*IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*
*e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 18, Issue 4, Ser. I (July. – August. 2023), PP 12-25*
*www.iosrjournals.org*

# Upgraded Telecommand Systems For Balloon Borne Payloads

P. R. Sandimani[1], S. M. Gharat[1], S. L. D'Costa[1], S. S. Poojary[1], S. B. Bhagat[1], H. Shah[1], R. B. Jadhav[1], B. G. Bagade[1], S. K. Ghosh[1], D. K. Ojha[1], J. P. Ninan[1]
[1]*(Tata Institute of Fundamental Research, Homi Bhabha Road, Colaba, Mumbai 400005, India)*

***Abstract:***
*A new realization of the ground segment sub systems of an existing telecommand system for balloon borne payloads, using modern technology is described in this paper. Existing telecommand system (Ghosh & Tandon, 1982), has been working from last many years during TIFR 100 cm (T100) far infrared balloon flights (Mookerjea B. et al., 2003; Kaneda et al., 2013; Suzuki et al., 2021). This system was designed and realized for on line payload control during balloon flight campaigns. Upgradation for the ground system is required considering the fact that, the older system uses many of the discrete components and some of the components are obsolete. Sometimes it becomes difficult to debug and repair the system in certain situations considering the lack of support of outdated and obsolete compilers, very old designs of Industry Standard Architecture (ISA) cards and their components etc. This becomes even more difficult when a balloon flight is under way and any of the malfunction is detected. We therefore designed and upgraded an encoder using the Universal Serial Bus (USB) interface between the PC and an encoder. An encoder uses a microcontroller interface with the discrete components based encoding hardware. Later we designed another two versions in which microcontroller alone has all the encoding intelligence. Encoders mentioned in this paper have been used during recent T100 far infrared balloon flights since February 2017 till now. We will continue using it during near future T100 balloon flights with proposed upgraded Fabry Perot Spectrometer (FPS) which will have a 5x5 pixels detector array. We also propose to make a standard telecommand bus which will be common to all the payloads, so that our telecommand encoder can be interfaced as a standalone unit.*

***Key Word:*** *Microcontroller; Encoder; ISA; T100; Balloon borne*

---

Date of Submission: 17-07-2023 Date of Acceptance: 27-07-2023

---

## I. Introduction

A telecommand system (TCS) plays an important role in a balloon borne scientific payload, which requires real time control of various experimental parameters. It transfers information reliably to the remote space borne payload from the ground station. A TCS consists of two major parts, viz. telecommand encoder (TCE) and telecommand decoder (TCD). The message to be transmitted is coded in the TCE at the ground station and transmitted via a S band radio link. The radio signals are degraded when they are received on board. The TCD deciphers messages from these degraded signals and makes a decision (according to pre-determined criteria) as to whether the message received is erroneous or not. If the message is erroneous, it is either rejected outright or error correction is performed if the number of erroneous bits is within some limit (the system described in this paper rejects erroneous messages). If the message is not erroneous, it is passed on to the relevant subsystem of the payload to be executed. A block diagram of the total information channel is shown in Figure no 1.

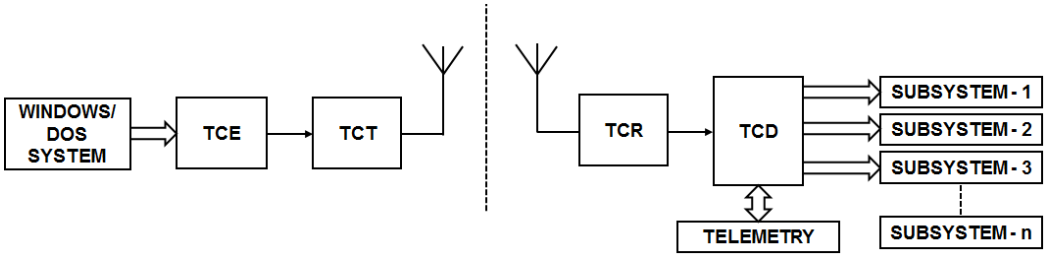


**Figure no 1:** Block diagram of total information channel
TCE: Telecommand Encoder, TCT: Telecommand Transmitter,
TCR: Telecommand Receiver, TCD: Telecommand Decoder

A new balloon payload was built for making far infrared (FIR) astronomical observations. This new payload required up to 12-bit data words to be transmitted from ground to the payload in order to control various parameters of the experiment. To meet this requirement, we have upgraded and fabricated new TCEs which are compatible with the specifications of the earlier telecommand transmitter (TCT). Upgraded encoders can be interfaced in both the DOS and Windows operating systems (OS).

New modern technologies allow us for the simpler implementation of the design, testing and maintenance. Upgraded ground systems have Peripheral Interface Controller (PIC) microcontroller, so that commands can be encoded with the help of microcontroller firmware. This system can be interfaced in both the DOS based system and Windows based system, which is at least at the same level as earlier or better in terms of the techonology. Identical encoding and redundancy have been used as in the upgraded versions of the encoders. This reduces the probability of wrong command operation as well as rejection of genuine commands.

Section II below explains the overview of the design scheme. Section III explains the current implementation of various versions of the encoders. One sub section explains the encoder which uses microcontroller and discrete component based hardware. Next sub section explains the inclusion of the ISA sandwich card. Subsequent sub sections explain other two of the versions of the encoder which uses sole microcontroller, which has inbuilt complex computing capability. Microcontroller details and some of the flowcharts are also explained further. Typical application of this system is explained in section IV. Validation and performance results are explained in section V.

## II. Overview of the design

Any TCS needs some of the primary requirements. This includes the environmental noise immunity, correct interpretation of the command considering the possibility of the errors introduced by noise at the on board receiver output and also the TCS should not have inevitable demands so that the efficiency of the system is decreased.

The system described in this paper transfers 18 bits of Pulse Code Modulation (PCM) data (the message) from the ground station to the payload. This data includes 6 bits which is treated as the subsystem ADDRESS and the remaining 12 bits are treated as DATA by the TCD. In order to achieve high error rejection through redundancy checks, several parity bits are added to the PCM data. Since the command transmission is not continuous, it is required to distinguish between three levels of information, viz. ZERO and ONE of the PCM data and NOTHING corresponding to the absence of data. This three level coding is achieved through pulse duration modulation (PDM). The PDM scheme is strictly maintained in the TCE. However, to achieve noise immunity, pulse duration is not demanded strictly in the TCD. Hence margins are allowed on both sides of the bit duration. The PDM output is used for amplitude shift keying (ASK) of an audio subcarrier. This subcarrier is fed to the S band transmitter for the final amplitude modulation (AM) and transmission. This scheme is described in detail in the published paper by Ghosh & Tandon (1982). Table no 1 shows some of the parameters related to the design scheme.

