltage levels returned for various gain settings during this interval. Tests have shown that this 'flat' value is much more consistant in gain setting algorithms than other techniques(such as ramping up from a zero value in range cell one). Likewise, the same method is used in range cells 63 and 64.

```
;
;       Local symbol definitions:
;
F0=%0                           ; Floating point reg. defns.
F1=%1                           ;
;
;       Local data blocks:
;

        .EVEN
DLGN:   .WORD   1.              ; Storage for <DELTA GAIN>
MU:     .WORD   14000           ; Nominal voltage level
NU:     .WORD   614.            ; Nu = (0.1)*Nominal voltage.
VDIFF:  .WORD   1.              ; Storage for Vmeas(N)-Vmeas(N-1)
UDIFF:  .WORD   1.              ; Storage for Vmeas-Mu.
VOLD:   .BLKW   64.             ; For last voltages.
        .PAGE

        .SBTTL  Subroutine GAIN entry point.
;
GAIN::  JSR     PC,$SAVAL       ; Save register status.
        CLR     R4              ; Clear index register.
        CMP     #1,PCNT         ; Is this the first pass?
        BNE     LOOP            ; NE=NO, so to second pass

        .SBTTL  Initial pass through GAIN routine.

        MOV     #64.,R0         ; Clear out scratch area for summation
        MOV     #SLOP,R1        ; of gains(for computation of mean).
1$:     CLR     (R1)+           ;
        MOV     GAINS(R4),GN(R4) ; Move gain to GN
        MOV     VMEAS(R4),VOLD(R4) ; Save voltage.
        ADD     #2,R4           ; Increment array pointer
        SOB     R0,1$           ;


        .SBTTL  Code for second and subsequent passes through GAIN routine.

;       Is gain at maximum and voltage still too low?

        MOV     #16.,R4
LOOP:   CMP     #255.,GAINS(R4) ; Check gain value
        BNE     L.2             ; If not equal to 255, continue
        CMP     VMEAS(R4),MU    ;
        BGT     L.2             ; Measured voltage is too big.
        JMP     ENDL            ; Else gain set, do next range

;       Compute difference in gains.

L.2:    MOV     GAINS(R4),DLGN  ;
        SUB     GN(R4),DLGN     ; DLGN=GAIN(N)-GAIN(N-1)
        TST     DLGN            ; Take absolute value.
        BPL     L.3             ;
        NEG     DLGN            ;

;       Compute UDIFF=V(N)-Mu.

L.3:    MOV     VMEAS(R4),UDIFF ;
        SUB     MU,UDIFF        ;
```

```
;       Compute difference in voltages.                                    (12)

        MOV     VMEAS(R4),VDIFF ;
        SUB     VOLD(R4),VDIFF  ; VDIFF=V(N)-V(N-1)
        TST     VDIFF           ; Take absolute value
        BPL     L.4             ;
        NEG     VDIFF           ;
L.4:    CMP     NU,VDIFF        ; Nu= (0.1)*Nomnl Nomnl=10000(8)
        BLE     L.5             ;
        MOV     NU,VDIFF        ;

;       Compute new S(n) =  ABS(<G(N)-G(N-1)>*<V(N)-MU> / <V(N)-V(N-1)>)

L.5:    SETF                    ;
        SETI                    ;
        LDCIF   DLGN,F0         ; F0 = DLGN(=GN(N)-GN(N-1)
        LDCIF   VDIFF,F1        ; F1 = VDIFF(=V(N)-V(N-1))
        DIVF    F1,F0           ; F0 = F0/F1 = DLGN/VDIFF
        LDCIF   UDIFF,F1        ; F1 = UDIFF(=V(N)-MU)
        MULF    F1,F0           ; F0 = UDIFF*(DLGN/VDIFF)
        STCFI   F0,DLGN         ;

;       Make sure ABS(DLGN) < 32.

        TST     DLGN            ;
        BGT     2$              ; Br if DLGN > 0.
        CMP     DLGN,#-32.      ; Is DLGN < -32 ?
        BPL     3$              ; No, continue.
        MOV     #-32.,DLGN      ; Yes, set to -32.
        BR      3$              ;
2$:     CMP     DLGN,#32.       ; Is DLGN > 32 ?
        BMI     3$              ; No, continue.
        MOV     #32.,DLGN       ; Else, set DLGN = 32.

;       Make sure DLGN is not zero.

3$:     TST     DLGN            ; DLGN must be non-zero.
        BNE     L.6             ;
        INC     DLGN            ; Set it to 1.
        TST     UDIFF           ; Make sure it's in correct direction.
        BPL     L.6             ;
        NEG     DLGN            ;

;       Compute new gain value.

L.6:    MOV     GAINS(R4),GN(R4) ; Save gain for next pass.
        SUB     DLGN,GAINS(R4)  ; New Gain = Old gain - DLGN
ENDL:   MOV     VMEAS(R4),VOLD(R4) ; Save voltage for next pass.
        ADD     #2,R4           ; Increment array pointer
        CMP     #122.,R4        ; End of loop?
