ISSN 2039-7941
Anno 2018_Numero 395

rapporti tecnici INGV

Generatore di segnale *IRIG-B* per la sincronizzazione di strumentazione scientifica attraverso l'uso del *Network Time Protocol* (*NTP*)

395

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - Referente
Rossella Celi
Tel. +39 06 51860068
redazionecen@ingv.it

in collaborazione con:
Barbara Angioni (RM1)

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

Rappresentante legale: Carlo DOGLIONI
Sede: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2018_Numero 395

rapporti tecnici

INGV

GENERATORE DI SEGNALE *IRIG-B* PER LA SINCRONIZZAZIONE DI STRUMENTAZIONE SCIENTIFICA ATTRAVERSO L'USO DEL *NETWORK TIME PROTOCOL* (*NTP*)

Antonino Sicali, Pasqualino Cappuccio, Alfio Amantia

INGV (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Catania - Osservatorio Etneo)

395

Come citare: Sicali A., Cappuccio P., Amantia A., (2018). Generatore di segnale *IRIG-B* per la sincronizzazione di strumentazione scientifica attraverso l'uso del *Network Time Protocol* (*NTP*). Rapp. Tec. INGV, 395: 1-26.

Indice

Introduzione

Ogni sistema di acquisizione ha la necessità di sincronizzare le misure prodotte allo scopo di facilitarne l'analisi [Sicali et al, 2016/B]. La sincronizzazione solitamente avviene attraverso l'uso di sistemi *GNSS* (*Global Navigation Satellite System*) e nel loro rappresentante più conosciuto il *GPS*. Seppur quest'ultimo, risulti essere il più versatile e universale, ciò nonostante possiede alcune limitazioni che a volte ne impediscono l'utilizzo. I problemi maggiori derivano dall'antenna utilizzata per captare il segnale radio, vero punto debole di tutto il sistema. In luoghi molto ostili potrebbe essere difficile utilizzare e mantenere un'appendice esterna come l'antenna *GPS*, a causa soprattutto di scariche da fulmine. Diversamente, potrebbe essere più semplice mantenere una trasmissione su fibra ottica o cavo in rame, utilizzando sistemi già presenti, collaudati e più sicuri, eventualmente mantenuti e pagati da terzi. Il sistema proposto è consigliato soprattutto per quei sistemi, che non possono gestire direttamente l'*NTP* (*Network Time Protocol*) [RFC 5905, 2010] e hanno la necessità di un segnale *hardware* esterno di tipo IRIG-B [IRIG-B, 2004].

1. Il protocollo *NTP*

Oggi, molte applicazioni, come la sincronizzazione temporale, sono basate interamente sul *GPS* che è largamente utilizzato per le sue caratteristiche di stabilità e diffusione. Ovviamente anche il *GPS* potrebbe avere problemi di blackout temporanei, dovuti soprattutto a un segnale scadente, causato dalla presenza di alberi, dal maltempo o in generale dalle caratteristiche morfologiche del sito.

Quando si installa una stazione di monitoraggio in un luogo aperto e poco protetto dal punto di vista delle fulminazioni, un particolare non trascurabile è che il *GPS* ha bisogno di una sua antenna per funzionare. Quando il sistema ha già appendici esterne, inserirne un'ulteriore non cambia molto la situazione. Tuttavia, se il sistema non possiede già appendici esterne, l'inserimento di un'antenna potrebbe esporre alla fulminazione, come è successo per alcune installazioni sul M.te Etna. A tutela del buon funzionamento dei sistemi e come protezione dalle extracorrenti da fulmine possono essere utilizzati scaricatori di tipo I e II. Tali scaricatori però permettono solo di ridurre i problemi derivati da una scarica e non risolvono completamente il problema. Inoltre gli scaricatori hanno bisogno di una regolare manutenzione che non sempre può essere eseguita a causa soprattutto dell'inaccessibilità del sito durante buona parte dell'anno. Molte volte si riesce a garantire e mantenere un collegamento telemetrico ottimale, nonostante il difficile accesso di alcuni siti, anche grazie all'utilizzo di canali di comunicazione protetti. In questi luoghi, in cui la trasmissione non avviene via etere, ma attraverso un collegamento diretto in fibra ottica o cavo, magari mantenuti da terzi, conviene utilizzare un sistema alternativo al *GPS.* Quando le precisioni da raggiungere non sono particolarmente elevate e il tempo di acquisizione è maggiore di un secondo, si possono utilizzare per sincronizzare le misure i server *NTP* presenti su Internet. Con le tecnologie attuali si possono ottenere precisioni anche di 100 ms. Con il progredire continuo delle tecnologie di rete si riusciranno a raggiungere i 2^{-32} secondi teorici previsti dalle specifiche del protocollo *NTP*. Esiste inoltre una evoluzione del protocollo *NTP,* il *PTP,* la cui precisione reale su linea locale (*LAN*) può raggiungere attualmente anche i μs. Quest'ultima scelta può risultare utile per quei luoghi in cui esiste già un server locale che fa uso di *GPS* dovutamente protetto e la precisione richiesta nell'acquisizione è elevata.