For cyclic redundancy checks, parity bits are introduced into the PCM command frame. They are generated using a Bose Chaudhuri Hocquenghem (BCH) cyclic code. In the older versions of the TCE corresponding to the paper by Ghosh & Tandon (1982), we had chosen such a code because of the simplicity of hardware realization. However, in the upgraded version of the TCE (mentioned in this paper), we use the similar encoding scheme, but the design uses the intelligence of the microcontroller for encoding.

The (28, 18) BCH code we use has the following properties. The Hamming distance is five, which means it can at least detect all erroneous command frames having up to five error bits distributed over the PCM command frame. It can detect up to 10-bit burst errors. Also, if one uses random error correction logic, this code is capable of correcting up to two random error bits anywhere in the frame. However, since in our application, command rejection probability is very small, the error correction capacity of the code is not utilized. Cyclic codes have the property such that if the whole frame is shifted cyclically by any number of bits, the resulting frame is again a valid frame. However, to avoid misinterpretation of the command due to noise, a four-bit synchronization (SYNC) word is introduced at the beginning of the PCM command frame.

As an upgraded design, a completely new implementation has been described here. This design uses the modern technologies, software platforms and minimum hardware. We used the microcontroller as the heart of the TCE system. Various versions of the TCEs are explained in the following sections. This include the hybrid design which has both the microcontroller and the discrete components based hardware, and also the sole microcontroller based systems.

## III. Design parameters and encoder explanation

Following sections explain the design details of various versions of the upgraded TCEs. Table no 1, Table no 2 and Table no 3 below give the basic parameters of the TCS design scheme, coding and decoding.

**Table no 1:** Basic parameters of the TCS

| **Parameter** | **Value** |
|---|---|
| Signal to noise ratio at decoder input (30 kHz bandwidth): | > 13 dB |
| Decoder input filter specification: | Q = 5, BW = 1.25 kHz |
| Telecommand transmitter frequency: | S Band system (2.259 GHz) |
| Subcarrier frequency: | 6.25 kHz |
| PCM command bit rate: | 41.67 Hz |

**Table no 2:** For Coding (**t**: no. of subcarrier cycles in a bit duration)

| **Bit logic level** | **No. of subcarrier cycles in a bit duration** |
|---|---|
| ZERO | t=50 |
| ONE | t= 100 |
| NOTHING | t = 0 |

**Table no 3:** For Decoding (**r:** no. of subcarrier cycles detected in a bit duration)

| **No. of subcarrier cycles detected** | **Detected bit logic level** |
|---|---|
| 25 <= r < 75 | ZERO |
| 75 <= r < 125 | ONE |
| 0 < r < 25 or r >= 125 | ILLEGAL CHARACTER |
| *r=0* | GAP |

**Telecommand encoder versions**

Following sections describe various versions of the TCEs developed. The first version is named as the universal serial bus interfaced telecommand encoder (USB TCE). It has a microcontroller, which generates parity bits and also adds the SYNC word. Further discrete components based hardware does the PDM i.e. generates 50 or 100 pulses depending on the logic level of individual bits, as the final encoding. Here WINDOWS PC operates as a 'master' to send the command. In the second version of the encoder, an intermediate handshaking module has been designed to use the DOS PC as a 'master' to send the command. This module is sandwiched between the DOS PC and USB TCE mentioned above. Third version is named as the 'Mini TCE'. It has two microcontrollers. This encoder can be interfaced with both the DOS PC and the Windows PC, as a 'master'. However, only one of the masters can send the command at a time. Fourth version is named as the 'Micro TCE'. It uses a single microcontroller. This encoder can be interfaced only with the DOS PC.

**Universal Serial Bus interfaced Telecommand Encoder (USB TCE)**

Since USB TCE has been designed for the Windows OS, a Graphical User Interface (GUI) (shown in Figure no 2) has been prepared using Visual Basic software. It sends 6 bytes to the encoder module, corresponding to each of the command digits (octal) entered in the GUI. After sending 6 bytes of the command, GUI also sends a termination byte to the encoder module so that the encoder can receive all the 7 bytes and uses only 6 legitimate command bytes for encoding. FT245 is a Future Technology Devices International Ltd. (FTDI) made USB FIFO interface, which has been used to create a virtual COM port emulating the RS232 protocol. Visual basic software uses the D2XX driver for the USB to COM port interface. D2XX drivers allow direct access to the USB device through a Dynamic Link Library (DLL) file. Application software can access the USB device through a series of DLL function calls. We made the provision of sending individual commands by manually typing command digits in the GUI. This is specially for the data commands. We also made the provision for selecting commands from the macro library. This is for some of the ON/OFF commands. History of the commands sent can also be displayed on the GUI.

Block diagram of the USB TCE is shown in Figure no 3. A master clock generates the subcarrier of 6.25 kHz. We used a crystal to generate 8 MHz clock. It is divided by 128 to get 62.5 kHz. We further used a decade counter to get 6.25 kHz master clock. In the next stage, it is divided by 150 using cascade Johnson counters to generate a bit rate clock of 41.67 Hz.

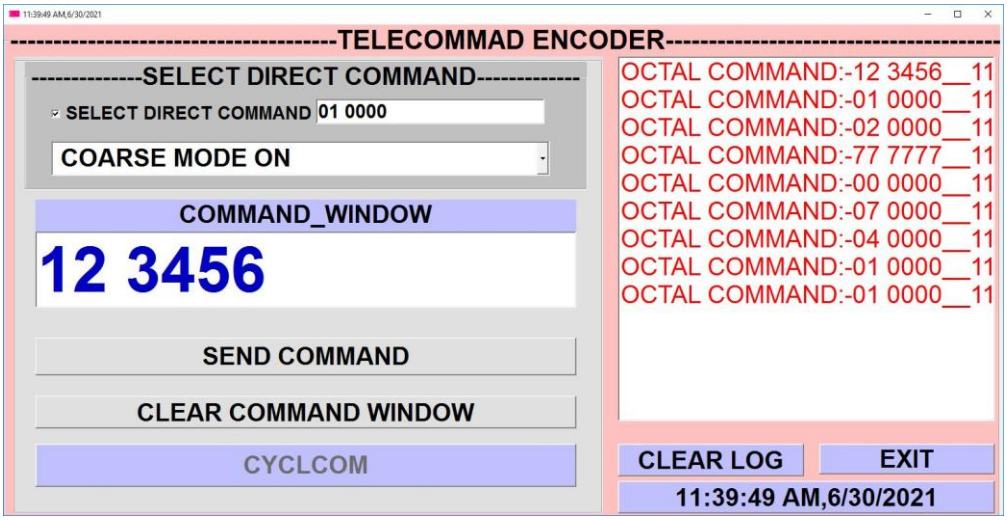


**Figure no 2:** GUI designed in Visual Basic.
Left hand side panel shows both the manual command and macro command options.
Right hand side panel shows the history of the commands which are sent successfully.

PIC receives 7 bytes through FT245 using some of the handshaking signals. It extracts 6 useful command bytes (octal digits). Thus PIC has 18 bit (3 bit per digit) information sent from the GUI. PIC sends these 6 digits to the 7 segment display as the initial confirmation of the successful command reception from the GUI.

