FCP

Five-bit Communications Protocol

Introduction

The *Five-bit Communications Protocol* (FCP) is a method of communicating, similar to Baudot or Morse code, which uses two states (expressed as '1' and '0') to represent characters in fixed five-bit code sizes and also defines a standard procedure for transmissions. This protocol is designed to be easily retained in memory. It can also, however, be used to send and receive transmissions effectively without requiring complete memorization of the codes by following a written reference sheet. The character sets can easily be recreated and written down on a sheet of paper with a minimal amount of memorization of the patterns in the character sets. This protocol can be used to communicate between two persons provided there is some way to represent two distinct states (expressed as '1' and '0') to send the five-bit characters.

The Five-bit Code

The two states ('1' and '0') can be represented by anything that can be used for communication: a high and low toned sound, a short and long pulse of sound or light, two different colored lights, waving a flag from a distance, or any other way that two distinct states can be represented. Using these two states in a group of five-bits to represent each character gives a total of 32 unique combinations. You can create a list of these 32 five-bit combinations by starting with the '00000' combination first and then counting in binary until you reach the combination '11111'. A quick way to do this is to write down each of the bit columns one at a time for all the 32 values. First you start on the right side with the last bit-column. Starting with '0' you will alternate between '0' and '1' (01010101...) until you reach the bottom of the 32 values. Move over to the left one column and again starting from the top with a '0' alternate between '0' and '1', but this time alternate in groups of two by writing two '0's then two '1's (00110011...) until you reach the bottom. For each column you move to the left you double the number of times you repeat the '0's or '1's before you alternate. So, for each column, the number of times you write a '0' or '1' before you alternate will be for the last column: 1, the next column to the left: 2, then 4, then 8, and ending with the first column repeating 16 times before alternating.

16	8	4	2	1	Alpha
0	**0**	0	0	0	Space
0	**0**	0	0	1	A
0	**0**	0	1	0	B
0	**0**	0	1	1	C
0	**0**	**1**	**0**	0	D
0	**0**	**1**	**0**	**1**	**E**
0	**0**	**1**	1	0	F
0	**0**	**1**	1	1	G
0	1	0	0	0	H
0	1	0	0	1	I

When receiving these five-bit characters from someone, you can use a reference chart of the five-bit character codes to determine which character is being sent without having to memorize the entire code chart. When receiving a five-bit code you can use a process of elimination to find what character is being sent to you. With each one of the five bits that come in from the sender you can eliminate half of the codes. For example, if you receive the five-bit value '00101', with the first bit '0' you can eliminate the bottom half of the 32 codes that all begin with '1' and focus on the upper half that start with '0'. When you receive the second bit '0' you can again eliminate half of the remaining 16 codes and focus on an increasingly smaller group. After receiving all five bits you will have narrowed it down to only one character that will remain. You can see this demonstrated in the table above where the bits in bold show how the character {E} was found by narrowing down the list. When you are sending five-bit characters you can also use the character chart in the opposite way without having to remember all of the codes. You will probably be able to memorize many of the five-bit codes and their values from frequent use, but memorizing the chart is not required in order to be able to use the protocol. Also, remember when sending five-bit characters that you place a pause, at least the length of sending one-bit, between each character that you send so that they will not all run together.

The Four Character Sets

Once you have created a list of all of the 32 five-bit codes, you can then list the four character sets to the right of these values. The first and default character set is the lower-case alpha set. The first character '00000' is the 'SPACE' character. From '00001' to the value '11010' are the letters A-Z in alphabetical order. The last five codes of the 32 values are reserved for special control purposes in all of the character sets (more on these later). The next character set is the upper-case alpha set. This is exactly the same as the lower-case alpha set, except that the letters A-Z are all upper-case. The third character set is the numeric set. The first ten characters are the decimal numbers 0-9 matching their equivalent binary numbers from the five-bit values. The remaining values in the numeric set are reserved for characters commonly used in mathematical expressions. The fourth and final character set is for special characters, which contains all of the non-control ASCII characters from a standard keyboard that are not already contained in the alpha or numeric character sets. Each special character or symbol is matched with an alpha character to help in remembering what symbol goes with each five-bit value.