2. Generatore di segnale *IRIG-B*

Il generatore proposto è composto da una parte *software* e da una *hardware*. La parte *software* richiede un sistema operativo *Unix-Like* (nello specifico può essere *Linux)* per funzionare e non deve necessariamente essere eseguita su un *personal computer*. Può andar bene qualsiasi dispositivo *hardware* che possa eseguire il software *NTP*, sia dotato di un orologio interno, di un collegamento di rete e di una scheda audio. La parte *hardware* (convertitore) è stata progettata per essere semplice e facilmente realizzabile in quanto usa componenti molto comuni [Sicali et al, 2016/A]. Per velocizzare la progettazione, i test e l'utilizzo a regime è stato utilizzato un normale *PC* con *Linux. Per* le installazioni *standalone* a energia solare si consiglia di usare un sistema a basso consumo e a *range* di temperatura esteso [Sicali et al., 2016/A].

2.1 Software

Dal sito dell'*NTP Project (R&D)* (www.ntp.org) si può prelevare il sorgente di un *software* chiamato *TG2*. Tale utility può generare, tra le altre cose, anche un segnale di tipo IRIG-B attraverso l'uso della scheda

audio di sistema. Normalmente l'*utility TG2* viene usata in modalità utente. Per le applicazioni geofisiche è importante che il *software* sia autosufficiente [Sicali et al., 2016/A], per questo l'utility è stata modifica affinché il funzionamento divenisse completamente automatico (*Unix daemon)*. Nella figura 1 sono illustrate, attraverso un diagramma di flusso, le modifiche apportate a *TG2* per utilizzarla come *daemon*. Il software in avvio controlla se è stata selezionata la nuova opzione introdotta '-e' questo permette di avviare il programma come *daemon*. Se è stata selezionata l'opzione viene passato il controllo alla funzione *daemon* altrimenti l'esecuzione continua normalmente. Se la modalità *daemon* viene richiesta la funzione omonima esegue un *fork* del processo. Successivamente il vecchio processo viene terminato mentre il nuovo continua la sua esecuzione. Infine i tre descrittori di file *standard* (*stdin*, *stdout* e *stderr*) vengono rilasciati e successivamente sostituiti da un unico descrittore che permette di effettuare un minimo di *logging* sull'esecuzione. Il *file* di *logging* viene creato nella *directory* di esecuzione del programma indicata dalla definizione *HOME_DIRECTORY*. Per una descrizione del software originale *TG2* e del suo utilizzo si può far riferimento alla documentazione del *NTP Project (R&D)* raggiungibile dalla home page del progetto [NTP Project (R&D)].

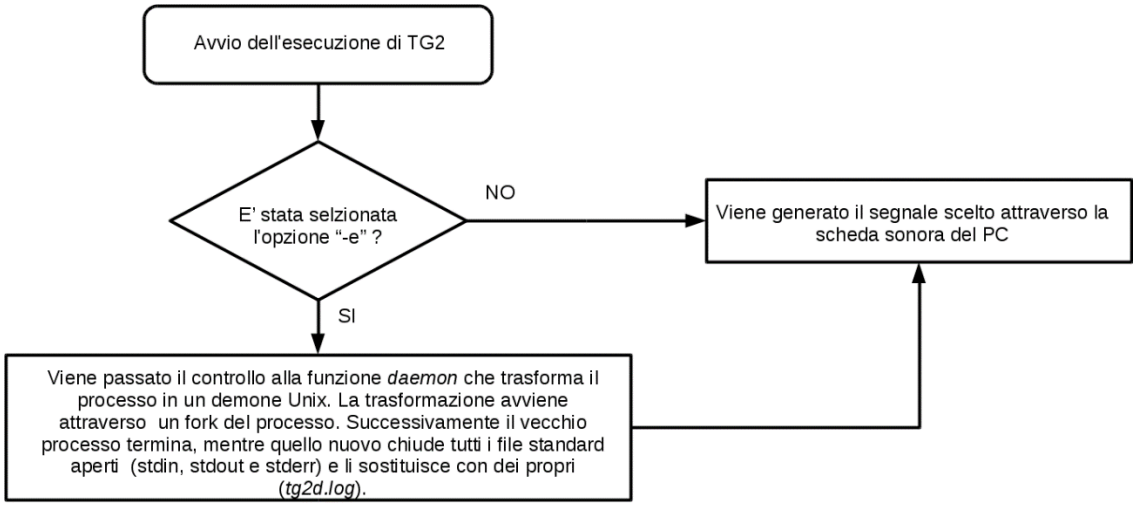

Figura 1. Schema a blocchi del software.