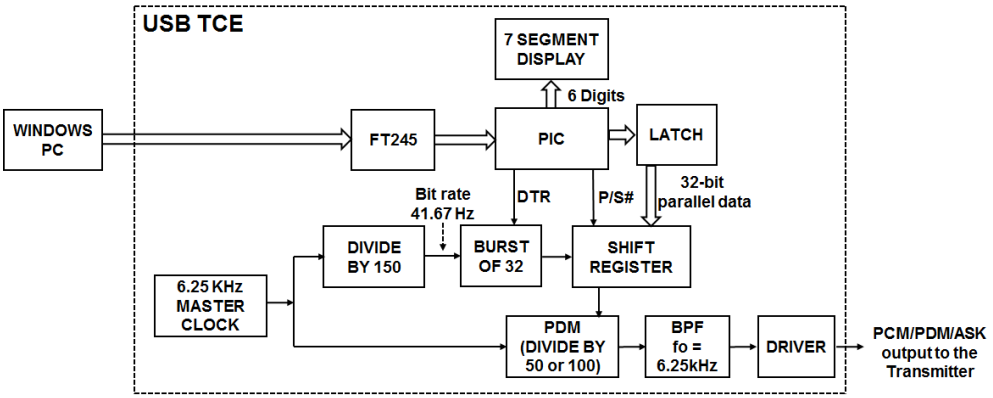


**Figure no 3:** Detailed block diagram of the USB TCE

PIC receives raw command bits from PC through FT245. PIC calculates the parity and also appends the SYNC bits. Remaining hardware and PDM modulates the bits depending on each bit's logic level. Encoded and modulated data bits are filtered by Band Pass Filter (BPF) and finally transmitted using the driver stage.

To implement the (28, 18) modified BCH cyclic code, the 1st (i.e. MSB), 3rd, 6th, 9th, 10th, 11th, 12th, 14th, 16th and 18th bits of the 18-bit command information at any time are added modulo two, to generate the parity bit and fed to the serial input (at the LSB end) of shifting subroutine of the PIC program. The modulo two addition is effected by cascaded exclusive OR operation of the PIC program. The parity generation logic is shown in Figure no 4. PIC also appends the command frame by 4-bit SYNC.

SYNC word is $_{lsb}1110_{msb.}$ MSB is sent first. Even though the parity generation logic is similar to the earlier designs, implementation of the microcontroller allows us to remove relatively bulky and multi component hardware. This makes the design very portable and lot of efforts of assembling various components and their testing have been reduced remarkably. This also reduces the probability of the component failure as very few components are used. The complete PCM command frame format (32 bits) as it appears at the shifting subroutine's output along with the SYNC bits is shown in Figure no 5.

These 32-bit information is divided into 4 bytes and are given to 4 latches. This is further given to the 32-bit shift register. PIC also generates the data ready (DTR) and Parallel/Serial select (P/S#) signals simultaneously. The DTR signal is given to the 'BURST OF 32' logic to generate 32 pulses corresponding to the bit rate of 41.67 Hz. The P/S# is given to the shift register, so that 32-bit parallel input can be loaded inside the shift register and further serial shifting can be initiated. Each bit is serially shifted out using the burst of 32 clocks.

Serial data is given to the PDM where, synchronized with the bit rate clock (41.67 Hz), a gate is opened for the subcarrier and the gated subcarrier pulses are counted. Depending on the bit value, after 50 or 100 cycles (corresponding to (1/3) *T* or (2/*3) T* respectively) the gate is closed. Up to this point all circuits are implemented using positive TTL logic. The resulting gated subcarrier (TTL levels) is band pass filtered *(Q=5,* BW=1.25 kHz) to remove the DC component and higher harmonics of the 6.25 kHz subcarrier. The impedance matched filter output is the final PCM/PDM/ASK output of the encoder which goes to the S band transmitter for the final AM and transmission.

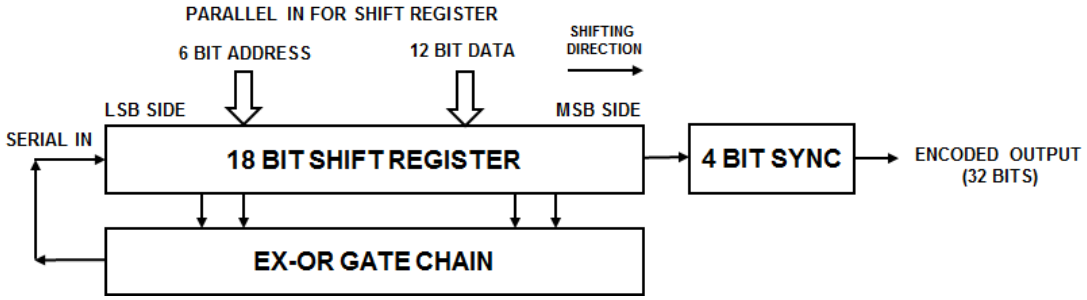


**Figure no 4:** Logic illustration of parity generation using the microcontroller program

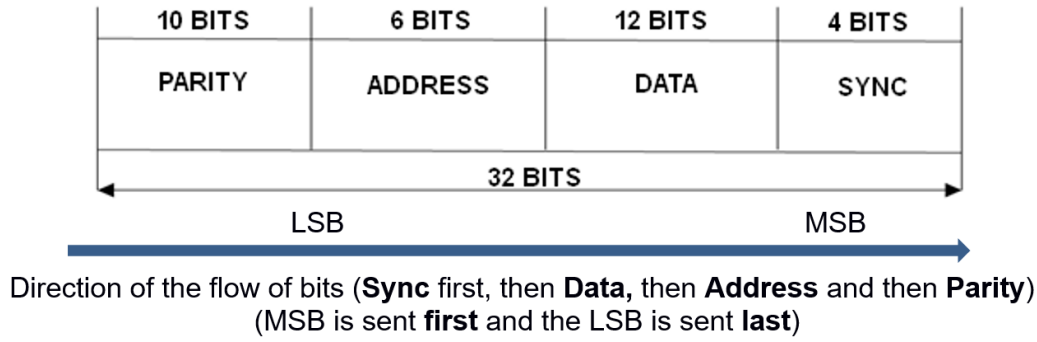


**Figure no 5:** Command frame format

**Intermediate handshaking module for 'DOS PC's ISA interface with the 'USB TCE'**

As mentioned in the last sub section, we have designed a USB TCE which is used with the Windows system for transmitting the commands to the payload. Since this encoder was designed exclusively for the Windows OS, we decided to make a sandwich board as an ISA Interface Card, which enables us to use the old DOS based system for sending the command using the newly designed encoder.

Figure no 6 shows the 'ISA interface card' designed as a sandwich card between the DOS system and the USB TCE and the Figure no 7 shows the handshaking signal details between the DOS PC and the 'ISA interface card'.