        BLT     DONE            ; LT=YES, so set new voltages
        JMP     LOOP            ; GE=NO, do next range cell
```

```
;
;
;       Done with this pass through all 64 range cells, so clean
;       up gains and check for last pass through subroutine.
;
DONE:   MOV     #GAINS,R0       ; Put plateau on beginning and end of array.
        ADD     #20,R0          ; Move range cell 9 into inner cells.
        MOV     #8.,R1          ; Need 8 moves.

4$:     MOV     (R0),-(R0)      ;
        SOB     R1,4$           ;

        MOV     #122.,R0        ; Move range cell 62 into 63 and 64.
        ADD     #GAINS,R0       ;
        MOV     (R0),2(R0)      ;
        MOV     2(R0),4(R0)     ;

        JSR     PC,AVE3         ; Do three point average
        JSR     PC,MXMIN        ; Assure  0 < gain < 256.
        CMP     #8.,PCNT        ; Finished with first 8 approximation passes?
        BGE     D.2             ; GE=NO, don't start sumation yet.

        MOV     #64.,R0         ; Sum up gain values for statistical smoothing
        MOV     #GAINS,R1       ;
        MOV     #SLOP,R2        ;
1$:     ADD     (R1)+,(R2)+     ; Store in SLOP scratch array.
        SOB     R0,1$           ;

D.2:    CMP     #40.,PCNT       ; Is pass counter=40.?
        BLE     D.1             ; GT=NO, set new voltages
        CLC                     ; Clear carry, not acceptable
        RTS     PC
;
;       Have completed the correct number of passes, so set mean
;       gain value for each range, put them into gain page array,
;       and signify to main task that gains are set.
;
D.1:    MOV     #64.,R5         ; Take mean of computed gain values.
        MOV     #SLOP,R4        ; Gains have been summed into this array.
        MOV     #GAINS,R2       ; Address of output array.
2$:     MOV     (R4)+,R1        ; Move each summed gain value into R1.
        ASH     #-5,R1          ; Divide by 32.
        MOV     R1,(R2)+        ; Save in array GAINS.
        SOB     R5,2$           ; For all 64 range cells.
        JSR     PC,MXMIN        ; Assure 0 < GAINS < 256.
        SEC                     ; Carry set, gains adjusted
        RTS     PC              ;
```

```
;
        .SBTTL  Routine for three point linear average
;
;       Title:  Subroutine AVE3
;

;
;       Function:       Does linear 3-point average of array GAINS.
;

;
;       Inputs: Array 'GAINS'
;
;       Outputs: Array 'GAINS'
;

;
;       Local Data Blocks:
;
GAVE:   .BLKW   64.             ; For temporary storage while averaging.
;

;
;       This routine takes a three point average of gains and deposits
;       the results into a second scratch array(to give genuine average).
;       Upon completion, the values in the scratch array are moved back
;       into the actual GAINS array for use by the receiver.
;
AVE3:   JSR     PC,$SAVAL
        MOV     #10.,R4         ; Start at range cell 6.
        MOV     #122.,R5        ; Stop at range cell 62.
AV.1:   MOV     GAINS(R4),R1    ; Get left value
        ADD     #2,R4           ;
        ADD     GAINS(R4),R1    ; Get middle value
        ADD     #2,R4           ;
        ADD     GAINS(R4),R1    ; Get right value
        ADD     #1,R1           ; For round-off.
        CLR     R0
        DIV     #3,R0           ; Divide by three
        SUB     #2,R4           ;
        MOV     R0,GAVE(R4)     ; Store at middle in output array
        CMP     R5,R4
        BGE     AV.1            ; GE=NO, continue
        MOV     #126.,R0        ; Three point average stores results in
MO.1:   TST     GAVE(R0)        ; GAVE, so move GAVE to GAINS
        BEQ     MO.2            ;
        MOV     GAVE(R0),GAINS(R0) ;
MO.2:   SOB     R0,MO.1         ;
        RTS     PC              ;
```

```
;                                                                    (15)
        .SBTTL  Routine for limiting range of gain values
;
;       Title:  Subroutine MXMIN
;

;
;       Function:       This routine limits the values put into the GAINS array.
;

;
;       Input: GAINS
;
;       Output: GAINS   Note: 0 < GAINS < 256
;
;       Gain returned from this routine must be greater than 0 in
;       order to prevent non-linearities caused by blanking in the
;       receiver for gain values of zero.
;

MXMIN:  MOV     R0,-(SP)        ;
        MOV     R1,-(SP)        ;
        MOV     #GAINS,R0       ;
        MOV     #64.,R1         ;
M.0:    CMP     #255.,(R0)      ; Greater than maximum allowable gain?
        BGE     M.1             ; If not, store new gain
        MOV     #255.,(R0)      ; If greater, set to maximum
M.1:    TST     (R0)            ; Check for positive gain value
        BGT     M.2             ;
        MOV     #1.,(R0)        ; If negative or zero, set to minimum
M.2:    TST     (R0)+           ;
        SOB     R1,M.0          ;
        MOV     (SP)+,R1        ;
        MOV     (SP)+,R0        ;
        RTS     PC              ;
        .END    START
```