HRFCP

Human Retainable Five-bit Communications Protocol

Introduction

This *Human Retainable Five-bit Communications Protocol* (HRFCP) draft is a method of communicating, similar to using Morse code, that uses two states (expressed as '1' and '0') to represent characters in fixed code sizes of five-bits and which also defines a standard procedure for transmissions. This protocol is designed to be easily retained by humans in memory. It can also, however, be used effectively to send and receive transmissions by following a written reference sheet without requiring complete memorization of the codes. The character sets can also be easily recreated and written down on a sheet of paper with little memorization of the patterns in the character set layouts. This protocol can be used to communicate between two persons where there is a 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'), used to send the five-bit codes, can be represented by anything that can be used for communications: 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 you can represent two distinct states with something. Using these two states in a group of five-bits to represent a character will give us a total of 32 unique combinations. You can create a list of these 32 combinations by starting with the '00000' combination first and counting in *binary numbers* (from 0 to 31) until you reach the combination '11111'. A quick way to do this is to write down one of the bit columns for all the 32 values at a time until you finish with the first column on the left side. First you start on the right side with the last bit-column of the five-bit values. 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 write two '0's before alternating and then two '1's (00110011) until you reach the bottom. For each column you move to the left you double the number of times in a row you repeat a '0' or a '1' before you alternate. So, for each column the number of times you keep writing either '0' or '1' before alternating will be for the last column 1, then the next column 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 possible character that will remain. You can see this example 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 without having to remember all of the values. First you will look for the character you want to send and then send the five '1's and '0's corresponding to character. When sending characters you should place a pause, at least the length of sending one-bit, between each five-bit character that you send. You will probably be able to memorize many of the five-bit codes and their values by heart from frequent use, but memorizing the chart is not required in order to be able to use the protocol effectively.

The Four Character Sets

Once you have created and listed all 32 five-bit values you can then list the four character sets to the right of these values. The first and default character set is the lowercase 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 characters of the 32 values are reserved for special purposes in all of the character sets (more on these later). The next character set is the uppercase Alpha set. This is exactly the same as the lowercase Alpha set, except that the letters A-Z are all uppercase. 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 last character set is for Special Characters, which contains all of the characters on a standard American-English keyboard that are not already contained in the Alpha or Numeric character sets. Each symbol is matched with an Alpha character to help in remembering what special symbol goes with each five-bit value.

Below are the Four Character Sets. *Click Here to see a stand-alone HRFCP Code Chart.*

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

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

The Five Special Purpose Characters

Of all of the four character sets, the lowercase Alpha set is the default set that you use to interpret the five-bit code values you receive. The last five special purpose characters are used for changing to another character set. To temporarily use one symbol from another character set all you do is send either the Shift, Numeric, or Special five-bit character, 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 following characters sent after that will be back in the default character set that was previously being used. (*One of the characters in the Special character set, the* {Spc}*, is used to spell-out a character that you want that isn't included anywhere else. If you ever receive* {Special}{Spc} *then the following Alpha characters sent 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.*) If you need to send more than one character at a time from another character set you can use the special purpose {Lock/UnLock} character ('11011'). After specifying what character set you will be using you send the {Lock} character to specify where the locking begins, and then after sending the characters from the character set that you wanted to use, you send the {Lock/UnLock} character again to indicate that you are ending the locked mode and switching back to the default character set. For instance, to send the word “HELLO” in all uppercase you would send {Shift}('11100') then {Lock}('11011') then the characters HELLO and then {UnLock}('11011') to return back to the default character set. While sending several characters in a locked mode, you may want to temporarily use a few characters from the default set without having to 'Lock' and 'Unlock' a character set multiple times. In this case you can, in the middle of a locked mode, just send the {UnLock} character then use a few characters from the default set and then just send the {Lock} character again by itself to relock and continue in the locked mode that you were in previously. For example, in sending a math expression like 10-3ax+5=30 you may just send {Num}{Lock}10-3{UnLock}AX{Lock}+5=30{UnLock} instead of sending {Num}{Lock} every time you switch back to the Numeric set after using a few characters from the default one. The last special purpose character '11111' is called the CONTROL character which can be used 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 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 will 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 will send the ending request sequence '010101' (the opposite 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 an '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 from 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 two {CTR} (or CONTROL) characters '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} characters to end the envelope. Once more, the double {CTR} characters are used to begin and 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”. If you are just sending a transmission from you to whoever happens to be receiving the transmission, you could just use a generic {T} character for the “To” field and an {F} character for the “From” field. A sample generic header or envelope for a transmission could look like this: {CTR}{CTR}T{CTR}F{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 Standard Transmission Formats

Three sample standard 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. 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}. The three formats included below are TXT, CHAT, and PIC.

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 several characters, you will 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 at the end of the current 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} confirmation character. Some examples of what might be sent to the message sender when an error occurs could be: “RPT ALL{CTR}”, “RPT 2ND WRD{CTR}”, or “TOO FAST{CTR}”. 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}.

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 after sending 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.

PIC

The PIC format can be used to send a diagram or picture to another person by following 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*, another {CTR} separator, and then the *yDimension*, and then ending the header with {CTR}{CTR}. A sample header could look like this: {CTR}{CTR}T{CTR}F{CTR}PIC{CTR}10{CTR}15{CTR}{CTR}. With the PIC 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 PIC format will start with zero 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 isn't writing down the drawing instructions you give but drawing it “live by ear”, the person on the other end may be creating the grid with the dimensions you gave. However, the receiver should let the sender know if it will be a while before everything is set up to begin receiving the instructions.

Once you are clear to begin sending the body of the PIC 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. You will also handle errors in the same way as you do in 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 PIC 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 'Locking' and 'UnLocking' character sets. Also with this format you will use the {CTR} character '11111' as the separator or spacer between values in the instructions, and is 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 PIC command line. Below are the standard drawing commands. [P.S. points or coordinates are indicated as (*x*,*y*)]

Line **–** L_*x1_y1_x2_y2* [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 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, for instance, to draw 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 set a certain numeric value to a series of Alpha characters, sort of like a variable, where that series of Alpha characters can be used in place of a numeric value in any drawing instruction. With the PIC format, when using a variable in a drawing instruction, you will need to use the {Shft} character (only once) before giving the Alpha representation of a set numeric value so that it isn't interpreted as from the Numeric character set. The {Shft} 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 PIC format transmission to give you a better idea of how it works:

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

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

{CTR}{CTR}T F PIC 40 50{CTR}{CTR} (*Transmission Header/Envelope*)

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

(*The end PIC command 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

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

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 (*code appears as* {S}_{H}{O}{M}{E}_{E}{Spc}_{A}{B}{Spc})

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*)

Click Here to see a stand-alone HRFCP Character Code Chart.

Click Here to Download this Document and the Code Chart in ZIP format.

Home

HRFCP

(Human Retainable Five-bit Communications Protocol)

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

Calling Codes

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

Human Retainable Five-bit Communications Protocol (HRFCP)

View HTML Explanation Document

View HRFCP Code Chart HTML Document

Download HTML Document and Code Chart in ZIP format

Community Survival Action Plan (CSAP)

View HTML document

Download HTML Document in ZIP format