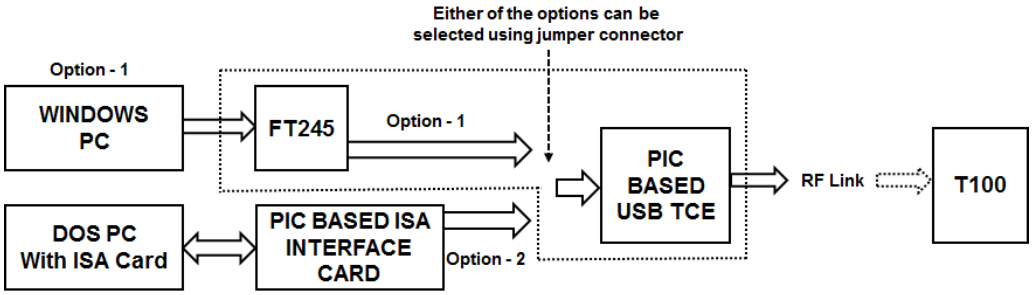


**Figure no 6:** Telecommand setup for both the Windows and DOS systems

Note that FT245 is the part of the USB TCE. A jumper connector is taken out for the desired option selection.

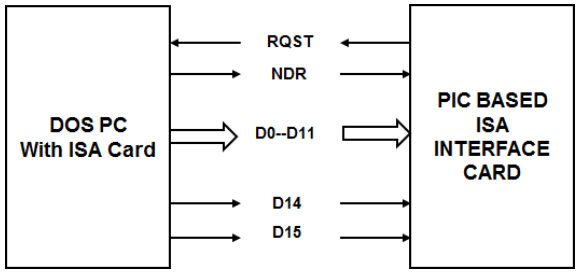


**Figure no 7:** Handshaking signals between 'DOS PC's ISA card' and 'PIC based ISA interface card'

DOS PC has an ISA card installed on the motherboard of the PC. This card was designed to send 18 bits of the raw command to the older version of the external encoder. A program was written in the assembly language to read the input command from the keyboard of the DOS PC and send command bits using the 12-bit data bus of the ISA card.

Assuming a command being XX YYYY (all octal digits), XX is the 6-bit Address and YYYY is the 12-bit Data. Flowchart (Figure no 8) shows the logic of the ISA card's assembly program.

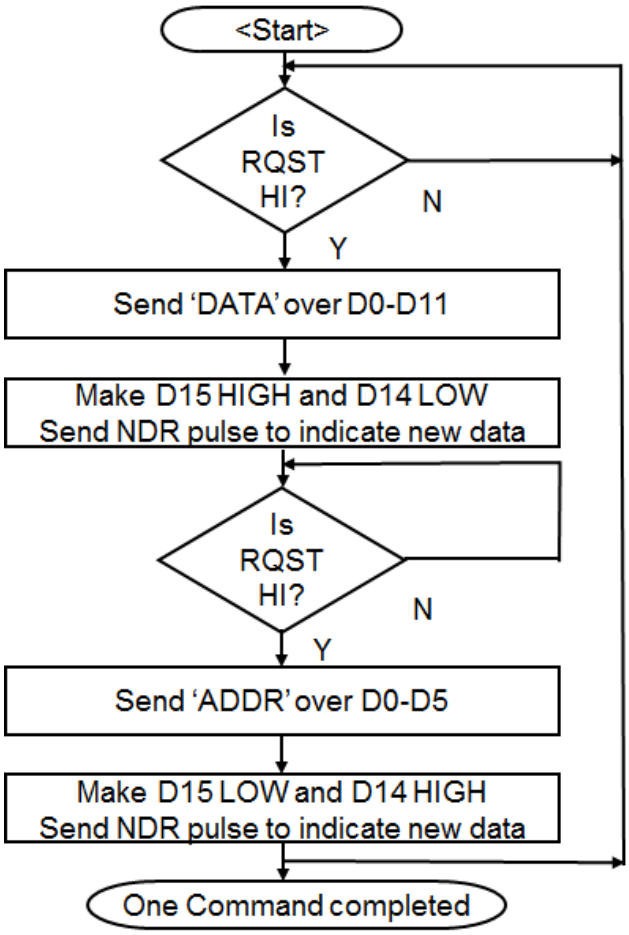


**Figure no 8:** Flowchart of the ISA card's assembly program

ISA interface card has a microcontroller (PIC18F4550), which receives signals from the ISA card of the DOS PC, using above mentioned handshaking signals. It receives 12-bit Data and 6-bit Address from the ISA card and separates them to form 6 individual digits (octal). It sends separated digits to the USB TCE system in the specific order along with strobe pulses with individual digits. It should be noted that the USB TCE can also receive these digits from the FT245 module as option 1, corresponding to the windows system, as explained in the last sub section on USB TCE. Rest of the encoding scheme inside the USB TCE remains the same. Here option 1 and option 2 are selected using jumper connectors so that either of the options can be selected for the encoding. Please refer to the appendix A for the detailed logic of the microcontroller program.

Software protection for the above mentioned ISA interface card and the USB TCE is briefly mentioned here. There are various data transfers and handshaking in the above mentioned encoder system. This includes data from the DOS PC to the ISA interface card, the ISA interface card to the USB TCE and the windows PC to the USB TCE. During actual command transfer from the master PC to the payload, there could be scenarios in which data transfer could get stuck due to malfunctioning of any of the buffer IC or latch IC or any of the handshaking signals. As the T100 payload receives a specific command and functions accordingly, in case of the above mentioned malfunction, care should be taken to avoid sending undesired command to the payload.

Enough care has been taken in the microcontroller firmware program and also in the Visual Basic GUI program, so that no command will be sent to the payload in case of malfunctioning of any of the hardware modules or in case of any of the missed data. Section V explains these safety measures in more detail.

**Mini TCE**

The microcontroller based encoders mentioned in the above sub sections which are titled as 'Universal Serial Bus interfaced Telecommand Encoder (USB TCE)' and 'Intermediate handshaking module for 'DOS PC's ISA interface with the 'USB TCE', use discrete components for the final encoding. Microcontroller is used to receive the command from either a Windows system or a DOS system and also to add the SYNC bits and to calculate the parity bits. Final PDM is done by discrete component based hardware. Here in this section, we describe another microcontroller based compact encoder which is named as 'Mini TCE'. It uses very minimal discrete components; hence the name is 'Mini'. All the encoding logic (including the PDM) has been embedded in the microcontroller firmware. This encoder can be interfaced with both the 'DOS OS' and the 'Windows OS'. This encoder uses two microcontrollers.

This section explains the 'Mini TCE' in detail. This encoder has a two stage processing. This is done by two microcontrollers, viz. uC1 and uC2. As shown in the following block diagram (Figure no 9), uC1 functions as an ISA interface card, similar to the design explained in the last sub section. This controller receives the command from the DOS PC, reformats parallel data into individual digits and sends 6 command digits and a termination digit to the second microcontroller uC2.