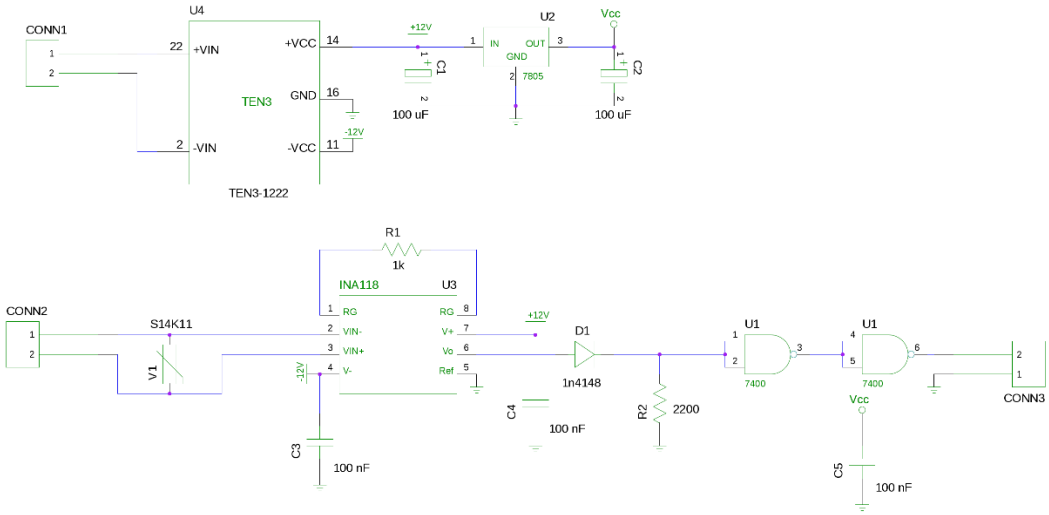

Figura 2. Schema elettrico del convertitore *IRIG-B TTL*.

2.2 Hardware

I segnali *IRIG-B* possono essere modulati attraverso una portante (*IRIG-B B12x*) oppure no (*IRIG-B B00x*). Il segnale generato da *TG2* attraverso la scheda audio del computer può essere di entrambi i tipi. Attraverso lo script riportato in appendice A1 (definizione *DAEMON_ARGS*) viene generato un segnale non modulato *IEEE-1344* (opzione -f 5) poiché lo strumento richiedeva quel tipo di standard. La scheda audio essendo un dispositivo analogico ha qualche difficoltà a produrre un segnale digitale. Il segnale risultante, riportato in figura 4 in blu, è molto sporco e potrebbe creare problemi. Per evitare tali problemi si è progettato il circuito riportato nella figura 2 che provvede a convertire il segnale analogico in uscita alla scheda audio in treno d'impulsi con livelli logici *TTL*. Il circuito proposto è composto da diverse parti, alcune di queste risultano essere alquanto importanti per la protezione della strumentazione dalle extra tensioni. Come si può osservare dallo schema elettrico riportato nella figura 3 esistono due alimentatori posti in cascata. Il primo alimentatore oltre ad assicurare un isolamento galvanico tra il circuito e la linea di alimentazione principale, permette di generare la tensione duale necessaria al funzionamento del differenziale per strumentazione in ingresso, un *INA118P*. Il secondo alimentatore stabilizza la tensione a 5 v per alimentare l'integrato digitale, in logica *TTL*, *7400*. Il segnale prodotto dalla scheda audio (traccia blu nella figura 4) viene squadrato grazie al guadagno elevato dell'amplificatore d'ingresso (figura 2). Il Segnale d'uscita viene ripulito dalla sua parte negativa e limitato in quella positiva dall'integrato digitale *7400*. Il *7400* può essere sostituito da qualsiasi circuito integrato della stessa serie. Lo si è scelto solo perché molto comune e presente in qualsiasi laboratorio o negozio di elettronica [Sicali et al., 2016/A]. La linea audio (*CONN2* dello schema riportato nella figura 2) viene protetta attraverso un *varistore* per evitare che eventuali transienti di tensione, proveniente dal PC e quindi dalla linea elettrica possano arrecare problemi alla strumentazione. Utilizzando il circuito su alcuni strumenti si è delineata la necessità di proteggere anche la linea di uscita *TTL* (*CONN3*) da transitori di tensione derivati dalla strumentazione non perfettamente protetta. In passato, in assenza della protezione della linea di uscita TTL, lo strumento ha provocato il suo stesso blocco. Questo perché un transitorio provocato dallo strumento ha danneggiato l'integrato *7400* e lo strumento non ha ricevuto il segnale di sincronizzazione *IRIG-B*, vitale per l'acquisizione.

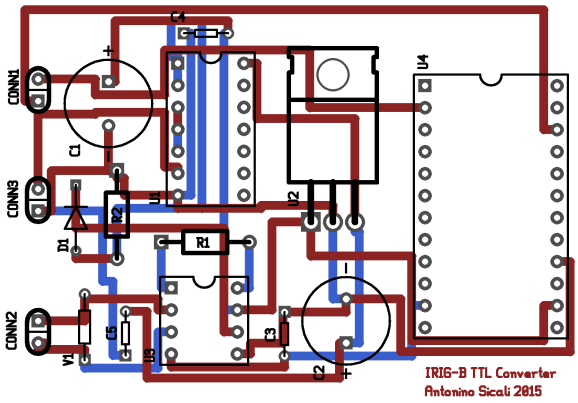

Figura 3. Esempio di *PCB* doppia faccia per il convertitore *IRIG-B TTL*.