On the next page are the Four Character Sets of the FCP Code Chart.

5-Bit	**Alpha**	**Alpha**	**Numeric**	**Alpha**	**Special Characters**
					(Spell-Out Char)
00000	[SPACE]	[SPC]	0	[SPC]	@ [AT SYMBOL]
00001	A	A	1	A	\ [BACK SLASH]
00010	B	B	2	B	, [COMMA]
00011	C	C	3	C	$ [DOLLAR]
00100	D	D	4	D	! [EXCLAMATION]
00101	E	E	5	E	/ [FORWARD SLASH]
00110	F	F	6	F	> [GREATER THAN]
00111	G	G	7	G	- [HYPHEN]
01000	H	H	8	H	: [COLON](i)
01001	I	I	9	I	; [SEMI-COLON](j)
01010	J	J	([OPEN PARENTH]	J	* [ASTREK/STAR]
01011	K	K	) [CLOSED PARENTH]	K	< [LESS THAN]
01100	L	L	÷ [DIVIDE](/)	L	" [DOUBLE QUOTE]
01101	M	M	× [MULTIPLY](*,·)	M	' [SINGLE QUOTE]
01110	N	N	- [NEGATIVE/MINUS]	N	| [PIPE]
01111	O	O	+ [PLUS]	O	. [PERIOD]
10000	P	P	. [POINT/PERIOD]	P	? [QUESTION]
10001	Q	Q	= [EQUALS]	Q	(LINE)[RETURN]
10010	R	R	[[OPEN BRACKET]	R	& [AMPERSAND]
10011	S	S	] [CLOSED BRACKET]	S	(LINE)[TAB]
10100	T	T	{ [OPEN BRACE]	T	_ [UNDERSCORE]
10101	U	U	} [CLOSED BRACE]	U	^ [CARET]
10110	V	V	^ [EXPONENT/CARET]	V	~ [TILDE]
10111	W	W	[UP ARROW]	W	% [PERCENT]
11000	X	X	[DOWN ARROW]	X	` [GRAVE ACCENT]
11001	Y	Y	[LEFT ARROW]	Y	# [POUND/NUMBER]
11010	Z	Z	[RIGHT ARROW]	Z	
11011	[Lower-Case/Default]				
11100	[Upper-Case]				
11101	[Numeric]				
11110	[Special]				
11111	-CONTROL-				

Calling Codes

101010	Request Start Call
111111	Answer Start Call
010101	Request End Call
000000	Answer End Call

International Characters (from Numeric Set)

P	[CEDILLA/RING/MISC.] (ex: Ç/Å)
V	[CIRCUMFLEX] (ex: Û)
W	[TILDE] (ex: Ñ)
X	[UMLAUT] (ex: Ö)
Y	[GRAVE ACCENT] (ex: È)
Z	[ACUTE ACCENT] (ex: Á)

The Five Special Purpose Codes

The lower-case alpha set is the default set that you use to interpret the five-bit code values you receive. The last five special purpose codes are used for changing to another character set. To use just one symbol at a time from another character set, all you need to do is send a Lower-Case, Upper-Case, Numeric, or Special five-bit shift code and the character sent immediately after that will be interpreted from that character set. For instance, to send an exclamation point you would send {Special}('11110') and then {E}. All of the characters following after that will be interpreted back in the character set that was previously being used. If you want to use more than one character from another character set you can lock to it by sending a pair of Lower-Case, Upper-Case, Numeric, or Special five-bit shift codes. All characters sent after locking to another character set will be interpreted from that set. To change back to the default set you would need to send two lower-case ('11011')('11011') five-bit shift codes to lock to that set.