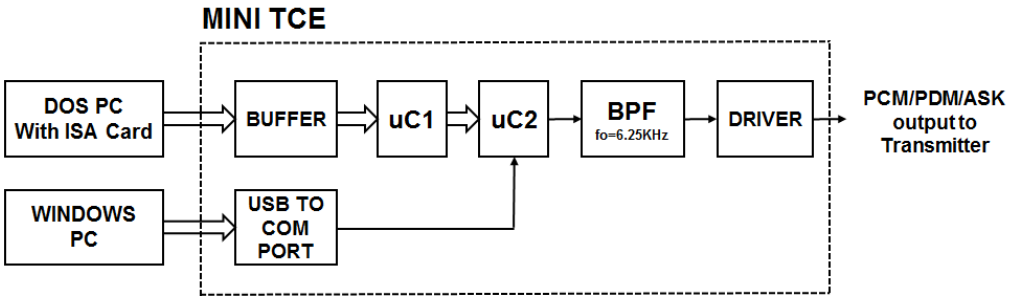


**Figure no 9:** Block diagram of the Mini TCE

uC1 receives the command from the DOS PC. uC2 receives the reformatted command from uC1. uC2 can (optionally) receive command from windows PC over RS232. uC2 calculates the parity and adds the SYNC bits. uC2 also does the PDM. Final encoded output is sent through the filter and driver stages.

uC2 receives the reformatted data and does the actual encoding. Program written for uC2 generates 4-bit SYNC, calculates the 10-bit parity and reformats the original 18-bit command into 32 bit. These 32 bits are further given to the another sub routine of the microcontroller program, which performs all the calculations required for the encoding i.e. generating 50 pulses for logic 0 and 100 pulses for logic 1. Internal timer based sub routines are used for these operations. Encoded data is given to the filter circuit and further given to the telecommand transmitter. Hence, all the blocks of the discrete components based encoder are replaced by uC2.

uC2 can also receive the command sent by the Windows PC using the USB based COM port. However, sub routines written for the uC2 for end to end encoding are similar as mentioned above. In case of the Windows PC option, uC1 does not play any role. It is to be noted that, there is no any jumper selection for selecting either the DOS PC option or the Windows PC option. Different I/O ports of the microcontroller uC2 are used for these options. However, care should be taken while sending the commands. Even if, both the DOS and the Windows systems are connected to the encoder for the provision of quick redundancy, only one of the systems should send the command, otherwise uC2 programs may crash and we may have to manually reset microcontrollers. Figure no10 shows the actual image of the Mini TCE hardware. Please refer to the appendix A for the detailed logic of both the microcontrollers (uC1 and uC2) programs.

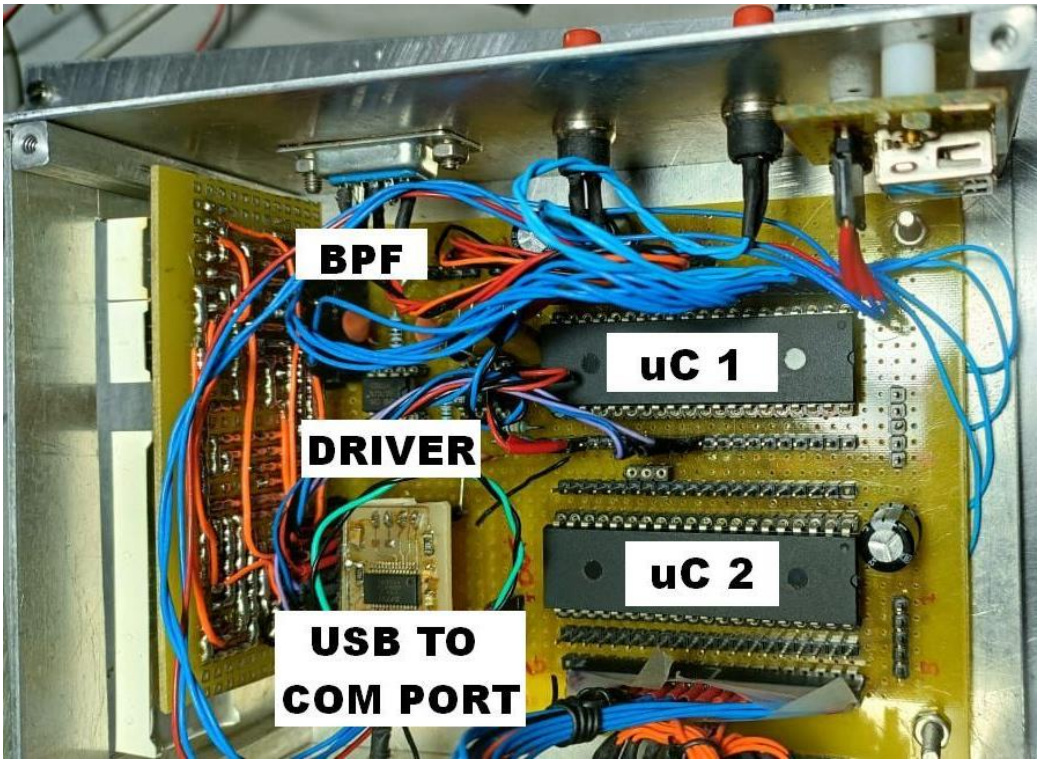


**Figure no 10:** Final assembled card of the Mini TCE

Following flowchart (Figure no 11) shows the basic flow of the operation of the Mini TCE

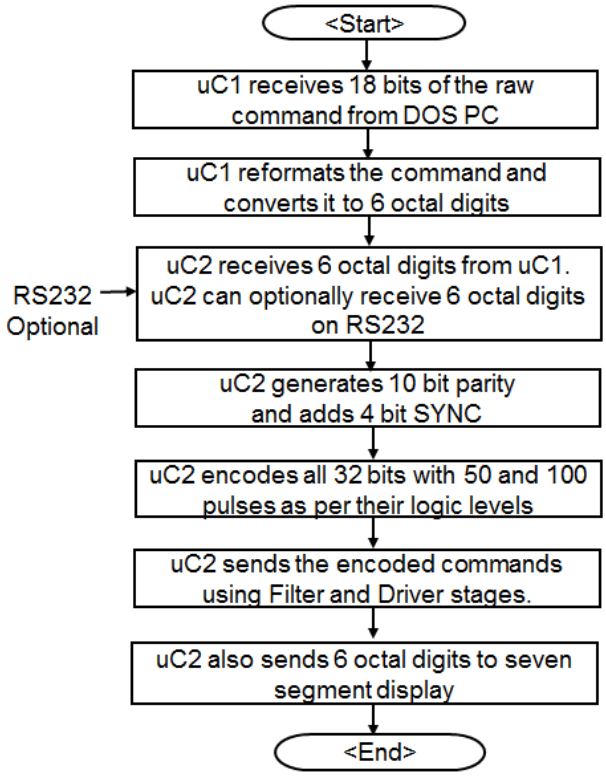


**Figure no 11:** Flowchart for the step by step operation of the Mini TCE.

**Micro TCE**

This is a slightly modified version of the above mentioned 'Mini TCE'. In the 'Mini TCE', two microcontrollers are used. One for receiving the command from the DOS PC and the second for actual encoding. In this version, microcontroller program has been modified so that, single microcontroller receives the command from the DOS PC and also does the calculations required for generating SYNC and parity, and it also does the final encoding. Hence, we name it as a 'Micro TCE'. This encoder can only be used with the DOS PC. The block diagram is shown in Figure no 12 below. Figure no 13 shows the final assembled circuit of the Micro TCE.