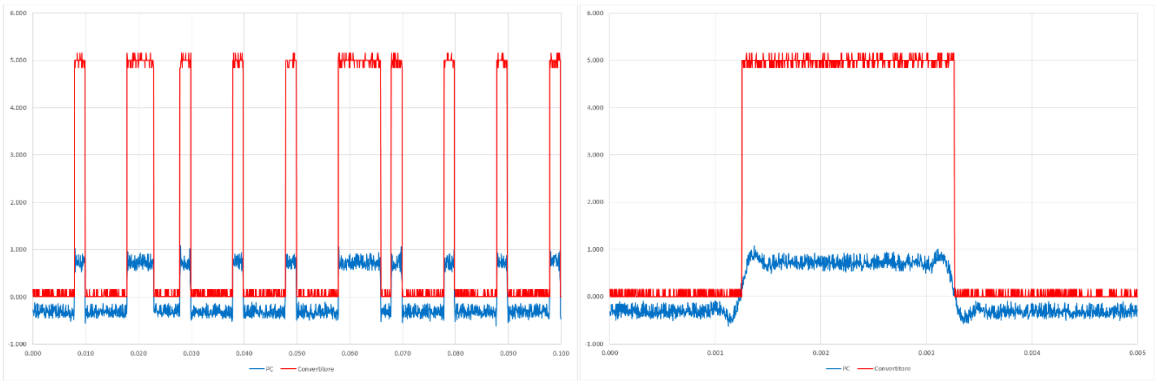

Figura 4. Esempio di segnali prodotti dal *PC* (blu) e corretto dal convertitore (rosso). Nel grafico a sinistra viene mostrato una porzione del treno d'impulsi. Nel grafico a destra viene mostrato un unico impulso. Si può notare la bipolarità della tensione e lo *slope* molto accentuato.

3. Installazione

3.1 Software

Dal sito www.ntp.org bisogna scaricare l'ultima versione del software. Al tempo dell'implementazione è stata usata la versione *4.2.8p4*. Nell'appendice A2 è riportata la patch da applicare al sorgente di *TG2* per ottenere la versione *daemon*. Per applicare la *patch* si può utilizzare il comando omonimo secondo la sintassi:

patch < tg2.patch

Successivamente all'applicazione della patch si possono eseguire i passi standard di compilazione previsti dal *NTP*, ovvero il *configure* seguito dal *make*. Alla fine bisognerà eseguire il *makefile* personalizzato riportato nell'appendice A3 attraverso la riga di comando:

make –f Makefile.tg2d

Verrà creato l'eseguibile che potrà essere usato dallo script riportato nell'appendice A1. Il software per funzionare ha bisogno che il sistema audio funzioni correttamente e che sia presente il file */dev/sound.* Se non è presente tale file si dovrà installare, poiché mancante, un modulo nel *kernel* utilizzando il comando:

modprobe snd-pcm-oss

L'*utility TG2* per generare il segnale *IRIG-B* leggerà l'orario dal computer locale che dovrà essere sincronizzato attraverso l'uso del software *NTPD*. Alla configurazione che solitamente si trova nel file */etc/ntp.conf* sarebbe opportuno inserire i due server *NTP* dell'Istituito Nazionale di Ricerca Metrologica [INRiM NTP] attraverso le righe:

server ntp1.inrim.it dynamic
server ntp2.inrim.it dynamic

Lo script riportato nell'appendice A1 verrà copiato durante l'esecuzione del *makefile* nella *directory /etc/init.d. Successivamente* collegato in due o più *directory* del tipo */etc/rcx.d* (per esempio *rc5.d e rc0.d*) per essere eseguito all'avvio e fermato durante lo *shutdown*. Potrà essere avviato, per esempio, come *S99* e fermato come *K00*. Lo script dell'appendice A1 potrebbe non essere compatibile con il sistema, per questo si può modificare il file standard */etc/init.d/skeleton. A* tal proposito bisognerà inserire soltanto il nome *TG2D* e le opzioni "-e –f 5" di esecuzione.

3.2 Hardware

L'installazione dell'hardware non richiede particolari accorgimenti e può essere accoppiato al resto dell'elettronica di sistema. La situazione cambia qualora l'installazione dell'hardware avviene in luoghi molto sensibili alle scariche elettrostatiche. Si deve prevedere per questi casi un inscatolamento in metallo (figura 5). La scatola esterna in metallo dovrebbe essere messa a terra come tutti i terminali degli scaricatori di linea per la protezione dalle extra tensioni. Nella figura 6 è riportato uno schema molto semplificato di come dovrebbero essere collegati i dispositivi di protezione. In tale schema vengono utilizzati due tipi di dispositivi di protezione: i *GDT* (*Gas Discharge Tubes*) e i *MOV* (*Metal Oxide varistore*). Lo schema di figura 6 è molto generale e non può soddisfare in pieno tutte le problematiche scaturite dalla protezione dalle extra tensioni. La progettazione di uno schema e l'utilizzo di un dispositivo rispetto a un altro dipende molto dal tipo di applicazione e dalle potenze in gioco. Per l'esposizione molto limitata del convertitore non è stato necessario progettare nulla di simile, sono stati sufficienti solo dei *MOV* di linea (figura 2). Se dovesse rendersi necessario inserire delle protezioni, poiché il circuito è molto esposto a sovratensioni, si possono eventualmente consultare i diversi testi che le varie case costruttrici di dispositivi di protezione mettono a disposizione del progettista. Da questi testi si possono ricavare schemi elettrici, consultare *application notes* e visionare risultati di prove condotte in laboratorio.