One of the characters in the Special character set, the {Spc}, is used to spell-out a character that is not included anywhere else. If you ever receive {Special}{Spc}, the following alpha characters will spell-out the character desired until another {Spc} is sent to indicate the end of defining the special character. For example, you could send {Special}{Spc}COPY{Spc} for the © symbol or {Special}{Spc}CENT {Spc} for the ¢ symbol. International characters can also be expressed by using one of six codes from the Numeric set (represented in alpha as P and V-Z) before any Lower-Case or Upper-Case character to modify it with an acute accent, grave accent, umlaut, tilde, circumflex, etc. Sending {Num}P before A will give you Å, {Num}V U gives you Û, {Num}W N gives you Ñ, {Num}X O gives you Ö, {Num}Y E gives you È, and {Num}Z A will give you Á.) These six codes in the Numeric set (P and V-Z) only behave as international character modifiers when they are accessed individually using a single Numeric shift code from within any other locked character set (such as the default Lower-Case set). Only when you are locked in the Numeric set will these six codes represent the four arrows (or mathematical variables w,x,y,z), the exponent, and decimal point characters. The exponent and decimal point characters in the Numeric set are redundant for convenience of use within mathematical expressions and are available in the Special character set for use individually from any other character set. The last special purpose code '11111' is called the CONTROL character. This can be used for various control purposes such as to indicate the end of a string (or packet) of characters being sent during a transmission, for confirming that a string of characters has been received and understood without errors, and for other control purposes that will be mentioned later.

Starting and Ending "Calls" or Transmissions

Besides the five-bit character sets, this protocol also includes a standard procedure for sending and receiving transmissions. To begin making a formal transmission you need to first request for someone to answer your “call”. You do this by sending one of the six-bit calling sequences to the other person. To request to send a transmission to someone you would send the starting request sequence '101010' every few seconds until you get the other person's attention and they respond with the answering sequence '111111'. Once you receive the answering sequence you know the other person is ready to receive your transmission. When you are done making your transmission and you want to end the “call”, you would send the ending request sequence '010101' (the inverse of the starting sequence) and the other person will respond with the ending sequence '000000' to confirm that the “call” is complete and no more transmitting is expected from you.

The Transmission Header or Envelope

Once a “call” has been answered with a '111111' from the other person, the first thing you will do is send header information or an envelope for your transmission. This includes who the transmission is “To” and “From” (in case the transmission is being relayed to someone else) and also which format you will be using during the transmission (which will be explained later). To send the envelope you first start by sending one {CTR} (or CONTROL) character '11111', then who it is “To” then a {CTR} spacer, who it is "From" then a {CTR}, and then a few Alpha characters representing the format you will be using, and then lastly two {CTR}{CTR} characters to end the envelope. To say it again another way, a single {CTR} character is used to begin the header, two {CTR}{CTR} characters are used to end the header, and a single {CTR} is used in between each field (or value) expressed in the header. With the “To” and “From” fields you can use whatever naming convention you want to specify who the transmission is “To” and “From”, such as a station identifier. If you are just sending a transmission from point-to-point where address information is not needed or applicable, you can send two {CTR}{CTR} characters to begin your envelope instead of one and then leave out the “To” and “From” fields. A sample header or envelope for a transmission could look like this: {CTR}TO{CTR}FROM{CTR}TXT{CTR}{CTR} or {CTR}*(address excluded)*{CTR}TXT{CTR}{CTR}. After sending the envelope, the person receiving the transmission will send the {CTR} character to indicate that it was received and understood and that the sender can begin sending the body of the transmission using the format specified in the header information.

Sample Transmission Formats

Three sample formats for use in transmissions are included below. When using a particular format you will place the format's name in the “Format” field in the transmission envelope. Any number of extra fields could also be included after the “Format” field in the transmission header that are specific to and defined by the format, each separated by a single {CTR} character. You could also use the extra fields to create layers of formats. For instance, you could have an encryption format called ENCRYPT where you define the next field to represent any other format that you want to use under the ENCRYPT format. An example might be {CTR}TO{CTR}FROM{CTR}ENCRYPT{CTR}TXT{CTR}{CTR}. The three sample formats included below are TXT, CHAT, and DRAW.