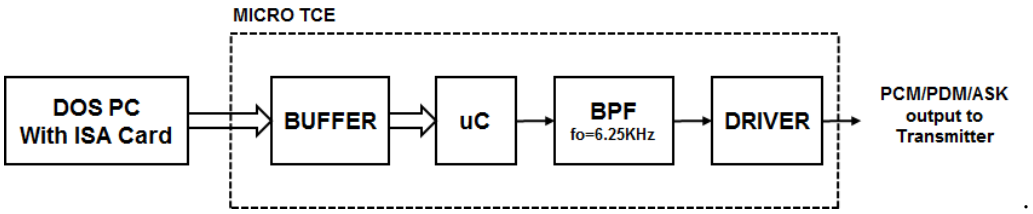


**Figure no 12:** Block diagram of the Micro TCE.

uC receives the command from the DOS PC. It calculates the parity and adds the SYNC bits. uC also does the PDM and finally command bits are transmitted using the BPF and driver stages.

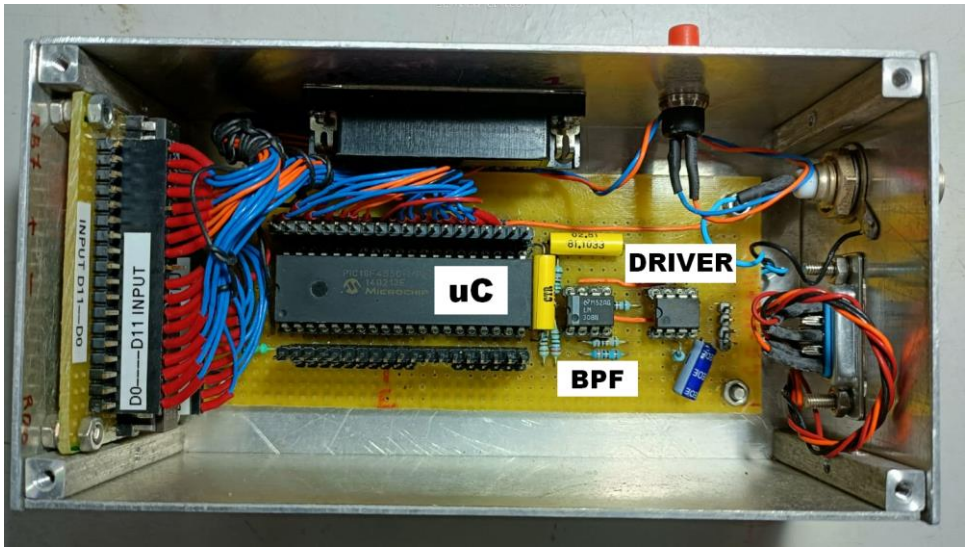


**Figure no 13:** Final assembled card of the Micro TCE

Following flowchart (Figure no 14) shows the basic flow of the operation of the Micro TCE

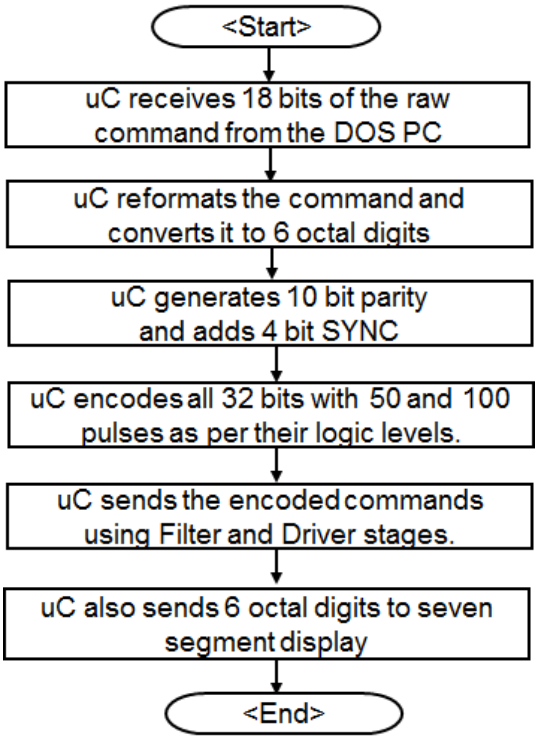


**Figure no 14:** Flowchart for the step by step operation of the Micro TCE

**Microcontroller and tools used**

We used the PIC 18F4550, an 8-bit microcontroller for the upgradation of the encoders. Pinout of the microcontroller is shown in Figure no 15. This family of devices offers the advantages of all PIC18 microcontrollers namely, high computational performance at an economical price with the addition of high endurance, and Enhanced Flash program memory. In addition to these features, the PIC18F4550 family introduces design enhancements that make these microcontrollers a logical choice for many high performance and power sensitive applications. This controller has 32 Kbytes flash memory, 2048 bytes SRAM, 256 bytes EPROM, 35 I/O ports, 13 numbers of 10 bit ADCs, 1 USART, 2 comparators and 3 number of 16 bit timers.

We used MPLAB IDE (Figure no 16) for developing the program for the microcontroller. We used the C18 compiler to generate the binary file. We used PICKIT 2 software for programming the microcontroller.