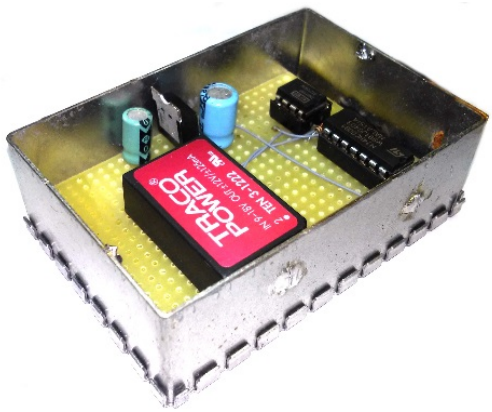

Figura 5. Esempio di alloggiamento per prevenire problemi di scariche elettrostatiche che potrebbero danneggiare la strumentazione e lo stesso convertitore.

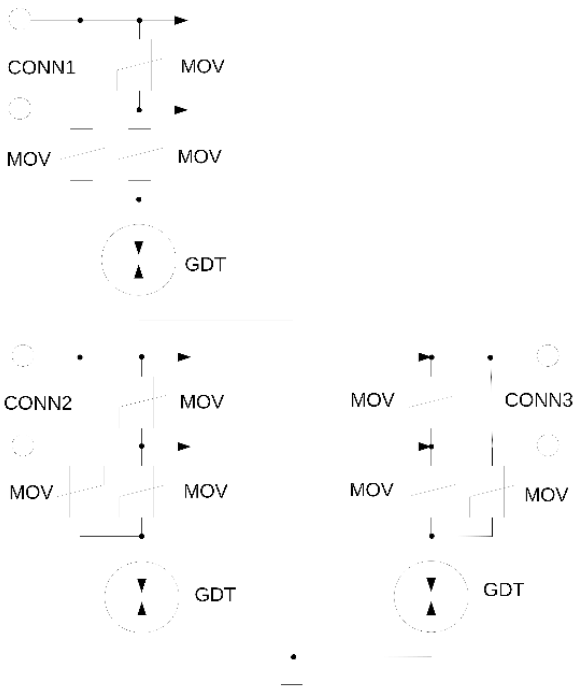

Figura 6. Schema elettrico tipo e molto semplificato utilizzante dispositivi di protezione per le extra tensioni (SPD).

Conclusioni

Per ogni tipo di applicazione sicuramente esiste una tecnologia che ne permette la messa in opera e lo sviluppo ottimale. La tecnologia dei *GPS* è perfetta per sincronizzare le misure prodotte da un sistema di acquisizione remota. Per quei casi in cui si può scendere ad un compromesso sulla frequenza di acquisizione, il sistema hardware/software proposto è una buona alternativa per salvaguardare la strumentazioni da danneggiamenti provocati dall'esposizione di antenne *GPS* a scariche atmosferiche.

Ringraziamenti

Volevamo ringraziare tutta la Segreteria di Redazione del CEN che si è dimostrata, ancora una volta, molto veloce ed efficiente. Un ringraziamento particolare è dovuto, alla dott.ssa Rossella Celi per la cordialità, la disponibilità e la professionalità ancora una volta dimostrata. Ringraziamo infine il Dott. Stefano Cacciaguerra per l'accuratezza, la precisione e la professionalità adottata durante la revisione.

Bibliografia

Sicali A., Amantia A., Cappuccio P., (2016). *Linee guida e criticità nella progettazione di sistemi per l'acquisizione di dati geofisici in prossimità di vulcani attivi.* Rapporti Tecnici INGV n°. 347, ISSN 2039-7941.

Sicali A., Cappuccio P., Amantia A., (2016). *Software per la sincronizzazione dei sistemi di acquisizione attraverso l'uso del GPS: implementazione hardware per l'architettura PC/104.* Rapporti Tecnici INGV n°. 356, ISSN 2039-7941.

Mills D., (2010). *Request for Comments 5905*, Network Working Group, University of Delaware ISSN 2070-1721.

Timing Committee, Telecommunications and timing group, Range Commanders Council (2004), *IRIG SERIAL TIME CODE FORMATS, IRIG STANDARD 200-04*. Secretariat, Range Commanders Council U.S. Army White Sands Missile Range, New Mexico 88002-5110.

Sitografia

NTP Project (R&D), http://www.ntp.org
INRiM NTP, http://rime.inrim.it/labtf/ntp/

Appendice A1. TG2.SH, init.d daemon script

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          tg2d
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO

# Do NOT "set -e"

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="WWV or IRIG signals generator"
NAME=tg2d
DAEMON=/usr/sbin/$NAME
DAEMON_ARGS="-e -f 5"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is present
# and status_of_proc is working.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   2 if daemon could not be started
        start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --
test > /dev/null \
                || return 1
        start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON -- \
                $DAEMON_ARGS \
                || return 2
        # Add code here, if necessary, that waits for the process to be ready
        # to handle requests from services started subsequently which depend
        # on this one.  As a last resort, sleep for some time.
}

