Computer Optics Vol. 2, No. 2, pp. 207–210, 1990
Printed in Great Britain

0955-355X/90 $10.00 + 0.00

MICROCOMPUTER-BASED SELF-CONTAINED SUBSYSTEM FOR PRIMARY DATA ACQUISITION IN DISTRIBUTED HETEROGENEOUS FILE TRANSMISSION SYSTEM KERMIT

L. I. BRUSILOVSKY, L. R. KONKINA and V. V. SKLYAROV

Abstract—The paper is devoted to the hardware and software of a self-contained subsystem for automated primary acquisition of data to be transmitted to an instrumentation/computing system. The transfer of data between the microcomputer-based acquisition and transmission station and an instrumentation/computing system is effected with the KERMIT transmission protocol.

INTRODUCTION

The paper describes CATS, a self-contained subsystem for automated primary acquisition of data to be transmitted to an instrumentation/computing system (ICS). The subsystem is intended to be a part of an integrated computer-aided system which performs the primary data acquisition and processing in computer optical (CO) element design, controls production and testing of CO components, analyses experimental data, etc.

HARDWARE AND SOFTWARE

The computer-aided primary data acquisition and transmission station (CATS) is based on the popular single-board microcomputer Elektronika/NC-80.

Any computer complete with an appropriate set of processing resources can serve as an ICS. In general, a CATS/ICS system comprises heterogeneous computing equipment. The transfer of data between CATS and ICS is effected with the KERMIT transmission protocol that has been chosen on the following grounds.

(i) KERMIT supports reliable data transmission between dissimilar computers.
(ii) KERMIT permits data transmission via standard communication lines (i.e. switched or dedicated communication lines) without the need for special network interfaces or expensive network adapters.

Having gained some experience in using this protocol with assorted computers and operating systems [1], we chose to build a CATS on the basis of Elektronika/NC-80 with the KERMIT protocol.

The CATS software may be broken down into three major sections:

— supervisor;
— object software that manages data acquisition from various input/output devices linked up with the station;
— networking software used to transmit data between CATS and ICS. This software implements the KERMIT protocol and will be described in more detail below.

KERMIT—FORMAL DESCRIPTION

The KERMIT protocol is designed to transfer data files between dissimilar computers via sequential communication lines [2].

KERMIT is a character-oriented protocol which supports character binary data transmission in both full- and half-duplex modes. It is worth noting that KERMIT can send both separate files

and groups of files by using generic names. The name of a file is represented in the standard format:

⟨name⟩·⟨type⟩

A unit of KERMIT data is a "packet", that is, a limited sequence of up to 94 characters, consisting of data fields, reserved and control fields to provide errorfree and complete data transmission. Every packet received under KERMIT should be acknowledged. This packet has the following format:

MARK | LEN | SEQ | TYPE | DATA | CHECK

Every field consists of ASCII characters, with the following Field Contents:

MARK: this character identifies the beginning of a packet: CTRL-A, by default;
LEN: packet length minus 2, that is, the number of characters within the SEQ, TYPE, DATA and CHECK fields, the maximum value being 94;
SEQ: sequential number of a packet modulo 64: each group of 64 packets is followed by SEQ reset to 0;
TYPE: packet type which may take on one of the following forms:
- D: data packet;
- Y: ack, successful transmission acknowledgement;
- N: nak, transmission is not acknowledged;
- S: send-init, initialization of sending data;
- B: break, termination of transaction;†
- F: file heading;
- Z: end of file;
- E: transmission error;
- T: reserved for interval use.

In addition, the following auxiliary types can be used in the SERVER mode:

- A: file attributes;
- R: initialization of receiving data;
- C: system commands;
- K: KERMIT command;
- G: generic KERMIT commands along with subcommands;
- I: initialization of data transmission process;

DATA: packet "contents" which is interpreted according to the packet type. For such types as C, K or G, the DATA field contains the text of the command;
CHECK: checksum for all fields except MARK and CHECK.

Thus, all KERMIT packet types are of the same format and they differ only in field contents. A special format is assigned to an S-packet containing data transmission parameters set at the beginning of transmission.

In addition to the packet format, KERMIT specifies the order of file transmission during a single transaction. Packets are sent in the following order.