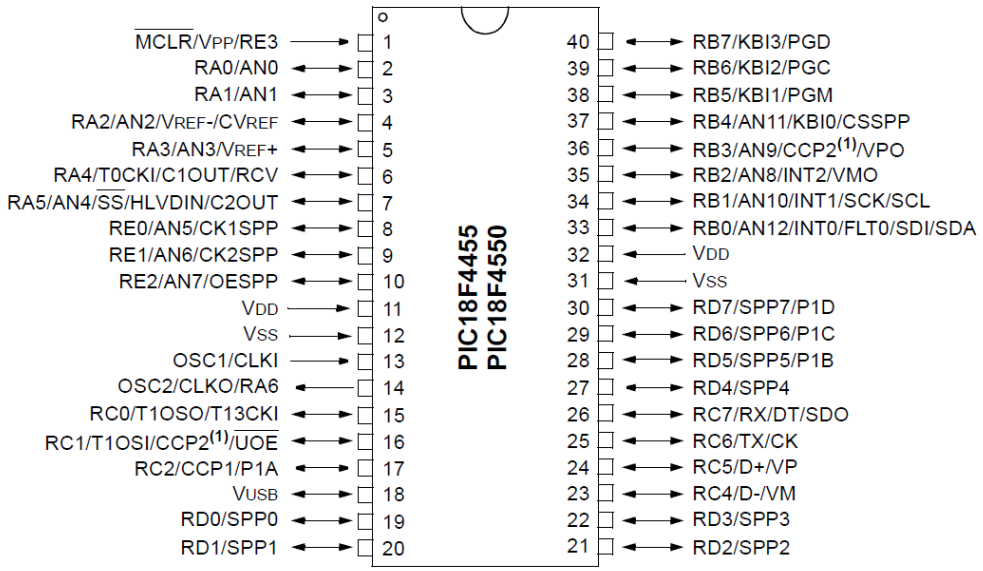


**Figure no 15:** Pinout of the 40 pin PDIP package of PIC 18F4550

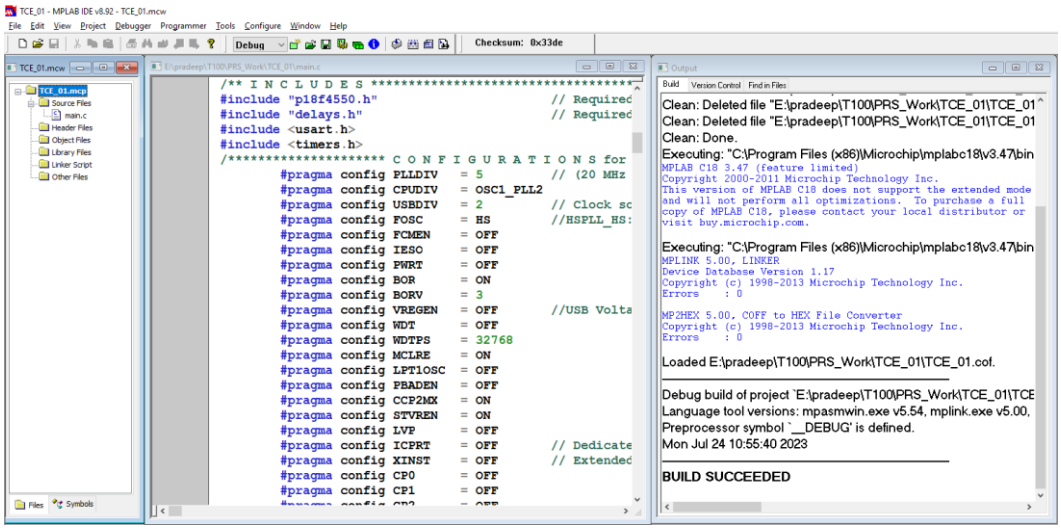


**Figure no 16:** Typical screenshot of the MPLAB IDE

## IV. Typical applications

The TCS described has been a part of a far infrared telescope (Daniel et al., 1979) for making astronomical observations from balloon altitudes. The telescope is oriented towards the celestial object under study. This orientation system requires the instrumental angular coordinates corresponding to the celestial object as inputs to be fed on line from the ground station. DATA commands are used to perform this task. Similarly, the telescope scan function parameters, detector gain, phase sensitive detector delay, etc. are a few examples of data commands.

There are certain operations in this experiment which require real time putting ON or OFF of various electronic switches. To name a few, these include putting ON or OFF the secondary mirror wobbling and certain optional power supplies, the discrete motion of a balancing weight to keep the telescope balanced and the latching operation of the telescope. In these cases, ON/OFF type commands are used.

Certain applications have irreversible operations to be performed by executing an ON/OFF command (e.g., cutting off the batteries at the end of the experiment). Extra safety is provided against the false execution of such an operation by demanding two commands to be received on board in a predetermined sequence within a certain time slot (typically 10 Sec.); if these conditions are violated the whole system ignores these commands.

Currently we have a scanning Fabry Perot Spectrometer (FPS) installed at the focal plane of the telescope. Spectrometer has a single pixel detector which is tuned to ~158 micron for [CII] line far infrared observation. This FPS is being upgraded which will have a 5x5 pixels array, which has much better resolution (R=10000). In the past, the Cassegrain focus of the T100 was populated with several far infrared Photometers (single as well as multi-channel systems) based on bolometers (cooled to 0.3 K) as the detectors as back ends. Considering all these options, our TCS system can serve all kinds of focal plane instruments. Accordingly, we anticipate this upgraded telecommand system will be used in the medium to long term future or as long as T100 remains operational.

Since a TCS is a very basic requirement for any of the balloon borne experiments, we have made it available for the proposed standard balloon service module, which provides support bus for all the payloads of all other experimental groups.

## V. Validation and performance results

Different versions of the TCEs mentioned in this paper have been tested and qualified at various stages during their developments. This includes the hardware tests, firmware tests, GUI software tests, probabilities of malfunctions, end to end tests with the payloads, etc. This is explained in details in the following paragraphs.

### GUI Test

For the GUI test, we used various dummy test commands with all the combinations of valid and invalid digits. GUI has been qualified to send only the digits from 0 to 7, as command format of the system is in the Octal form. A cyclic form of commands was also sent using GUI and qualified successfully. These commands were seen on the digital display of the TCEs and also observed on the oscilloscope.

**Firmware Test**

As mentioned in the section III, enough care has been taken in the microcontroller firmware program and also in the Visual Basic GUI program, so that no unintentional command can be transmitted in case of malfunctioning of any of the hardware modules. Following paragraphs explain the protection part in some detail.

As a part of the protection, an encoder module receives 7 octal digits instead of actual useful 6 digits. An additional character (as a termination digit) is sent as the 7$^{th}$ digit from the master PC. This digit is used as the confirmation of the faithful transmission of the command from either the 'ISA interface card' or from the windows PC to the Encoder unit. GUI software in the visual basic has been written accordingly.

As can be seen in the Figure no 17, 7 digits are sent to the TCE which include the termination digit 'A' (0x0A in Hexadecimal). These signals are sent from the 'ISA interface card' to the TCE. This applies when the DOS based system is used. Similarly, a termination digit is sent from the Visual basic GUI, when encoder is operated from the windows OS.

As shown in Figure no 17, actual command to be sent is 12 3456 from the DOS PC. Here, 12 is the address part and 3456 is the data part of the command. The ISA interface card receives the data and address part using handshaking signals, as mentioned above in respective sections. It separates all the 6 digits. It first sends data part, viz. digits 3,4,5,6 and then the address part, viz. digits 1 and 2 and finally sends the termination digit 'A' to the USB TCE. While sending each of the seven digits to the USB TCE, a strobe pulse is also generated, so that the USB TCE unit can receive and latch each of the digits.