#
# Function that stops the daemon/service
```

```
#
do_stop()
{
        # Return
        #   0 if daemon has been stopped
        #   1 if daemon was already stopped
        #   2 if daemon could not be stopped
        #   other if a failure occurred
        start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile
$PIDFILE --name $NAME
        RETVAL="$?"
        [ "$RETVAL" = 2 ] && return 2
        # Wait for children to finish too if this is a daemon that forks
        # and if the daemon is only ever run from this initscript.
        # If the above conditions are not satisfied then add some other code
        # that waits for the process to drop all resources that could be
        # needed by services started subsequently.  A last resort is to
        # sleep for some time.
        start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec
$DAEMON
        [ "$?" = 2 ] && return 2
        # Many daemons don't delete their pidfiles when they exit.
        rm -f $PIDFILE
        return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
        #
        # If the daemon can reload its configuration without
        # restarting (for example, when it is sent a SIGHUP),
        # then implement that here.
        #
        start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name
$NAME
        return 0
}

case "$1" in
  start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  status)
        status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
        ;;
  #reload|force-reload)
        #
```

```
        # If do_reload() is not implemented then leave this commented out
        # and leave 'force-reload' as an alias for 'restart'.
        #
        #log_daemon_msg "Reloading $DESC" "$NAME"
        #do_reload
        #log_end_msg $?
        #;;
  restart|force-reload)
        #
        # If the "reload" option is implemented then remove the
        # 'force-reload' alias
        #
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
          *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
  *)
        #echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
        echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
        exit 3
        ;;
esac

:
```

Appendice A2. Modifiche (tg2.patch) da applicare a TG2.C (prelevabile dal sito www.ntp.org)

```
--- tg2.c       2014-12-19 12:56:52.000000000 +0100
+++ new_tg2.c   2015-12-03 09:38:31.699413400 +0100
@@ -235,6 +235,11 @@
 #define        ISSUE           (23)
 #define        ISSUE_DATE      "2007-02-12"

+#define PID_FILE "/var/run/tg2d.pid"
+#define HOME_DIRECTORY "/var/tg2d"
+void startDaemon(void);
+void stopDaemon(void);
+
 #define        SECOND  (8000)                  /* one second of 125-us samples */
 #define BUFLNG (400)                   /* buffer size */
 #define        DEVICE  "/dev/audio"    /* default audio device */
@@ -538,6 +543,49 @@


 /*
+ * Start Daemon
+ */
+
+void startDaemon(void)
+{
+    pid_t pid;
+
+    pid = fork ();
+    if (pid == -1) exit(EXIT_FAILURE);
+        else if (pid != 0)
+            exit (EXIT_SUCCESS);
+
+    if (setsid ( ) == -1) exit(EXIT_FAILURE);
+
+    if (chdir(HOME_DIRECTORY) == -1) return exit(EXIT_FAILURE);
+
+    close(STDIN_FILENO);
+    close(STDOUT_FILENO);
+    close(STDERR_FILENO);
+
+    char filename[256];
+    sprintf(filename,"%s/tg2d.log",HOME_DIRECTORY);
+    open("/dev/null",O_RDWR);
+    open(filename,O_RDWR);
+    open(filename,O_RDWR);
+
+    FILE *handle=fopen(PID_FILE,"wt+");
+    fprintf(handle,"%d\n",getpid());
+    fclose(handle);
+
+    setvbuf(stdout, NULL, _IONBF, 0);
+}
+
+/*
+ * Stop daemon
+ */
+
+void stopDaemon(void)
+{
+    unlink(PID_FILE);
+}
+
```

```
+/*
  * Main program
  */
 int
@@ -550,7 +598,7 @@
        audio_info_t info;     /* Sun audio structure */
        int     rval;           /* For IOCTL calls */
 #endif
-
+       int                     DaemonRunning=0;               /* Running as
daemon */
        struct  timeval  TimeValue;                            /* System clock at
startup */
        time_t                  SecondsPartOfTime;             /* Sent to gmtime()
for calculation of TimeStructure (can apply offset). */
        time_t                  BaseRealTime;                  /* Base realtime so
can determine seconds since starting. */
@@ -663,17 +711,17 @@

        /* String to allow us to put out reversed data - so can read the binary
numbers. */
        char    OutputDataString[OUTPUT_DATA_STRING_LENGTH];
-
+
        /* Number of seconds to send before exiting.  Default = 0 = forever. */
        int             SecondsToSend = 0;
        int             CountOfSecondsSent = 0;        /* Counter of seconds */
-
+
        /* Flags to indicate whether to add or remove a cycle for time
adjustment. */
        int             AddCycle = FALSE;              // We are ahead, add cycle
to slow down and get back in sync.
        int             RemoveCycle = FALSE;   // We are behind, remove cycle to
slow down and get back in sync.
        int             RateCorrection;                // Aggregate flag for
passing to subroutines.
        int             EnableRateCorrection = TRUE;
-
+
        float   RatioError;