(1) The sending program (Sender) sends an S-packet specifying the transmission parameters; a receiving program (Receiver) acknowledges receipt of the packet by issuing a Y-packet.
(2) Sender transmits an F-packet (the file heading) which contains the file name within the data field; Receiver acknowledges the packet reception by commandeering an idle data field or a name (in the data field) under which the file is to be stored.
(3) Sender transmits the file contents by D-packets, each being acknowledged by the receiver.
(4) Once the transmission of the file is completed, Sender issues a Z-packet (end-of-file). If there are more files to be sent, then the process will commandeer another F-packet; otherwise, Sender transmits a B-packet (end-of-transaction).

† Here, by "transaction" we mean a session of packet transmission which begins with an S-packet and terminates with a B- or E-packet. A session involves one-way transmission of one or more files.

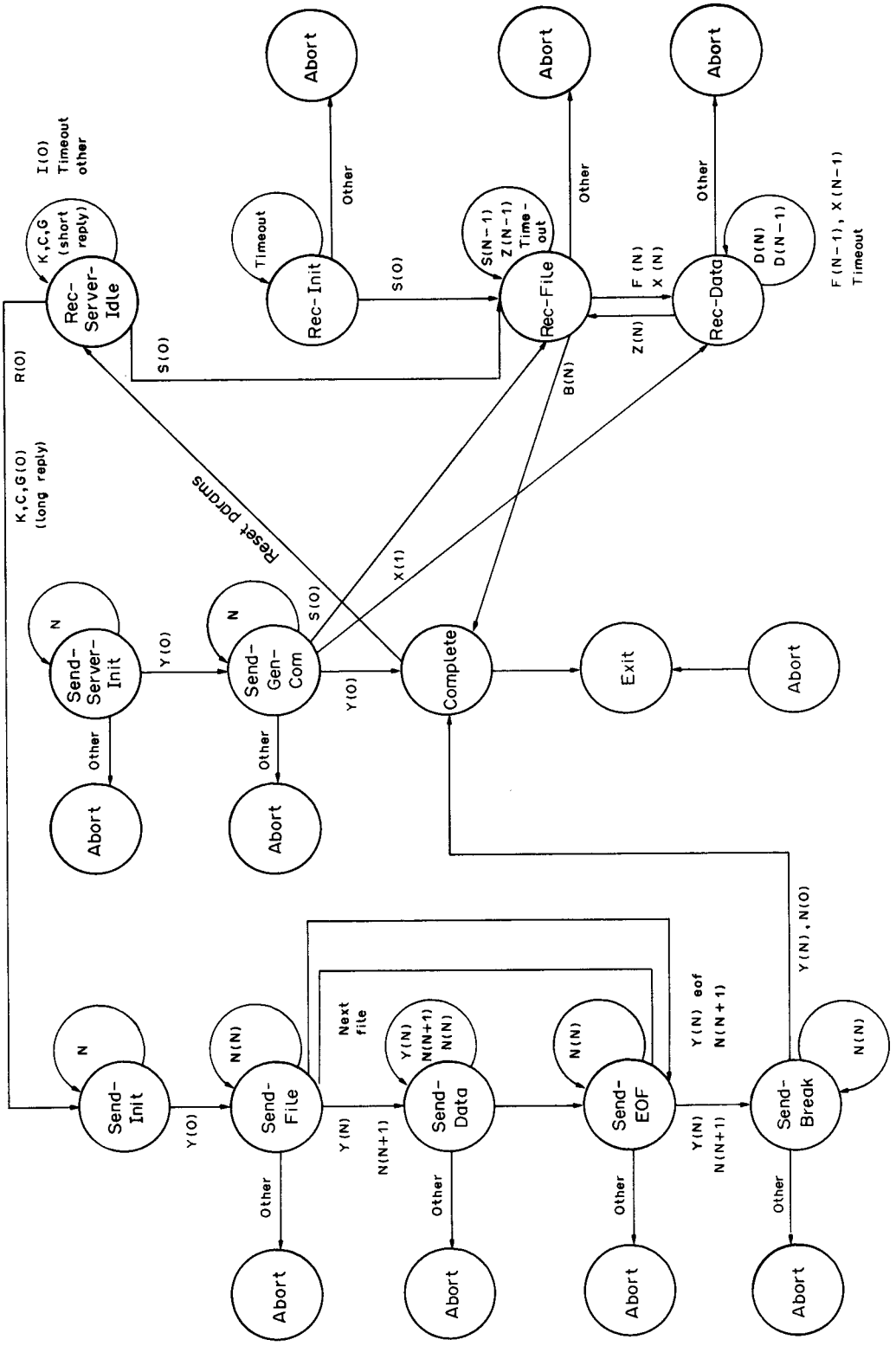
Rec-Server-Idle
Rec-Init
Rec-File
Rec-Data
Send-Server-Init
Send-Gen-Com
Send-Init
Send-File
Send-Data
Send-EOF
Send-Break
Complete
Exit
Abort
Reset params
K,C,G (short reply)
I(0) Timeout other
K,C,G(0) (long reply)
R(0)
S(0)
Timeout
Other
S(N-1) Z(N-1) Time-out
F(N) X(N)
Z(N)
B(N)
D(N) D(N-1)
F(N-1), X(N-1) Timeout
X(1)
Y(0)
N
N(N)
Y(N) N(N+1)
Next file
Y(N) eof N(N+1)
Y(N), N(O)
Y(N) N(N+1)