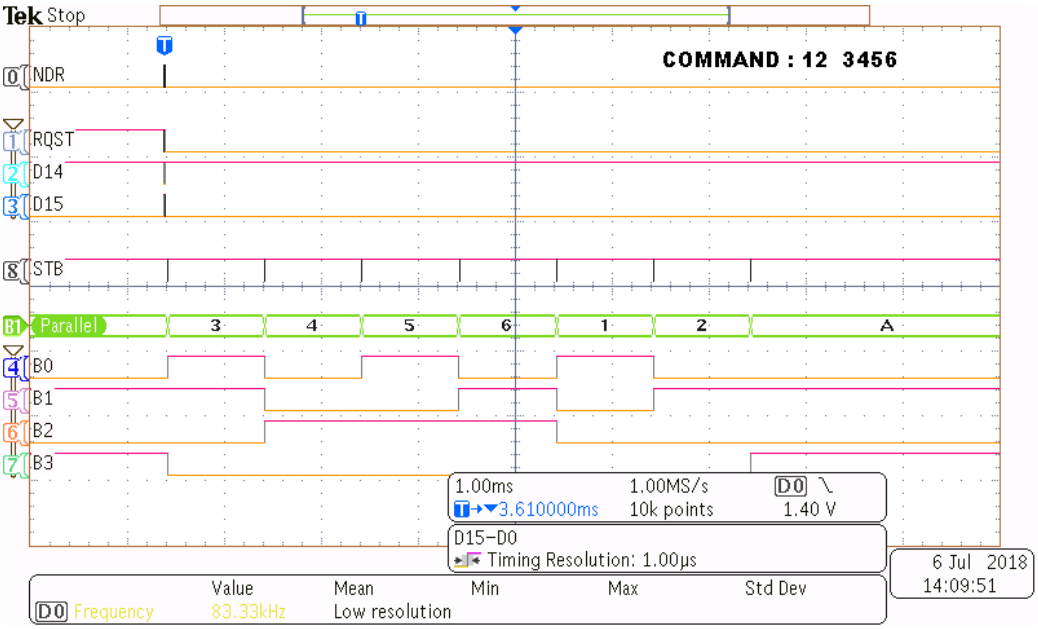


**Figure no 17:** Signals (including a Termination byte) between the ISA interface card and the TCE

A timer is also used for an additional protection. Command is sent in two parts from the DOS PC ISA card to the microcontroller, i.e. 'data' and 'address'. When 'data' part of the command is received by the microcontroller, a timer is initialized. 'Address' part of the command should be received by the microcontroller before the timer overflows. Otherwise, earlier received 'data' part of the command is discarded and the microcontroller goes to the quiescent state and looks for the new command.

**Laboratory Test**

Tests mentioned below include the encoders output voltage stability, fluctuation of the center frequency of the band pass filter due to temperature, handling of the ideal mode of the encoder during power ON, application of the encoder using any of the COM port interface module, protection of the microcontroller circuit from unintentional source current, etc.

There is a band pass filter stage which has the center frequency of 6.25 kHz for all the versions of the encoders. Since this filter is tuned to operate at a fixed center frequency, a stable capacitor in the filter stage should be used so that the TCE output voltage would not fluctuate. Fluctuation in the output peak to peak voltage would malfunction the on board TCD by incorrect decoding and subsequently undesired functioning of the payload. Various capacitor types were studied and optimized and finally we decided to use the CTR type capacitor. Complete encoder setup was tested extensively multiple times and it was observed that output voltage was stable to the desired voltage level.

When encoder is powered ON, encoder is not stabilized to the ideal mode, hence does not give the desired output for the very first command. This is because of the possibility of the improper power on reset (POR) of the microcontroller and also due to absence of the termination character which is required to initiate the encoding algorithm to start from the correct position. Users need to give any one (or few) of the test commands to the encoder, without connecting the encoder to the payload, so that proper initialization takes place in the microcontroller program logic.

During the initial period of the development of these encoders, parity generation logic was implemented in the Visual Basic program. Encoded 32 bits (18-bit command, 10-bit parity and 4-bit SYNC) were sent using the D2XX mode to the microcontroller for further modulation. In the later modifications, parity generation logic was implemented inside the microcontroller firmware itself. Here, Visual Basic program simply sends 6 octal digits of commands and a termination byte to the microcontroller. This enables us to use a simple HyperTerminal of the PC or any of the COM port interface module, to send raw command digits to the encoder module without using the Visual Basic GUI.

When an encoder is used with the DOS PC, during initial testing it was observed that, microcontroller card gets powered up because of the source current of the ISA card installed in the DOS PC. This could lead to the unintentional operation of the microcontroller even in absence of the actual power supply to the encoder card. To avoid this, we added a unidirectional buffer stage at the input of the microcontroller. This makes the microcontroller to operate only in presence of its own power supply.

The TCE, being not a balloon borne equipment, need not be tested environmentally (at low temperatures and pressures). In order to confirm the proper functioning of the TCE hardware, it was tested in the lab at room temperature along with the emulated TCD simulator unit at the receiving end. The TCD simulator unit is an exact duplication of the onboard TCD.

**Preflight and in flight**

The T100 telescope interfaced with the FPS was flown with a balloon to an altitude of 31 km on 5 times during February 2017 to February 2023 from Hyderabad (latitude = 17.5', longitude = $78.5^0$). During initial couple of balloon flights, upgraded command system explained in this paper was kept as a redundant for the old discrete component based external encoder. As a regular practice, we carry out a complete telescope diagnosis test (TSD) to check the status of the telescope. Upgraded TCEs mentioned in this paper were qualified and used regularly during the TSD tests and also during recent balloon flights successfully. A typical view of the balloon flight setup on the launch field is shown in Figure no C1 under the appendix C.

**Oscilloscope screenshot of the final encoder output.**

Figure no 18 shows the final output of the encoder and some of the variables used and mentioned in flowcharts in the appendix section, viz. byte_counter, pulse_counter, bit_counter, IDLE etc.


**byte_counter** : To count 6 digits of the command and a Termination byte.
**pulse_counter** : To count either 50 or 100 pulses
**bit_counter** : To count 32 encoded bits of the command
**IDLE** : indicates that no any pulses are sent. This is to cover the remaining part of the BIT time.

**Figure no 18:** Oscilloscope screenshot of the encoder output.

## VI. Conclusion

From the above discussion, we conclude that all the versions of the TCEs have been successfully developed and applied for the telecommand interface with the T100 payload. All the TCEs have been qualified and tested successfully during the ground tests and also during last couple of balloon flights of the T100 far infrared telescope. Since these TCEs have given satisfactory performances, we will use them regularly for future balloon flights.

## References

[1]. S. K. Ghosh And S. N. Tandon, "A Telecommand System For Balloon Payloads", 15 755, J. Phys. E: Sci. Instrum., 1982
[2]. H. Kaneda, T. Nakagawa, S. K. Ghosh, Et Al., "Large-Scale Mapping Of The Massive Star-Forming Region RCW38 In The [CII] And PAH Emission", 556, A92, A&A, 2013
[3]. B. Mookerjea, S. K. Ghosh, H. Kaneda, Et Al., "Mapping Of Large Scale 158 Microns [CII] Line Emission: Orion A", 404, 569, A&A, 2003
[4]. T. Suzuki, S. Oyabu, S. K. Ghosh, Et Al., "[CII] Emission Properties Of The Massive Star-Forming Region RCW 36 In A Filamentary Molecular Cloud", A&A, 2021