TXT

The TXT format is used to send a one-way text message to someone, similar to sending an e-mail. In this format, after receiving back the {CTR} confirmation character after sending the envelope, you begin the body of the transmission by just starting to send a string of characters from the text message you want to send. Every once in a while, after sending a string of several characters, you will want to send the {CTR} character to check and see if the receiver is getting everything. If everything is okay then the receiver will send back the {CTR} character to indicate everything is fine and that you can continue sending another string of characters. Once you are finished sending the last set of characters in the text message, and after you get the confirmation {CTR} character back from your last string, you will end the transmission in the standard way by the person who initiated the “call” sending '010101' and the receiver replying back '000000' to end the “call”. If the receiver of the text message gets confused or misses something during a transmission and an error occurs, instead of just replying back with only the {CTR} confirmation character after receiving a string, the receiver will first send a string of characters telling the sender what went wrong and what to do to correct it before sending the {CTR} character. Some examples of what might be sent back to the message sender when an error occurs could be: “RPT ALL{CTR}”, “RPT 2ND WRD{CTR}”, or “TOO FAST{CTR}”. If the sender makes an error in the transmission, the sender should send two {CTR} characters then an error message string ending with {CTR}. The receiver will then send a {CTR} confirmation character and the sender will continue with the transmission again. Remember that the {CTR} character sent at the end of a string of characters does not mean a carriage or line return. If you want to indicate a line return in your message you need to send {Special}{R}. Also, a MAIL format could be developed for retrieving multiple relayed messages that may be waiting for someone at another station.

CHAT

The CHAT format is used to chat back and forth between you and the other person who answered your “call”. In this format, after receiving back the {CTR} confirmation character following the envelope, you begin the body of the format by just starting to send a string of characters of a line of dialog to the other person. When you are finished talking you send the {CTR} character to indicate you are done and then the other person will begin sending you a line of dialog in the same manner. You continue back and forth like this until you are finished chatting and you then end the “call” in the standard way. Any errors you encounter in the CHAT transmission can just be corrected by including your error reply as part of the dialog of your CHAT session.

DRAW

The DRAW format can be used to send a diagram or picture to another person by using a grid and sending standard drawing commands to recreate a drawing on the other side. This format is more complicated than the simple TXT and CHAT formats, but it may be useful to have a standard way to send something graphically to someone, like a map or simple picture, if you would ever need to use it.

This format has two extra fields included after the “Format” field in the transmission envelope that indicates the dimensions of the grid that will be used in recreating the drawing. After the “Format” field in the header you will have another {CTR} separator followed by the *xDimension(horizontal)*, another {CTR} separator, and then the *yDimension(vertical)*, and then ending the header with {CTR}{CTR}. A sample header could look like this: {CTR}TO{CTR}FROM{CTR}DRAW{CTR}10{CTR}15{CTR}{CTR}. With the DRAW format you will automatically interpret the characters in the two "X" and "Y" fields as being in the Numeric character set since those values are expected to be numeric in this format. The *x* and *y* dimensions in DRAW format will start with the origin in the upper left-hand corner of the grid. The *x* value will run along the upper part of the grid increasing from left to right, and the *y* value will run down the left side of the grid increasing from top to bottom. After sending the transmission envelope remember to be patient in receiving back the confirmation {CTR} character. If the other person is not writing down the drawing instructions but drawing it “live by ear”, the person on the other end will be busy creating the grid with the dimensions you gave. The receiver should send back a string telling the sender how long it will take to create the grid followed by the {CTR} confirmation character after receiving the envelope if necessary.