@@ -691,12 +739,13 @@
        strlcpy(device, DEVICE, sizeof(device));
        Year = 0;
        SetSampleRate = SECOND;
-
+
 #if    HAVE_SYS_SOUNDCARD_H
-       while ((temp = getopt(argc, argv,
"a:b:c:df:g:hHi:jk:l:o:q:r:stu:xy:z?")) != -1) {
+       while ((temp = getopt(argc, argv,
"a:b:c:def:g:hHi:jk:l:o:q:r:stu:xy:z?")) != -1) {
 #else
-       while ((temp = getopt(argc, argv,
"a:b:c:df:g:hHi:jk:l:o:q:r:stu:v:xy:z?")) != -1) {
+       while ((temp = getopt(argc, argv,
"a:b:c:def:g:hHi:jk:l:o:q:r:stu:v:xy:z?")) != -1) {
 #endif
+                printf("<%c>",temp);
                switch (temp) {
```

```
                case 'a':       /* specify audio device (/dev/audio) */
@@ -709,7 +758,7 @@
                        InsertLeapSecond = FALSE;
                        DeleteLeapSecond = TRUE;
                        break;
-
+
                case 'c':       /* specify number of seconds to send output for
before exiting, 0 = forever */
                        sscanf(optarg, "%d", &SecondsToSend);
                        break;
@@ -741,7 +790,7 @@
                        InsertLeapSecond = TRUE;
                        DeleteLeapSecond = FALSE;
                        break;
-
+
                case 'j':
                        EnableRateCorrection = FALSE;
                        break;
@@ -753,7 +802,7 @@
                                {
                                RemoveCycle = TRUE;
                                AddCycle = FALSE;
-
+
                                if   (Verbose)
                                        printf ("\n> Forcing rate correction
removal of cycle...\n");
                                }
@@ -763,7 +812,7 @@
                                        {
                                        RemoveCycle = FALSE;
                                        AddCycle = TRUE;
-
+
                                        if   (Verbose)
                                                printf ("\n> Forcing rate
correction addition of cycle...\n");
                                        }
@@ -859,7 +908,9 @@
                case 'z':       /* Turn on Debug output (also turns on Verbose
below) */
                        Debug = TRUE;
                        break;
-
+                case 'e':       /* Run as daemon */
+                        DaemonRunning=1;
+                        break;
                default:
                        printf("Invalid option \"%c\", aborting...\n", temp);
                        exit (-1);
@@ -1052,6 +1103,9 @@
         * Unless specified otherwise, read the system clock and
         * initialize the time.
         */
+
+       if (DaemonRunning) startDaemon();
+
        gettimeofday(&TimeValue, NULL);                 // Now always read the
system time to keep "real time" of operation.
```

```
        NowRealTime = BaseRealTime = SecondsPartOfTime = TimeValue.tv_sec;
        SecondsRunningSimulationTime = 0;      // Just starting simulation,
running zero seconds as of now.
@@ -1101,7 +1155,7 @@
                    Year / 10, DayOfYear, Hour, Minute, Year % 10);
                if  (Verbose)
                    {
-                   printf("\n Year = %2.2d, Day of year = %3d, Time
= %2.2d:%2.2d:%2.2d, Code = %s",
+                   printf("\n Year = %2.2d, Day of year = %3d, Time
= %2.2d:%2.2d:%2.2d, Code = %s",
                                Year, DayOfYear, Hour, Minute, Second, code);

                                if  ((EnableRateCorrection) || (RemoveCycle) ||
(AddCycle))
@@ -1164,7 +1218,7 @@
                                if  ((CountOfSecondsSent != 0) &&
((TotalCyclesAdded != 0) || (TotalCyclesRemoved != 0)))
                                    {
                                    RatioError = ((float) (TotalCyclesAdded -
TotalCyclesRemoved)) / (1000.0 * (float) CountOfSecondsSent);
-                                   printf (" Adjusted by %2.1f%%, apparent
send frequency is %4.2f Hz not %d Hz.\n\n",
+                                   printf (" Adjusted by %2.1f%%, apparent
send frequency is %4.2f Hz not %d Hz.\n\n",

        RatioError*100.0, (1.0+RatioError)*((float) SetSampleRate),
SetSampleRate);
                                    }
                                }
@@ -1374,7 +1428,7 @@
                                    "%01d%03d%02d%02d%01d", Year / 10,
                                    DayOfYear, Hour, Minute, Year % 10);
                                if  (Verbose)
-                                   printf("\n Year = %2.2d, Day of year = %3d,
Time = %2.2d:%2.2d:%2.2d, Code = %s",
+                                   printf("\n Year = %2.2d, Day of year = %3d,
Time = %2.2d:%2.2d:%2.2d, Code = %s",
                                                Year, DayOfYear, Hour, Minute,
Second, code);

                                if  ((EnableRateCorrection) || (RemoveCycle) ||
(AddCycle))
@@ -1383,7 +1437,7 @@
                                    if  ((CountOfSecondsSent != 0) &&
((TotalCyclesAdded != 0) || (TotalCyclesRemoved != 0)))
                                        {
                                        RatioError = ((float)
(TotalCyclesAdded - TotalCyclesRemoved)) / (1000.0 * (float)
CountOfSecondsSent);
-                                       printf (" Adjusted by %2.1f%%,
apparent send frequency is %4.2f Hz not %d Hz.\n\n",
+                                       printf (" Adjusted by %2.1f%%,
apparent send frequency is %4.2f Hz not %d Hz.\n\n",