Fig. 1.

If a fatal transmission error occurs, an E-packet is sent causing abnormal termination of the transaction.

The KERMIT protocol can support data transmission in two modes, the basic mode and the SERVER mode. SERVER is the KERMIT routine initiated at the far end of the line. In the basic mode, the user issues control commands from the receiving and sending ends while KERMIT is initiated at both ends of the line. In the SERVER mode the transmission process is completely controlled from one end of the line once the routine has been initiated at the far end in the SERVER mode. In the course of a transaction, SERVER receives all commands in C-, K-, R- or G-packets.

Figure 1 shows the state diagram of a receive/transmit node operating in the SERVER mode. Circles represent states of the node, and arrows identify state-to-state transitions with the cause of a transition indicated next to the arrow. A transition may be caused either by a timeout or by receiving a data, control or acknowledgement packet.

The initial state of the node is defined by a user command issued upon the start of the KERMIT routine. If the command is SERVER, then at the outset the node is set to REC-SERVER-IDLE, that is, the state between transactions in which SERVER waits for a control command. GET and HOST commands send the node into the SEND-GEN-COM state or the SEND-SERVER-INIT state in which a control command is sent to SERVER.

States shown in the diagram:

REC-FILE:	waiting for an F-packet (file handling);
REC-DATA:	waiting for a date or EOF packet;
SEND-INIT:	initialization of sending data;
SEND-FILE:	sending an F-packet;
SEND-DATA:	sending a data package;
SEND-EOF:	sending an EOF-packet (end-of-file);
SEND-BREAK:	sending a B-packet (end-of-transmission);
COMPLETE:	successful termination of transmission;
ABORT:	abnormal termination of transmission.

The SERVER mode is convenient since it does not require on-line user interaction at the far end. It is of great importance for our task since the station must operate off-line, without intervention of the user.

CATS SOFTWARE IMPLEMENTATION

The CATS networking software implements the SERVER mode of KERMIT.

To meet the memory saving requirement of the software (64 K-byte addressed memory space), the data transmission routine only provides the key features of the protocol, such as execution of SEND, RECEIVE, GET or CONNECT commands in the SERVER mode as well as a number of commands required for on-line conversation with a remote SERVER. But the routine is open to expansion, so the designer of CATS may realize the extensive capabilities of KERMIT by using Elektronika/NC-80.3 complete with a 256 K-byte addressed memory.

The CATS software is developed and debugged on the basis of ICS with the Micropower Pascal environment, which gives an opportunity to write self-contained application programs. Then, a finished application may be loaded to the CATS to be executed off-line, with the loading carried out via the ICS-CATS communication lines, or directly from PROMs where CATS applications are wired.

REFERENCES

1. L. I. Brusilovsky, E. A. Otlivanchik *et al.* Use of file transmission package KERMIT to interface conventional automation systems based on heterogeneous computing equipment. *Komputernaya Optika* No. 2, MTsNTI, Moscow (1987).
2. Frank da Cruz. *Kermit Protocol Manual*, 5th edn. CUCCA, New York, 3 April (1984).