Once you are clear to begin sending the body of the DRAW format, you will send one of the standard instructions at a time, similar to the way you send a string of characters using the TXT format. Remember to be patient in receiving back the {CTR} confirmation character from the receiver after sending each instruction if the person is doing it “live by ear”. The seven standard drawing instructions are listed below. Each instruction begins with a single alpha character to indicate what drawing action to take, followed by a set of numeric values specifying coordinates or points on the grid. With the body of the DRAW format you will automatically interpret the five-bit character values received (after the initial alpha character instruction) as being in the numeric character set to save transmission time from frequently shifting character sets. Also with this format you will use the {CTR} character '11111' as the separator or spacer between values in the instructions (represented below as an “_”). Since a single {CTR} character is used as a spacer between values, you will need to use double control characters {CTR}{CTR} at the end of each DRAW command line. You will handle errors in the same way as you do in the TXT format (except if the sender makes an error then three {CTR} characters should be used).

Below are the standard drawing commands. [Points or coordinates are indicated as (*x*,*y*)]

Line **–** L_*x1_y1_x2_y2...* [Two or more (*x, y*) coordinates connected by a line, ex: L_5_7_20_35 means draw a line from point (5,7) to (20,35)]

Rectangle **–** R_*x1_y1_x2_y2* [ex: R_5_7_20_35 means draw a rectangle with the upper-left corner at point (5,7) and the lower-right corner at point (20,35)]

X Mark Spot **–** X_*x_y* [ex: X_5_7 means mark a spot at the point (5,7)]

Circle **–** C_*x_y_radius* [ex: C_20_35_5 means draw a circle with the center at point (20,35) and that has a radius of 5 units.]

With the next two instructions you will not interpret the characters (following the initial alpha character instruction) as in the numeric set but instead in the default alpha set since these instructions deal with text.

Write **–** W_*text* [ex: W_HOME means to write the text label "HOME" on the last object drawn. For instance, you could label a line "Main Street", a marked spot "Home", a rectangle "Village", or a circle "Lake".]

Text Instruction **–** T_*text instruction* [ex: T_COLOR RED could mean to change to a red color before drawing the next instruction. The *Text Instruction* command is used to perform any other special drawing instruction that cannot be done with the standard commands or to do a standard instruction in a special way, such as drawing the next line command as a zig-zag line instead of a straight line.]

The last command is *Set Value*. This may be used to associate a certain numeric value to a series of alpha characters, sort of like a variable or symbol representing a point on the grid, where that series of alpha characters can be used in place of a numeric value in any drawing instruction. With the DRAW format, when using a variable in a drawing instruction, you will need to use an alpha shift character (only once) before giving the alpha representation of a set numeric value so that it is not interpreted as from the numeric character set. An alpha shift character will let you know that the characters following (up to the next {CTR} separator) are not a numeric value but a variable (or representation) of a numeric value.

Set Value **–** S_*AlphaCharacters_NumericValues...* [ex: S_a_5_7 means make the Alpha character "a" represent "5_7"]

Below is an example DRAW format transmission to give you a better idea of how it works:

'**101010**' (*Request Start Call*)

'**111111**' (*Answer Start Call*)

{CTR}TO_FROM_DRAW_40_50{CTR}{CTR} (*Transmission Header/Envelope*)

{CTR} (*Confirmation sent back to begin the body of the transmission*)

(*The DRAW command end line double control characters* {CTR}{CTR}*and the confirmation* {CTR} *character that is returned from the receiver will not be shown for the instructions below.*)

L_12_32_20_50

L_10_17_4_12_30_27 (*Draw a line from* (10,17) *to* (4,12) *to* (30,27).)

W_PATH (*write the label* “PATH” *to the last object drawn.*)

S_a_3_4

S_b_5_7

S_c_10_20

L_a_b (*expanded as* L_“3_4”_“5_7”)

L_b_c

L_a_c

S_home_50_120

X_home (*expanded as* X_“50_120”)

W_HOME

C_b_5 (*code appears as* {C}_{Shft}{B}_{5})

W_LAKE

T_FILL IN LAKE BLUE (*a special text instruction*)

L_home_a (*code appears as* {L}_{Shft}{H}{O}{M}{E}_{Shft}{A})

R_20_10_a (*expanded as* R_20_10_“3_4”)

'**010101**' (*Request End Call*)

'**000000**' (*Answer End Call*)