        RatioError*100.0, (1.0+RatioError)*((float) SetSampleRate),
SetSampleRate);
                                        }
                                    }
@@ -1611,7 +1665,7 @@
                                        {
                                        if  (RateCorrection < 0)
```

```
                                                        {       // Need to remove
cycles to catch up.
-                                                       if  ((HexValue & arg) !=
0)
+                                                       if  ((HexValue & arg) !=
0)
                                                                {
                                                                if  (Unmodulated)
                                                                        {
@@ -1629,7 +1683,7 @@
                                                                        }

        strlcat(OutputDataString, "x", OUTPUT_DATA_STRING_LENGTH);
                                                                }
-                                                       else
+                                                       else
                                                                {
                                                                if
        (Unmodulated)
                                                                        {
@@ -1652,7 +1706,7 @@
                                                        {       // Else clause for
"if  (RateCorrection < 0)"
                                                        if  (RateCorrection > 0)
                                                                {       // Need to
add cycles to slow back down.
-                                                               if  ((HexValue &
arg) != 0)
+                                                               if  ((HexValue &
arg) != 0)
                                                                        {
                                                                        if
(Unmodulated)
                                                                                {
@@ -1670,7 +1724,7 @@
                                                                        }

        strlcat(OutputDataString, "+", OUTPUT_DATA_STRING_LENGTH);
                                                                }
-                                                               else
+                                                               else
                                                                        {
                                                                        if
        (Unmodulated)
                                                                                {
@@ -1692,7 +1746,7 @@
                                                        else
                                                                {       // Else
clause for "if  (RateCorrection > 0)"
                                                                // Rate is OK, just
do what you feel!
-                                                               if  ((HexValue &
arg) != 0)
+                                                               if  ((HexValue &
arg) != 0)
                                                                        {
                                                                        if
(Unmodulated)
                                                                                {
@@ -1706,7 +1760,7 @@
                                                                        }
```

```
        strlcat(OutputDataString, "1", OUTPUT_DATA_STRING_LENGTH);
                                                                }
-                                                           else
+                                                           else
                                                                {
                                                                if
        (Unmodulated)
                                                                    {
@@ -1725,7 +1779,7 @@
                                                  }       // End of true clause for
"if  ((FrameNumber == 5) && (BitNumber == 8))"
                                              else
                                                  {       // Else clause for "if
((FrameNumber == 5) && (BitNumber == 8))"
-                                                 if  ((HexValue & arg) != 0)
+                                                 if  ((HexValue & arg) != 0)
                                                      {
                                                      if  (Unmodulated)
                                                          {
@@ -1739,7 +1793,7 @@
                                                      strlcat(OutputDataString,
"1", OUTPUT_DATA_STRING_LENGTH);
                                                      }
-                                                 else
+                                                 else
                                                      {
                                                      if      (Unmodulated)
                                                          {
@@ -2178,12 +2232,13 @@
                                printf ("> Hmm, time going backwards?\n");
                        }
                }       // End of true clause for "if  (EnableRateCorrection)"
-
+
        fflush (stdout);
        }
-
-
+
+
 printf ("\n\n>> Completed %d seconds, exiting...\n\n", SecondsToSend);
+       if (DaemonRunning) stopDaemon();
 return (0);
 }

@@ -2209,12 +2264,12 @@
        peep(5, tone, HIGH);            /* send seconds tick */
        peep(25, tone, OFF);
        peep(code - 30, 100, LOW);      /* send data */
-
+
        /* The quiet time is shortened or lengthened to get us back on time */
        if  (Rate < 0)
                {
                peep( 990 - code, 100, OFF);
-
+
                TotalCyclesRemoved += 10;

                if  (Debug)
@@ -2457,6 +2512,7 @@
```

```
        printf (  "\n          -b yymmddhhmm                  Remove leap second
at end of minute specified");
        printf (  "\n          -c seconds_to_send             Number of seconds
to send (default 0 = forever)");
        printf (  "\n          -d                             Start with IEEE
1344 DST active");
+     printf (  "\n          -e                            Running as daemon");
        printf (  "\n          -f format_type                 i = Modulated IRIG-
B 1998 (no year coded)");
        printf (  "\n                                         2 = Modulated IRIG-
B 2002 (year coded)");
        printf (  "\n                                         3 = Modulated IRIG-
B w/IEEE 1344 (year & control funcs) (default)");

IRIG-B 1998 (no year coded)");
```

Appendice A3. Makefile.tg2d

```
all:    tg2d

tg2d:
        make tg2
        -killall tg2d
        -/etc/init.d/tg2d stop
        -mkdir /var/tg2d
        chmod 777 /var/tg2d
        cp -f tg2d.sh /etc/init.d/tg2d
        chmod 755 /etc/init.d/tg2d
        cp -f tg2 /usr/sbin/tg2d
        -update-rc.d tg2d defaults
        /etc/init.d/tg2d start
        ps -aux | grep tg2d
clean:
        rm -f tg2d
        rm -f tg2d.o
```

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/it/le-collane-editoriali-ingv/quaderni-di-geofisica.html

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/rapporti-tecnici-ingv.html

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

Miscellanea INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/miscellanea-ingv.html

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli, ecc.

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia